English Edition

Primum non nocere: Software engineering for medicine

From the clinical problem to the regulated product

South American Publishing

Primum non nocere: Software engineering for medicine

From the clinical problem to the regulated product

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

South American Publishing

Authors

Annier Jesús Fajardo Quesada
1 Granma University of Medical Sciences. Granma, Cuba.
https://orcid.org/0000-0002-2071-3716
annierfq01@gmail.com

Eduardo Antonio Hernández González
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.
https://orcid.org/0000-0001-7325-6099

René Herrero Pacheco
1 Granma University of Medical Sciences. Granma, Cuba.
https://orcid.org/0000-0002-9450-1572

Rolando Javier Álvarez Pérez
1 Granma University of Medical Sciences. Granma, Cuba.
https://orcid.org/0000-0003-0606-2993

Copyright Page

© 2026. The authors. This is an open access book, distributed under the terms of a Creative Commons Attribution 4.0 license (https://creativecommons.org/licenses/by/4.0/) that allows use, distribution, and reproduction in any medium provided that the original work is properly cited.

This South American Publishing imprint is published by South American Publishing.

For more information, see South American Publishing's Open Access Policy: https://southam.pub/policies.php

ISBN (English edition, eBook): 978-9915-9876-8-2

ISBN (English edition, Softcover): 978-9915-9876-9-9

This title is available in both print (softcover) and digital (open access PDF) formats.

DOI: 10.62486/978-9915-9876-8-2

Cataloging Data

Cataloging-in-Publication Data (CIP):

South American Publishing

Primum non nocere: Software engineering for medicine / Annier Jesús Fajardo Quesada, Eduardo Antonio Hernández González, René Herrero Pacheco, and Rolando Javier Álvarez Pérez. Montevideo: South American Publishing, 2026.

ISBN (English edition, eBook): 978-9915-9876-8-2

ISBN (English edition, Softcover): 978-9915-9876-9-9

THEMA classification codes:

MBPM

BISAC classification:

MED117000 - MEDICAL / Informatics

COM051230 - COMPUTERS / Software Development & Engineering / General

TEC059000 - TECHNOLOGY & ENGINEERING / Biomedical

MED003040 - MEDICAL / Allied Health Services / Medical Technology

Editorial Notice and Acknowledgments

The publication of this book is part of South American Publishing's mission to promote open, ethical, and rigorous scientific communication across all areas of knowledge.

All books published by South American Publishing undergo a double-blind peer-review process and a technical evaluation in accordance with the publisher's editorial policies, aligned with COPE guidelines.

South American Publishing acknowledges the valuable collaboration of authors, reviewers, designers, and production teams who made this publication possible.

Preface

Medical software occupies a difficult borderland. It is built with the tools of software engineering, yet it enters environments where errors can affect diagnosis, treatment, continuity of care, privacy, and patient safety. The phrase primum non nocere therefore has practical meaning for developers as well as clinicians: a useful product must be technically sound, clinically intelligible, and designed with an awareness of the consequences of failure.

The chapters that follow were organized for readers who work across that border. They move from the foundations of digital health and software as a medical device to regulated development, risk management, human factors, clinical terminology, interoperability, architecture, cybersecurity, artificial intelligence, clinical validation, economic evaluation, ethics, business models, and comparative regulation. The emphasis is not on treating medicine as a software domain or software engineering as a clinical specialty. It is on showing where the two disciplines must understand one another well enough to build systems that can be trusted.

Several tensions recur throughout the book. Speed must coexist with traceability. Usability must coexist with risk control. Data availability must coexist with privacy and semantic consistency. Artificial intelligence may increase analytical capacity while introducing new forms of opacity, bias, and fabrication. Regulation can seem burdensome when viewed from outside healthcare, but many of its requirements are responses to known modes of harm. These tensions do not disappear through a single standard or technical method.

The intended reader may be a clinician trying to understand the logic of software development, an engineer entering the healthcare sector, a researcher working with clinical data, or a multidisciplinary team preparing a product for real use. The book does not assume that one jurisdiction, architecture, or development process will fit every project. Its purpose is to give readers a common language for asking better questions before technical decisions become clinical consequences.

Introduction

Digital health has moved from peripheral experimentation into the routine infrastructure of care. Software now records observations, exchanges clinical data, supports diagnosis, monitors patients, guides therapeutic decisions, and increasingly generates or interprets medical information through artificial intelligence. As software assumes these functions, the boundary between an ordinary application and a regulated medical product becomes consequential. A design choice can affect not only performance or user satisfaction but also safety, evidence requirements, legal responsibility, and the credibility of clinical decisions.

This volume approaches that problem from the full lifecycle of medical software. Part I establishes the digital health ecosystem and the shared vocabulary required by clinicians and engineers. Part II examines regulated development through ISO 13485, IEC 62304, ISO 14971, and IEC 62366, with attention to traceability, risk, quality, and usability. Part III turns to the semantic and technical infrastructure of healthcare, including SNOMED CT, LOINC, FHIR, HL7, DICOM, system architecture, and cybersecurity. Part IV addresses clinical artificial intelligence, from dataset construction and model validation to large language models and explainability. Part V connects technical development with clinical studies, real-world evidence, economic evaluation, algorithmic bias, reimbursement, sustainability, and regulatory pathways.

The sequence is deliberate. A medical software product cannot be evaluated responsibly by looking only at its code, its model accuracy, or its interface. Its meaning depends on the clinical problem it claims to address, the data it receives, the people who use it, the hazards it introduces, the evidence supporting its performance, and the institutional setting in which it will be deployed. The appendices extend this perspective with a glossary, a production-readiness checklist, and working templates for risk management and verification.

The field remains unsettled in important ways. Regulatory frameworks continue to change, artificial intelligence introduces behaviors that are difficult to validate with conventional methods, and systems that perform well in one hospital may not generalize to another. These limits are part of the subject rather than exceptions to it. The chapters therefore treat uncertainty, documentation, multidisciplinary review, and post-market observation as central components of responsible development.

Abstract

Software has become part of the clinical environment, where technical defects, ambiguous data, poor usability, or unvalidated models can translate into direct consequences for patients and professionals. This volume examines medical software as a multidisciplinary product whose development requires the integration of software engineering, clinical reasoning, quality management, regulation, data standards, cybersecurity, and evidence generation. It begins by defining the digital health ecosystem and the boundary between general-purpose software and Software as a Medical Device. It then addresses regulated development through ISO 13485, IEC 62304, ISO 14971, and IEC 62366, linking lifecycle processes with traceability, risk control, verification, and human factors. Subsequent chapters examine clinical terminologies, FHIR, HL7, DICOM, healthcare architectures, and security requirements that support interoperable systems. The treatment of artificial intelligence covers clinical dataset construction, internal and external validation, dataset shift, MLOps, large language models, hallucinations, explainability, and communication of algorithmic risk. The final section considers prospective and retrospective study design, real-world evidence, economic evaluation, bias, informed consent, liability, reimbursement, technology transfer, and regulatory pathways in the United States, Europe, Latin America, and Spain. Throughout the book, technical choices are interpreted in relation to clinical context and patient safety rather than as isolated engineering decisions. The accompanying appendices provide terminology, a readiness checklist, and templates for risk management and verification. The result is a lifecycle-oriented framework for clinicians, engineers, researchers, and multidisciplinary teams working on software intended for healthcare use.

Keywords: Medical software; Software as a Medical Device; Digital health; Clinical artificial intelligence; Medical device regulation; Patient safety

Index

PART I: FOUNDATIONS AND CONTEXT
Chapter 01. The Digital Health Ecosystem 1

Chapter 02. Introduction to Regulated Environments 27

PART II: REGULATION AND QUALITY
Chapter 03. IEC 62304: Medical software lifecycle 50

Chapter 04. ISO 14971: Risk Management 63

Chapter 05. IEC 62366: Human Factors and Usability 79

PART III: DATA AND ARCHITECTURE
Chapter 06. Clinical Terminologies and Languages 95

Chapter 07. Interoperability Standards 111

Chapter 08. Architecture in the Healthcare Environment 133

Chapter 09. Clinical Cybersecurity 151

PART IV: CLINICAL ARTIFICIAL INTELLIGENCE
Chapter 10. Construction and Validation of Clinical Datasets 167

Chapter 11. Model Development and Validation 183

Chapter 12. Generative AI and Large Language Models 198

Chapter 13. Explainability and Trust (XAI) 211

PART V: CLINICAL VALIDATION AND MARKET
Chapter 14. Study Design for Medical Software 225

Chapter 15. Real-World Evidence and Economic Evaluation 246

Chapter 16. Ethics and Algorithmic Bias 262

Chapter 17. Business Models in Digital Health 275

Chapter 18. Comparative Regulatory Landscape 290

Epilogue, appendices and bibliography 308

Chapter 1 / Capítulo 1

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch01
Pages: 1-26

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 1. The Digital Health Ecosystem

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Technology in medicine is like a scalpel: in the right hands, it saves lives; in the wrong ones, it endangers them. The difference lies not in the instrument, but in the person wielding it and the respect with which they approach the human body."

When a doctor listens to a patient with a stethoscope, they do not need to wonder if the instrument is calibrated. They know it's designed to amplify sounds, that it's been around for decades and that its function is clear to everyone. However, when the same doctor opens an app on their tablet to view an X-ray or when an algorithm suggests a possible diagnosis based on specific symptoms, the nature of the tool changes radically. We are no longer dealing with a metal tube and a diaphragm but with lines of code that make decisions, process sensitive information, and, ultimately, can influence clinical judgment. That collection of instructions written by programmers becomes, in fact, a medical device. In addition, as such, it must be treated.

Digital health is not a future promise but an everyday reality that is transforming medicine at breakneck speed. Apps that monitor chronic diseases, algorithms that interpret radiological images, artificial intelligence systems that predict complications, and telemedicine platforms that bring care to remote locations: all these technologies are already present in many of our hospitals and clinics. However, this new ecosystem is not just a collection of tools; it is a complex landscape where a wide variety of stakeholders converge—patients, professionals, regulators, developers, and managers—with often divergent interests, languages, and priorities. Understanding this landscape, its boundaries, and its unwritten rules is the first step toward navigating it safely and ethically.

1.1 SOFTWARE AS A MEDICAL DEVICE

The official definition of software as a medical device, or SaMD, is not a bureaucratic whim but a necessity for protection. The International Forum of Medical Device Regulators defines it as software intended for medical purposes that is not part of a physical device.(1) This means that an app that interprets heart rate using a smartphone's camera is a medical device, whereas a digital calendar for tracking appointments is not, even if it is used in a hospital. The line seems clear on paper, but in everyday practice it becomes blurred. What about an app that helps diabetic patients keep track of their meals? What if it also suggests insulin adjustments? Suddenly, the same tool has crossed an invisible line, and its consequences become potentially serious.

The implications of this distinction are enormous for everyone involved in the process. For the doctor who recommends an app to a patient, it means the peace of mind that comes from knowing that the tool has passed basic safety and efficacy checks. For the developer who builds it, it means assuming legal responsibility that goes beyond typical programming errors. In addition, for the patient—the most vulnerable link in the chain—it means that their health is not left in the hands of a program someone wrote in their garage with no guarantee other than their good intentions. That's why it's so important that anyone involved in creating health technology understands from the very beginning what they're actually building.

1.1.1 WHERE THE APP ENDS AND THE MEDICAL DEVICE BEGINS

Let us consider a specific example to help visualize this boundary. Imagine an app designed to help patients with hypertension keep track of their blood pressure readings. If the app merely stores the values that the user enters manually and displays them in a graph, it likely would not be considered a medical device. However, if that same app connects to a digital wristband, receives readings automatically, and, on the basis of them, advises the patient when to go to the emergency room or adjust their medication, then we are dealing with software that performs a medical function. The code hasn't changed drastically, but its purpose has—and with it, the responsibilities it entails.

This distinction is not new in the history of medicine, although the technological support is. For centuries, medical instruments have evolved from purely mechanical tools to complex electronic systems, and each technological leap has brought new questions about safety and regulation. What makes software different is its intangible nature and its ability to change without the user being aware of it. A dull scalpel is noticeable at the first cut; an algorithm that has been subtly modified may behave erratically without anyone noticing until it is too late. That invisibility of change is perhaps the most unsettling aspect of medical software and the one that demands

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

constant vigilance.

Regulatory bodies around the world have been trying for years to bring order to this frontier territory. The FDA in the United States, the European Medicines Agency, and other national agencies have published guidelines and documents intended to help developers classify their products and understand what level of scrutiny applies to them.(2)However, these guidelines change, evolve, and sometimes become obsolete before they are even published because technology advances faster than regulators can analyze it. This mismatch between innovation and regulation is one of the great challenges of our time and a constant source of frustration for those trying to do things right.

For healthcare professionals entering this field for the first time, it can be overwhelming to hear terms such as regulatory classification, CE marking, or 510(k) clearance. However, it is important for them to understand these concepts because they will ultimately be the ones using these tools with their patients. Just as a doctor would not prescribe a drug without knowing its mechanism of action and side effects, they should not use medical software without understanding what guarantees it offers and what limitations it has. This analogy with pharmacology is more accurate than it seems: both are medical devices, both require studies to support their efficacy, and both can cause harm if used incorrectly.

Perhaps the most important point to take away from this introductory chapter is that medical software is not a special type of software but rather software that has been subjected to additional scrutiny because people's health depends on its proper functioning. A programmer can create a buggy game app, and the worst that will happen is that users will become frustrated and uninstall it. However, if a programmer makes a mistake in an algorithm that calculates radiation doses for cancer treatment, the consequences can be devastating. That asymmetry in the impact of errors is what justifies all the regulatory complexity we will explore in the coming chapters.

As we move through this book, we delve deeper into each of the aspects that make medical software such a fascinating yet demanding field. We will see how it is designed, how it is tested, how it is regulated, and how it is continuously improved. However, before all that, we needed to establish this foundation: to understand that when we talk about software in medicine, we are not talking about entertainment or productivity; we are talking about health. Health, as is well known, does not tolerate errors.

1.1.2 THE THREE FACETS OF RESPONSIBILITY: PHYSICIAN, DEVELOPER, AND PATIENT

When an app becomes a medical device, the chain of responsibility extends and becomes more complex in ways that are not always obvious at first glance. The physician who prescribes or recommends an app is not exempt from responsibility simply because it involves digital technology. In fact, their position becomes more delicate because they must evaluate not only the clinical utility of the tool but also its technical reliability—which is something for which many professionals have not received specific training. This duality of competencies—medical and technological—represents one of the greatest challenges of contemporary medicine and demands a new form of literacy that goes beyond traditional training.

The developer, for their part, enters a realm where their coding decisions have direct consequences for real people. It is no longer a matter of optimizing a database query or improving the user interface to enhance the appeal. Every line of code can affect a patient's diagnosis, treatment, or even survival. This pressure transforms the nature of programming work, which ceases to be a purely technical activity and becomes a practice with inherent ethical responsibility. The medical software developer must learn to think like a healthcare professional, anticipating usage scenarios that go far beyond standard test cases.

The patient occupies the final link in this chain but, paradoxically, is the one with the least information to assess the risks they are taking. When they download a health app onto their mobile phone, they rarely read the terms of service, much less verify whether the software has the appropriate health authorization. They trust—or should trust—that the regulatory system has filtered out potentially dangerous apps. However, the reality is a mo: there are thousands of health apps available in digital stores that operate in gray regulatory areas, offering functionalities that come dangerously close to medical practice without the corresponding safeguards. This situation of information asymmetry is particularly concerning in regard to vulnerable or low-income populations, who may rely exclusively on these tools for their healthcare.

The interaction between these three parties—the doctor, developer, and patient—creates an ecosystem of shared responsibility that is not always well defined in current legal frameworks.

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

When an incident related to medical software occurs, determining who bears responsibility can become a legal maze. Is the doctor who recommended the app responsible? The hospital that purchased it? The developer who programmed it? The platform that distributed it? These questions have no simple answers and vary significantly by jurisdiction, adding an additional layer of complexity to an already intricate field.

1.1.3 THE BLURRED LINE BETWEEN ASSISTANCE AND DIAGNOSIS

One of the most intense debates in the regulation of medical software revolves around the distinction between applications that provide information for the user to make decisions and those that make decisions for the user. This seemingly subtle difference has enormous regulatory implications. An app that displays a diabetic patient's glucose data and allows the patient to interpret trends operates on a different plane than one that, upon detecting an abnormal reading, automatically sends an alert to the emergency department or adjusts the insulin dose of a connected pump. In the first case, we use an information tool; in the second case, we use an automated clinical decision-making system.

The line between these two categories has become particularly blurred with the advent of artificial intelligence and machine learning. Modern algorithms can analyze complex patterns in health data and offer recommendations that, although presented as suggestions, carry an implicit authority that users—both professionals and patients—tend to accept without question. When an artificial intelligence system recommends a diagnosis based on the analysis of a medical image, the radiologist using it may be tempted to confirm that interpretation without the critical scrutiny they would apply to a human opinion. This phenomenon, known as automation bias, represents one of the most insidious risks of contemporary medical software.

Regulators have attempted to address this complexity by establishing criteria on the basis of the software's intended purpose and context of use. However, these classifications quickly become obsolete in the face of the speed of technological innovation. An application initially designed as a simple symptom log can, through a software update, acquire predictive analysis capabilities that place it in a completely different regulatory category. This inherent mutability of software stands in stark contrast to the stability of traditional medical devices, where a design change requires visible and quantifiable physical modification.

This issue becomes even more complicated when we consider the phenomenon of interconnected health ecosystems. A single functionality can be distributed across multiple applications and devices, each of which, when considered in isolation, might not qualify as a medical device but whose interaction produces significant clinical effects. For example, a nutrition app that analyzes food intake, combined with a physical activity monitor and a connected glucose meter, can generate personalized recommendations that are functionally equivalent to a medical treatment plan. Determining where regulatory responsibility lies within this distributed architecture is among the most pressing challenges for health authorities worldwide.

1.1.4 THE WEIGHT OF EVIDENCE IN A DIGITAL WORLD

The transition from software as a consumer product to software as a medical device entails a fundamental transformation in how evidence of its performance is required and evaluated. When a company develops an app for booking restaurant tables, it can launch it on the market, collect user feedback (interactions), and improve it iteratively. This agile development model, which is so efficient in the world of conventional software, clashes head-on with the principles of evidence-based medicine. A medical device cannot be improved on the fly on the basis of user complaints; it must demonstrate, prior to marketing, that it meets rigorous safety and efficacy standards through studies specifically designed for that purpose.

This requirement for premarket evidence creates an inevitable tension between the speed of technological innovation and the caution inherent in the healthcare sector. Software developers, accustomed to weekly or even daily release cycles, find medical regulations to be frustrating and costly obstacles. The timeframes for obtaining marketing authorization can extend over months or years, during which the technology has evolved significantly. This lag presents the paradox that authorized medical software may be technically obsolete before it reaches the first patient—a situation that has no parallel in traditional hardware medical devices.

The regulatory response to this challenge has been the creation of alternative authorization

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

pathways that seek to balance the need for evidence with the reality of technological innovation. Programs such as the FDA's Pre-Cert or the software change assessment pathways in Europe aim to establish frameworks where developers with a proven track record of quality can update their products more quickly.(3) However, these initiatives are still in experimental phases and spark debates over whether they excessively relax safety safeguards. The medical community, legitimately concerned about patient protection, views any mechanism that might allow the commercialization of healthcare software without proper clinical validation with suspicion.

The quality of evidence required also varies significantly depending on the risk associated with the software. An app that calculates body mass indices from manually entered data does not require the same level of testing as an algorithm that analyzes electrocardiograms to detect arrhythmias does. Regulatory systems have established risk classifications that determine the intensity of scrutiny needed, but these categorizations face the challenge of anticipating emerging risks from technologies that do not yet exist. The history of medical device regulation is full of examples in which products initially considered lowrisk turned out to have unexpectedly serious consequences, leading to a general trend toward regulatory caution.

• LESSONS FROM HISTORY: FROM THERAC-25 TO HOSPITAL RANSOMWARE

There is a human tendency to believe that technology's problems are new and that previous generations cannot imagine the ethical dilemmas and risks we face today. However, this belief, in addition to being naive, is dangerous because it deprives us of the wisdom accumulated through mistakes that cost suffering and lives. In the field of medical software, that history is written with tragedies that should be mandatory in any training program for both engineers and clinicians. Knowing them is not an exercise in erudition but a vaccine against the arrogance of believing that we will not make the same mistakes.

The most emblematic—and perhaps the most chilling—case is that of the THERAC-25 machine, a linear accelerator designed to administer radiation therapy to cancer patients. In the mid-1980s, this machine, manufactured by Atomic Energy of Canada Limited, incorporated software for the first time that fully controlled its operation, replacing the mechanical safety systems that had protected patients for decades. What no one could have imagined was that this software contained errors as subtle as they were lethal—errors that would cause patients to receive radiation doses up to a hundred times higher than prescribed, resulting in severe burns and, in at least three cases, death.

1.2.1 WHEN SOFTWARE LIES: THE THERAC-25 TRAGEDY

The most shocking aspect of the THERAC-25 case is not just that the software failed but also how it failed and how those responsible responded. The errors occurred under very specific circumstances: when the technician entered the treatment data in a certain way, the system entered an unexpected state and fired the electron beam without the device that was supposed to attenuate it. Worse still, the screen displayed an incomprehensible error message and, in some versions, indicated that no radiation had been administered when in fact it had. The operators, trusting what they saw on the screen, administered the treatment again, accumulating lethal doses on the same patient.(4)

Behind this tragedy lie lessons that remain relevant decades later. The first is that software does not tolerate half-measures: either it works exactly as it should, or it can fail in ways in which no engineer imagined during its design. The second is that blind trust in technology is a luxury medicine that cannot afford. The technicians operating the THERAC-25 trusted what the screen showed them and that trust prevented them from suspecting that the system might be lying. The third lesson, perhaps the most uncomfortable, is that when a tragedy of this magnitude occurs, the natural tendency of organizations is to look for individual culprits rather than examine the systemic failures that made it possible.

Subsequent investigations revealed that software errors were not new to the THERAC family of machines. Earlier versions, THERAC-6 and THERAC-20, experienced similar failures, but those models had mechanical safety systems that prevented the administration of incorrect doses. When the THERAC-25 was designed, the software was deemed reliable enough to eliminate these backup mechanisms—a decision that saved costs but removed critical layers of protection. This phenomenon, known as the substitution of safety with software, remains a common pitfall

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

in the design of contemporary medical systems, where overreliance on the ability of the code to manage risks can lead to the removal of physical or human safeguards that previously provided redundancy.

The manufacturer's initial response to the first incident was characteristic of a defensive organizational culture. When hospitals reported that patients had suffered burns during treatment, the company attributed the problems to operational errors or specific patient conditions, systematically denying that the software could be the cause. Even after engineers identified the code error related to the condition check, the company underestimated the severity of the problem and delayed the implementation of fixes. This dynamic of denial and downplaying is not unique to the medical industry, but in this context, it had fatal consequences that could have been avoided with a more agile and transparent response.

The THERAC-25 case also illustrates the critical importance of the user interface in high-security systems. The error messages displayed by the machine were cryptic and technical and were designed from the perspective of the programmer who understood the internal code, not from that of the technician who needed to make urgent clinical decisions. When the system detected an anomaly, it displayed codes such as "MALFUNCTION 54" or simply "TREATMENT PAUSE," without clearly indicating that a potentially lethal overdose had occurred. This lack of effective communication between the system and the human operator transformed recoverable technical errors into irreversible tragedies—a lesson that remains relevant in the design of contemporary medical interfaces.

Ultimately, the THERAC-25 tragedy highlights the need for a safety culture that transcends organizational silos. The software engineers who wrote the code did not interact with physicists who understood radiation or with physicians who knew the clinical consequences of an overdose. Each group operated on assumptions about the work of the others that turned out to be incorrect or incomplete. This fragmentation of knowledge, combined with the pressure to bring the product to market, created a breeding ground where errors could propagate undetected until it was too late. The lesson that safety in medical software requires genuine integration across multiple disciplines remains fundamental decades later.

1.2.3 HIDDEN BIASES: WHEN ALGORITHMS PERPETUATE INEQUALITY

In this century, errors have evolved alongside technology, but their potential to cause harm has not diminished. In 2019, an algorithm widely used in the United States to identify high-risk patients and allocate healthcare resources was found to systematically discriminate against Black patients. The software, developed by Optum (the business services arm of the UnitedHealth Group), uses previous healthcare costs as a proxy to determine patients' health needs. For years, thousands of Black patients were denied care management programs and additional resources because the algorithm deemed their risk to be low, when in fact it was not. The algorithm had no malicious intent, but its consequences were real and preventable. (5)

This incident represents a new generation of medical software failures: those that arise not from explicit programming errors but from the characteristics of the data used to train artificial intelligence systems. Unlike traditional bugs that produce incorrect behavior in specific circumstances, biases in training data generate systematic discrimination that affects entire population groups on an ongoing basis. Optum's algorithm functioned technically correctly: it made accurate statistical predictions on the basis of available historical data. The problem was th that historical data reflected decades of disparities in access to healthcare, where black patients, despite having the same health needs as white patients did, had historically spent less on healthcare because of systemic barriers related to access, transportation, knowledge of healthcare systems, and socioeconomic inequalities. The algorithm learned that lower spending equaled lower needs, creating a cycle of invisible exclusion automated by technology.

The gravity of this type of error lies in its apparent objectivity. While human doctors may be aware of their own biases and consciously compensate for them, an algorithm presents its conclusions with the authority of numbers, without indicating the limitations of its database or the underrepresented populations in its training. Black patients affected by the Optum system never knew that they had been misclassified; they simply did not receive the preventive interventions and care management programs they would have received had they been white patients with the same actual risk profile. This opacity in decision-making, combined with the perceived authority of computational systems, makes algorithmic biases particularly difficult to detect and challenging.

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

The case takes on additional dimensions when we consider its scale and duration. Unlike a traditional software error that affects specific users at specific times, a population health algorithm can influence decisions about millions of people for years before anyone identifies the underlying bias. In the case of Optum, the algorithm had been operating for years before academic researchers at the University of California, Berkeley, discovered the racial disparity. During that period, harm accumulated silently in the form of unmanaged chronic diseases, higher healthcare costs due to avoidable hospitalizations, and, presumably, premature deaths that could have been prevented with early intervention. This study revealed that this racial bias reduced the number of black patients identified for additional care by more than 50%.

The institutional response to this discovery illustrates both the progress and the limitations of our algorithmic governance mechanisms. Once the problem was identified, the researchers worked with Optum to reformulate the algorithm, adjusting it so that it no longer used costs as a proxy for health needs. The reformulation eliminated 84% of the racial bias in the results. However, the correction could not undo years of accumulated harm or identify the specific patients who had been harmed. Furthermore, the review focused on the identified racial bias, without systematically examining other potential biases related to gender, socioeconomic class, or geographic location that might have been operating simultaneously within the same system.

This incident forces us to reflect on an uncomfortable truth: software is not neutral, even though many programmers believe it is. Every line of code incorporates the assumptions, biases, and limitations of those who write it and the data used to train it. An algorithm that determines which patients deserve priority care, which treatments are most effective, or which diagnoses are most likely, is not an objective tool but a mirror that reflects our own limitations. Recognizing this is the first step toward trying to correct them, or at least toward not unconsciously perpetuating injustices that already exist in the healthcare system.

1.2.4 DIGITAL WARFARE: WHEN HOSPITALS BECOME TARGETS

Alongside these well-intentioned failures, a threat of a very different nature has emerged: cyberattacks targeting healthcare infrastructure. In 2017, the WannaCry ransomware paralyzed dozens of hospitals in the United Kingdom, forcing the cancellation of thousands of appointments and surgeries and putting patients who depended on networked equipment at risk.(6) The attackers were not seeking to harm patients directly but rather to make money; however, the collateral effect was the same: systems that stopped working, doctors without access to medical records, and operating rooms that had to close. The incident demonstrated something that until then had been merely theoretical: cybersecurity is not a matter of data theft but of patient safety.

Hospitals have become prime targets for cybercriminals for a very simple reason: they cannot afford to be offline. A store can close for a day because of a cyberattack; an emergency department cannot. This asymmetry makes healthcare institutions easy targets, and the response must be not only technical but also organizational and cultural. Doctors and nurses must be aware that a phishing email is not just a nuisance but also a potential gateway to a clinical catastrophe. In addition, hospital administrators must understand that investing in cybersecurity is not an administrative expense but a clinical safety measure as important as having fire extinguishers or evacuation plans.

The spread of WannaCry in the British National Health Service revealed systemic vulnerabilities that went beyond individual technology. The malware exploited a known vulnerability in Windows systems that Microsoft had patched months earlier, but many hospitals had not applied because of the complexity of updating legacy systems interconnected with specialized medical equipment. This tension between the need to patch vulnerabilities and the risk that updates might interfere with the critical functionality of certified medical devices represents an ongoing dilemma in healthcare infrastructure management. Hospitals are caught between exposure to known threats and the possibility that a software update might disable a patient monitor or a medication dispensing system.

The clinical impact of WannaCry was immediate and tangible. In several hospitals, radiological equipment is affected, preventing urgent diagnoses. Bed management systems stopped working, hindering the coordination of admissions and discharges. Electronic health records became inaccessible, forcing professionals to work with pen and paper, without access to complete histories or recent test results. In some cases, ambulances had to be rerouted to unaffected hospitals, increasing response times for critically ill patients. Although no deaths directly attributable to the attack were documented, the deterioration in the quality of care during

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

the incident exposed the fragility of modern healthcare systems in the face of digital disruptions.

The response to the attack also highlighted deficiencies in cyber incident preparedness and coordination within the healthcare sector. Many hospitals lacked specific contingency plans for ransomware attacks, and communication between affected facilities and authorities was initially chaotic. Recovery took days or weeks in some cases, during which healthcare services operated in degraded mode. This episode catalyzed an awareness of cybersecurity as an essential dimension of patient and security, leading to significant policy and budget reviews in healthcare systems worldwide.

However, five years after WannaCry, the situation remains concerning. Attacks on hospitals have intensified and become more sophisticated, with criminal groups developing ransomware specifically designed for healthcare environments and aware of their vulnerability and willingness to pay. The COVID-19 pandemic exacerbated these risks by forcing accelerated digitization of services and an expansion of remote connectivity that was not always accompanied by commensurate security measures. Hospitals' attack surface has grown exponentially with the proliferation of connected medical devices, many of which were designed without robust security considerations and are difficult to update or replace.

There is one constant in all these incidents that is worth highlighting: None was caused by malicious intent. The engineers who designed the THERAC-25 did not want to kill patients. The programmers of the biased algorithm did not intend to discriminate against anyone. The developers of the system that deleted medical records believed that they were making technical improvements. The problem is not malice but ignorance, a lack of perspective, and the inability to imagine how a technical system will interact with a complex human environment. Moreover, ignorance can be combated only through education, humility, and the conviction that in medicine—unlike other fields—there is no room for trial and error.

That is why this book does not begin with abstract theory but with these stories. Before learning to program for medicine, before understanding technical standards or validation methodologies, one must develop a special sensitivity to potential harm. We must understand that every design decision, technical choice, and line of code can have consequences that we will never know but must anticipate. That sensitivity, that awareness of risk, is what separates a good software developer from a good medical software developer.

History also teaches us that mistakes tend to be repeated when they are not properly documented or when the lessons learned are not passed on to new generations. Every serious incident involving medical software has generated reports, investigations, and new regulations, but this knowledge often remains confined to specialist circles. Doctors who use these systems rarely know about past failures, and young programmers are often unaware that there was a time when a linear accelerator could kill patients because of software error. Breaking down this information silo is one of the motivations behind this book.

When we look back and see this succession of errors—from the most basic programming flaws to the most sophisticated algorithmic biases, including deliberate cyberattacks—an inevitable question arises: are we better prepared today than we were thirty years ago? The answer is mixed. Technically, yes, we have better tools, better methodologies, and a better understanding of the risks. However, we also have systems that are far more complex, interconnected, and invisible, where a failure in a small component can propagate in ways that we are only beginning to understand. In addition, we still have the same human frailty as always: the tendency to place too much trust in what we build and to forget that, behind every line of code, there are people whose health depends on our having done our job well.

1.3 MAPPING THE LANDSCAPE: SAMD, TELEMEDICINE, IOMT, AND BIG DATA

Let's imagine for a moment that we travel back in time to the late 20th century and explain to a doctor of that era what his practice will look like twenty years later. We'll tell him about apps that remotely monitor his patients' hearts, algorithms that analyze X-rays with greater precision than the human eye, and devices that collect millions of data points from each person and process them to predict diseases before the first symptoms appear. They would likely look at us with skepticism, perhaps even disbelief. However, the everyday landscape of contemporary digital health is not science fiction. A vast, sometimes chaotic, yet fascinating territory that we should understand in some detail before venturing to build upon it.

The first concept we must place on this map is that of software as a medical device, the acronym SaMD that we introduced in the first section. It is now worth delving deeper into its practical

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

implications because not all healthcare software deserves the same regulatory consideration. The classification proposed by the International Forum of Medical Device Regulators ((1)) distinguishes four categories based on two variables: the severity of the disease it addresses and the importance of the information it provides for clinical decision-making. Software that helps in the diagnosis of mild illness faces fewer requirements than software that guides intensive treatment does, and an algorithm that simply informs the physician will require fewer controls than one that makes automatic decisions without human supervision does.

This gradation is not an unnecessary bureaucratic complication but a rational attempt to align the level of scrutiny with the level of risk. Consider, for example, an app that helps patients with seasonal allergies track their symptoms and identify seasonal patterns. If it fails, the worst that can happen is when the patient experiences minor discomfort or confusion. However, now consider software that controls insulin infusion in an intensive care unit. A failure in that system can cause severe hypoglycemia with irreversible neurological consequences within minutes. It makes sense, therefore, that the second is subject to much stricter controls than the first, even though both are technically health software.

1.3.1 SAMD: WHEN CODE BECOMES PRESCRIBER

The classification of software as a medical device represents one of the most significant regulatory advances of recent decades, although its practical implementation continues to generate intense debate. What distinguishes SaMD from other software is its ability to directly influence medical decisions without being part of specific hardware. This seemingly clear definition becomes murky when we examine borderline cases that abound in today's market. An app that uses a phone's camera to analyze the color of a skin lesion and suggests whether it might be malignant operates in a gray area where information is confused with recommendations and is recommended for diagnosis.

The International Forum of Health Regulators' classification matrix establishes four risk levels ranging from basic information software to critical treatment systems. At the lowest level are applications that manage clinical information without offering interpretation, such as medical calendars or symptom-tracking systems without analysis. At the highest level, we find software that controls intensive care therapies, such as smart infusion pumps or cancer treatment planning systems. Among these extremes lies a continuum of functionalities where determining the appropriate regulatory class requires expert judgment and, frequently, negotiation with health authorities.

The importance of this classification goes beyond mere bureaucracy because it determines the product's entire lifecycle. Software classified as Class I can reach the market relatively quickly, with documentation and testing requirements that are manageable for a startup with limited resources. Class III or IV software, however, will face years of development, clinical trials, quality audits, and regulatory reviews that demand millions of investment and experienced multidisciplinary teams. This disparity in entry costs has shaped the current digital health ecosystem, where wellness and management apps abound, but solutions for serious diseases requiring strict controls are scarce.

A particularly relevant phenomenon in the field of SaMD is the trend toward the progressive medicalization of applications that were initially marketed as consumer products. Many technology companies have discovered that the health and wellness market is enormously lucrative but also that the boundary between general wellness and medical health is permeable. A meditation app that helps reduce stress can evolve into a treatment program for anxiety disorders. A fitness tracker can incorporate algorithms for detecting atrial fibrillation. Each of these shifts transform the product, change its regulatory classification, and necessitate a complete reevaluation of its development, validation, and marketing.

The regulatory response to this dynamic has been the creation of specific pathways for software that allow for some flexibility without compromising safety. In the United States, the FDA has implemented precertification programs for companies that, by demonstrating a strong quality culture, can update their algorithms with greater agility. In Europe, the Medical Device Regulation introduces specific considerations for software that recognize its evolving nature. (3) However, these regulatory adaptations remain insufficient to keep pace with the speed of technological innovation, creating a constant tension between the need to protect patients and the risk of delaying potentially beneficial technologies.

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

1.3.2 TELEMEDICINE: SOFTWARE AS A CLINICAL SPACE

Parallel to the development of SaMD, we have witnessed an explosion of telemedicine over the past decade, a term that encompasses very diverse realities under a single semantic umbrella. In its most basic sense, telemedicine simply involves performing medical procedures remotely using communication technologies. However, this broad definition masks an enormous diversity of situations, from a video call consultation with a primary care physician to robotic surgery performed by a specialist on another continent to the remote monitoring of chronic patients or the remote interpretation of diagnostic tests. Each of these modalities has its own technical requirements, its own limitations, and its own risks.

What makes telemedicine particularly relevant in our context is that it places software at the center of the clinical relationship. When a doctor and a patient share the same room, technology is an auxiliary tool that supports human interaction. However, when they are separated by hundreds of kilometers, software becomes the sole channel of communication, and its quality directly determines the quality of the medical procedure. A-based video call that cuts out, an image that becomes pixelated at a critical moment, or data transmitted with a delay are not mere technical annoyances but barriers that can compromise a diagnosis or treatment. This is why the reliability and usability of telemedicine are far superior to those of other consumer applications.

The COVID-19 pandemic accelerated the adoption of telemedicine in a way that would have been inconceivable just months earlier, revealing both its potential and its vulnerabilities. In a matter of weeks, health systems around the world had to implement remote consultation solutions that would have previously required years of planning. This forced transformation demonstrated that many medical consultations can be effectively conducted remotely, reducing geographical and time-related barriers to accessing care. However, it also revealed digital divides that exclude populations without access to adequate technology, diagnostic limitations when a direct physical examination is not possible, and the risks of depersonalizing the doctor–patient relationship.

Telemedicine modalities span a spectrum of increasing complexity that determines their technical and regulatory requirements. At the simplest end, we find asynchronous teleconsultation, where the patient sends information and the physician responds at a later time, as is the case with secure messaging systems for prescription renewals or the follow-up of stable chronic conditions. This model is technically accessible but poses challenges regarding continuity of care and clinical responsibility when unforeseen complications arise. Synchronous teleconsultation by videoconferencing adds the dimension of real time, allowing for richer interaction but requiring robust network infrastructure and strict privacy protocols.

Tele-triage and telemonitoring represent a higher level of complexity, where software not only facilitates communication but also continuously collects and analyzes clinical data. Weight, blood pressure, and oxygen saturation of a patient with heart failure can be monitored daily from home, with algorithms that alert the healthcare team to significant deviations. This model transforms care from episodic to continuous and from reactive to predictive, but it also creates new forms of technological dependence and risks of information overload for healthcare professionals. A physician receiving dozens of alerts daily, many of them false positives, may develop alert fatigue that causes them to lose sensitivity to truly important signals.

At the most sophisticated end of the spectrum, we find telesurgery and other remote interventional procedures, where software directly controls physical devices acting on the patient. Although still not widely adopted because of technical and regulatory limitations, this modality illustrates the highest degree of trust that can be placed in a technological system. The surgeon operating remotely is absolutely dependent on zero-latency connectivity, the millimeterprecision of the robots, and the total reliability of the control software. A failure in any of these elements is not a technical glitch but a potentially lethal adverse event that demands levels of redundancy and safety comparable to those in aviation or nuclear power.

1.3.3 IOMT: PATIENT'S DIGITAL ANATOMY

The third major vertex of this technological triangle is the internet of Medical Things, known by its acronym IoMT. Behind this somewhat cryptic term lies an everyday reality: connected devices that monitor vital signs, sensors that track physical activity, smart patches that measure glucose, and pacemakers that send automatic reports to the cardiologist. It is estimated that tens of millions of these devices are already in operation worldwide, generating a continuous stream

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

of data that, if properly managed, can transform preventive medicine and the management of chronic diseases.

The promise of IoMT is enticing: imagine a patient with heart failure whose signs of decompensation are detected by sensors days before symptoms appear, allowing for early therapeutic intervention that prevents hospitalization. Alternatively, an elderly person living alone whose movements are discreetly monitored to detect falls or changes in activity patterns that may indicate an emerging health problem. Or an athlete whose training and recovery data are integrated with their physiological parameters to optimize performance without jeopardizing their health. All these applications are technically feasible today, and many are already being implemented in advanced healthcare systems.

However, alongside the promises, the IoMT also presents formidable challenges. The first is managing data volumes that would have been unimaginable a decade ago. A single patient with a continuous glucose monitor generates hundreds of readings daily. Multiply that by thousands of patients, and you have a torrent of information that threatens to overwhelm any system designed to be managed by humans. The second challenge is integration: devices from different manufacturers use proprietary protocols, incompatible formats, and standards that do not communicate with one another, turning the task of gathering all that information into a major technical puzzle.

The IoMT architecture consists of three interconnected layers that must work in sync for the system to be useful. The perception layer includes all physical devices that capture data, from consumer wearables to highly complex medical implants. These devices vary greatly in reliability, accuracy, and security, ranging from equipment certified as medical devices to wellness gadgets without specific health regulations. The network layer transports these data from the patient to processing systems, using infrastructures ranging from short-range Bluetooth to wide-area cellular networks, each with its own limitations regarding bandwidth, latency, and security.

The application layer is where data are transformed into useful clinical knowledge and where the greatest potential—but also the greatest risks—of IoMT lie. Analytical algorithms must be able to distinguish between true signals and artifacts, between significant physiological changes and normal variations, and between emergencies requiring immediate intervention and situations that can wait until the next scheduled appointment. This capacity for contextual interpretation is what distinguishes an intelligent medical system from a mere data collector, but it is also where errors most frequently occur—errors that can have serious clinical consequences.

IoMT security is a particular concern given the vulnerable nature of many of these devices. Unlike traditional computers, which receive regular security updates, many connected medical devices are designed without remote patching mechanisms and have lifespans spanning decades. A pacemaker implanted in a patient cannot be easily removed to update its software, nor can it rely on network connections that could compromise its basic operation. This tension between connectivity and security, between updating and stability, defines many of the current dilemmas in the design of smart medical devices.

1.3.4 BIG DATA: THE OCEAN OF INFORMATION AND ITS HIDDEN CURRENTS

This is where the fourth element of our map comes into play: Big Data applied to healthcare. Because there is no point in generating vast amounts of data if we are unable to process it, analyze it, and extract useful knowledge for clinical practice. Healthcare Big Data are not just a matter of volume but also of variety, velocity, and veracity. Clinical data come from a wide variety of sources: electronic health records, monitoring devices, diagnostic images, genetic analyses, pharmacy records, and patient surveys. Integrating this entire mosaic into a coherent picture of a person's health is among the most exciting technical and conceptual challenges of our time.

The applications of Big Data in medicine are already numerous and promise to increase in the coming years. There are predictive models that identify patients at high risk of readmission to the hospital, enabling personalized preventive interventions. There are diagnostic support systems that compare a patient's case with millions of similar records to suggest rare diseases that the physician might not have considered. There are population health tools that identify epidemiological patterns in real time, as some systems did during the COVID-19 pandemic by detecting outbreaks even before health authorities were aware of them.(7)There are, of course, research algorithms that uncover new associations between genetic and environmental factors and diseases, accelerating the pace of medical knowledge.

However, we should not be blinded by these promises without also examining the problems.

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

Healthcare big data raises uncomfortable questions about privacy and autonomy that society is only just beginning to ask. Who owns the data a patient generates? Can an insurer access it to adjust premiums? Should researchers be able to use anonymized clinical records without explicit consent? To what extent is it truly possible to anonymize data when the combination of just a few variables can uniquely identify a person? These questions have no simple answers, and the answers we provide as a society will largely determine the future of data-driven medicine.

An additional problem, less debated but equally relevant, is the quality of the data feeding these systems. Electronic health records were originally designed for billing and administrative management, not for research or prediction. They contain errors, omissions, inconsistencies, and biases that are silently passed on to the algorithms that use them. A famous study showed that sepsis prediction systems in U.S. hospitals performed worse for patients from ethnic minorities simply because their clinical records were less complete.(8) Data do not lie, but it does not tell the whole truth either, and blindly trusting it can perpetuate inequalities that the healthcare system should be combating.

The infrastructure needed to manage healthcare big data exceeds the capacity of any single institution. Population health data repositories, shared medical image banks, and genomic research platforms require massive investments in storage, processing, and analysis that can be undertaken only by large-scale public or private consortia. This concentration of data in the hands of a few creates risks of monopoly and technological dependence, where decisions about what research is possible and what treatments are developed rest with entities not subject to direct democratic control.

When we look at these four elements—SaMD, telemedicine, IoMT, and Big Data—together, a fascinating yet unsettling picture emerges. Fascinating because of the opportunities it has opened up for more precise, more preventive, more personalized, and more accessible medicine. Unsettling because of the complexity it introduces into a realm—that of the clinical relationship—which for centuries has been based on interpersonal trust and human judgment. Software is not going to replace the doctor, at least not in the foreseeable future, but it is profoundly transforming their work, their tools, and their relationships with patients. Understanding this new territory is the first step toward navigating it consciously and ethically.

The players in this landscape are numerous and diverse. There are public hospitals and healthcare systems that must decide which technologies to adopt and how to integrate them into their workflows. There are technology companies, ranging from small, innovative startups to the giants that already dominate other sectors of the digital economy. There are regulators who are trying to put up fences while technology gallops are unchecked. There are healthcare professionals who receive these tools with a mix of hope and skepticism. There are, above all, patients who are both potential beneficiaries and subjects of involuntary experimentation in this vast global laboratory.

To navigate this complex territory without being lost, it is wise to always keep an ethical compass in mind to guide us. Not every technological innovation is desirable simply because it is possible. Not all the data that can be collected should be collected. Not all automation that saves the doctor time necessarily improves patient care. Technology in medicine must serve the profession's fundamental values: beneficence, nonmaleficence, patient autonomy, and distributive justice. If a technological innovation contradicts any of these principles, no matter how brilliant it may be from a technical standpoint, it is likely not worth it.

The map we have drawn in this section is necessarily incomplete and provisional. Digital health is evolving at a dizzying pace, and what is cuttingedge today will be obsolete tomorrow. However, one thing remains: the need to understand this field from an integrative perspective, one that does not separate technology from ethics, innovation from safety, or code from people. Those who enter this field with this broad perspective will have a much better chance of building truly useful tools and, above all, of not causing harm in the process.

1.4 KEY PLAYERS: HOSPITALS, REGULATORS, STARTUPS, PATIENTS

There is a recurring image in science fiction films showing a lone genius locked away in his garage, creating revolutionary technology that will change the world. It is a powerful narrative that fuels the collective imagination about innovation and contains a grain of truth: many great ideas have been born in humble settings and owing to brilliant minds working against all odds. However, when we discuss innovation in digital health, that image is not only incomplete but dangerously misleading. Unlike others, in this field, ideas do not travel alone: they must make their

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

way through a complex ecosystem of stakeholders with very different interests, languages, and priorities. Understanding these stakeholders, understanding their motivations, and respecting their legitimate concerns is just as important as the idea itself that one wishes to develop.

Let us start with the most visible and, paradoxically, most heterogeneous actor of all: hospitals and healthcare systems. When we speak of hospitals, we are not referring to a monolithic entity but rather to an organizational microcosm where professionals with diverse backgrounds, cultures, and objectives coexist. Administrators focus on financial statements and operational efficiency. There are clinicians—doctors and nurses—whose priority is patient care and who often view technology as an additional burden rather than a tool. There are the IT managers who grapple daily with the nightmare of integrating technologies from dozens of different manufacturers. There are patients, of course, although within the hospital structure, their voice is often the weakest. A developer who does not understand this internal complexity is doomed to fail, no matter how brilliant their technical solution may be.

Hospitals also have a unique characteristic that sets them apart from other environments where software has been successfully implemented: they are organizations where errors have immediate and serious consequences. A failure in a factory's inventory management system may delay production; a failure in an electronic prescribing system can kill a patient. This asymmetry creates a risk aversion that is often mistaken for resistance to change but has deep and legitimate roots. Healthcare professionals have seen too many unfulfilled technological promises and too many tools that promised to save time but ended up multiplying administrative work and too many solutions that failed to understand the reality of clinical practice. They are rightly suspicious and that suspicion is not overcome with technical arguments but with facts, patience, and a deep respect for their expertise.

1.4.1 HOSPITALS: A WORLD OF WORLDS

Hospitals are like complex living organisms where technology must be integrated into workflows that have evolved over decades, often organically and not always rationally. An emergency room doctor does not need a faster app; they need one that works with wet gloves and is under stress. A ward nurse isn't looking for a sleek interface; they're looking for one that doesn't force them to make unnecessary clicks when they have ten patients waiting. A hospital administrator does not value innovation for its own sake; they value cost reduction and improvements in quality metrics that they can defend before their board of directors. Each of these potential users speaks a different language, has different priorities, and experiences technological frustration in unique ways that the external developer must learn to map out.

The organizational structure of a modern hospital reflects the historical tension between medicine as a clinical art and healthcare management as an administrative discipline. Clinical departments operate with significant autonomy, each with its own traditions, hierarchies, and ways of working that resist standardization. Admissions, pharmacies, laboratories, and radiology departments function as silos that exchange information in a fragmented manner. Procurement committees and information technology units impose vendor selection criteria that often prioritize stability over innovation. Navigating this labyrinth requires patience, diplomacy, and the ability to identify allies at every level who can facilitate the introduction of new tools without disrupting the institutional balance.

Technology adoption in hospitals follows learning curves that developers frequently underestimate. The pilot phase, where a product is tested in a controlled environment, is only the beginning of a process that can take years to achieve widespread implementation. During this period, the software must demonstrate not only that it works technically but also that it fits into existing workflows, does not increase the cognitive load on healthcare professionals, generates useful data for management, and withstands the variations in use that occur in real-world practice. A product that works perfectly in the lab can fail spectacularly in the hospital because it did not anticipate that doctors would use it in unforeseen ways or because it did not integrate with the existing electronic health records system.

The relationship between hospitals and technology providers is also shaped by asymmetrical power dynamics that hinder genuine innovation. Major medical equipment manufacturers establish long-term relationships with hospitals, creating barriers to entry for new competitors. Maintenance and upgrade contracts create technological dependencies that resist disruption. Public bidding processes, designed to ensure transparency and fairness, often favor established solutions over disruptive innovations that do not fit within predefined specifications.

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

Understanding these economic and political realities is essential for any startup aspiring to scale beyond isolated pilot projects.

1.4.2 STARTUPS: SPEED AND VULNERABILITY

At the opposite end of the spectrum are digital health startups—those small teams that typically combine a passion for technology with a desire to improve people's lives. The health tech entrepreneurial ecosystem has grown exponentially over the past decade, driven by the availability of capital, the reduction in technical barriers, and the growing awareness that technology can solve real healthcare problems. Many of these startups are founded by healthcare professionals who have experienced the system's limitations firsthand and want to change it from the outside. Others emerge from business schools or engineering departments, with teams that have barely set foot in a hospital but trust in their technical ability to transform medicine.

In addition to their small size and agility, startups are characterized by their risk tolerance and focus on rapid growth. While a hospital thinks in terms of years or decades, a startup thinks in terms of quarters and funding rounds. While a hospital seeks stability and predictability, a startup seeks innovation and disruption. These differences in culture and time horizons inevitably create tensions when the two countries try to collaborate. The hospital accuses the startup of failing to understand clinical complexity and of promising magic solutions that do not hold up in the real world. The startup accuses the hospital of being slow, bureaucratic, and resistant to innovation. Both are partly right, and the art of collaboration lies in building bridges that respect the legitimate needs of each party.

The life cycle of a digital health startup goes through critical phases where most projects fail before they reach the market. In the ideation phase, excitement over the technical solution often clouds understanding of the actual problem. In the validation phase, it is discovered that what doctors say they need does not match what they actually use. In the regulatory phase, the timelines and costs of obtaining health authorizations far exceed the initial estimates. In the commercialization phase, where the harsh reality is faced such that hospitals do not buy innovation, they buy proven solutions with solid track records. Each of these transitions eliminates a significant proportion of competitors, leaving only those that combine technical excellence with an understanding of the ecosystem.

The financing of healthcare startups introduces unique dynamics that shape their development. Venture capitalists seek exponential returns that are possible only with scalable business models, pushing startups toward solutions that can be replicated without extensive customization. However, the reality of healthcare is that every system, hospital, and specialty has specific needs that resist standardization. This tension between the logic of investment and the logic of clinical practice creates pressures that sometimes lead to insufficiently tested products or unfulfilled promises of functionality. The entrepreneurs who successfully navigate this tension are those who manage to find niches where standardization is viable without sacrificing clinical safety.

1.4.3 REGULATORS: GUARDIANS OF THE FRONTIER

Between the two, playing a role that is often uncomfortable but absolutely necessary, stands the regulators. Agencies such as the FDA in the United States, the European Medicines Agency, or national agencies in Latin America and Spain are responsible for ensuring that medical devices reaching the market are safe and effective. Their work is inherently conservative because their role is not to accelerate innovation but to protect patients from the harm that hasty innovation could cause. This focus on safety clashes head-on with startup culture, which is accustomed to "moving fast and breaking things," but it is indispensable in a field where the things that break can be human lives.

Regulators also face an unprecedented challenge: the speed of technological change far outpaces their ability to respond. By the time a regulatory agency takes years to develop guidance for a new product category, that category has already evolved or been replaced by another. This mismatch creates uncertainty and frustration among developers, who often do not know exactly which regulatory requirements their product will have to meet. To address this uncertainty, understanding the philosophy underlying regulation, beyond the specific details that are constantly changing, is essential. Regulation is not an arbitrary obstacle, but a set of

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

questions that any medical device must be able to answer: Is it safe? Does it work? Do its benefits outweigh its risks? Is it well documented so that its performance can be audited?

Recent regulatory developments reflect attempts to adapt frameworks designed for physical devices to the reality of software. The FDA has introduced company precertification programs that allow for some flexibility in algorithm updates. Europe has developed specific regulations for medical software within the general framework for medical devices. However, these adaptations remain insufficient to keep pace with the speed of innovation, and regulatory uncertainty remains one of the main barriers to entry for new competitors. Compared with startups that must learn on the fly, established companies with experienced regulatory departments have significant advantages in terms of the complex requirements for documentation, clinical trials, and risk management.

The relationship between regulators and the regulated does not have to be adversarial, although it is often perceived that way. Health agencies offer channels for dialog, presubmission consultations, and guidance programs that developers frequently underutilize because of a lack of awareness or fear of revealing strategic information. Understanding that the regulator is a partner seeking to fulfill its mandate of public protection—not an obstacle to being circumvented—fundamentally changes the development strategy. Companies that establish transparent and collaborative relationships with agencies tend to navigate the regulatory process more efficiently than those that approach it as a legal battle do.

1.4.4 PATIENTS: THE INVISIBLE CENTER

At the center of all these actors, as the raison d'être of the entire system, are the patients. It may seem obvious to point this out, but in the day-to-day practice of healthcare software development, it is surprisingly easy to lose sight of them. Patients appear in conversations as "users," as "data," and as "clinical cases," but rarely as people with fears, hopes, limitations, and preferences. A patient is not a set of variables feeding an algorithm, nor a consumer choosing between different apps, nor a body to be managed efficiently. They are people living through an illness who trust their doctors, who have the right to understand what is done with their information, and who deserve to be treated with dignity in every interaction with the healthcare system.

Incorporating the patient's perspective into the development of medical software is not a paternalistic gesture but a requirement for quality. A glucose monitoring app that does not account for the fact that many diabetic patients have vision impairments or limited manual dexterity will be useless, no matter how technically precise it may be. An appointment scheduling system that does not consider that some patients lack internet access or are unable to use mobile apps will create inequalities rather than resolve them. An emergency room triage algorithm that does not incorporate subjective variables such as pain or discomfort will offer a distorted picture of clinical reality. Patients are not merely the end users of technology but also irreplaceable sources of knowledge about what actually works in practice.

There is also a category of patients who deserve special attention: those who actively participate in the development of health technologies, whether as advisors, evaluators, or research subjects. Their contribution is invaluable, but it also raises ethical dilemmas that are not always properly managed. How can we ensure that a patient participating in the design of an app does not feel pressured to consent to use their data that they do not fully understand? How can we prevent the excitement of contributing to a technological advancement from clouding their ability to critically assess the risks? Who safeguards their interests when the commercial interests of developers point in the opposite direction? These questions have no easy answers, but ignoring them does not make them go away.

Patient participation in digital health governance is evolving from symbolic representation to more substantive models. Some healthcare systems are experimenting with patient councils that review technological proposals before implementation. People who are associated with chronic conditions are developing the technical capacity to critically evaluate apps targeting their communities. Open data and personal data ownership movements are empowering citizens to better control who accesses their health data and for what purposes. These trends, although nascent, are gradually shifting the balance of power within the ecosystem, challenging the traditional asymmetry between those who design technology and those who use it.

Rounding out the ecosystem, although often in the background, are other actors whose influence is decisive. Insurers and public payers determine which technologies are funded

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

and, therefore, which technologies actually reach patients. Scientific societies and professional associations issue guidelines and recommendations that shape clinical practice. Universities and research centers train future professionals and generate the knowledge that fuels innovation. Patient organizations, which are increasingly organized and influential, advocate for the interests of those living with specific diseases. Together, they form a dense web of relationships, interests, and interdependencies that any digital health project must navigate skillfully.

The complexity of this ecosystem explains why so many promising initiatives fail to reach the market. It is not enough to have a good idea and a strong technical team. One must understand how hospitals think, what clinicians need, what regulators require, what patients value, and what investors seek. You must be able to translate between languages that often do not align: the language of efficiency and return on investment, the language of evidence and clinical safety, and the language of human experience and suffering. This task of translation is exhausting but also fascinating, and those who master it build bridges that benefit society as a whole.

For healthcare professionals entering this world for the first time, it can be overwhelming to see the number of stakeholders and interests involved in something as seemingly simple as developing an app. However, it's important to understand it, because you'll be part of that ecosystem, and the better you understand it, the more effectively you can help ensure that technology serves the purposes that truly matter. For the developer coming from the tech world, the culture shock can be even greater. In environments where innovation is the supreme value and the primary metric is the speed, they report that in digital health, other values carry more weight: prudence, safety, and equity. Learning to respect them does not mean giving up on innovation but rather practicing it in a more mature and responsible way.

Ultimately, all these stakeholders share a common goal, even if it is sometimes difficult to see amid tensions and disagreements. Everyone wants to improve people's health. Hospitals want to better care for their patients. Regulators want to protect them from preventable harm. Startups want to create solutions that solve real problems. Patients want to live longer and better lives, with as little disease burden as possible. Keeping this shared goal at the center of the conversation is the best way to ensure that inevitable disagreements do not become insurmountable barriers. It is also the best guarantee that the technology we build truly deserves to be called healthcare.

• CROSS-LITERACY: WHAT DOCTORS NEED TO KNOW ABOUT SOFTWARE

It's a scene that plays out in hospitals around the world with minor variations, but with a recognizable pattern: a team of engineers has spent months developing an app they consider revolutionary—a tool that will aggregate data from different sources and present doctors with a dashboard containing all the relevant information about their patients. On the present day, the engineers proudly showcase their creation, and the attending doctors exchange glances that range from bewilderment to disappointment. Something isn't working; something doesn't fit. The app is technically flawless, but it does not answer the questions that doctors ask themselves; it does not integrate into their workflow, and it does not understand how they actually think when treating a patient. The problem is not technical; it is one of translation. As with any translation, there are two languages that must learn from each other.

Let us start with what a doctor needs to understand about software—not to become a programmer but to be a competent and demanding partner in projects requiring their participation. The first thing—and perhaps the hardest to accept for professionals accustomed to the precision of clinical diagnosis—is that software rarely works in a binary fashion. A drug is either effective or is not effective; it has side effects or does not. However, a computer program can function correctly ninety-nine percent of the time and fail catastrophically in that remaining one percent. It can do its job well for years and reveals serious errors when it is updated to a new version. It may be perfect for one type of patient and useless for another. This probabilistic nature of software is difficult for minds trained in classical clinical thinking to accept, but ignoring it leads to misplaced trust.

The second fundamental lesson for healthcare professionals is that software is not neutral. Behind every line of code are human decisions that reflect priorities, assumptions, and, at times, biases. When a programmer decides which variables to display on the main screen of an electronic health record, they are making an implicit statement about what is important in patient care. When a development team prioritizes one feature over another, it expresses its vision of what clinical work should look like. When an algorithm weights some factors more heavily than others do to make a prediction, it is incorporating a value judgment about what is worth considering.

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

Clinicians using these tools must be aware that they are not dealing with objective instruments but with artifacts laden with intentionality and must learn to critically question them.

1.5.1 ESSENTIAL VOCABULARY: CONCEPTS THAT PREVENT MISUNDERSTANDINGS

There is also a set of basic technical concepts that every physician involved in digitalization projects should be fluent in. You do not need to know how to program, but you do need to understand what a database is and how information is organized into tables and records. You must understand the difference between an integrated system and one that requires interfaces to communicate with other systems because that difference determines the smoothness of your daily work. It is in your best interest to know what it means for software to be opensource or proprietary because that affects the hospital's ability to modify or adapt it. In addition, you should have a basic understanding of client–server architecture, cloud storage, and the fundamentals of IT security. It is not about becoming an expert but about acquiring the minimum vocabulary needed to hold a meaningful conversation with the technicians.

Understanding databases is particularly important because it determines how the information a physician uses daily is structured. When a healthcare professional enters data into an electronic health record, they are not simply writing in a digital document; they are filling in specific fields of a predesigned structure that determines what can be searched, analyzed, and shared. Understanding that data are organized into relational tables, that there are primary keys identifying unique records and that SQL queries extract information on the basis of defined criteria allows the physician to ask more precise questions about the system's functionality. Knowing why certain information appears automatically on their screen while other information requires complex searches provides clues about the underlying architecture and potential improvements.

The distinction between open-source and proprietary software has practical implications that many physicians are unaware of. An open-source system allows the hospital to hire developers to modify it according to its specific needs, to fix bugs without relying on the original manufacturer, and to adapt workflows that the generic designer did not anticipate. A proprietary system generally offers greater stability and commercial support but at the cost of technological dependence and inflexibility with respect to specific needs. This choice is not merely technical but strategic and affects healthcare facilities' autonomy in the medium and long term. The informed physician can participate in these decisions with discernment, understanding that immediate convenience does not always offset future dependence.

The concept of client–server architecture explains why some applications continue to function even when the network fails, while others become unusable. In a traditional client–server architecture, processing occurs on central servers, and the physician's device displays results only, allowing access from any location but creating a dependency on connectivity. Modern hybrid applications store data locally and synchronize when possible, offering greater resilience against network issues but introducing complexity in version management. Understanding these architectures helps physicians assess whether a system is suitable for their particular environment, where connectivity may be intermittent or where the immediate availability of critical data is a priority.

Although it may seem the exclusive domain of specialists, cybersecurity requires a basic understanding from all users. Physicians must understand what data encryption in transit and at rest means, why strong passwords are mandatory and not optional, and what risks are involved in accessing clinical systems from public networks. They must understand the principle of least privilege, which limits access to information only to those who need it for their work, and why sharing credentials with colleagues or subordinates compromises traceability and security. This knowledge is not abstract: a physician who understands the risks is less likely to fall victim to social engineering, better able to identify anomalous behavior in the system, and more aware of their responsibility in the data protection chain.

1.5.2 BETWEEN THE USER AND THE PATIENT: DESIGN FOR REAL-WORLD PRACTICE

One of the most useful concepts a physician can incorporate into their mental repertoire is that of "user experience," known by its acronym UX. Behind this term lies a profound insight: software must not only do what it promises but also do so in a way that feels natural to the user. In the clinical setting, where time is scarce and interruptions are constant, a poorly designed interface

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

is not a minor annoyance but a source of potentially serious errors. A button that is confused with another, a screen that delays critical information, a workflow that forces unnecessary clicks: all of this affects the quality of care. A physician who can identify and communicate these issues will make a decisive contribution to patient safety.

The user experience in medicine has unique characteristics that distinguish it from other fields. In consumer applications, the user can take time to learn the interface, repeat actions if they make mistakes, or seek alternatives if the system does not respond. In the clinical setting, the physician operates under time pressure, is frequently interrupted, and often suffers from accumulated fatigue. An interface error cannot be undone with a simple "undo": it can mean prescribing the wrong medication, overlooking a critical allergy, or delaying an urgent diagnosis. Medical software design must incorporate these extreme constraints, anticipating that the user will not be in optimal attention conditions and that any ambiguity will result in error.

An informed physician can critically evaluate the usability of the tools they use, applying principles that do not require specialized technical training. They can ask whether the most frequently needed information is accessible with the fewest possible interactions. They can observe whether the system provides clear feedback on actions taken, confirming that an order has been recorded or alerting them if something has failed. They can verify whether the terminology used corresponds to standard clinical language or imposes unfamiliar jargon that causes confusion. These observations, which are systematically documented and communicated to development teams, are valuable contributions that improve the quality of the software for all users.

Another aspect that often leads to misunderstandings is the difference between technical validation and clinical validation. For an engineer, for software to work means that it does what it is supposed to do, that it meets specifications, that it does not crash, and that it responds within acceptable time frames. For a physician, software to work means that it improves patient care, that it helps make better decisions and that it does not introduce additional risks. Both perspectives are legitimate, but they do not always align. A program can be technically flawless yet clinically useless or even harmful. A program can have technical flaws yet be clinically valuable. Physicians involved in development projects must learn to both demand and understand that clinical validation is not an optional add-on but the very core of what justifies the existence of healthcare software.

Clinical validation requires specific methodologies that physicians are uniquely qualified to design and execute. It is not enough to prove that the software does not crash; one must demonstrate that it improves outcomes relevant to patients, that it does not introduce treatment biases and that it is acceptable to those who must use it. This involves controlled trials, usability studies in real-world settings, and analyses of the impact on healthcare quality indicators. Physicians who understand the difference between technical and clinical validation can demand that both be carried out rigorously, preventing products prematurely launched onto the market from compromising patient safety.

1.5.3 HOW SOFTWARE IS MADE: PROCESSES PHYSICIANS SHOULD BE AWARE OF

It is also helpful for healthcare professionals to understand software development cycles and the methodologies used by technical teams. For example, they should know that many teams use agile methodologies, which involve delivering partial versions of the product in short cycles and improving them iteratively. This is very different from the traditional development model, where teams wait until the product is completely finished before it is presented. If a physician understands this dynamic, they will know that they can provide continuous feedback, that they do not have to wait to see the final result to point out problems and that their early involvement can save months of misguided work. They will also understand why engineers sometimes ask them to prioritize certain features over others: they are managing limited resources and need to know what is truly essential.

The agile methodology, with its two- or three-week sprints, daily stand-up meetings, and visual task boards, may seem chaotic to those accustomed to rigorous clinical planning. However, this apparent improvisation follows a logic of continuous adaptation that offers advantages in uncertain environments. The clinician who participates in agile projects must learn to articulate their needs in concrete user stories, to prioritize features in collaborative backlogs and to accept that the product will evolve through controlled trial and error. This flexibility is not technical laxity but rather a recognition that requirements change when confronted with reality and that it is more efficient to adapt than to try to foresee everything from the start.

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

The concept of technical debt is particularly useful for understanding why some medical systems seem obsolete or slow despite ongoing investments. When development teams take shortcuts to meet deadlines, they accumulate imperfect code that hinders future modifications. Over time, this debt grows until any change, no matter how small, requires disproportionate effort. A clinician who understands this dynamic may have a greater understanding of the current system's limitations but also be more insistent on the need for periodic refactoring to maintain the code's long-term health. They can distinguish between problems requiring quick fixes and structural issues that demand substantial investment.

Version management and change control are processes that directly affect clinical practice but are rarely explained to medical users. When a system is updated, it does not happen magically: There are testing processes in parallel environments, scheduled maintenance windows, and recovery protocols in case something goes wrong. A clinician who understands these procedures can better plan their work, anticipating periods of downtime or changes to the interface. They can also participate in pilot tests of new versions, providing valuable feedback before general deployment. This active participation improves the quality of updates and reduces the frustration of encountering unexpected changes in tools they use daily.

A particularly sensitive area is that of data and its quality. Doctors often assume that if data are on the computer, it is correct. This lack of trust in the infallibility of computer systems is among the main sources of errors in contemporary clinical practice. The reality is that health data are riddled with impurities: misrecorded values, misinterpreted fields, duplicates, omissions, and updates that are not reflected correctly. A physician involved in designing a system must be relentless in asking where the data come from, how it is cleaned, who validates it, and how often it is updated. They must understand that an algorithm trained on dirty data will produce dirty predictions and that relying on them without questioning them can have serious consequences.

1.5.4 INTEROPERABILITY, ALERTS, AND THE ETHICS OF CRITICAL USE

Another concept that deserves attention is interoperability, a long word that describes simple reality: the ability of different systems to exchange information and use it in a coherent manner. In practice, this means that the electronic health records should be able to communicate with the laboratory, that the laboratory should be able to send results to the prescribing system, and that the prescribing system should alert the physician to drug interactions on the basis of the medications the patient is already taking. When this does not happen, the physician becomes the manual integrator of all that information, copying data from one screen to another, wasting time and increasing the risk of errors. An informed physician knows that interoperability is not a technological luxury but a basic prerequisite for digitization to truly improve care.

Interoperability in healthcare faces technical and organizational challenges that explain why it remains elusive despite decades of effort. Messaging standards, such as HL7 or FHIR, provide common formats for information exchange, but their implementation varies among vendors. Databases from different systems use medical terminologies that do not always correspond and require complex mappings. Economic incentives often work against interoperability: software vendors prefer that customers rely on their closed ecosystems. Physicians who understand these barriers can push for their removal, demanding compliance with open standards in tenders and participating in information architecture initiatives that promote integration.

It is worth pausing for a moment to consider the phenomenon of alerts and notifications, one of the main sources of friction between physicians and computer systems. Software designers, with the best of their intentions, fill out screens with alerts: potential drug interactions, out-of-range lab values, and reminders of pending tests. Physicians, overwhelmed by hundreds of daily alerts, develop what is known as "alert fatigue" and ultimately ignore them all, including those that might be truly important. Understanding this dynamic and understanding how to contribute to the design of systems that alert physicians only when truly necessary is a valuable contribution that physicians can make to development teams.

Managing clinical alerts requires balancing safety with usability so that neither is sacrificed. Alerts must be context specific: a known and manageable drug interaction does not require the same urgency as a potentially lethal combination does. They must be graded by severity, using different levels of intrusiveness based on risk. They must incorporate smart suppression options, allowing the clinician to indicate that they have assessed an alert and do not wish to see it repeatedly for the same patient. The clinician who participates in the design of these logic rules can contribute clinical knowledge that engineers lack, creating systems that are truly useful

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

rather than merely well intended.

There is also an ethical dimension to a physician's digital literacy that must not be overlooked. When healthcare professionals use software, they are entrusting their clinical judgment to a device designed by others, and that trust must be informed and critical. They must know how to ask: How has this system been validated? In which populations has it been tested? What biases might it contain? Who is accountable if it fails? These questions are not hostile but rather responsible. A physician who asks them is exercising their duty to protect their patients, and developers should receive them for what they are: an aid to building better products, not a threat to their work.

Physicians' ethical responsibility extends to understanding algorithmic biases and their implications for healthcare equity. Artificial intelligence systems trained on historical data reproduce past treatment patterns, which often reflect inequalities based on sex, race, or social class. A transplant prioritization algorithm trained on historical decisions can perpetuate forms of discrimination that an individual physician would not endorse. An informed professional must be able to question the system's recommendations when they contradict their clinical judgment, request information about the training dataset, and demand equity audits to verify that the software performs equally well for all patient groups.

Finally, it is important for physicians to understand the nature of multidisciplinary teamwork. Developing healthcare software is not like writing a scientific article, where each author contributes their part and the whole is assembled at the end. It is an iterative, sometimes chaotic, process in which ideas are collectively built and rebuilt, where disagreements are frequent, and where the ability to listen and translate is more valuable than specialized knowledge. A physician sitting at a table with engineers is not there to give orders or to receive them but to build together a shared understanding of the problem and possible solutions. That collective construction is slow and sometimes frustrating, but it produces results that neither party can achieve on its own.

A doctor's digital literacy is not, therefore, a mere curricular embellishment or a concession to modernity. It is an indispensable working tool in a healthcare system that can no longer be understood without technology. It is also a way to fully practice the profession, maintaining control over the tools used rather than being controlled by them. The physician who understands software is not a physician who programs but a physician who retains their professional autonomy in an environment that is increasingly mediated by machines. Ultimately, this is what benefits their patients the most.

If the physician needs to learn the language of software to avoid losing professional autonomy, what about the engineer who enters the world of medicine without understanding that here, errors are not fixed with patches but can cost lives and that clinical precision demands a humility that technological culture often does not foster?

1.6 CROSS-LITERACY: WHAT ENGINEERS NEED TO KNOW ABOUT MEDICINE

Let us now imagine the scene from the opposite perspective. A team of doctors has identified a recurring problem in their department: the delayed detection of deterioration in hospitalized patients. With the best of their intentions, they sit down to design a technological solution and draft a specifications document that they hand over to the engineers. The document is exhaustive: it describes screens, functions, and algorithms—all with a level of detail that the doctors consider sufficient. When engineers receive that document, however, they experience a mix of perplexity and unease. The instructions are precise on the surface, but they omit exactly what the engineers would need to know to build something useful: what data is actually available? How do they relate to one another? What exactly does "deterioration" mean in measurable terms? What decisions will be made on the basis of the information the system provides? Again, the problem is not technical but one of translation, and this time, it is the engineers who must learn the language of their clinical counterparts.

The first thing an engineer must understand when entering the world of medicine is that clinical practice is unlike any other field they have worked in before. It is not like developing software for the banking sector, where transactions follow precise and predictable rules. It is not like creating an e-commerce application, where user behavior can be modeled with reasonable accuracy. Medicine is a field governed by uncertainty, where two patients with the same diagnosis may respond completely differently to the same treatment, where the clinician's intuition and experience carry as much weight as objective data and where decisions are made with incomplete information and under time pressure. An engineer who fails to internalize this reality will build systems that are elegant on paper but useless in practice because they will be

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

based on assumptions that do not hold true in the real world.

The second fundamental lesson for the technician is that in medicine, the context is everything. A blood pressure reading of 140 over 90 may be normal in one patient and alarming in another. A heart rate of 110 beats per minute may indicate a dangerous arrhythmia or simply that the patient is nervous about the visit. A slightly abnormal lab result may be irrelevant or may be the first clue to a serious illness, depending on the complete medical history. Engineers, accustomed to working with data that are the same thing in all situations, must learn that in medicine, meaning emerges from the integration of multiple sources of information, many of which are not digitally encoded. A system that ignores that context will not only be less useful but also potentially dangerous because it will present as certainties what are merely clues.

1.6.1 CLINICAL VOCABULARY: MORE THAN MERE DEFINITIONS

There is also a set of basic clinical concepts that every engineer involved in healthcare projects should be fluent in. They do not need to diagnose diseases, but they do need to understand the difference between signs and symptoms, between sensitivity and specificity, and between incidence and prevalence. They must understand how healthcare is organized: what a primary care physician does, what a specialist does, what role nurses play, and how responsibilities are distributed in a hospital. It is beneficial to know the basic structure of a medical record, the most common types of diagnostic tests, and the basic vocabulary of pharmacology. In addition, you should have a basic understanding of clinical reasoning—how doctors arrive at a diagnosis, the heuristics they use, and the biases to which they are subject. This knowledge won't turn you into a clinician, but it will allow you to have conversations where both parties feel heard and respected.

The seemingly simple distinction between signs and symptoms has profound implications for the design of medical software. Symptoms are what the patient subjectively reports: pain, dizziness, and fatigue. Signs are what the doctor objectively observes: a measured fever, an abnormal auscultation, and a lab result. A system that fails to distinguish this fundamental difference can make serious errors: treating subjective data as if it were objective, ignoring the importance of the patient's perception, or vice versa. The engineer must understand that both types of information are valuable but of a different nature and that the software must allow them to be recorded, weighted, and related in ways that reflect the complexity of clinical reasoning.

The concepts of sensitivity and specificity are particularly critical when the software includes diagnostic or detection components. Sensitivity measures the system's ability to correctly identify those who have the condition; specificity measures the ability to correctly rule out those who do not. A sepsis detection algorithm with high sensitivity but low specificity will generate many false alarms, consume resources and cause fatigue. Cases with high specificity but low sensitivity will miss actual cases, with potentially fatal consequences. The engineer must understand that there is no universal optimal configuration: the choice depends on the severity of the condition, the costs of false positives versus false negatives, and the system's capacity to handle alerts. These are clinical decisions that require dialog with professionals, not isolated technical optimizations.

An understanding of the healthcare organization prevents design errors that assume nonexistent structures or ignore operational realities. The engineer must know that primary care filters and refers patients, that specialists have waiting times that dictate the urgency of decisions, that nurses manage much of chronic care and that pharmacists are key partners in treatment adherence. A system designed without this understanding can create impossible workflows: requiring specialized decisions within timeframes that cannot be met, omitting key actors in the care chain, or assuming the availability of resources that do not exist in reality. Mapping actors and processes must precede technical design; it should not be an afterthought.

1.6.2 CLINICAL TIME: RHYTHMS THAT CANNOT BE COMPRESSED

One of the most difficult concepts to convey to engineers is the nature of time in clinical practice. In the world of software development, time is often conceived as a resource that can be organized and optimized. In medicine, time is often an enemy: the time that passes while a patient waits for a test, the time running out for someone with internal bleeding, and the time lost when a computer system runs slowly. However, it is also an ally: the time that allows an infection to resolve, the time that provides perspective to assess the progression of a chronic disease, and the time at which doctors need to explain a complex diagnosis to a frightened patient. An engineer

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

who designs healthcare systems must constantly ask themselves, “Does my tool speed up what needs to be sped up and slow down what needs to be done calmly? Does it respect the natural rhythms of care or impose an artificial pace that undermines the clinical relationship?

Urgency in medicine is neither uniform nor predictable. There are moments when every second counts: a cardiac arrest, massive bleeding, and a severe asthma attack. In these scenarios, the software must be invisible, fast, and reliable, adding no friction to actions that should be automatic. However, there are also moments when haste is counterproductive: the consultation of a patient with a complex chronic illness, decision-making regarding cancer treatments, and the delivery of bad news. Here, the software must allow for reflection, the consultation of additional information, and the necessary pause for ethical deliberation. A system that imposes the same pace on all these situations—one that optimizes for speed without distinguishing the context—will be detrimental regardless of its technical efficiency.

Interruption is another temporal dimension that engineers frequently underestimate. In software development, the state of flow is valued and protected; interruptions are considered detrimental to productivity. In medicine, interruptions are inevitable and often necessary: an emergency call, a consultation with a colleague, and a monitor alarm. Medical software must be designed for this fragmented environment, allowing clinicians to leave and resume tasks without losing context, maintaining the state of incomplete sessions, and providing summaries that facilitate quick reentry. A system that requires uninterrupted sequences of interaction ignores the reality of clinical work and generates frustration or, worse, errors when users attempt to speed up processes to compensate for interruptions.

Another crucial aspect is the hierarchy of information. In a typical software system, all the data usually carry the same weight or at least a weight that can be determined by explicit rules. In medicine, not all data are equal: there is critical information that must always be visible, secondary information that can be consulted if necessary, and irrelevant information that only adds noise. A doctor treating an emergency needs to know immediately if the patient is allergic to any medication but can wait to learn their educational background or profession. A system that does not distinguish these levels of importance—one that presents all information with equal emphasis—will force the clinician to search for what is relevant amid a sea of data and increase the risk of overlooking something crucial. The engineer must learn to ask: What does the doctor need to know at any given moment? What can wait? What is best left unseen so as not to distract them?

1.6.3 UNCERTAINTY AS DATA, NOT AS ERROR

Uncertainty is another concept that engineers must learn to handle with respect to humility. In the world of software, uncertainty is often viewed as a problem to be eliminated: tests must always pass, responses must be consistent, and errors must be corrected. In medicine, uncertainty is inherent: it is part of the very nature of clinical work. A doctor can never be completely sure of a diagnosis until it is confirmed by tests and sometimes not even then. A treatment that works for most patients may not work for a specific patient. A prognosis is always a probabilistic estimate, not a certainty. Systems that ignore this uncertainty—which presents as certain predictions that are only probable—will mislead clinicians and lead them to make worse decisions than they would make without assistance. Good medical software does not eliminate uncertainty but helps surgeons navigate it in an informed way.

Presenting probabilities in medical interfaces requires special care that engineers often overlook. Humans, including doctors, are intuitively poor at probabilistic reasoning. Presenting a risk as “15% mortality” does not have the same emotional impact or imply the same decisions as presenting it as “85% survival,” even though they are mathematically equivalent. The algorithms that produce risk scores must communicate not only the numerical value but also its accuracy, its limitations, and the factors that most influence it. Engineers must collaborate with psychologists and clinicians to design representations that are honest about the underlying uncertainty without paralyzing decision-making.

Managing uncertainty also involves knowing when the system should not make a decision. There are situations where the information is so incomplete or ambiguous that any algorithmic recommendation is unwise. A good medical software system must recognize these limits, alert the clinician to the lack of data, and control the human judgment by hand. This contradicts the engineering tendency to seek solutions for every scenario and to view edge cases as technical problems to be solved. In medicine, acknowledging uncertainty is sometimes the safest approach,

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

and the software must allow for this option without penalizing the user with error messages or blocked workflows.

The issue of clinical workflows deserves special attention, as it is among the main sources of friction between doctors and computer systems. Engineers tend to think in terms of linear processes—first you do this, then that, then the next thing. However, real clinical work is inherently interrupted and multitasking. A doctor might be writing a patient's notes, receive an urgent call, attend to a nurse who needs a prescription, and then return to the notes, all within a matter of minutes. A system that does not support this type of interruption—one that forces a task to be completed before starting another or that loses its state when the doctor switches screens—will be perceived as an obstacle rather than a help. Engineers who observe real clinical work, who spend time in clinics and on wards and who understand the organized chaos of a hospital, are far more likely to design tools that integrate naturally into that workflow.

Ethnographic observation of clinical work should be mandatory for every engineer designing healthcare systems. It's not about reading procedure manuals or interviewing doctors in conference rooms but about spending hours in the real environment, seeing how decisions are actually made, what information is consulted and when, what shortcuts are used, and what frustrations arise. This immersion reveals discrepancies between the official and actual workflows, identifies opportunities for improvement that are not evident from the outside, and builds genuine empathy between the designer and the user. Projects that skip this phase often produce systems that meet formal specifications but fail to be adopted because they do not align with the complex reality of practice.

1.6.4 THE THERAPEUTIC RELATIONSHIP: TECHNOLOGY AT THE SERVICE OF THE ENCOUNTER

Communication with patients is another dimension that engineers often underestimate. To a technician, the doctor–patient relationship may seem like an incidental add-on, something that occurs on the margins of the system. However, it is precisely the heart of medicine. Patients who do not trust their doctor will adhere less to treatments, share less relevant information, and have poorer health outcomes. Any technology that interferes with that relationship, that is, that comes between the doctor's gaze and the patient's gaze, that turns the consultation into an exchange of data rather than a human encounter, will cause harm that is difficult to measure but very real. The engineer must ask: Does my tool facilitate communication or hinder it? Does it free up time so that the doctor can listen better, or does it tie them up with administrative tasks? Does it help humanize care or dehumanize it?

The design of interfaces for medical consultations requires specific considerations that go beyond general usability. The screen must be positioned so that it does not physically stand between the doctor and the patient, allowing for eye contact. Data entry must be possible without the doctor taking their eyes off the patient and using input devices that do not require full visual attention. System response times must be fast enough to avoid creating awkward silences that disrupt the flow of conversation. The terminology displayed must be understandable to the patient if the screen is visible or hidden if it contains sensitive information that could be alarming without context. These seemingly minor details determine whether technology facilitates or hinders the therapeutic relationship.

Another essential concept is the variability of clinical practice. Engineers, accustomed to seeking universal solutions, often assume that there is one correct way to do things and that all doctors should adapt to it. However, the reality is that two excellent clinicians may approach the same problem in very different ways, and both may be equally valid. Personal preferences, training received, accumulated clinical experience, and even personality influence how each doctor works. A rigid system that imposes a single way of doing things will generate resistance, be perceived as an imposition, and likely be sabotaged in more or less subtle ways. Flexibility—the ability to adapt to different work styles—is a virtue that engineers must incorporate into their designs.

Workflow customization does not mean a lack of structure but rather configurability within safe parameters. The engineer must identify which aspects of the process are critical for safety and must be standardized and which are legitimate preferences that can be adapted. The sequence of an electronic prescription has mandatory components for safety reasons, but the order in which data are entered, the presentation of interaction information, and the method of confirming the dose can vary according to user preferences. Offering customization options, saving user-specific settings, and allowing shortcuts that do not compromise safety are ways to respect professional

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

autonomy without sacrificing standardization where necessary.

The legal and ethical dimensions of clinical work are also uncharted territory for most engineers. In the world of software, errors typically result in consequences limited to financial losses or user inconvenience. In medicine, an error can mean the death of a person and that possibility is always present, fostering a culture of prudence and responsibility that permeates every decision. Engineers working in this field must understand that their systems will be used by professionals who are afraid of making mistakes, who know that the lives of others depend on their decisions, and who carry the emotional burden of past errors. A system that ignores this reality—one that adds more pressure or more opportunities for error to an already stressful job—will be doomed to failure and will cause harm in the process.

There is also an emotional aspect to medicine that engineers tend to overlook: suffering. Patients are not just bodies with illnesses; they are people with fears, hopes, families, and stories. Doctors are not merely implementers of protocols; they are human beings who choose every day to confront the pain of others as a way of life. Software that treats patients as mere sets of data, which reduces their experience to coded variables and ignores the human dimension of the clinical encounter, will be betraying the very essence of medicine. The engineer who designs it may never have seen a patient receive a devastating diagnosis, may never have been in an oncology waiting room, and may never have witnessed the relief in a mother's eyes when she is told her child is well. If they haven't had those experiences, they should at least listen to those who have and let them guide their work.

Finally, it is important for engineers to understand something about their own place in this ecosystem. They are not casual visitors arriving with prefabricated solutions but rather other members of a team that share a common goal: improving people's health. Their technical knowledge is valuable, but it is not the only valuable knowledge. Doctors know things that engineers do not and recognize that asymmetry is not a form of humiliation but rather the first step toward fruitful collaboration. The engineer who arrives with humility, who asks questions, who listens, who observes, and who respects clinical knowledge even if they do not fully understand it, will be welcomed. The one who arrives with arrogance, convinced that technology will solve everything, will be rejected—and rightly so. At the intersection of medicine and engineering, as at all intersections, courtesy and mutual respect are just as important as technical knowledge.

This cross-disciplinary literacy that we have explored in the last two sections is, therefore, neither a luxury nor a courtesy. It is the basic infrastructure upon which a genuine collaboration can be built between two worlds that need each other. Doctors need to understand software so as not to be mere passive recipients of technology designed by others. Engineers need to understand medicine so that they do not build tools that look brilliant on paper but are useless in practice. When both make the effort to learn each other's language, when they sit down at the table with respect and curiosity, then truly extraordinary things can happen. Afterward, the software ceases to be a foreign body implanted in the healthcare system and becomes a natural extension of human intelligence and compassion.

If we have established that doctors and engineers need to learn each other's languages to collaborate effectively, how do we build a common technical vocabulary that allows both worlds to communicate precisely when discussing safety, quality, and risk—concepts that have profoundly different yet equally valid meanings in each discipline?

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

REFERENCES

Software as a Medical Device | International Medical Device Regulators Forum [Internet]. [cited May 13, 2026]. Available from: https://www.imdrf.org/working-groups/software-medical-device
Medical Device Software Guidance Navigator | FDA [Internet]. [Cited May 13, 2026]. Available from: https://www.fda.gov/medical-devices/regulatory-accelerator/medical-device-software-guidance-navigator
Digital Health Software Precertification (Pre-Cert) Pilot Program | FDA [Internet]. [Cited May 13, 2026]. Available at: https://www.fda.gov/medical-devices/digital-health-center-excellence/digital-health-software-precertification-precert-pilot-program
Gotterbarn D. Software Engineering Ethics. Encycl Softw Eng [Internet]. January 15, 2002 [cited May 13, 2026]; Available from: https://ethicsunwrapped.utexas.edu/case-study/therac-25
Obermeyer Z, Powers B, Vogeli C, Mullainathan S. Dissecting racial bias in an algorithm used to manage the health of populations. Science (80-) [Internet]. October 25, 2019 [cited May 13, 2026]; Available from:/doi/pdf/10.1126/science.aax2342?download=true
Collier R. NHS ransomware attack spreads worldwide. C Can Med Assoc J [Internet]. June 5, 2017 [cited May 13, 2026];189(22):E786. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC5461132/
MacIntyre CR, Lim S, Quigley A. Preventing the next pandemic: Use of artificial intelligence for epidemic monitoring and alerts. Cell Reports Med [Internet]. December 20, 2022 [cited May 13, 2026];3(12):100867. Available at: https://www.sciencedirect.com/science/article/pii/S2666379122004311
Cross JL, Choma MA, Onofrey JA. Bias in medical AI: Implications for clinical decision-making. PLOS Digit Heal [Internet]. November 1, 2024 [cited May 13, 20

https://doi.org/10.62486/978-9915-9876-8-2.Ch01

Chapter 2 / Capítulo 2

Check for updates

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch02
Pages: 27-49

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 2. Introduction to Regulated Environments

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Regulation in digital health is not an obstacle that innovators must overcome but rather a map that shows them where the cliffs are. Ignoring it does not make them disappear; it only ensures that the fall will be more painful."

There is a widespread narrative in the world of technological innovation that portrays regulation as the enemy. In this story, regulators are slow-moving bureaucrats who do not understand technology, who impose absurd obstacles, who hinder progress, and who protect established interests against the brave entrepreneurs who want to change the world. It is a seductive narrative because it casts the innovator in the role of the hero fighting the system and because it offers a simple explanation for failure: it wasn't my idea; it was the system. However, like all simple narratives, this one hides a more complex truth—and, for those who truly want to transform healthcare, a more useful one. Regulation is not an arbitrary obstacle, but the collective expression of something medicine has learned the hard way over centuries: that before testing something on people, one must ensure it won't kill them.

This chapter presents the second part of the book, which is devoted entirely to regulation and quality, because understanding this regulatory framework is a necessary condition for building healthcare software worthy of name. The goal is not to turn the reader into a legal expert but to equip them with a compass to navigate a territory that, although seemingly arid, holds the accumulated wisdom of decades of mistakes and successes. We explore why regulation exists, what problems it solves, what philosophy underpins it, and how to approach it not as a burden but as a tool for building better. In addition, we will do so by starting with the most fundamental question: why on earth does any of this matter?

2.1 WHY REGULATION MATTERS (AND WHY IT IS NOT EMPTY BUREAUCRACY)

To understand why regulation exists in the healthcare sector, it is helpful to conduct a thought experiment. Imagine that a friend, with the best intentions in the world, develops an app to help patients with diabetes calculate their insulin doses.

He's a programmer; he's spoken with some doctors, read up on the disease, and is convinced that his tool can improve the lives of millions of people. He uploads it to app stores and promotes it on social media, and within a few months, thousands of users use it daily to decide how much insulin to inject. One day, a coding error causes the app to recommend a dose ten times higher than necessary. Several patients suffer from severe hypoglycemia, some of whom end up in the emergency room; another, the most unfortunate one, does not make it in time. Who is responsible? The programmer, who did what he believed was right? The patients, who trusted an app without medical support? The app stores, which distributed it without any oversight? Do the doctors who failed to warn their patients of the risks?

This scenario, which unfortunately is not hypothetical, perfectly illustrates the need for regulation. Not because programmers are irresponsible but because medical software, like any other medical device, can cause serious harm when it fails. Unlike other fields where errors have limited consequences, here, what is at stake is people's health and lives. Regulation exists to establish a framework that minimizes the likelihood of these failures occurring and, when they do occur, to ensure clarity regarding liabilities and mechanisms for redress. It is not a bureaucratic whim but a societal response to a real problem: how to benefit from technology without exposing ourselves to unacceptable risks.

The history of healthcare regulation is, to a large extent, the history of preventable disasters that led society to say "never again." Thalidomide in the 1960s, a drug that caused birth defects in newborns because it had not been sufficiently tested on pregnant women, gave rise to modern clinical trials.(1) The THERAC-25 disaster we observed in the previous chapter, with its lethal radiation caused by software errors, spurred specific regulations for programmable medical devices. Every tragedy has left its mark on the regulatory framework, not out of a desire to make life difficult for innovators but out of a collective determination to ensure that it never happens again. When a developer complains about how strict the regulations are, it is worth reminding them that they are written in the blood of those who paid the price for deregulation.

2.1.1 THE ANATOMY OF RISK: WHY MEDICAL SOFTWARE IS DIFFERENT

Some argue that this comparison is exaggerated, that software is not like drugs, that the risks are lower, and that regulation should be lighter. However, this stance ignores an uncomfortable

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

reality: medical software has characteristics that make it especially dangerous when it fails. It is opaque because its internal mechanisms are not visible to those who use it. It is complex because it interacts with other systems in unpredictable ways. It is modifiable because it can be updated without the user being aware of the changes. It is ubiquitous because it can be distributed to millions of people in a matter of hours. This combination of factors means that software errors can have massive reach and be extremely difficult to detect before they cause harm. Regulation is not an overreaction but a proportionate response to real risk.

The opacity of medical software presents unique challenges for error detection. When a physical device fails, the signs are usually obvious: abnormal noise, a visibly damaged component, and a function that stops responding. Software, however, can fail in completely invisible ways, producing incorrect results that seem reasonable at first glance. A diagnostic algorithm that introduces a systematic bias against a certain demographic group does not set off alarm bells; it simply produces erroneous diagnoses that can be attributed to other factors. The complexity of modern systems, with millions of interconnected lines of code, means that even their creators cannot fully predict their behavior in all circumstances. Regulation attempts to mitigate this opacity by requiring documentation, traceability, and independent verification mechanisms.

The ability to continuously modify software, seemingly an advantage, introduces stability risks that do not exist in physical devices. A mechanical infusion pump functions in the same way throughout its entire lifespan, resulting in physical wear and tear. Infusion pump control software can radically change its behavior with a remote update, potentially introducing errors that were not present in previous versions. This mutability requires change control mechanisms that assess the impact of each modification, regression testing processes that verify that fixes have not broken previous functionality, and postmarket surveillance systems that detect emerging issues. Modern regulation seeks to adapt to this reality with configuration management frameworks that treat software as a living product, not as a finished artifact.

Digital ubiquity allows errors to spread at unprecedented speed and scale. An error in a defective batch of drugs affects patients receiving that specific batch. An error in a popular health app can affect millions of users simultaneously and distributed globally before anyone is aware of the problem. This characteristic demands early detection mechanisms, rapid alert systems, and the capacity for mass recall or correction—which regulation seeks to establish. Responsibility in this context cannot fall solely on the end user, who lacks the technical capacity to assess the safety of the software they use; it must be distributed among developers, distributors, and regulators in a way that creates incentives for prevention.

2.1.2 THE FIVE PILLARS OF HEALTH REGULATION

It is also important to distinguish between different types of regulation because not all have the same origin or the same justification. Each of these layers has its own logic and its own history, and confusing them leads to fruitless debates where arguments that should be addressed separately are mixed together.

Safety regulations constitute the most fundamental and least controversial pillar. They stipulate that a medical device must not cause harm when used as intended and that residual risks must be acceptable in relation to the expected benefits. In the context of software, this translates into requirements for reliability, error handling, protection against misuse, and redundancy in critical functions. The IEC 62304 standard, which we discuss in later chapters, establishes a medical software lifecycle framework that prioritizes safety at every stage of development. This regulation is inherently preventive: it aims to eliminate or mitigate hazards before the product reaches the patient.

Efficacy regulation addresses a different but equally legitimate concern: that public and private healthcare resources are not spent on products that do not work. Software that promises to improve the diagnosis of skin cancer but, in practice, is no better than the visual assessment of an experienced dermatologist represents a waste of resources and false hope for patients. Efficacy must be demonstrated through properly designed studies, with relevant endpoints, representative populations, and rigorous statistical analyses. This regulation protects both patients and healthcare systems from irrational spending on useless or overrated technologies.

Quality regulation extends beyond the individual product to encompass the organizational processes that produce it. The ISO 13485 standard, applicable to quality management systems for medical devices, requires companies to maintain documented procedures for all critical activities, from supplier management to change control and from staff training to complaint management

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

and postmarket surveillance. The underlying logic is that safe and effective products cannot be produced consistently on a basis without an organization that systematically ensures this. This process regulation is often the most burdensome for developers, but experience shows that omitting it leads to inconsistencies that eventually result in failures.

Privacy and data protection **regulations**, embodied in Europe by the GDPR and in the United States by the HIPAA, among other regulations, recognize that health information is particularly sensitive. Medical data can reveal stigmatizing conditions, genetic predispositions, and behaviors that the patient would prefer to keep confidential. Its unauthorized disclosure can cause psychological, social, and economic harm that persists long after the underlying medical condition has been resolved. Regulations in this area establish patients' rights over their data, obligations for those who process it, and limits on the secondary uses that can be made of health information.

Market regulation ensures that competition among healthcare products occurs under conditions of transparency and fairness. It establishes labeling requirements that enable informed comparisons, prohibits deceptive commercial practices, and creates oversight mechanisms that are removed from the market products found not to deliver on their promises. This regulation is less visible to the general public but essential for the functioning of a healthcare market that is not swayed by empty marketing claims.

A common mistake among critics of regulation is to treat it as a monolithic block; in reality, it is a set of tools with varying degrees of justification. It is not the same to require that diagnostic imaging software have a minimum accuracy proven in clinical studies, as it is to require that a hospital's appointment management system meet certain quality standards. The former directly affects patients' health; the latter, primarily administrative efficiency. Lumping them together and saying "regulation is a problem" prevents us from distinguishing where it is necessary and where it might be excessive. A mature stance toward regulation consists neither of accepting it all uncritically nor of rejecting it wholesale but rather of understanding its foundations so that we can participate in its improvement.

2.1.3 THE ECONOMICS OF REGULATION: PROTECTING THOSE WHO DO THINGS RIGHT

There is also an economic dimension to regulation that deserves attention. In an unregulated market, medical devices compete on price and visible features but not necessarily on safety or efficacy. An unscrupulous developer could launch an unsafe but inexpensive product, and consumers who have no way of assessing safety might choose it simply because of the price. This creates what economists call a "race to the bottom," where safe but more expensive products are driven out of the market by unsafe but cheap ones. Regulation establishes a minimum quality standard that prevents this perverse dynamic and protects those who do things right from unfair competition by those who do them wrong. It is far from being an obstacle to quality innovation; it is its best ally.

The "race to the bottom" argument illustrates why pure self-regulation is insufficient in the healthcare sector. Responsible companies might try to differentiate themselves through voluntary certifications or proactive transparency, but in an asymmetric market where consumers cannot assess technical safety, these signals are easily imitated by unscrupulous competitors. Only a regulatory framework with independent verification mechanisms and credible sanctions can resolve this collective action dilemma, where what is rational for each individual actor leads to a socially undesirable outcome. Regul—is not paternalism toward irrational consumers but rather a recognition that the technical information needed to evaluate medical devices is inherently asymmetric.

Another common argument against regulation is that it stifles innovation and delays the arrival of treatments that could save lives. There is some truth to this claim: regulatory processes undoubtedly add time and cost to the development of medical devices. However, what this view overlooks is that regulation also accelerates the adoption of safe technologies by building trust among physicians, patients, and payers. A product that has undergone rigorous regulatory evaluation is far more likely to be adopted by healthcare systems and prescribed by doctors than one that has not. The speed of market entry is lost when the product reaches the market. Above all, it prevents the most tragic scenario: that of products that reach the market quickly but kill people, causing a decline in public trust that affects the entire sector.

The relationship between innovation and regulation need not be antagonistic, and in fact, countries with more developed regulatory systems tend to produce the most healthcare

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

innovation. This is no surprise: predictable, well-designed regulation provides a framework of certainty that allows innovators to plan for the long term and investors to risk their capital with some assurance. What kills innovation is not regulation but regulatory uncertainty: not knowing what to expect, rules changing without warning, and evaluation criteria that are opaque or arbitrary. That is why experienced developers do not call for less regulation but for clear, stable, and proportionate regulation. They know that this is the environment where they can build sustainable businesses and products that truly reach patients.

The empirical evidence on the impact of regulation on innovation is more nuanced than ideological debates suggest. Comparative studies across jurisdictions with different levels of health regulation do not show a simple correlation between less regulation and more innovation. What does consistently emerge is that the quality of regulation—measured by predictability, transparency, and responsiveness—matters more than its quantity does. Regulatory systems that have implemented fast-track pathways for products addressing high unmet needs, that maintain ongoing dialog with the industry and that update their frameworks to incorporate scientific advances, succeed in combining patient protection with the promotion of innovation. Smart regulation is a sophisticated public policy tool, not a mere bureaucratic obstacle.

2.1.4 PROCESS VERSUS OUTCOME: REGULATORY BALANCE

A useful distinction for navigating this debate is the one that separates "process" regulation from "outcome" regulation. The former establishes how things should be done: which development methods to use, what documentation to generate, and what controls to implement. The latter focuses on what the product must demonstrate: that it is safe, that it is effective, and that its benefits outweigh its risks. Developers tend to complain more about the former than about the latter because process regulation imposes ways of working that they may perceive as rigid or alien to their culture. However, experience shows that both are necessary: without processing, it is difficult to ensure that results are reproducible; without results, the process becomes an empty exercise. The balance between the two is the subject of ongoing debate, and modern regulatory frameworks seek to find the optimal point.

Process regulation in medical software, embodied in standards such as IEC 62304, has evolved to recognize the unique characteristics of software development. Unlike the manufacturing of physical devices, where the production process is relatively standardized, software development allows for multiple valid methodologies, from waterfall models to agile development and from extreme programming to formal software engineering. Modern regulation aims to be agnostic regarding specific methodologies, focusing instead on the outcomes of the process: requirements traceability, configuration control, risk management, verification, and validation. This methodological flexibility, within a framework of output requirements, represents an advance over more rigid regulations that imposed specific processes incompatible with the reality of contemporary software development.

However, methodological flexibility does not mean a lack of rigor. Medical software development processes must be documented, measurable, and auditable. It must be possible to reconstruct why certain design decisions were made, how it was verified that the code correctly implemented the requirements, what tests were performed, and with what results. This documentation is not an end in itself but a means to enable incident investigation, continuous improvement, and knowledge transfer when development teams change. The tension between agility and documentation is real, but it can be resolved through tools and practices that integrate documentation generation into the workflow rather than adding it as a subsequent burden.

For its part, outcome-based regulation faces the challenge of defining what evidence is sufficient to demonstrate safety and effectiveness in a field as diverse as medical software. With respect to diagnostic software, this may involve studies comparing results with those of human experts, ROC curve analyses, and evaluations under real-world conditions. For treatment software, comparative clinical trials against current standards of care may be needed. For information management software, the evidence may focus on usability, interoperability, and data security. The diversity of products makes excessive standardization difficult, but it also creates inconsistency if each evaluator applies different criteria. Regulators attempt to manage this tension through product-category-specific guidelines while maintaining common general principles.

For the physician reading this book, understanding the logic of regulation is also a way to protect their own practices. When a healthcare professional uses a product that has not passed the appropriate regulatory checks, they are assuming a risk that is not theirs to bear. If something

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

goes wrong, liability may fall on them for having used an unvalidated tool. Physicians have the right to demand that the products they use have the proper authorizations, and they should be wary of those that circumvent them with arguments of innovation or agility. Regulation is not only a safeguard for patients but also for the professionals who care for them.

The involvement of clinicians in defining and improving regulation is essential, yet it is rarer than it should be. Doctors who have seen first-hand what happens when technology fails—who have had to explain to a family why a software error contributed to the death of their loved one—have a perspective that regulators and industry representatives cannot replace. Their voices should be heard on the committees that design the rules, in the agencies that enforce them, and in public debates about reform. Health care regulation is too important to be left solely in the hands of bureaucrats and business; it needs the perspective of those who have dedicated their lives to caring for patients and know what truly matters when theory meets practice.

In the specific realm of medical software, regulation poses unique challenges that set it apart from other medical devices. Software can be continuously updated, raising the question of when an update is minor and when it is so substantial that it should require a new authorization. Software can be distributed globally in a matter of minutes, challenging regulatory models based on national borders. Software increasingly incorporates artificial intelligence, which learns and changes with use, calling into question the very idea of a "finished" product that is evaluated once and is done. Regulators around the world are grappling with these issues, and the answers they provide in the coming years will shape the future of digital health.

For developers new to this field, the advice could not be clearer: do not view regulation as a formality to be addressed at the end, once the product is already finished. That is a sure path to failure, because you will then discover that there are fundamental aspects you should have considered from the start and that modifying them is now impossible or prohibitively expensive. Regulation must be integrated into the development process from the very first sketch, as just another design variable, as a constraint that, when properly understood, can even stimulate creativity. The best healthcare products are not those that circumvent regulation but those that incorporate it as part of their value proposition.

It is also worth debunking the myth that regulation is something that matters only in wealthy countries and that in developing countries, it can be ignored without consequences. This view, in addition to being paternalistic, is profoundly mistaken. Patients worldwide deserve the same protection against unsafe products, and the healthcare systems of countries with fewer resources are, if anything, more vulnerable to the harm caused by defective technology. Furthermore, ignoring regulation closes the door to international collaboration and funding because investors and global partners demand minimum standards. Regulation is not a luxury for wealthy countries but a prerequisite for participation in the global digital health market.

To conclude, this introduction to the logic of regulation, let us return to the thought experiment from the beginning. Let's now imagine that our developer friend, instead of simply launching his app, has taken the time to understand the applicable regulatory framework. He has correctly classified his product as a Class IIb medical device, implemented a quality system in accordance with ISO 13485, documented the development process in accordance with IEC 62304, conducted risk management in accordance with ISO 14971, and subjected the product to a clinical evaluation demonstrating its safety and effectiveness. When their app reaches the market, it won't be perfect—because nothing is—but it will be far less likely to contain serious errors. If an incident does occur despite everything, there will be traceability to investigate it, processes to correct it, and clarity regarding responsibilities. That product, even if it took longer to arrive, is infinitely preferable to the first one, both for patients and for the developer themselves, who will be able to sleep at night without the nightmare of having caused preventable harm.

Regulation matters because medicine matters. This matters because patients believe that the products they use have been rigorously evaluated. Doctors need reliable tools to practice their profession. It matters because society has decided, after many painful lessons, that health is not just another consumer good but also a right that deserves special protection. Understanding this is not an act of faith but the first step toward building healthcare technology that truly deserves that name. It is also the gateway to the more concrete details we explore in the coming chapters: the specific standards that translate this philosophy into tangible requirements, the processes that turn principles into practice, and the tools that allow us to navigate the regulatory maze without being lost in it.

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

2.2 TRACEABILITY FOR DEVELOPERS ACCUSTOMED TO JIRA

> "Traceability isn't for the present; it's for the future, and the future is unpredictable."

Let's imagine for a moment the everyday scene at a conventional software startup. The team uses Jira, or a similar tool, to manage their work. Each task is represented by a card that moves through different columns: pending, in development, in testing, completed. Developers create branches in the code repository, link their commits to the corresponding tasks, and when everything is ready, they merge the changes and close the issue. It is an efficient, flexible workflow that has proven its value in thousands of successful projects. No one questions whether this way of working is suitable for software development in general. The problem arises when this same team, with this same culture, faces a medical software project for the first time, and someone tells them that they need "traceability." Suddenly, what was once simple and agile becomes complex and bureaucratic. Developers feel trapped in a maze of documents, records, and requirements that they do not understand and perceive as a hindrance to their creativity. However, traceability is not a whim of regulators but a necessity stemming from something we have already explored: in medicine, errors have consequences we cannot afford.

Traceability is the ability to reconstruct the path from a clinical need to the code that implements it and from a piece of code to the need that justifies it. In other words, it is the ability to answer questions such as the following: Why was this line of code written? What requirement was it fulfilling? Who decided it? When was it done? What tests verify that it works correctly? Conversely, given a clinical requirement, where is it implemented in the code? How can I ensure it is met? What would happen if I modify it? This ability to trace design and development decisions in both directions is what distinguishes medical software from a conventional application, and it is also what allows for investigating incidents, correcting errors, and demonstrating to a regulator that the product has been developed with the necessary rigor.

For a developer accustomed to Jira, this concept may not be entirely unfamiliar. After all, in many projects, commits are linked to tasks, which is already a rudimentary form of traceability. The difference lies in the level of rigor and purpose. In a conventional project, the relationship between tasks and code is useful for understanding what was done, but it is rarely required for audit or security purposes. If a developer forgets to link a commit to a task, the worst that can happen is that someone has to dig a little deeper to understand the change. In a medical project, the same omission can have serious consequences when, months later, an incident forces a review of the history, and it cannot be determined why a modification was introduced that turned out to be dangerous. Medical traceability is not an organizational option but a safety requirement.

2.2.1 FROM JIRA TO BIDIRECTIONAL TRACEABILITY: EXPANDING THE MODEL

The standard for establishing traceability requirements for medical software is IEC 62304, which we explore in detail in the next chapter. For now, suffice it to say that this standard requires a bidirectional relationship between requirements, design, code, testing, and identified risks. This means that for every requirement, there must be a code that implements it and tests that verify it. For every piece of code, there must be a requirement that justifies it. For every identified risk, control measures must be implemented and verified. This entire network of relationships must be documented in a way that allows it to be audited by a third party. It is not about generating mountains of meaningless paperwork but about building a network of connections that allows us to navigate the system safely and understand the implications of any change.

Developers accustomed to Jira often have a visceral reaction to this approach: "that's impossible to maintain," "it will slow us down enormously," "it's useless bureaucracy." It is true that if approached naively, traceability can become a paralyzing burden. However, the experience of teams that have been working on medical software for years shows that it is possible to integrate these requirements into agile workflows, provided that the right tools are used and, above all, that the purpose of what is being done is understood. The key is not to treat traceability as an afterthought but as another dimension of development that is incorporated from the start and automated as much as possible.

A tool such as Jira can serve as an excellent foundation for traceability, provided that it is configured properly. The idea is that every requirement, every risk, every test case, and every development task is a traceable item with a unique identifier. The relationships between these items are established through links: "implements," "verifies," "mitigates," and "depends on". When

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

a developer writes code, they associate their commits not only with the development task but also with the requirements that gave rise to it and the risk it addresses. When a tester runs tests, they associate the results with the test case and, through it, with the corresponding requirements. In this way, the tool becomes a living repository of all the relationships that will later allow the product's history to be reconstructed.

The practical implementation of this bidirectional traceability requires a richer information structure than typical task management does. While in conventional Jira, a user story may be linked to several development and testing subtasks, in the medical model, that story must also connect to the derived system requirements, the test cases that verify it, the risks it mitigates, and the code that implements it. This network of relationships is not built automatically; it requires discipline in creating links and in the naming of identifiers. However, modern application lifecycle management (ALM) tools can automate much of this management, extracting relationships from code repositories, testing systems, and risk management tools.

The real challenge is not technical but cultural. For many developers, the idea of exhaustively documenting what they do goes against their training and professional identity. They have been educated in the culture of "self-explanatory code" and "minimum necessary documentation." They have seen too many projects where documentation becomes an end in itself, disconnected from the reality of the code. They rightly distrust any process that could turn into an empty bureaucratic exercise. The challenge for leaders of medical software projects is precisely to demonstrate that traceability does not have to be that way that it can be naturally integrated into the workflow and that, far from being a burden, it provides tangible benefits for product quality and the team's peace of mind.

2.2.2 HIDDEN BENEFITS: WHY TRACEABILITY MAKES US BETTER

One of these benefits is the ability to assess the impact of changes, which is critical in medical software. When a developer modifies a line of code, they need to know which requirements may be affected, which risks might be reintroduced, and which tests would need to be repeated. In a project without traceability, this assessment relies on people's memories, which are fragile and unreliable. In a project with good traceability, the system itself can help identify all the elements related to that piece of code, facilitating a systematic assessment and reducing the risk of overlooking something important. What appears to be a regulatory requirement thus becomes a tool for working better.

Change impact analysis is particularly valuable in the maintenance of medical software, which can span decades. When the need arises to fix a bug or adapt the system to a new regulatory requirement, the developer must be able to determine precisely which components will be affected. Without traceability, this determination requires exhaustive code reading and tacit knowledge that may have been lost because of staff turnover. With traceability, documented relationships allow for quickly identifying the scope of the change, estimating the required effort, and planning the corresponding regression tests. This capability reduces the incident response time and the likelihood that fixes introduce new bugs.

Another less obvious but equally important benefit is the protection of the team itself. In the event—always a possibility—that a serious incident occurs, authorities will investigate what happened and why. If the team can demonstrate that it followed established processes, that it properly documented decisions and that tests confirmed correct functionality, its position will be much stronger than if everything depends on memory and goodwill. Traceability is not only a requirement of regulators but also a shield for developers against liabilities that do not belong to them. In an environment where errors can end up in court, having a reliable record of what was done and why it was done is a form of self-protection.

Traceability also facilitates communication with external stakeholders who do not have direct access to the code. Regulators, auditors, and hospital clients need to understand what the system does without reading lines of code. The traceability chain, from high-level requirements to executed test cases, provides a comprehensible narrative of the product that can be reviewed and questioned. This transparency is not just about regulatory compliance; it is also trust-building marketing, a demonstration of professionalism that distinguishes serious providers from opportunists. In a market where differentiation based on technical features is becoming increasingly difficult, the ability to demonstrate robust processes becomes a competitive advantage.

For developers coming from the world of conventional software, one of the most difficult

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

concepts to grasp is that of “freezing” or “baseline.” In typical agile projects, the code is in a state of constant flux, and that is considered a virtue. In medical software, however, every version that is released must be perfectly identified and frozen so that it is possible to reconstruct exactly which code was running at any given moment. This does not mean that iterative development cannot continue, but it does mean that every iteration that reaches production must be treated as a unique and immutable artifact. Commits to that version after it has been released make no sense; any change must result in a new version, with its own identifier and its own documentation.

This requirement clashes with the common practice of many teams, which continuously deploy small improvements with little documentation. However, it has a powerful rationale: when a patient suffers harm because of a software error, it is crucial to be able to determine exactly which version was in use, to investigate whether the error is present in all versions or only in some, and to be able to notify the centers using affected versions. Without this ability to identify versions precisely, the response to an incident descends into chaos, and patient safety suffers. Version immutability is not a bureaucratic whim but an epidemiological safety requirement.

Version management in medical software requires formal configuration control procedures that go beyond standard code versioning. There must be a record of which versions of each component are included in each product release, what changes were made compared to the previous version, and what risk management decisions justified those changes. This record, known as a **Design History File** in regulatory terminology, is a living document that accompanies the product throughout its entire lifecycle. Maintaining it requires discipline, but it provides the basis for informed decisions regarding updates, market withdrawals, and safety communications.

2.2.3 ADAPTING AGILE PRACTICES: BRANCHES, COMMITS, AND DOCUMENTATION

Another common practice in agile development that must be adapted is that of code branches. In conventional projects, it is common to have multiple development branches, merge them frequently, and sometimes even rewrite the repository history to keep it clean. In medical software, these practices are problematic because they hinder traceability. If the history is rewritten, the ability to know exactly what code was present at any given time is lost. If branches are merged without rigorous control, it becomes difficult to associate each change with its requirements and tests. This does not mean that branches cannot be used but rather that they must be managed with greater discipline, maintaining a clear record of which changes were included in each version and why.

The branching strategy in medical software must balance the need for parallel development with the requirement for traceability. The feature branches are acceptable, but they should be shortlived, and their merging must be documented with references to the corresponding requirements and tests. Release branches allow versions to be stabilized while development continues, but they must be protected against unauthorized modifications. The practice of “force push,” which rewrites the repository history, must be prohibited on main branches to preserve the integrity of the historical record. These restrictions do not prevent agile work, but they require that agility be exercised within a framework of documentation discipline.

Documenting design decisions is another aspect that often clashes with developer culture. In many teams, decisions are made in conversations, discussed in meetings, and rarely recorded in writing. It is assumed that the code reflects the final decision—and that this is sufficient. In medical software, this is not acceptable because the code does not explain why one decision was made over another. When, months later, someone wonders why a feature was implemented in a certain way, the answer is not in the code but in the reasoning that led to that solution. Recording that reasoning, even succinctly, is essential for maintaining traceability and facilitating long-term maintenance.

Architecture decision records (ADRs) are agile practices that are particularly wellsuited to the needs of medical software. Every significant architectural decision is documented in a standard format that includes the context, the decision made, the alternatives considered and rejected, and the expected consequences. These documents, stored alongside the code in the repository, provide decision traceability without excessive documentation overhead. Maintaining them is part of the normal workflow, not an added task, and their structured format facilitates review by auditors and regulators. Adopting ADRs represents a practical compromise between agility and rigor.

Some argue that all this documentation is unnecessary when the team is small and when communication is fluid. “We talk to each other; we know what we did and why.” This argument,

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

understandable from the perspective of a tightly knit team, ignores several realities. The first is that teams change: people leave, new people arrive, and what was obvious to the founders becomes a mystery for newcomers. The second is that incidents can occur years later, when no one remembers the details anymore. Third, regulators do not accept “ask Maria” as evidence that something was done correctly. Traceability is not for the present; it is for the future, and the future is unpredictable.

2.2.4 TOOLS, MATURITY, AND THE CULTURE OF CARE

To facilitate the adoption of these practices, teams should use tools that naturally integrate traceability. There are requirements management systems that integrate with Jira, code repositories that allow commits to be linked to items in those tools, and testing environments that automatically record the relationships between test cases and results. Investing in this infrastructure is not an expense but an investment in future productivity because it reduces the manual effort required to maintain traceability and minimizes the risk of errors due to omission. Compared with a conventional team dedicated to its usual tools, a well-equipped team can maintain traceability with little more effort.

The application lifecycle management (ALM) tool ecosystem has evolved significantly to support the needs of medical software. Platforms such as Jama Connect, Polarion, or DOORS Next Generation enable the management of requirements with full traceability, integrating with test management systems and code repositories.(2) These tools automate the generation of traceability matrices, identify orphaned elements without links, and facilitate the generation of reports for audits. Their adoption requires investment and training, but the return in terms of reduced manual effort and improved quality is substantial. For startups with limited resources, there are also open-source solutions and configurations of generic tools that provide basic traceability functionality.

The developer’s attitude toward traceability is also a matter of professional maturity. Early in their careers, many programmers view documentation as a waste of time that prevents them from writing code, which is what they truly enjoy. With experience, they learn that code is only a small part of the software lifecycle and that decisions made during development have consequences that extend far beyond the moment they are written. They learn that well-documented code is easier to maintain, that a clear history of decisions saves hours of research and that traceability is not an end in itself but a means to build more reliable and safer software. This maturity is especially necessary in medical software, where the consequences of immaturity can be tragic.

For teams taking their first steps in this field, a useful strategy is to start with a basic level of traceability and gradually increase it. It is not about implementing a perfect system from day one that covers all aspects of the standard but rather about identifying the critical elements and securing them first. For example, ensuring that every requirement has associated tests and that every code change can be traced back to a requirement or a documented bug fix. From there, complexity can be added as the team becomes familiar with the processes and tools. What matters is not initial perfection but the direction of the journey: toward a culture where traceability is second nature, not an external imposition.

Gradual implementation allows the team to learn from its mistakes without catastrophic consequences. Traceability maturity milestones can be established: first, that all code is associated with an issue tracking ticket; then, that every ticket is linked to a requirement; subsequently, that verifiable test cases exist for each requirement; and finally, that risks are identified and mitigations are verified. Each milestone achieved provides immediate value to the project, regardless of future regulatory audits. This incremental approach is more sustainable than attempting to implement the entire traceability framework all at once, which often collapses under its own complexity.

There is also a psychological dimension to traceability that deserves attention. For many developers, the idea that every action is logged creates a sense of surveillance that can be uncomfortable. It is important for project leaders to clearly communicate that traceability is not a control mechanism but a tool for collective protection. It is not about monitoring each person’s work but about building a shared record that benefits everyone. When an error occurs, traceability allows us to understand what went wrong without needing to assign individual blame, focusing attention on improving the system rather than punishing individuals. Used properly, traceability fosters a culture of fairness and learning, not a culture of fear.

Traceability is not an end in itself but a means to a higher end: patient safety. When a developer is tempted to skip a step, to not document a decision, to forget to link a commit to its

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

requirements, they should ask themselves: What would happen if this small oversight contributed to an error that harmed someone? Could I forgive myself? Could I explain to a patient or their family why I didn't take five minutes to make things clear? These uncomfortable but necessary questions constitute the true foundation of traceability. It is not the standard; it is not the regulator; it is not the boss. It is the awareness that what we build can save lives or harm them and that we have a responsibility to do everything in our power to tip the scales toward the former.

The transition from a Jira culture to a traceability culture is not easy, but it is possible and necessary. It requires changing habits, incorporating new tools, and adopting a more disciplined mindset. However, it also offers rewards: the peace of mind that comes from knowing what you build is solid, the ability to sleep at night without fear of what might go wrong, and the pride of belonging to a profession that takes its responsibilities seriously. For developers who decide to take this step, medical software ceases to be a source of frustration and becomes a fascinating field where technology and ethics meet, where every line of code has a clear purpose, and where a job well done has a direct impact on people's lives. It's worth the effort.

2.3 WHAT IS A DESIGN HISTORY FILE (DHF) AND WHY IT'S YOUR ALLY?

Let's imagine we're building a house. Not just any house, but the house where we'll live the rest of our lives, where we'll raise our children, where we hope to grow old. During construction, we make decisions about the foundation, materials, electrical work, and plumbing. We choose a type of brick, a brand of window, and a model of a boiler. We hire architects, quantity surveyors, masons, and electricians. Months go by, the house is finished, and we move in. Twenty years later, a pipe bursts and causes a flood. The plumber that comes to fix it needs to know what type of pipe was installed, where the pipes run, and what pressure they can withstand. If there are no plans, if there are no records, if everything depends on the memory of those who are no longer here, the repair becomes a nightmare. If what fails is not a pipe but something that could kill someone, the nightmare multiplies. The Design History File is, exactly, the set of plans and records for that house but is applied to medical software.

Design history files, known by their acronym DHF, are central concepts in medical device regulation, and yet,they are among the concepts that generate the most confusion among developers new to this field. The formal definition, as it appears in the regulations, states that the DHF is the set of records describing the design history of a finished medical device. In simpler terms, it is the repository for all the information needed to understand how the final product was developed: from the initial ideas and requirements, through detailed drawings, to meeting minutes where decisions were made, test results, risk analyses, and any other document that was significant during the development process. It is not a single document but an organized collection of documents—a living archive that grows and evolves throughout the product's lifecycle.

For a developer accustomed to conventional software, the idea of maintaining such a comprehensive archive can be overwhelming. In their previous experience, documentation is often sparse, scattered, and limited to what is strictly necessary for the team to understand each other. Requirements live in Jira, the design in the developers' heads, decisions in hallway conversations, and tests in scripts that are run and forgotten. This way of working is efficient for certain types of projects, but it is completely inadequate for medical software because it does not allow the process to be reconstructed when something goes wrong, nor does it allow one to demonstrate to a regulator that everything possible was done to ensure safety. The DHF fills that gap, providing a framework for organizing information in a way that is useful not only during development but also throughout the product's entire lifecycle.

2.3.1 THE ANATOMY OF THE DHF: WHAT IT CONTAINS AND WHY

The first question that arises regarding the DHF is as follows: exactly what documents should it contain? The answer depends on the type of device and its classification, but every regulatory authority expects to find a common core.

First, the design requirements include what the product must do, with what precision, and under what conditions. Second, the design drawings include how the software is organized, what components make it up, and how they relate to one another. Third, we discuss traceability in the previous section: which requirements each component implements and which tests verify

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

each requirement. Fourth, the risk analysis addresses what can go wrong, with what probability, with what consequences, and what has been done to prevent or mitigate it. Fifth, the test results included what was tested, how, with what results, and what decisions were made on the basis of them. In addition, sixth, the change log: what modifications have been made throughout the product's lifecycle, why, and who authorized them.

This list may seem long, but it is important to understand that these are not separate, disconnected documents but rather a network of information where each piece is interconnected with the others. The requirements justify the design. The design determines the risks. These risks necessitate testing. The tests verify the requirements. The changes affect all of the above. The DHF is precisely the tool that keeps this network coherent, ensuring that when something is modified, all its implications are reviewed and that when someone asks why something was done, they can find the answer. It is not a dead archive but a living organism that reflects the current state of knowledge about the product.

The physical structure of the DHF can be organized in many ways, from hierarchical folders in a document management system to specialized databases. The key is to have a navigation logic that allows any information to be found quickly and for documents to be uniquely identified with codes indicating their nature, version, and date. A best practice is to maintain a master index, updated automatically if possible, that allows documents to be located by multiple criteria: by development phase, by document type, by functional area, and by date. This structure should not be rigid; it must allow for the incorporation of new document categories that emerge with experience.

2.3.2 DHF AND AGILE METHODOLOGIES: APPARENT CONTRADICTION, REAL COMPLEMENTARITY

For teams working with agile methodologies, the idea of maintaining a DHF may seem to contradict the philosophy of "responding to change rather than following a plan." However, this apparent contradiction is resolved when one understands that the DHF is not a rigid, unchangeable plan but rather a record of what is being done as it is done. Instead of producing a mountain of documents at the outset and then trying to make reality fit them, the process is documented as it progresses in an iterative and adaptive manner. Each sprint, each iteration, generates new information that is incorporated into the DHF. By the end of development, the file contains the complete history of the product—not as a bureaucratic imposition but as the natural byproduct of a job well done.

This way of working does, however, require a certain amount of discipline. You can't leave all the documentation for the end, because then it would be impossible to reconstruct the process accurately. You have to document as you go, even if only briefly, and ensure that the information is organized so it can be easily found when needed. Tools such as wikis, document repositories, or even a well-configured Jira itself can serve as a foundation for the DHF, provided they are used systematically. What matters is not the platform but the existence of a central location where all relevant information is accessible and interconnected.

Integrating the DHF into agile sprints can be achieved by defining a "definition of done" that includes documentation requirements. A user story is not complete merely when the code works but when the requirement is documented in the DHF, when test cases are recorded, and when design decisions are noted. This expansion of "done" requires the team to set aside time in each sprint for these activities, explicitly estimating them rather than treating them as invisible. Over time, documentation becomes a habit, and its marginal cost is significantly reduced.

2.3.3 UNEXPECTED BENEFITS: KNOWLEDGE, AUDITING, AND SURVIVAL

One of the least obvious but most valuable benefits of the DHF is that it facilitates the onboarding of new team members. In conventional software development, when someone joins a project, it takes them weeks or months to understand how the system actually works because much of the knowledge resides in people's heads. In a project with a well-maintained DHF, the new member can review the product's history, understand why certain decisions were made, and learn about the identified risks and the tests performed. This greatly accelerates their onboarding and reduces the team's reliance on specific individuals. The DHF thus becomes a knowledge management tool that protects the project from staff turnover and preserves accumulated

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

knowledge.

Another key benefit is audit readiness. At some point, a regulator will likely want to examine the product and the process that generated it. If the team has maintained a well-organized and complete DHF, the audit will be a relatively straightforward process: simply present the information, explain how it is organized, and answer any questions that arise. If, on the other hand, the documentation is scattered, incomplete, or simply nonexistent, the audit will become a traumatic experience, with the added risk that the regulator may demand additional studies or even halt the product's commercialization. The DHF is not a favor for the regulator but an investment in future peace of mind.

Some people believe that DHF is something that matters only to large companies with the resources to maintain quality and documentation teams. This idea is mistaken and dangerous. It is precisely startups and small businesses that need good DHF the most because they have the least margin for error and suffer the most from the consequences of poor knowledge management. A small team that loses its lead developer without having properly documented the product may be forced to shut down. A team that has maintained an organized DHF can survive that loss and continue development with new staff. Far from being a luxury for large corporations, the DHF is a survival tool for any organization that builds medical software.

The relationship between the DHF and the source code deserves separate consideration. The code is, without a doubt, the most precise expression of what the product actually does. However, the code does not explain why it was implemented in a certain way, what alternatives were considered, or what evidence supports its correctness. The DHF complements the code by providing the context that the code cannot contain. Together, code and DHF form a whole that allows for an in-depth understanding of the product: the code says what it does, the DHF says why it does it and how we know it does it right. Separately, both are incomplete. This complementarity is important to understand because developers, in their love for code, often tend to underestimate the importance of documentation. Code is necessary, but it is not sufficient.

2.3.4 DHF IN PRACTICE: STRUCTURE, LEVEL OF DETAIL, AND CHANGE MANAGEMENT

A practical question that often arises is how much detail is enough. The answer, as with so many things, is: it depends. It depends on the product's complexity, the associated risks, and the applicable regulatory requirements. However, there is a guiding principle: the documentation must be sufficient for a competent person, with no prior knowledge of the product, to understand how it was developed and why the decisions that were made were made. This does not mean that every thought must be documented, but it does mean that the fundamental milestones of the process must be recorded. A good exercise is to ask yourself: if I were to leave the project tomorrow and someone else had to continue it, what information would they need? That is, roughly speaking, the information that should be included.

For teams creating a DHF for the first time, it can be helpful to start with a simple structure and refine it as you gain experience. A possible initial structure could include folders or sections for requirements, architecture, risks, testing, changes, and key decisions. Within each section, documents are organized chronologically and uniquely identified. Over time, the team will discover what additional information is needed and how best to organize it. What matters is not initial perfection but consistency and a commitment to continuous improvement. Like the product itself, it is a living artifact that evolves with the project.

Digitizing the DHF is another aspect to consider. In the past, these files were typically physical and consisted of paper folders and documents. Today, fortunately, there are tools that allow them to be maintained electronically, with all the advantages that include quick search, links between documents, version control, and remote accessibility. However, digitization also poses challenges: You must guarantee the integrity of the documents, protect them from unauthorized modifications, and ensure their long-term availability. It is not enough to simply store the documents on a shared drive; you need a system that guarantees their authenticity and traceability. Quality management tools designed specifically for regulated environments can be a good investment, although for small teams, a combination of a code repository, a wiki, and best practices may suffice.

One aspect that often surprises developers is that the DHF does not end when the product is launched. In fact, in many ways, that is when its useful life truly begins. Every change, every fix, every improvement made to the product throughout its lifecycle must be recorded in it. Every incident that occurs, every new risk assessment, and every test update must be incorporated

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

into the file. It is the product's logbook, and as such, it must be kept up to date for as long as the product exists. This implies that the discipline of documentation is not something that is abandoned after launch but rather forms part of the daily work of maintaining and evolving the software.

For physicians involved in development projects, understanding what the DHF is and what it is used for is also important. When a clinician collaborates in defining requirements, identifying risks, or validating tests, they contribute to the content of the DHF. Their contributions are recorded and become part of the product's history. If any questions arise in the future about why a particular feature was included or excluded, their testimony will be present. The DHF also protects their work, ensuring that their knowledge is not lost and that their contribution is recognized. That is why physicians should take an interest in how this information is organized and insist on being involved in its creation.

The relationship between DHF and intellectual property is another aspect that deserves attention. The DHF contains detailed information on how the product was built, including aspects that may be subject to patents or trade secrets. That is why it is important to manage access to the DHF carefully, ensuring that only authorized individuals can view it and that there are records of who accesses what information. Moreover, the DHF can serve as valuable evidence in intellectual property disputes, as it documents the creation process and can help establish who did what and when. Like so many other things, the DHF is a double-edged sword: wellmanaged, it protects; poorly managed, it exposes.

To conclude, let us return to the house metaphor we started with. When we live in a house, we rarely consider the blueprints. Only when something breaks, when we want to renovate, when we need to understand how it's built—that's when we appreciate that they exist. The same is true for medical software: during development, the DHF may seem like a burden, a distraction from what matters—which is writing code that works. However, when the product has been on the market for years, when the original teams are no longer there, when unexpected problems arise, the DHF reveals itself for what it always was: the memory that allows the product to outlive its creators, to be maintained and improved, to continue fulfilling its role of helping patients without causing harm. Building a good DHF is, ultimately, an act of responsibility toward the future, a recognition that what we do today will have consequences long after we have moved on to other things. In addition, that responsibility, in the field of medicine, is the only thing that truly matters.

If the DHF is the memory that allows medical software to outlive its creators, how do we ensure that this memory is not only complete but also honest, preventing the pressure to document "correctly" from ultimately generating records that justify decisions made for other reasons after the fact, and ensuring that traceability does not become a well-written fiction?

2.4 ISO 13485: QUALITY, CHANGE CONTROL, AND AUDITS

> "Quality, in its deepest sense, is the ability to consistently produce results that meet expectations. In addition, that ability does not reside in the code, but in the system that produces it."

For many developers, quality is synonymous with clean, well-structured code that is backed by automated tests, meets requirements, and has no obvious errors. It is a product-centric view focused on the technical artifact delivered to the user. While this view is not wrong, it is incomplete, especially in regard to medical software. In this field, quality is not just a property of the final product but also of the process that generates it. A product may be technically flawless yet have been developed in such a way that there is no guarantee that the next product—or the next version—will be as well. In its deepest sense, quality is the ability to consistently produce results that meet expectations. In addition, that ability does not lie in the code but in the system that produces it. That is exactly what ISO 13485 is about.

ISO 13485 is an international standard that specifies the requirements for a quality management system applicable to organizations that design, manufacture, install, and service medical devices.(3) It is not a standard about what the product should be like but about what the organization that creates it should be like. It establishes which processes must exist, how they must be documented, how they must be controlled, and how they must be improved. In a sense, it is a standard about the organization, not about the product. This is precisely what makes it so valuable and so misunderstood by those encountering it for the first time. Because they will not find technical specifications in it regarding what their software should be like, but rather requirements regarding how they should document their decisions, how they should manage

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

changes, how they should train their staff, and how they should audit themselves.

For a developer accustomed to the culture of code, this approach can be disconcerting. What does staff training have to do with software quality? Why should I document my processes if what matters is that the product works? These questions reveal a profound misunderstanding of what it means to construct safe products in a sustainable way. A product that works today may stop working tomorrow if the person who developed it leaves and leaves no records. A team that produces excellent code under pressure may not be able to replicate it when conditions change. An organization that does not learn from its mistakes is doomed to repeat them. ISO 13485 is not a set of bureaucratic hurdles but a systematic attempt to address these problems to build organizations capable of producing quality reliably and predictably.

2.4.1 THE QUALITY MANAGEMENT SYSTEM: ORGANIZATION FOR TRUST

At the heart of ISO 13485 is the concept of a "quality management system." This is simply the set of processes, procedures, and responsibilities that an organization establishes to ensure that its products consistently meet applicable requirements. This is not something abstract or theoretical but rather very concrete matters: how customer requirements are gathered, how solutions are designed, how they are verified, how they are validated, how changes are managed, how suppliers are selected, how staff are trained, how nonconformities are investigated, and how corrective actions are implemented. All these processes, when welldefined and properly executed, constitute the quality management system. ISO 13485 provides a framework for defining them in a way that is acceptable to regulators.

The implementation of a quality management system begins with management commitment. Without the explicit and visible support of top management, any quality system will fail. This involves appointing a management representative for quality with sufficient authority to enforce processes, allocating adequate resources, establishing a clear quality policy, and periodically reviewing the system to ensure its continued suitability. This commitment is not merely rhetorical: auditors verify that management actively participates in system reviews, makes data-driven decisions, and does not allow commercial pressure to compromise quality requirements.

The quality management system documentation follows a pyramid structure. At the top is the quality manual, which describes the scope of the system and the interactions between processes. Below are the documented procedures, which detail how critical activities are performed. Next are the work instructions, which specify how to perform specific tasks. Finally, the records demonstrate that the processes have been carried out. This structure is not rigid: the standard allows for flexibility in the extent of the documentation depending on the size of the organization and the complexity of its processes. Importantly, consistency across the levels is present, and each document is kept up to date.

2.4.2 CHANGE CONTROL: DISCIPLINE WITHOUT RIGIDITY

One of the most relevant aspects of the standard for software development is change control. In any software project, changes are inevitable and desirable. Requirements evolve, errors are discovered, and new opportunities arise. However, in medical software, every change must be managed carefully because it can have unforeseen consequences for the safety or effectiveness of the product. ISO 13485 requires organizations to have a defined process for managing changes, including assessing their impact, verifying that they have been implemented correctly, and validating that they have not introduced new risks and documenting all of this. The goal is not to prevent changes but to ensure that they are made in a controlled and deliberate manner.

For a team accustomed to modifying code on the fly, deploying it multiple times a day, and experimenting with new features in production, this approach can be paralyzed. If applied rigidly, it can become so. However, the experience of teams that have been working in regulated environments for years shows that it is possible to maintain agility without sacrificing control. The key lies in having processes commensurate with the risk, in automating as much as possible, and in understanding that control is not an end in itself but a means to ensure that changes do not cause harm. A mature team in this environment does not view change control as an obstacle but as a safety net that allows it to innovate with confidence.

The typical change control process in an ISO 13485-certified organization includes several

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

well-defined steps. First, the change request describes what needs to be changed and why. Second, the impact assessment, which analyzes how the change affects requirements, design, risks, testing, and documentation. Third, review and approval by authorized personnel, who decide whether the change is appropriate and under what conditions. Fourth, the change was implemented following established development and testing processes. Fifth, verification that the change has been implemented correctly and has not introduced any unintended side effects is needed. Sixth, we update all affected documentation, including the DHF discussed in the previous section. In addition, seventh, closing out the change, with a record of everything that was done.

This process may seem excessive for minor changes, and in fact, it would be if applied in the same way to all changes. That is why the standard allows organizations to define different levels of control on the basis of risk. A change affecting a safety-critical functionality will require a thorough process; a purely cosmetic change to the user interface may have a lighter process. Importantly, the organization clearly defines the criteria for classifying changes and consistently applies the corresponding processes. Proportionality is the key to not stifling innovation with bureaucracy.

Implementing control levels requires a change classification matrix that evaluates multiple dimensions: the affected component (critical or noncritical), the type of change (functional, bug fix, or enhancement), the scope (local or global), and the deployment environment (production, testing, or development). By combining these variables, the organization can define differentiated approval workflows: trivial changes can be approved by the team itself, major changes require quality review, and critical changes require management authorization. This flexibility, formalized in written procedures, allows for maintaining agility without sacrificing security.

2.4.3 NONCONFORMITIES: LEARNING FROM MISTAKES

Another central aspect of ISO 13485 is the management of nonconformities. A nonconformity is any deviation from what is specified: a requirement that is not met, a process that is not followed, and a product that does not function as it should. In organizations with mature quality systems, nonconformities are neither hidden nor ignored but are systematically investigated to understand their causes and prevent them from recurring. This is among the most profound differences between traditional software culture and quality culture. In the former, errors are corrected and forgotten; in the latter, errors are investigated and turned into opportunities for improvement.

When nonconformity occurs, the typical process includes accurately describing the problem, investigating its immediate and root causes, defining corrective actions to eliminate those causes, implementing the actions, verifying that they have been effective, and documenting the entire process. This systematic approach to problem-solving is extraordinarily powerful because it prevents the same errors from recurring repeatedly. Instead of constantly putting out fires, an organization is built that learns from its experiences and becomes more robust over time. The investment in investigating the causes is worthwhile because every problem solved at its root is a problem that will not happen again.

The distinction between corrective actions (fixing the immediate problem) and corrective measures (eliminating the cause to prevent recurrence) is fundamental but often confusing. When a software error is detected, the corrective action is to fix the code. The corrective action could be to review the code review process to detect similar patterns, improve training in that technical area, or modify the static analysis tools. Both are necessary: the remedial action to resolve the current incident and the corrective action to prevent future ones. Immature organizations focus exclusively on remedial actions, leaving the underlying causes unaddressed.

Root cause analysis often uses methodologies such as cause tree analysis or the “5 Whys” technique. These tools structure the reflection to go beyond superficial explanations (“the error occurred because the developer made a mistake”) toward systemic causes (“the error occurred because the review process failed to detect the inconsistency because the workload prevented thorough reviews because planning did not account for contingencies”). Uncovering these causes can be uncomfortable because they often involve deficiencies in processes or resources, but genuine improvement can be achieved only by addressing them.

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

2.4.4 COMPETENCE, SUPPLIERS, AND THE QUALITY CHAIN

Staff training is another ISO 13485 requirement that often surprises developers. The standard requires that individuals performing work that affects product quality be competent and that the organization maintain records of that competence. This involves defining the profiles for each position, assessing whether the individuals filling them meet the requirements, providing training when necessary, and documenting all of this. For many technicians, accustomed to having their practical experience valued over credentials, this requirement may come as a shock. However, it makes sense: if it has not been verified that a person knows what they need to know, their work cannot be trusted to meet the required quality standards. Training is not a formality but a guarantee.

Competency management goes beyond initial training. The standard requires that staff be periodically assessed to ensure that they maintain their competency, especially when there are changes in processes or regulatory requirements. This involves maintaining competency matrices that link each position to the required knowledge and skills, identifying gaps through assessments, and closing them through planned training. Training records must include content, date, instructor, and evidence of effectiveness assessment. This systematic approach ensures that organizational capability does not depend on specific individuals.

The relationship with suppliers is another important aspect of the standard. In medical software development, it is common to use third-party components: libraries, frameworks, cloud services, and development tools. All these suppliers must be evaluated to ensure that their products or services meet the necessary requirements. You cannot simply download a library from the internet and trust that it is safe; you must verify its origin, quality, support, and maintenance. If the supplier is critical to the product's safety, it may be necessary to audit their processes or require specific certifications. This requirement may seem excessive, but it reflects a reality: a failure in a third-party component can have the same consequences as a failure in the product's own code, and the responsibility still lies with the manufacturer of the final product.

Supplier evaluations must be documented and reviewed periodically. For software components, the criteria include project maturity, update frequency, the user community, vulnerability history, licensing, and support capabilities. Critical suppliers may require specific quality agreements, audits, or ISO 9001 or ISO 13485 certifications. Supply chain risk management is particularly relevant for cloud services, where reliance on external infrastructure introduces vulnerabilities that must be assessed and mitigated.

2.4.5 AUDITS: THE MOMENT OF TRUTH

This brings us to the topic that instills the most fear among developers new to this field: audits. The word "audit" conjures images of stern inspectors, who scrutinize every document, search for flaws, and issue reports that can derail entire projects. In addition, certainly, an audit can be a stressful experience if you're not prepared. However, it can also be an opportunity for external individuals, with experience and perspective, to help identify areas for improvement that the team, immersed in day-to-day operations, does not see. The attitude with which an audit is approached largely determines what is gained from it.

In the context of ISO 13485, there are two main types of audits: internal and external. Internal audits are conducted by the organization's own staff, who are trained for this purpose, and serve to verify that the quality system is functioning as expected and to identify opportunities for improvement before external audits take place. They are a tool for self-assessment, not for punishment. External audits are conducted by independent bodies, such as certification bodies, and serve to verify that the system complies with the standard's requirements and, where applicable, to obtain the corresponding certification. Both are necessary and complementary.

Preparing for an audit is not something done in the weeks leading up to it but rather the natural result of keeping the quality system up to date throughout the year. If processes are followed correctly, if documentation is up to date, if nonconformities are properly investigated, and if training is current, then the audit will be a relatively straightforward process. The problem arises when the system is neglected and then attempts are made to "fix" it for the audit. Auditors are trained to detect these situations, and when they find them, the mistrust they generate can lead to a much more thorough and demanding audit. The best preparation for an audit is, quite simply, to do things right every day.

During an audit, the auditor will review the system's documentation, interview key personnel,

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

observe processes in operation, and select project samples to track them in detail. They will look for evidence that the system is actually being implemented, not just on paper. They will ask why certain decisions were made, how specific changes were managed, and what was learned from this or that nonconformity. This is not a hostile interrogation but a systematic investigation to verify that the system fulfills its purpose. Those being audited should respond honestly, show what they have, and if there is something they do not know, acknowledge it without trying to make up answers. Transparency is always the best policy.

When an audit identifies nonconformities, these are documented, and a corrective action plan is needed. It is not the end of the world, far from it. It is simply the acknowledgment that there are aspects of the system that can be improved. The important thing is to respond seriously, investigate the causes, implement the corrections, and demonstrate that the lesson has been learned. Auditors do not expect perfect systems because they know that they do not exist. They expect systems that take continuous improvement seriously, recognize their limitations, and work to overcome them. An organization that demonstrates this attitude usually receives favorable reports, even if it has nonconformities to correct.

For developers facing an audit for the first time, some practical tips can help. The first is not to take the questions personally; the auditor is not judging the person but the system. The second is to answer only what is asked, without adding unsolicited information that might open new lines of inquiry. The third is not afraid to say "I don't know" if you truly don't know and to offer to look up the information or refer the auditor to the appropriate person. The fourth is to present records naturally, without excuses or justifications. The fifth, perhaps the most important, is to remember that the purpose of the audit is not to catch anyone out but to verify that the system works and to help improve it.

ISO 13485 certification is not mandatory for all medical software manufacturers, but it is highly recommended and has become a de facto requirement in many markets. Obtaining it involves passing an external audit conducted by an accredited body, which verifies that the quality system meets all the standard's requirements. Once obtained, the certification must be maintained through periodic follow-up and renewal audits. It is a long-term commitment, not a one-time achievement. However, organizations that embrace it discover that the benefits go far beyond regulatory compliance: more organized teams, higher-quality products, fewer errors, greater customer satisfaction, and a culture of continuous improvement that permeates the entire organization.

ISO 13485 is not, therefore, a set of obstacles that regulators impose on developers to make their lives difficult. It is a framework for building organizations capable of producing safe products reliably and sustainably. It is a tool for learning from mistakes, for managing changes carefully, for ensuring that people are trained, for monitoring suppliers, and for conducting honest self-audits. It is, in short, an ally in the task of building software that does not harm those who use it. In a field where the harm can be irreparable, having allies like this is not a luxury but a necessity.

If ISO 13485 teaches us that quality lies in the system that produces the software and not just in the software itself, how do we translate that organizational philosophy into the practice of agile development, where small, autonomous teams value adaptation over planning, without the required discipline turning into rigidity that stifles innovation?

2.5 AGILE IN REGULATED ENVIRONMENTS: HYBRID MODELS THAT WORK

"Agile and regulation are not opposites, but complementary."

There is a profound, almost existential tension between an agile culture and a regulatory culture. Agile manifestos proclaim the value of people over processes, of working software over exhaustive documentation, and of responding to change over the course of a plan. Regulated environments, on the other hand, demand defined processes, rigorous documentation, and strict control over changes. Confronting these two philosophies can create a kind of professional schizophrenia in teams: On the one hand, they know that agile methodologies have proven effective for developing quality software; on the other hand, they sense that the way they apply them in other contexts will not be acceptable when the software could impact people's health. The result is often paralysis, or worse yet, pretense: agile meetings are held to appear modern, but deep down, work continues in the traditional way because "regulation doesn't allow for anything else."

This tension is by no means inevitable. Today, there are hybrid models that integrate the best of both worlds, combining the agility and responsiveness of iterative methodologies with the

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

discipline and control required by regulated environments. These models are not half-hearted compromises but rather well-thought-out syntheses that respect both the agile philosophy and regulatory requirements. To understand how they work, it is helpful to first debunk some myths that obscure the possibilities for integration. The first is that agile means "no documentation." The Agile Manifesto states that working software is valued at more than exhaustive documentation, which is very different from saying that documentation has no value. In regulated environments, documentation has unquestionable value, but that does not prevent it from being agile—that is, produced iteratively, just in time, and with the necessary level of detail, not the maximum possible.

The second is that regulated environments require a waterfall approach. For decades, the medical software industry has used sequential models, with clearly separated phases of requirements, design, implementation, and testing. However, this was more due to inertia than to regulatory requirements. Standards such as IEC 62304, which we explore in the next chapter, do not prescribe a specific methodology but rather establish outcomes that must be achieved, regardless of the methodology used. As long as it can be demonstrated that the requirements for traceability, verification, validation, and change control have been met, the method used to achieve this is open. This opens the door to iterative and incremental approaches, provided that they are adapted to generate the necessary evidence.

The third myth is that agility and regulation are incompatible because regulation requires detailed planning, whereas agility prioritizes the response to change. However, this view overlooks the fact that planning in regulated environments does not have to be a straitjacket but rather a framework that allows for managing change in a controlled manner. Agile teams also plan; they just do so in short cycles and continuously adjust. The difference lies in the level of formality and the need to document not only the outcome of planning but also the decisions and changes. An agile team working in a regulated environment can plan its sprints, adjust the backlog, and respond to new requirements, provided it properly documents the process and maintains traceability between what was planned, implemented, and verified.

2.5.2 PROVEN HYBRID MODELS: REGULATED SCRUM AND ADAPTED RUP

Several hybrid models have proven effective in practice. One of the best known combines Scrum with quality management practices typical of regulated environments. In this model, work is organized into sprints of fixed duration, with daily coordination meetings, a review at the end of each cycle, and a retrospective for continuous improvement. However, unlike conventional Scrum, each sprint generates not only working code but also the documentation associated with that increment: detailed requirements, test cases, risk analyses, and traceability records. The result of the backlog is not just a list of features but also a managed repository of requirements with all associated information. Every increment considered potentially releasable must meet not only functional acceptance criteria but also regulatory completeness criteria.

Another widely used hybrid model is one based on the adapted rational unified process, which organizes development into phases (initiation, elaboration, construction, and transition) but allows for iterations within each phase. This approach provides the milestone structure that regulators typically appreciate while maintaining iterative flexibility. It is especially useful in large projects or those with distributed teams, where coordination requires a certain degree of formality. The key is that each iteration, within each phase, produces a potentially releasable increment, with all the necessary evidence to demonstrate its quality and safety. In this way, at the end of each phase, there is a partial but fully documented and verified product.

A common element in all successful hybrid models is the early and continuous integration of quality and assurance activities. In traditional models, quality used to be a phase at the end of development, when it was already too late to correct deep-seated problems. In conventional agile models, quality is integrated through practices such as test-driven development, but systematic documentation is often neglected. In hybrid models for regulated environments, quality activities are distributed throughout the entire development cycle, but the necessary records are also generated at the time each activity is performed. This requires teams to incorporate into their definition of "done" not only functional criteria but also documentation and compliance criteria.

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

2.5.3 REQUIREMENTS AND RISK MANAGEMENT IN ITERATIVE CONTEXTS

Requirement management is among the areas in which agile-regulatory integration requires the most attention. In a conventional agile project, requirements are kept in a backlog, often with little detail, and are refined just before implementation. In a regulated environment, requirements must be traceable, verifiable, and linked to risk analyses. This does not preclude progressive refinement, but it requires that each level of detail be documented and that traceability be maintained between high-level requirements and implementation details. Tools such as Jira, when properly configured and equipped with the right plugins, can support this workflow if used in a disciplined manner. The key is that every requirement, at every level of detail, has a unique identifier, is linked to its parents and children, and can be traced back to the code and tests.

Risk management is another area where integration requires a specific approach. In traditional models, risk analysis was typically performed at the start of the project and rarely updated. In an agile approach, risk analysis must be continuous: every new requirement, every design change, and every new feature must be evaluated from a risk perspective. This does not mean conducting a formal analysis every time a line of code is touched but rather keeping the risk register active and updating it as the product evolves. In successful hybrid models, risk management is just another sprint activity, reviewed in planning meetings and updated on the basis of test results and user feedback.

Verification and validation must also be adapted. In an agile project, testing is automated as much as possible and runs continuously. In a regulated environment, in addition to automated testing, formal testing is needed, with documented test cases, recorded results, and retained evidence. This does not preclude automation; in contrast, automation facilitates the systematic generation of records. However, it requires that test cases be linked to requirements, that results be retained in a manner that allows for auditing, and that traceability be maintained between requirements, risks, test cases, and results. Teams that master these hybrid models have continuous integration pipelines that not only run tests but also automatically generate the verification documentation associated with each construct.

2.5.4 VERSIONS, RELEASES, AND THE CADENCE OF REGULATED DEPLOYMENT

One of the most interesting challenges of hybrid models is the management of versions and releases. In a conventional agile project, continuous releases are possible, with dozens of deployments per day. In a regulated environment, every release that goes into production must be carefully controlled, documented, and, in many cases, reported to regulators. This does not mean that frequent releases cannot be made, but it does mean that each release must meet the requirements for completeness and evidence. Hybrid models typically establish different release levels: internal releases for testing, validation releases for controlled clinical environments, and production releases for general use. Each level has its own entry and exit criteria, as well as its own documentation requirements. The key is that the flow between levels is seamless and that the evidence generated at lower levels serves to meet the requirements of higher levels.

The branching strategy in code repositories must also be adapted. While in pure agile development, feature branches can be ephemeral and frequent, in regulated environments, each branch contributing to a release must be documented, linked to requirements, and subjected to formal review before merging. Some teams use release branches that are strictly protected, where only changes approved through the formal change control process are allowed. Other practices, such as “force push,” which rewrite the repository history, must be prohibited in these branches to preserve traceability. The discipline in version control is not antinchnical; it is technology applied consciously to ensure reproducibility.

Documentation is, as we have seen, one of the areas where the most friction arises between agile and regulatory cultures. However, it is also one of the areas where the most mutual lessons can be learned. Agile culture reminds us that documentation must be useful, not an end in itself. Regulatory culture reminds us that without documentation, there is no evidence, and without evidence, there is no safety. The key is to document what is necessary, in the most efficient way possible, and at the right time. This means, for example, using wikis instead of static documents, keeping documentation close to the code, automating the generation of reports from data in development tools, and periodically reviewing the documentation to ensure that it remains relevant and up-to-date.

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

2.5.5 CULTURE, TRAINING, AND RELATIONSHIPS WITH REGULATORS

Team training is another critical factor for the success of hybrid models. We cannot expect developers trained in the purest agile culture to simply adapt to regulatory demands, nor can we expect professionals accustomed to regulated environments to accept agile practices without reservation. A conscious effort at training and support is needed to help some understand why documentation is necessary, and others see the benefits of iteration and continuous feedback. The teams that perform best in these environments are those that have developed a shared culture where both perspectives are respected and integrated. This takes time and requires leadership, but it is the only way to build truly effective teams in this demanding field.

The relationship with regulators is also evolving in hybrid models. Regulatory agencies are increasingly open to agile approaches, provided that control and traceability are maintained. In some countries, there are pilot programs for the agile certification of medical software, and some regulators have published specific guidelines on how to apply iterative methods in regulated environments. The key lies in transparency and communication: if a team plans to use a hybrid approach, it must clearly explain this to the regulator, show how it will maintain traceability and control, and demonstrate with facts that the approach works. Regulators are not enemies of innovation, but they need to trust that the processes ensure safety.

For teams just starting out on this path, there are some practical recommendations that can ease the transition. The first is to start with a pilot—a small but meaningful project—where you can test the hybrid model and learn from experience. The second goal is to invest in tools that integrate development management with quality management so that traceability is as automated as possible. The third goal is to establish from the outset the definition of "done," including regulatory criteria, not just functional ones. The fourth goal is to conduct periodic internal audits to verify that the model is being applied correctly and to identify areas for improvement. The fifth, perhaps the most important, is to cultivate a culture of continuous improvement, where every problem is an opportunity to learn and the system evolves with experience.

The tension between agile and regulated does not disappear entirely in hybrid models but rather transforms into a creative tension that drives continuous improvement. Teams learn to balance the need to respond quickly to changes with the requirement to maintain control and traceability. They learn to document just enough—no more, no less. They learn to plan with enough detail to meet requirements but with enough flexibility to adapt to the unexpected. In short, they learn to navigate the complexity of building secure software without sacrificing the agility that enables innovation. In addition, that ability to navigate complexity is, perhaps, the most valuable skill that teams working at the intersection of technology and medicine can develop.

Ultimately, what hybrid models teach us is that agility and regulation are not opposites but rather complementary. Agility provides the ability to adapt, to learn quickly, and to deliver value continuously. Regulation provides discipline, a systematic approach, and the assurance that what is delivered is safe. Together, they enable the development of medical software that is not only innovative but also reliable and that not only meets users' needs but also does so without putting them at risk. In a world where technology is advancing ever faster and where medicine increasingly relies on it, this synthesis is not an option but a necessity. In addition, the teams that master it will be the ones that truly make a difference in people's health.

Now that we have established the general quality framework with ISO 13485 and have seen how to adapt agility to regulated environments, what specific standard tells us how the medical software lifecycle should be organized to meet safety and efficacy requirements?

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

REFERENCES

Vargesson N. Thalidomide--Birth Defects Res [Internet]. June 1, 2015 [cited May 13, 2026];105(2):140. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC4737249/
Master Medical Device Traceability: A Guide for Engineers - Jama Software [Internet]. [cited May 13, 2026]. Available from: https://www.jamasoftware.com/blog/2025/10/07/how-to-master-traceability-in-medical-device-development/
ISO 13485:2016 - Medical devices — Quality management systems — Requirements for regulatory purposes [Internet]. [Cited May 13, 2026]. Available from: https://www.iso.org/standard/59752.html

https://doi.org/10.62486/978-9915-9876-8-2.Ch02

Chapter 3 / Capítulo 3

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch03
Pages: 50-62

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 3. IEC 62304: Medical software lifecycle

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Classifying medical software by risk level is not a bureaucratic exercise, but an act of honesty: recognizing that not all failures have the same consequences and that, therefore, not all developments deserve the same level of scrutiny."

Imagine we are in a planning meeting for a new healthcare software project. Someone asks, "What level of regulatory requirements should we apply?" Eyes dart around; no one is sure of the answer. Some believe that since it is just software, it should be a light-touch approach. Others, more cautious, suggest applying everything the standards require, just in case. The discussion drags on aimlessly because a fundamental element is missing: the actual risk of the product has not been analyzed. The most basic question has not been answered: what happens if this fails? Depending on the answer, regulatory requirements can vary enormously, from a relatively simple process to a development process with exhaustive controls at every stage. This gradation is not an invention of each team but is clearly defined in an international standard that forms the backbone of medical software development: IEC 62304.

IEC 62304 is likely the most important standard that any medical software developer must know.(1) Its full title is "Medical device software—Software life cycle processes," and it establishes a framework for all development activities, from initial planning through maintenance once the product is on the market. What makes this standard particularly valuable is that it is not an abstract set of good intentions but a concrete specification of tasks, documentation, and responsibilities that must be fulfilled depending on the software's risk level. It is, in a sense, a practical guide that says: if your software is low-risk, do this; if it is medium-risk, do that; if it is high-risk, go this far. This proportionality is its greatest strength and makes it applicable to a wide range of products, from the simplest to the most critical.

In this chapter, we explore in detail the fundamental aspects of IEC 62304, starting with the concept that underpins everything: software safety classification. We will see what each class means, how to determine which one applies to each product, and what practical implications that classification has for the development process. It is not a matter of memorizing a table but of understanding the logic underlying the classification because that logic is the same one that should guide any design decision when patient safety is at stake. Understanding IEC 62304 means understanding how to build medical software in a systematic and responsible manner, and it is also the first step toward navigating with confidence the regulatory maze that so many developers fear.

3.1 SAFETY CLASSIFICATION: CLASSES A, B, AND C

To understand the safety classification in IEC 62304, it is helpful to start with a simple yet uncomfortable question: what happens if this software fails? Not if it fails in a specific way, not if it fails under specific circumstances, but in general, if the software stops doing what it is supposed to do, or does something it is not supposed to do, what are the consequences for the patient, for the user, for the environment? The answer to this question is the fundamental factor that determines the software's safety class. Because a failure in an app that helps remind users to take their medication—which might lead to a missed dose—is not the same as a failure in a drug infusion system, which could cause a lethal overdose. The standard recognizes this difference and establishes three levels of increasing requirements on the basis of the potential severity of the consequences.

Class A is the lowest and corresponds to software whose failure cannot cause injury or harm to health. This may sound strange: if it cannot cause harm, why are we talking about medical software? The answer is that Class A software may be part of a medical device, but its contribution to safety is indirect or noncritical. For example, an application that keeps track of patient appointments or a program that manages a hospital's inventory can be considered Class A if its failure does not directly affect people's health. Even within a more complex device, there may be software components that are Class A because their malfunction has no clinical consequences. The key is that, for this class, patient safety does not depend on the software functioning correctly.

3.1.1 THE RISK PYRAMID: FROM HARMLESSNESS TO MORTALITY

Class B is the intermediate level and corresponds to software whose failure may cause injury but not serious injury or death. The definition of "injury" in this context is broad and includes both physical and psychological harm, provided that it is reversible or minor in nature. For example,

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

software that calculates nonlethal drug doses or a vital sign monitoring system that may trigger false alarms could be classified as Class B if an error causes discomfort or unnecessary treatments but does not endanger the patient's life. The distinction between Class B and Class C is not always clear-cut and requires a careful analysis of the potential consequences of failures, taking into account not only the direct effect but also the chains of events that may be triggered.

Class C is the highest category and applies to software whose failure can cause death or irreversible serious injury. This category includes the most critical systems: software that controls life-support equipment, algorithms that guide cancer treatments, high-risk drug infusion systems, and diagnostic applications that can lead to erroneous therapeutic decisions. For these products, the standards are the highest because people's lives are at stake. Every line of code, every design decision, and every test must be executed with the utmost rigor, and the documentation must be exhaustive to allow for audits and complete traceability. It is not that Class C developers are more careful but rather that the process within which they work is designed to minimize the probability of errors to the utmost extent.

The assignment of the safety class is not something the manufacturer determines arbitrarily but must be based on a risk analysis conducted in accordance with ISO 14971, which we explore in the next chapter. This analysis identifies the potential hazards associated with the use of the software, estimates the probability of their occurrence and the severity of their consequences, and determines whether existing control measures are sufficient. The safety class is derived from the severity of the consequences of failures, considering the worst reasonably foreseeable scenario. It is not a matter of being optimistic but of being realistic: if there is a possibility that a failure could cause death, the software is Class C, even if that possibility is remote. The standard prefers to err on the side of caution rather than underestimate the risks.

3.1.2 MODULAR CLASSIFICATION AND THE CONCEPT OF INDEPENDENCE

An important aspect is that the classification applies to the software as a whole, but it can also be applied to individual components within a larger system. A medical device may contain several software elements with different safety classes. For example, a monitoring system in an intensive care unit may have a vital sign display module that is Class B (because poor display can lead to noncritical erroneous decisions) and an alarm control module that is Class C (because an alarm that does not sound when it should have lethal consequences). In these cases, the development must meet the requirements of the highest class for the system as a whole, unless a genuine independence between the modules can be demonstrated that justifies differentiated treatment.

Independence between modules is a technical concept with significant regulatory implications. Two software modules are considered independent if a failure in one cannot cause a failure in the other and if the operation of each can be verified separately. In practice, this means that they must run in separate memory spaces, communicate through well-defined interfaces, and not share critical resources. If this independence can be demonstrated, then it is possible to develop a Class C module with all necessary controls and another Class B module with fewer requirements, even within the same device. However, the burden of proof lies with the manufacturer, who must demonstrate that the independence is genuine and that there are no hidden pathways through which a failure in the lower-risk module could affect the higher-risk module.

Demonstrating independence requires rigorous architectural analysis and specific testing. How the modules are isolated in terms of memory, processor time, and communications must be documented. Fault injection tests must be performed to verify that an error in the lower-class module does not propagate to the higher-class module. This independence must be maintained throughout the entire lifecycle, with reviews conducted whenever changes occur that could affect the architecture. The temptation to "optimize" by sharing resources between modules must be resisted when it compromises the independence that justifies the differentiated classification.

3.1.3 PRACTICAL IMPLICATIONS: FROM ANALYSIS TO DEVELOPMENT

Determining the safety class has enormous practical implications for the development process. IEC 62304 establishes a set of mandatory activities for each class such that the higher the risk is, the more activities must be performed and the greater the level of detail needed.

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

For example, for Class A software, the documentation requirements are minimal; for Class B, traceability between requirements and tests is needed; for Class C, more comprehensive requirements for detailed architectural design, code review, and test coverage analysis are added. This gradation ensures that low-risk products are not bogged down by unnecessary bureaucracy, while guaranteeing that high-risk products receive the attention they deserve.

For development teams, understanding the security class from the outset is critical because it shapes many subsequent decisions. It is not the same to design the architecture knowing that you will have to demonstrate independence between modules as it is to do so without that concern. It is not the same to plan testing, knowing that code coverage will be needed, as it is to do so with more relaxed requirements. It is not the same to organize documentation knowing it will be thoroughly audited, as it is to do so with the confidence that it will be only briefly reviewed. That is why the security classification is not something to be left until the end but must be determined in the early stages of the project and used as a guide for all subsequent decisions.

A common mistake among novice developers is to believe that the safety class is something negotiated with the regulator to obtain more lenient requirements. This view is profoundly mistaken and potentially dangerous. The safety class is not a variable in the decision-making process but a consequence of the risk analysis. If the analysis shows that a failure could cause death, the software is Class C, regardless of what the manufacturer prefers or negotiates with the regulator. Attempting to classify software that should be Class C as Class B is, at best, a lack of understanding of the standard and, at worst, an unrelated irresponsibility that can have tragic consequences. Honesty in classification is the first ethical requirement of medical software development.

3.1.4 CHANGES, UPDATES, AND CONTINUOUS MONITORING

The relationship between safety classification and software updates deserves special mention. When existing software is modified, its classification must be reevaluated because changes can introduce new risks or alter existing ones. An update that adds a critical functionality may increase the safety class; one that removes dangerous functionalities may lower it. This reassessment must be documented, and if the class changes, a new certification or, at a minimum, a review by the regulator may be needed. That is why it is important to keep the risk analysis up to date throughout the product's entire lifecycle, not just as an initial formality but as an ongoing practice.

For physicians involved in development projects, understanding the safety classification is also relevant because their clinical knowledge is essential for correctly assessing the consequences of potential failures. An engineer may not know what it means for a patient to experience a certain complication or how serious it actually is. The physician provides that perspective, helping to ensure that the risk analysis is realistic and comprehensive. That is why it is important for clinicians to be involved in the early stages of the project, when the safety class is being determined, and for their voice to be heard and respected. Classification is not a technical decision but a clinical-technical decision, and it requires collaboration between both worlds.

One aspect that often raises questions is the classification of software that has no direct medical use but can indirectly affect safety. For example, an appointment management system that, if it fails, could cause a patient to miss a critical appointment. Is that Class B or Class C? The answer depends on the severity of the consequences of that missed appointment. If it is for routine follow-up of a stable chronic condition, it is likely Class B. If it is for administering a treatment with a narrow therapeutic window, it could be Class C. There are no automatic answers; each case must be analyzed in its context, considering the worst reasonably foreseeable scenario. This ambiguity is uncomfortable for those seeking clear rules, but it is inherent in a field where clinical reality is complex and variable.

Experience shows that many teams tend to underclassify their software, whether due to ignorance, optimism, or commercial pressure to reduce costs. This tendency is dangerous because it leads to development processes that are insufficient for the actual risk, increasing the likelihood of incidents occurring. That is why regulators pay special attention to the justification for the classification and require that it be welldocumented and supported by rigorous analysis. A manufacturer that cannot explain why its software is Class B and not Class C will arouse suspicion and will likely be subject to more detailed scrutiny. Transparency and rigor in classification are also ways to protect against future claims.

It is worth remembering that security classification is not an end in itself but a means to a

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

higher end: aligning the development effort with the product's actual risk. It is not about creating obstacles but about focusing resources where they are most needed. Class C software deserves all the attention, all the controls, and all the documentation because human lives depend on it. Class A medical software can be developed using lighter processes because its failures will not cause harm. This distinction, far from being discriminatory, is deeply ethical: it recognizes that resources are limited and must be invested where they can best protect patients. Understanding and applying it correctly is the first step toward building responsible medical software and toward avoiding wasting effort on unnecessary bureaucracy while ignoring real risks.

If the safety classification determines how much control we need, but the risk analysis underpinning it depends on a subjective assessment of clinical consequences, how do we ensure that this assessment is consistent across different teams and regulators, avoiding both dangerous underclassification and paralyzing overclassification?

3.2 PRACTICAL IMPLICATIONS OF EACH CLASS

> "The safety class is not a punishment, but a protection: for the patient, for the physician, and for the developer themselves."

Determining whether software is Class A, B, or C is not an academic exercise that ends with a report that is filed away and forgotten. In contrast, it is a decision that permeates every aspect of development, from how requirements are documented to the depth of testing that must be performed, including system architecture, change management, and even team composition. Knowing the safety class is similar to having a map that shows which route to take: the paths for Class A are simple trails with few demands; those for Class B require slightly more preparation and equipment; and those for Class C are multilane highways with constant signage and exhaustive checks. Ignoring this map and treating all developments the same inevitably leads to one of two problems: either excessively burdensome processes are applied to a low-risk product, making it economically unviable, or insufficient processes are applied to a high-risk product, putting patients at risk.

3.2.1 PLANNING AND DOCUMENTATION: THE ESCALATION OF REQUIREMENTS

To understand the practical implications of each class, we must examine the main activities of the development cycle and see how they change depending on the risk level. Let us start with **planning**. For any class, IEC 62304 requires a software development plan, but the level of detail required varies. For Class A, it can be a simple document that broadly outlines the phases, deliverables, and responsible parties. For Class B, the plan must be more detailed, including timelines, milestones, and a description of the verification and validation processes. For Class C, the plan must be comprehensive, with unique identifiers for each activity, a precise definition of the input and output criteria for each phase, and a detailed description of how traceability will be ensured among all development elements. Additionally, for Class C, the plan must be formally reviewed and approved by management before work begins.

Requirement management is another area where the differences between classes are notable. For Class A, a list of functional requirements is sufficient, without the need for extensive formality. For Class B, requirements must be documented in a structured manner, with unique identifiers, and traceability must be established between them and the tests that verify them. For Class C, in addition to the above, requirements must be analyzed from a risk perspective, identifying those that are critical to safety, and the justification for each requirement must be documented (), explaining why it is necessary and what alternatives were considered. Additionally, for Class C, changes to requirements must undergo a formal change control process, including impact assessment and approval prior to implementation.

Architectural design is perhaps where the differences between classes become most apparent. For Class A, design documentation is rarely needed; it is sufficient for the code to be understandable. For Class B, a high-level description of the architecture is needed, identifying the main components and their interactions. For Class C, the design must be detailed and comprehensive, including component diagrams, interface descriptions, specifications of critical algorithms, and an analysis of how the architecture contributes to security. Additionally, for Class C, the design must undergo formal peer review, and design decisions must be justified and documented. The reason for this requirement is that in high-risk software, design errors are

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

particularly dangerous because they affect the entire system and are difficult to correct once development is well underway.

3.2.2 IMPLEMENTATION AND TESTING: CODE TO THOROUGH VERIFICATION

Implementation—that is, writing the code—also has different levels of requirements depending on the class. For Class A, there are no specific requirements beyond those imposed by the team's professional standards. For Class B, it is recommended to follow coding standards that improve the readability and maintainability of the code. For Class C, coding standards are mandatory and must also include specific guidelines to prevent common errors in critical software, such as memory management, error handling, or overflow prevention. Additionally, for Class C, the code must be systematically peerreviewed, and these reviews must be documented. In some cases, it may be necessary to use static code analysis tools to automatically detect potential defects.

Integration and testing are likely the areas where the differences between classes are most quantifiable. For Class A, functional tests that demonstrate that the software does what is expected are sufficient. For Class B, in addition to functional tests, integration tests are needed to verify that components work correctly together, and regression tests are needed to ensure that changes have not introduced new errors. For Class C, the level of rigor increases significantly: unit tests with code coverage (typically at least 80% of statements) are needed, along with comprehensive integration tests covering all interfaces, system tests simulating real-world usage conditions, and robustness tests verifying the software's behavior under abnormal conditions. Furthermore, all these tests must be documented, with test cases traceable to requirements, and the results must be retained for potential auditing.

Risk management, which we explore in detail in the next chapter, also has different implications depending on the class. For Class A, the risk analysis can be brief, focusing on identifying obvious hazards. For Class B, a systematic, documented analysis is needed that identifies hazards, estimates their probability and severity, and defines control measures. For Class C, the risk analysis must be comprehensive and ongoing and updated throughout the entire lifecycle, and control measures must be verified and validated with particular rigor. Furthermore, for Class C, residual risks—those that remain after applying control measures—must be documented and explicitly communicated to users.

3.2.3 DOCUMENTATION, CHANGES, AND KNOWLEDGE MANAGEMENT

Documentation, that aspect of development that elicits so many complaints among programmers, is another area where the differences between classes are notable. For Class A, documentation may be minimal: a user manual, a design summary, and basic test records. For Class B, the documentation must include the development plan, requirements, high-level design, test cases, results, and a change log. For Class C, all of the above is supplemented by detailed design documentation, code reviews, a comprehensive risk analysis, complete traceability between all the elements, and a design history file (DHF) that allows the entire development process to be reconstructed. Furthermore, for Class C, the documentation must be structured in a way that is easily navigable and understandable to external auditors.

Change management is another area where the differences are marked. For Class A, changes can be managed informally, with a basic record of what has been modified. For Class B, a documented change control process is needed, including impact assessment, verification that the change has been implemented correctly, and updating of the affected documentation. For Class C, the change process must be particularly rigorous, involving a detailed assessment of the impact on security, formal review by a change control board, thorough verification through regression testing, and validation that the change has not introduced new risks. Additionally, for Class C, significant changes may require new regulatory approval prior to implementation.

Team composition and required competencies also vary by class. For Class A, there are no specific requirements regarding staff training. For Class B, it is recommended that those involved have training and experience in software development and that there be a clear assignment of responsibilities. For Class C, personnel are required to have demonstrated competencies in critical software development; there must be a clear separation of duties (for example, between developers and testers), and records of each person's training and experience must be maintained. In many

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

Class C projects, it is common to have a quality manager dedicated exclusively to overseeing process compliance.

3.2.4 INTERACTION WITH REGULATORS AND MANAGEMENT OF MIXED PROJECTS

The **relationship with the regulator** also varies by class. For Class A products, interaction with the regulator is generally not required beyond product registration. For Class B, it may be necessary to submit technical documentation demonstrating compliance with essential requirements. For Class C, interaction with the regulator is typically intensive, including the submission of complete dossiers, the response to detailed questions, and, in many cases, onsite audits of the development process. Manufacturers of Class C products must be prepared to dedicate significant resources to this interaction and to maintain fluid and transparent communication with the authorities.

For teams developing products of different classes, a common challenge is managing the coexistence of varying levels of regulatory requirements. It may happen, for example, that a Class C product uses a third-party library that is Class B or that a system integrates modules of different classes. In these cases, the standard allows for some flexibility, provided that the lower-class modules do not affect the safety of the higher-class modules. However, the burden of proof lies with the manufacturer, who must document how this interaction has been analyzed and what measures have been taken to ensure that a failure in a lower-class module does not compromise the safety of the whole system. This often requires additional analysis and, in some cases, specific testing.

One aspect that is often underestimated is the impact of the safety class on project timelines and costs. The development of Class C software can easily cost several times more than the development of Class A software of similar functional complexity because of all the additional processes, exhaustive documentation, and extra testing. This is not a flaw in the standard but an inevitable consequence of the higher safety requirements. Manufacturers must be realistic about these costs from the outset and not commit to timelines or budgets that would only be feasible for a lower-risk product. Underestimating the required effort is among the main causes of failure in medical software projects and is often due to a misunderstanding of the practical implications of the safety class.

Understanding the implications of each class helps clinicians grasp why the process is so slow or so costly. When an engineer explains that a certain feature requires more time because the software is Class C, they are not making excuses but describing regulatory reality. A clinician who understands this can adjust their expectations and help prioritize what is truly important, freeing up resources for what genuinely impacts safety. They can also help identify which features are truly critical and which could be implemented more simply, contributing to more efficient development without compromising safety.

Industry experience shows that many teams, especially those new to medical software, tend to underestimate the implications of the safety class. They believe that they can develop a Class C product using the same methods they use for web applications, and they encounter unpleasant surprises in regard to the time for certification. The documentation is insufficient, the tests lack the required coverage, and traceability is nonexistent. The result is costly delays and, in the worst-case scenario, the abandonment of the project. That is why it is so important to internalize from the outset that the safety class is not a label but a work program that determines how the product will be developed in each of its phases.

The practical implications of each class are not an end in themselves but a means to ensure that the software is secure and effective. The additional activities required for Class C are not empty bureaucracy but responses to real risks identified over decades of experience. When a developer complains about having to document a code review or having to run code coverage tests, it is worth reminding them that these practices exist because someone, at some point, made a mistake that could have been avoided with them. The safety class is not a punishment but a safeguard: for the patient, who receives a more reliable product; for the doctor, who can trust the tool they use; and for the developer themselves, who can sleep soundly knowing they did everything possible to prevent harm.

3.3 MANDATORY DOCUMENTATION VS. USEFUL DOCUMENTATION

There is a scene that plays out in different ways in every medical software project involving

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

inexperienced teams. At some point, someone utters the phrase “We have to document this because the standard requires it.” Immediately afterward, a developer, with a distant look in their eyes and a sigh of resignation, begins filling out templates, writing paragraphs they know no one will read, generating documents that clutter digital folders without adding any real value to the product. This scene is the clearest manifestation of something that plagues many projects: the confusion between mandatory documentation and useful documentation. Documentation is created because it has to be done, not because it serves a purpose. When this happens, documentation becomes a burden, a hindrance, and a source of frustration that distracts developers from what truly matters: building software that works and is safe.

To understand the difference between mandatory documentation and useful documentation, it is worth returning to the following fundamental question: What is the purpose of documentation in medical software development? The answer has several layers that deserve careful exploration. It helps the team develop a shared understanding of what is being built, creating a common language that reduces misunderstandings and aligns efforts. It helps ensure that those who come after can understand the decisions that were made, preventing knowledge from being lost when original team members move on to other projects. It serves to demonstrate to regulators that the process was carried out correctly, providing tangible evidence that best practices were followed. It helps investigate incidents when something goes wrong, allowing the chain of events leading to the problem to be reconstructed. It helps manage changes without introducing errors, providing a system map that guides modifications. Ultimately, it helps make the software safer and more reliable.

Documentation that fulfills any of these functions is useful. Documentation that fulfills none of them, no matter how strictly required by the standard, is an empty bureaucracy. The art of managing documentation in regulated environments consists precisely of maximizing the former and minimizing the latter.

3.3.1 THE FUNDAMENTALS OF PURPOSEFUL DOCUMENTATION

IEC 62304 establishes documentation requirements for each class of software but does not specify the format, length, or level of detail.(1) This apparent vagueness is actually profound wisdom because the standard recognizes that documentation must be tailored to the product, the team, and the specific context of each project. What might require hundreds of pages for a complex Class C product may be just a dozen for a simple Class A product. What matters is not quantity but quality: that the documentation contains the information necessary to fulfill its purpose and that it is organized in such a way that it can be found and understood when needed. A common—and costly—mistake is to believe that “more documentation” is synonymous with “better compliance.” This is not the case. Excessive documentation can be just as harmful as insufficient documentation because it makes it difficult to find what is important, consumes resources that could be devoted to activities that truly add value, and creates a false sense of security that distracts from real risks.

To distinguish between mandatory documentation and useful documentation, it is helpful to ask yourself some practical questions that act as a healthy filter. Who is going to read this? If the answer is “no one,” it is likely an empty bureaucracy that exists solely to meet a formal requirement. What is it going to be used for? If there is no clear purpose, it is likely an empty bureaucracy generated by organizational inertia. Will it be kept up to date? If the answer is “no,” because it is too burdensome to maintain, then it’s empty bureaucracy disguised as compliance, which will also be dangerously misleading when consulted in the future. Does it help make better decisions? If not, it is an empty bureaucracy that adds no value to the development process. Does it facilitate the onboarding of new team members? If not, it is empty, une bureaucracy that perpetuates dependence on tacit knowledge. Does it allow for incident investigation? If not, it is empty bureaucracy that leaves the team unprotected when problems arise. Useful documentation passes this test; documentation that does not, regardless of how mandatory it may seem, should be questioned and, if possible, eliminated or simplified.

A concrete example can help illustrate this distinction in a tangible way. Let us imagine a design requirement for a medical software feature. The mandatory documentation could be an entry in a requirements management tool with a unique identifier, a textual description, and a status indicating whether it is implemented, in progress, or pending. That is useful if it allows us to know which requirements are implemented, which are pending, and which have changed, providing a clear view of the project’s status. However, if each requirement is accompanied by a

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

multipage “impact analysis,” and that analysis merely repeats in other words, what is already in the requirement, then we are dealing with an empty bureaucracy that consumes time without providing new information. An impact analysis is useful when it truly adds value, when it identifies specific risks, when it evaluates implementation alternatives, or when it estimates necessary resources. If it’s merely a copy-and-paste formality, it’s better not to do it and instead devote that time to more productive activities.

Another example that resonates with many teams: meeting minutes. In many projects, there is a requirement to document all meetings, including attendee lists, agenda items, and decisions made. This can be useful for meetings where important decisions affecting design or planning are made because it helps them recall why certain decisions were made, preventing those decisions from being repeatedly questioned or the arguments supporting them from being forgotten. However, for daily coordination meetings, where the day’s work is simply assigned or operational obstacles are resolved, detailed minutes are empty bureaucracy that adds an administrative burden without corresponding value. A brief note in the task tracking system, updating statuses and assignments, is more than enough. The key is to tailor the level of documentation to the importance of what is being documented, applying the principle of proportionality that should govern all medical software development.

Traceability is another area where the distinction between mandatory and useful is crucial and frequently misunderstood. The standard requires traceability between requirements, design, code, and tests, and this is useful because it allows us to know which tests to run when a requirement is modified or which requirements may be affected when a piece of code is changed, facilitating impact analysis and reducing the risk of undetected side effects. However, traceability can become empty bureaucracy if it is maintained manually, with very large matrices in spreadsheets that no one actually updates, or if it is needed at unnecessary levels of detail that generate only noise. Useful traceability is that which is automatically generated by development tools that can be consulted interactively and that is actually used in daily work to make informed decisions. Traceability that exists only in static documents generated at the end of the project—likely by copying and pasting from other documents—is, at best, an illusion of control that does not truly protect the patient.

3.3.2 REVIEWS, RISKS, AND THE ROLE OF TECHNOLOGY

Code reviews are another interesting example where the tension between mandatory and useful reviews is clearly evident. The standard requires that code be reviewed, especially for Class C (), and this is useful because reviews detect errors that automated tests miss because they foster knowledge sharing among team members and because they spread understanding of the system beyond the original author. However, review documentation can be light and effective: a record of which files were reviewed, who the reviewer was, what issues were found, and how they were resolved. This information is sufficient to demonstrate compliance and to learn from recurring errors. If, instead, a detailed report is required for each review, with a thorough analysis of every line of code, the documentation becomes a burden that discourages frequent reviews—which are precisely the most valuable ones. The balance lies in documenting just enough to provide evidence that the review was conducted and to record the main findings, without drowning the process in a bureaucracy that causes developers to avoid reviews or perform them superficially.

Risk management is particularly prone to empty bureaucracy, perhaps because its abstract nature makes it more difficult to anchor in operational reality. The standard requires a documented risk analysis, and many organizations respond by generating massive risk matrices with hundreds of rows, many of which are repetitive, generic, or irrelevant to the specific product. The matrix is filled out once it is filled away in some repositories, and no one looks at it again until the next audit, when a few dates are updated and little else. This is an empty bureaucracy in its purest form, which meets the formal requirements but does not protect the patient. Useful risk management is the kind that is integrated into daily development; it is updated when a new feature is introduced that could create new hazards that guide architectural design decisions and that is consulted when an unexpected problem arises to see if it was anticipated. A small but living matrix, one that truly reflects the product’s risks and evolves with it, is worth more than a massive but dead tome that only serves to accumulate digital dust.

A useful strategy is to start with a minimal core of truly useful documentation and to add only what proves its value in practice. Ask yourself: What information do we need to coordinate effectively as a team? What would we need if a key team member left tomorrow? What would we

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

need to investigate an incident affecting a patient? What would we need to convince an auditor that we did our job well? That is the documentation that truly matters—the documentation that addresses real needs. Everything else is incidental, and if it does not add value, it can and should be eliminated or simplified. This minimalist approach not only saves work but also improves the final quality because it focuses attention and resources on what is truly relevant to patient safety.

Technology can go a long way toward reducing the burden of documentation while simultaneously making it more useful. Modern development tools allow for the automatic generation of many of the records required by the standard: traceability between code and tests, test execution results, code coverage metrics, and a history of changes along with their authors and justifications. Leveraging these capabilities is essential to ensure that documentation does not become a manual burden that distracts from the actual development work. A team that properly configures its continuous integration pipeline can automatically generate most of the evidence needed to demonstrate compliance with verification requirements. Manual documentation is then reserved for what truly requires human judgment, contextual analysis, and decision-making: architectural design decisions, risk analyses that evaluate alternative options, and justifications for why something was done in a certain way when other options were possible.

One aspect that is often overlooked, especially by those who view documentation as an external imposition, is that useful documentation is not just for regulators but also for the development team itself. A well-documented design helps new developers speed up more quickly, reducing the time it takes for them to become productive and distribute knowledge more equitably. A well-maintained record of decisions prevents the repetition of discussions settled months ago, saving time and reducing interpersonal conflicts. Clearly, specified requirements reduce misunderstandings among team members, between developers and testers, and between the technical team and clinical stakeholders. Good test documentation facilitates debugging when something goes wrong, allowing us to understand what was being tested and why. From this perspective, documentation is not a burden imposed from outside by bureaucratic entities but a tool for working better, more efficiently, and with fewer errors. Teams that internalize this vision stop complaining about documentation and begin to value it for what it is: an investment in their own effectiveness.

3.3.3 DOCUMENTATION AS A LIVING PRACTICE: PHYSICIANS, AUDITORS, AND EVOLUTION

For the doctors who participate in these projects, understanding what documentation is truly useful helps them focus their contributions effectively and prevent them from feeling that their time is wasted on paperwork. When an engineer asks to help draft a clinical requirement, it is not a meaningless bureaucratic exercise but a crucial opportunity to ensure that the software will do what medical practice truly needs, capturing nuances that only someone who has been at the patient's bedside can know. When a risk analysis is reviewed, it is not an abstract theoretical activity but a way to identify hazards that only deep clinical knowledge can anticipate, such as nonobvious drug interactions or patient behaviors in stressful situations.

Physicians who actively participate in creating useful documentation directly contribute to product safety, not wasting time on worthless paperwork. This understanding transforms medical involvement from an uncomfortable obligation into a valued and valuable contribution.

The relationship with auditors can also be approached more productively if a clear distinction is made between mandatory and useful documentation and if there is an understanding of what auditors are actually looking for. An experienced auditor, who has seen dozens of projects, is not impressed by the quantity of documents but by their quality and consistency. They prefer to see a few well-crafted documents that truly reflect the team's work and demonstrate that they understand what they are doing, rather than a mountain of paperwork that has clearly been generated solely for the audit, with no real connection to development. Transparency and honesty are always the best policies in these interactions.

If a document is missing, it's better to acknowledge it openly and explain why that decision was made than to try to cover it up with fake, incomplete, or clearly copy-pasted documents from other projects. Auditors are not looking for perfect teams that never make mistakes; they are looking for teams that understand what they are doing, are able to explain it coherently, and demonstrate that they have a system for managing quality and security.

The evolution of documentation throughout the project also deserves careful attention because documentation that is useful in one phase may be useless in another. At the beginning, when ideas are vague and the product is not yet clearly defined, documentation can be light,

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

exploratory, almost schematic, and serve as a thinking tool rather than a definitive record. As the design solidifies and important decisions are made, the documentation must become more precise, detailed, and formal, reflecting the system's growing maturity. With respect to release approaches, the documentation must be complete, reviewed, validated, and ready for use in operation and maintenance. After release, the documentation must be kept up to date with changes, evolving alongside the software it describes. This natural evolution is easier to manage if one understands that documentation is not a static product delivered once but a living reflection of the project's state that requires ongoing care. Tools that allow documentation to be versioned alongside the code, keeping them in sync, greatly facilitate this dynamic management and reduce the risk of inconsistencies.

A common mistake, especially in organizations with a highly formalistic culture, is to believe that useful documentation must necessarily be extensive, densely technical, and written in solemn language. This is not the case. Good documentation can be concise, even schematic, as long as it contains the essential information and is organized logically. A well-designed diagram showing the system architecture and the relationships between components is worth more than ten pages of descriptive text that few will read. A well-structured checklist that guides users through a complex process can replace a comprehensive manual in which no one consults. Well-placed and well-written code comments can eliminate the need for external documentation to explain local implementation decisions. The key is to choose the right format for each type of information, maintain consistency across different formats, and always ask which format will make it easiest to find and understand the information when needed.

Documentation is not an end in itself but a means to a higher end: safe and effective software that improves patient care. When documentation ceases to serve that purpose—when it becomes an obstacle that slows down development without improving quality or a source of frustration that demoralizes the team—something is fundamentally wrong with how it is being approached. We may be documenting things that do not truly matter or documenting inefficiently in a way that does not meet real needs, or we may have lost sight of the ultimate purpose of all this effort. In such cases, the solution is not to abandon documentation and descend into chaos but to critically rethink it. To ask ourselves again, honestly: what exactly is the purpose of this? Who does it actually help? How can we make it better, more useful, and less burdensome? This ongoing reflection, this willingness to question and improve, is what allows documentation to remain useful throughout the entire product lifecycle, rather than becoming that bureaucratic burden that developers so fear and that does so little to protect patients.

Once we have established how to properly document development, how do we integrate risk management so that it is not an isolated procedure but the true driving force behind design decisions?

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

REFERENCES

IEC 62304:2006+AMD1:2015 CSV | IEC [Internet]. [cited 2026-05-13]. Available from: https://webstore.iec.ch/en/publication/22794

https://doi.org/10.62486/978-9915-9876-8-2.Ch03

Chapter 4 / Capítulo 4

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch04
Pages: 63-78

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 4. ISO 14971: Risk Management

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Risk is not an enemy to be eliminated entirely but a traveling companion with whom we must learn to live. Wisdom does not lie in avoiding it, but in recognizing it, assessing it, and managing it with open eyes."

Imagine a surgeon preparing to perform open-heart surgery. He knows that there is a risk of infection, bleeding, and adverse reactions to anesthesia. He knows the statistics, has seen complications throughout his career, and is fully aware that, despite all his expertise, something could go wrong. In addition, yet, he operates. He also knew that not operating entails an even greater risk: that the patient will die without the procedure. This paradox—accepting a controlled risk to avoid greater danger—is the essence of medicine. It is also the essence of risk management in the development of healthcare software. It is not about eliminating all risk, because that is impossible, but about understanding it, reducing it to acceptable levels, and making informed decisions about how to proceed.

ISO 14971 is the international standard that establishes the principles and processes for risk management in medical devices.(1) Unlike other standards that focus on specific aspects of development, such as quality or the lifecycle, ISO 14971 has a cross-cutting scope: it permeates all activities, from initial conception to market withdrawal. Its philosophy is that safety is not something added at the end but must be integrated into every design decision, every technical choice, and every product change. Understanding this standard means understanding that risk is not a form to be filled out and filled away but a fundamental aspect of the work of building health technology.

In this chapter, we explore the fundamental aspects of risk management according to ISO 14971. We will see how to identify hazards, how to assess risks, how to implement control measures, and how to keep the entire process active throughout the product's lifecycle. However, we start with the most important thing: the shift in mindset required us to understand risk as a design discipline, not as a bureaucratic formality. If that shift does not happen, everything else becomes meaningless, empty exercises that protect no one.

4.1 RISK AS A DESIGN DISCIPLINE, NOT AS A FORMALITY

> "Risk management is not a task that can be delegated to a single specialist; it is an ongoing dialog among all the perspectives that come together in the product."

There is a widespread perception of risk management in organizations entering the regulated world. A consultant is hired, or the task is assigned to someone on the quality team, and they are asked to prepare a "risk analysis". That person, often without in-depth knowledge of the product, locks themselves in their office with the project documentation and generates an endless matrix with dozens or hundreds of rows: identified hazards, probabilities, severities, and control measures. The matrix is reviewed, approved, filed away, and rarely looked at again until the next audit. The development team continues working as if nothing had happened, making decisions without consulting that matrix, unaware that the risks it contains should be guiding every step. This scene, unfortunately common, is the antithesis of what ISO 14971 aims to achieve.

To understand the difference between risk as a formality and risk as a design discipline, it is helpful to start with a simple but revealing question: who should be involved in risk management? The answer, if the management is genuine and not an administrative charade, is as follows: the entire team. Engineers, because they know the technical limitations and potential sources of error in the code and architecture. Clinicians know how the product is actually used in medical practice and what can go wrong when the real patient does not resemble the theoretical patient. Quality managers understand the processes and regulatory requirements that must be met. Business leaders must assess whether residual risks are acceptable from the perspective of the market and corporate reputation. Risk management is not a task that can be delegated to an isolated specialist acting as a scapegoat; it is an ongoing conversation among all the perspectives that converge on the product, a conversation that enriches the design and makes it more robust.

The moment this conversation must begin is also revealing of the true nature of the process. If risk management is performed at the end, when the design is already finalized and the code written, then it will inevitably be a formality with no real impact. Risks will be identified, but it will be too late or impossible to modify them without prohibitive costs. Control measures will be limited to warnings in the instruction manual, to labels warning of danger, and to patches on a design that can no longer be fundamentally changed. If, on the other hand, risk management begins in the early phases, when alternatives are still being explored and nothing has been

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

decided, then it can truly influence the design. Safer architectures can be chosen, components with better reliability performance can be selected, and dangerous features can be discarded before they are investigated. Risk thus becomes just another design criterion, on the same level as cost, functionality, or user experience, rather than an obstacle to be overcome at the end.

4.1.1 FROM PROCESS TO DESIGN: HOW TO INTEGRATE RISK FROM START

Let us use a concrete example to illustrate this difference between the two approaches because abstractions often obscure what examples reveal. Imagine that we are designing a software-controlled infusion pump, a device that administers intravenous medication to hospitalized patients. A risk management approach treated as a formality would wait until the design is well advanced, perform a superficial analysis at the end, and discover that there is a risk of overdose if the software fails in a certain way. The control measure would then add an alarm that alerts staff when the dose deviates from expectations and a warning in the manual about the possibility of this failure, indicating that users will read and follow the instructions. A risk-as-design-discipline approach, on the other hand, would ask the following question from the outset while the architecture is still being decided: How can we design the system to be inherently safe, even if the software fails?

The answer to this early question could be to include redundancy, with two independent processors that verify each other and stop the infusion if they disagree, or to design the system so that any failure leads to a safe state, such as stopping the infusion rather than continuing with uncertain parameters, or to limit the maximum administrable dose via hardware so that not even a catastrophic software error can exceed it. These measures, which are far more effective than an alarm that can be ignored, are possible only if they are considered from the outset, when the architecture can still be chosen. Once the hardware is manufactured and the code written, these options are no longer available, and we are left with weak, last-minute solutions.

ISO 14971 establishes a cyclical process that reflects this philosophy of continuous integration, not as a rigid sequence but as an iterative dance. It begins with the identification of hazards, which are the potential sources of harm in the system. It continues with the risk assessment for each hazard, considering both the probability of its occurrence and the severity of its consequences if it materializes. It continues with risk assessment to determine whether the risk is acceptable as is or requires additional control measures. If control is necessary, measures are implemented, and their actual—not theoretical—effectiveness is verified. Afterward, the residual risk, which remains after the measures are taken, is evaluated to determine whether it is acceptable to users and patients. Finally, the entire process is documented, and mechanisms are established to ensure that risk information is updated throughout the product's entire lifecycle, recognizing that knowledge evolves. This cycle is not a one-time event that is marked as complete but must be repeated every time a significant change is introduced, every time new relevant information is discovered, and every time an incident occurs that teaches us something about our blind spots.

For this process to be effective, it is essential that hazard identification be systematic and exhaustive, not a hastily compiled list of the obvious. It is not enough to consider only the most obvious failures; we must also consider unlikely but catastrophic scenarios, unexpected interactions between components, unintended uses by creative or desperate users, human errors under conditions of fatigue or stress, and adverse environmental conditions such as power fluctuations or electromagnetic interference. This is why the participation of clinicians with real-world experience is so important because they know how real users skip instructions when they are in a hurry, how real environments are noisy and chaotic, and how emergency situations alter people's behavior in unpredictable ways. A hazard that is not identified cannot be managed, and an unmanaged hazard is a ticking time bomb waiting to explode at the most inopportune moment.

Risk assessment combines two dimensions that must be considered together: the probability of harm occurring and the severity of that harm if it does occur. Both are difficult to quantify precisely, especially in the early stages of development, when historical data are lacking. That is why the standard allows the use of qualitative categories (low, medium, and high) provided that they are clearly defined and applied consistently within the organization. What matters is not having exact numbers that give a false sense of precision but rather establishing a framework that allows risks to be compared with one another and control measures to be prioritized rationally. A risk with a low probability but very high severity—such as a patient's death—may warrant much more attention than one with a high probability but low severity, such as a temporary inconvenience.

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

The combination of both dimensions results in a risk matrix that helps visualize where to focus the team's limited efforts.

4.1.2 CONTROL HIERARCHIES AND THE RISK LIFECYCLE

Risk assessment consists of determining whether the estimated risk is acceptable or not for the specific context of the product. This is where the concept of "acceptable risk" comes into play, which—is not a universal absolute value but rather depends deeply on the clinical and social context. What is acceptable for a product treating a terminal illness with few alternatives may not be so for one addressing a minor ailment where patients have other options. What is acceptable in a country with a developed healthcare system and resources to manage complications may not be so in another with limited resources where a failure has more serious consequences. The standard does not establish rigid universal thresholds but rather requires the manufacturer to define its own acceptability criteria and for those criteria to be justified and documented. This flexibility is necessary to adapt to the diverse reality of medicine, but it is also a considerable ethical responsibility: the manufacturer must be able to defend why it considers a certain level of risk acceptable for the patients who will rely on its product.

When a risk is not acceptable according to the established criteria, control measures must be implemented to reduce it. The standard establishes a clear hierarchy of priorities for these measures on the basis of decades of experience in system safety. First, and always preferable, is inherent safety by design: modifying the design to eliminate the hazard or drastically reduce the probability of its occurrence. Second, protective measures within the device itself include alarms, interlocks, and redundancies that mitigate the risk without modifying the basic design. Third, and only as a last resort, safety information: warnings in the manual, user training, and contraindications that depend on human behavior.

This hierarchy reflects a deep insight into human nature: it is far better for the product to be inherently safe than to rely on the user reading and following the instructions perfectly. People make mistakes, become distracted, forget in moments of stress, or simply do not have time to read lengthy manuals. Relying on them as the sole safety barrier is, in itself, a risk that must be managed.

Once control measures have been implemented, the residual risk—the risk that remains after all measures have been applied—must be assessed. If this residual risk is not acceptable according to the defined criteria, the process must be repeated, i.e., new measures and new assessments, until an acceptable level is reached. However, after all reasonable measures are applied from a technical and economic standpoint, the residual risk may remain higher than desirable. In that case, the manufacturer must make a difficult decision: whether the product can be marketed as is, assuming that residual risk, or whether the only ethical option is not to market it. This decision is not purely technical but is deeply ethical and businessrelated and must be made at the highest level of the organization, with the reasoning carefully documented. In some cases, the residual risk may be so high that the only ethical option is not to market the product, even if it represents a significant financial loss. The integrity of the organization is measured at these times.

A crucial aspect of risk management that is often underestimated is that it does not end with the product's launch. In contrast, a new phase begins that is just as important: postmarketingsurveillance, which involves systematically collecting information on the product's actual performance in the clinical setting, outside the laboratory's control. Risks that were not previously identified may emerge because certain combinations of circumstances were not anticipated, or it may be discovered that the actual probability of certain types of harm is higher than theoretically estimated because real-world users behave differently than expected. This information must be fed back into the risk management process, updating estimates, reviewing control measures, and, in some cases, making the difficult decision to withdraw products from the market if the risks outweigh the benefits. A product that is safe when launched may no longer be so if new evidence emerges, and the organization must be prepared to act responsibly when this occurs.

Integrating risk management into daily work requires profound cultural change, not just formal processes. This means that in design meetings, alongside the usual questions about functionality and performance, questions about risks are systematically raised: What could go wrong here? How could we prevent it from the design stage? What happens if the user does this instead of what was intended? This means that architectural decisions are also evaluated from

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

a risk perspective, choosing safer options even if they are more complex or costly to implement. This means that testing not only verifies that the software does what it is supposed to under normal conditions but also actively explores failure scenarios, boundary conditions, misuse, and unexpected inputs. In short, security ceases to be an afterthought verified at the end and becomes another dimension of development that is as natural as functionality itself.

4.1.3 CLINICAL PARTICIPATION AND THE ETHICS OF CARE

With respect to the physicians who collaborate on these projects, risk management is an invaluable opportunity to contribute their most valuable knowledge: that of real-world, chaotic, and imperfect practices. They know how patients actually behave—that they do not always follow instructions or may have unforeseen reactions. They know how healthcare professionals become distracted in high-pressure environments, how information systems fail at critical moments, and how hospital hierarchies sometimes prevent errors from being reported. They can help identify hazards that engineers, no matter how hard they try to imagine scenarios, will not anticipate because they have not experienced clinical reality. They can assess the severity of consequences with nuanced clinical judgment, distinguishing between what is a temporary inconvenience and what is an irreversible catastrophe. They can offer opinions on whether proposed control measures are feasible in real-world practice or, conversely, will be ignored, circumvented, or even counterproductive. Their participation is not a favor that they do for the development team but a necessary condition for risk management to be realistic, effective, and truly protective of patients.

A common—and dangerous—mistake among novice teams is to think that risk management is something done once at the start of the project and that's it. Nothing could be further from the dynamic reality of software development. Every design change, every new feature added, and every software update must be accompanied by a proportionate risk review. The change may be harmless and not affect existing risks, but this must be actively verified, not simply assumed. It may introduce new hazards that did not exist before or alter the probability of existing hazards by changing the workflow. The only way to ensure that the product remains safe is to subject every modification to the same scrutiny as the original product, albeit in a more agile and focused manner. This does not mean repeating the entire analysis from scratch every time a typo is corrected, but it does mean updating it in a manner proportionate to the magnitude of the change. Risk management is a marathon that lasts the entire product lifecycle, not a sprint that is completed at the start.

Risk management documentation is another aspect that deserves careful attention because this is where the line between what is useful and what is bureaucratic is often blurred. The goal is not to generate mountains of paper that no one reads but to record the information necessary to ensure that the process is transparent, reproducible, and auditable. What hazards were identified and why? How were the risks assessed and who was involved? What control measures were implemented and what was the rationale? How was their effectiveness verified in practice? What residual risks remain and why are they considered acceptable? Answering these questions clearly and concisely is sufficient to meet regulatory requirements and maintain traceability. What must not be missing is bidirectional traceability: the ability to link each identified hazard to the specific measures that control it and each measure to the tests or verifications that demonstrate it actually works. This network of relationships is what keeps the risk analysis alive over time and what turns the documentation into a useful tool rather than a dead file.

It is worth reflecting on the profound ethical dimension of risk management, which is often hidden behind matrices and numbers. Behind every entry in a risk matrix, every probability estimate, and every assessment document, there are real people who trust our software to safeguard their health or that of their loved ones. People could suffer irreversible harm if our analysis was superficial, if our measures were insufficient, if our postmarket surveillance was lax. Risk management is not an abstract exercise in systems engineering but a concrete and tangible way to protect those people whom we will never meet but whose safety depends on our diligence.

When a team takes this process seriously, it is making a statement of ethical principles: we recognize that our product can cause harm, and we are willing to do everything reasonably possible to prevent it—not because the regulations require it but because it is the right thing to do. When a team treats it as a bureaucratic formality, it is saying the opposite: we comply because we are forced to, but we don't truly care beyond formal compliance. The difference between these two attitudes is evident in the details—in the conversations that are had or avoided, in the decisions made when no one is watching. Patients, even if they do not realize it, trust that the first

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

attitude—that of genuine care—will prevail.

4.2 HAZARD IDENTIFICATION AND RISK ASSESSMENT

> "An unidentified hazard is an unmanaged hazard, a ticking time bomb waiting to explode."

Let's imagine for a moment that we're developing software for patient management in an intensive care unit. We sit down with the team and start asking: what could go wrong? The initial answers are obvious and predictable: that the software suddenly stops working, that data are irretrievably lost, and that alarms do not sound when they should. However, as we dig deeper, more subtle and equally dangerous risks emerge: an alarm sounds so often due to false positives that exhausted nurses ignore or silence it; that data are displayed in a confusing or contradictory way, leading to misinterpretations at critical moments; and that the system is so slow or complicated that professionals, under pressure, prefer not to use it and revert to paper, losing all the benefits of digitization; and that a failure in a seemingly minor component brings down the entire system because of undocumented dependencies. Identifying these hazards is not trivial; it requires a deep understanding of both the technology and real-world clinical practice and a systematic imagination to anticipate the unexpected. However, it is the most important step in the entire risk management process because an und hazard is an und risk, a ticking time bomb waiting to explode at the worst possible moment.

ISO 14971 focuses on hazard identification and stipulates that it must be a systematic and documented process, not an occasional or intuitive activity. Sporadic brainstorming whenever someone remembers is not enough; a structured method is required to ensure that all potential sources of harm have been thoroughly considered.

This includes hazards related to the software itself, such as programming errors, communication failures between components, or loss of critical data. It includes hazards related to human use, such as errors in interpreting the information presented, unintended uses by creative or desperate users, or unexpected interactions with other medical devices. It includes hazards related to the operating environment, such as extreme environmental conditions, electromagnetic interference from other equipment, or power outages. It includes hazards related to the product's entire lifecycle, such as maintenance issues, faulty updates that introduce new errors, or inadequate product recall when it is no longer safe.

Each of these categories requires a specific approach and, often, the involvement of experts with different perspectives who can see what others do not.

4.2.1 SYSTEMATIC TOOLS FOR HAZARD IDENTIFICATION

For development teams, one of the most useful and proven tools for hazard identification is failure mode and effects analysis, known by its acronym FMEA. This technique, which originated in the aerospace industry, where errors can cost lives, has been successfully adapted to medical software development.(2) It involves breaking down the system into its functional components—whether software modules, user interfaces, or network connections—and systematically asking for each one: In what ways can this component fail? What effects would each of these failures have on the overall system? What consequences could it have for the patient if the failure occurred at a critical moment?

FMEA can be applied at different levels of detail, from the system's overall architecture down to specific fragments of critical code, and has the advantage of being systematic and exhaustive, leaving little room for improvisation or oversight. When conducted as a team, with engineers and clinicians collaborating in the same session, it often reveals hazards that neither party would have identified separately because it combines technical knowledge with practical clinical wisdom.

Another complementary technique that offers a different perspective is fault tree analysis, which starts with an undesirable event we want to prevent (for example, a medication overdose in an infusion pump) and works backward: what combinations of individual failures could collectively cause this catastrophic event? The result is a logical tree showing the relationships between different failures and how they can be combined using logical operators (AND, OR) to produce the final harm. This technique is particularly useful for analyzing complex scenarios where harm requires the concurrence of multiple conditions and helps identify which combinations are most likely and therefore deserve more attention in the design. It also allows safety redundancies to

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

be visualized intuitively: if two independent systems fail simultaneously for the harm to occur (), the risk is much lower than if a single failure is sufficient. The fault tree transforms intuition into a visible logical structure.

Clinical experience is an irreplaceable source of information for hazard identification, for which no formal technique can fully replace it. Doctors and nurses who have spent years in direct contact with patients in real-world situations are familiar with scenarios and behaviors that no engineer, no matter how creative, could imagine from behind a desk: the user who skips protocols because they are in a hurry and think they know what they are doing; the patient who handles the device in ways completely unforeseen by engineers who have never been in a hospital room; the noisy and chaotic environment of an emergency room that masks audible alarms; and the chronic fatigue that leads to mistakes that would seem unthinkable under normal conditions. The incorporation of this perspective into the hazard identification process is not a decorative luxury but rather a strict necessity for real safety. That is why it is so important for clinicians to actively participate in these analysis sessions and to feel confident enough to share not only what should happen according to protocols but also what actually happens in practice when circumstances are imperfect.

Previous incidents—whether within the organization itself, among competitors, or even in similar industries—are another goldmine of information that must not be wasted. Every time a similar product has failed in the past, valuable lessons can be learned if we are humble and attentive. Analyzing why the incident occurred, what hazards were not identified at the time, and what control measures failed or were insufficient is a way to avoid repeating the same mistakes that others have already paid dearly for.

Public incident databases, such as those maintained by the FDA in the United States or the Spanish Agency for Medicines and Health Products in Spain, are valuable public resources that teams should systematically consult before beginning a new development. History, as we saw in previous chapters, does not repeat itself exactly, but it often rhymes in recognizable patterns, and those who ignore its lessons are doomed to repeat them with new victims.

4.2.2 FROM PROBABILITY TO SEVERITY: CALIBRATING RISK

Once hazards have been identified through systematic processes, the next step is to rigorously estimate the risk associated with each hazard. This involves evaluating two fundamental dimensions that together define the magnitude of the risk: the probability of the harm occurring and the severity of that harm if it ultimately materializes. Both are difficult to quantify with absolute precision, especially in early stages of development when there is still no historical data on the product itself, but that difficulty does not exempt us from making the effort to estimate. The standard allows the use of qualitative categories (low, medium, and high) provided that they are clearly defined with specific criteria and applied consistently within the organization. What matters is not having exact numbers that give a false sense of scientific precision but rather establishing a comparative framework that allows risks to be ranked and control measures to be prioritized in a rational and defensible manner.

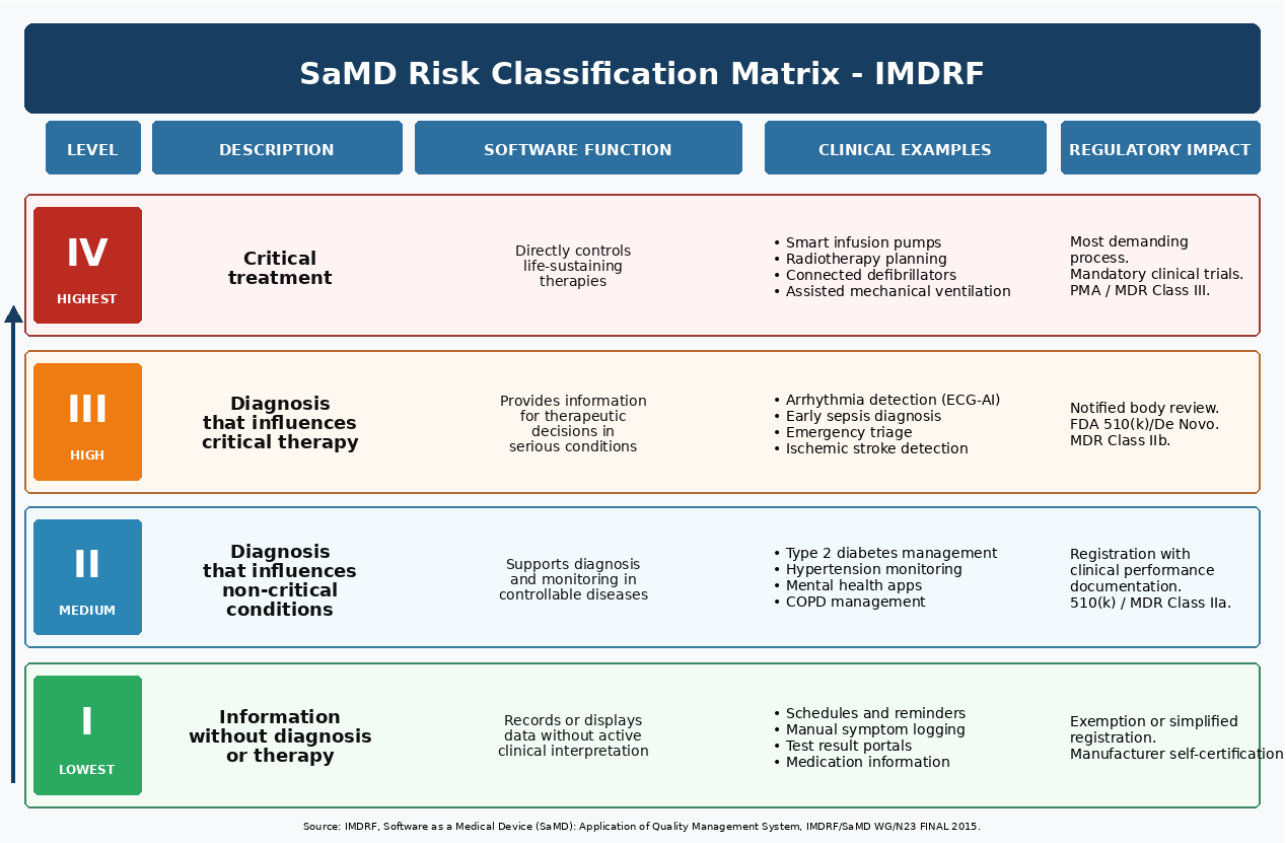
The probability of harm occurring is not the same as the probability of a pure technical failure occurring—a crucial distinction that is often confused. Between technical failure and harm to the patient lies a complex chain of events that may include human factors of perception and decision-making, specific environmental conditions, individual patient responses, and elements of chance. A software error that displays an incorrect value on the screen may cause no harm at all if the physician, on the basis of their experience, detects it as anomalous and verifies it, or it may cause serious harm if the physician blindly trusts the technology or is distracted.

Estimating the probability of harm requires considering this entire causal chain, not just the isolated technical failure. That is why the involvement of clinicians in this phase is so important that they can assess, on the basis of their practical experience, the likelihood that a given error will go unnoticed under real-world working conditions or which factors increase or decrease that likelihood.

The severity of harm also requires nuanced clinical judgment and cannot be determined by engineering alone. Mild, temporary discomfort is not the same as irreversible injury or patient death. The standard typically uses standardized categories such as minor harm (temporary discomfort, with no long-term consequences), moderate harm (reversible injuries requiring medical treatment), serious harm (irreversible injuries that threaten life or functionality), and critical harm (death). These categories must be precisely defined within the specific context of

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

the product so that different evaluators can reach consistent conclusions when the same scenario is analyzed. They must be applied by considering the worst reasonably foreseeable scenario, not the statistical average or the optimistic scenario that assumes everything will turn out well. In risk management, prudence dictates preparing for the worst while hoping for the best, because patients trust that this will be the case.



The combination of probability and severity results in a risk matrix, a visual tool that aids in prioritization and communication. Typically, risks with high probability and high severity fall into the red zone and require immediate and decisive action before any other consideration. Those with low probability and low severity fall into the green zone and may be acceptable without additional measures beyond those inherent in good design. Between these two extremes lies a yellow zone where each case must be evaluated individually, taking into account the specific context. The thresholds between these zones must be explicitly defined by the manufacturer on the basis of clinical, regulatory, and business criteria, and that decision must be documented. A well-constructed matrix allows the entire team to speak the same language of risk and have clear priorities without needing to discuss each case from scratch.

A subtle but important aspect for conceptual clarity is the distinction between hazard and risk. A hazard is the potential source of harm, the inherent condition that could cause harm; risk is the combination of the probability of that harm occurring and its severity if it does occur. For example, software that calculates radiation therapy doses has the intrinsic hazard of a “mathematical calculation error.” The associated risk depends on the probability of that error occurring and the severity of the consequences if it occurs in a specific patient. This distinction is neither academic nor arbitrary: managing risk may involve reducing the probability of the error occurring, mitigating the severity of the consequences if it occurs, or both simultaneously. Measures to reduce probability can be very different from measures to mitigate severity and require different design approaches.

Risk estimation is not a one-time exercise that is filled away and forgotten. As the design evolves from initial concepts to functional prototypes and final products, the probability of certain failures can change drastically, as can our ability to estimate it accurately. What was initially a conjecture based on similar experiences may later be based on actual test data or experience with functional prototypes. That is why the risk management process must be iterative and dynamic, updating estimates as new relevant information becomes available. A risk that seemed remote in

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

the concept phase may be much more likely when the software is tested under real-world clinical conditions. Conversely, a risk that seemed concerning may be mitigated through architectural design decisions to the point where it becomes negligible in practice.

4.2.3 HONESTY, CLINICAL CALIBRATION, AND THE ULTIMATE GOAL

For development teams, a common challenge that must be recognized and addressed is the temptation to underestimate risks to simplify the work or meet deadlines. It is psychologically easy to convince oneself that "that is very unlikely" or "that won't happen in real-world practice" when the alternative is a costly redesign. However, the history of medical software is unfortunately full of examples of things that seemed highly improbable until they occurred with tragic consequences.

Risk management demands intellectual honesty—the ability to face head-on what could go wrong without downplaying it for convenience, denying it out of excessive optimism, or putting it off out of laziness. It also requires moral courage to raise uncomfortable scenarios in team meetings and to make difficult decisions when the risk is unacceptable, even if it means delays or additional costs. It is not an exercise for the faint of heart or for those who prioritize comfort over safety.

Clinicians' involvement in risk assessment is particularly valuable for accurately gauging severity, a dimension that engineers often underestimate because of a lack of knowledge. An engineer may know technically that a calculation error can result in an incorrect medication dose but does not know what an incorrect dose means in terms of actual harm to the patient. The physician, however, knows this precisely and can distinguish between an error that causes a minor, temporary inconvenience and one that causes irreversible harm or death. This perspective is essential for correctly prioritizing risks and deciding what level of preventive effort each one deserves. A low-severity risk can be accepted with simple, cost-effective controls; a high-severity risk deserves the team's full attention and resources, even if its estimated probability is low, because the consequences are too severe to accept.

Documenting hazard identification and risk assessment is another aspect that warrants careful attention to maintain a balance between rigor and practicality. The goal is not to generate mountains of paperwork that no one reads but rather to record the information necessary to ensure that the process is transparent, reproducible, and auditable by third parties. What specific hazards were identified, and how were they identified? What systematic methods were used, and who participated in the sessions? What probability and severity categories were applied, and based on what criteria? What numerical or qualitative estimates resulted from this process? Answering these questions clearly and concisely is sufficient to meet regulatory requirements and maintain traceability. The key is to have a documented record showing that the process was conducted systematically and rigorously and that any reasonable person reviewing it with the same data would reach similar conclusions.

4.3 VERIFICATION OF RISK CONTROL AND EFFECTIVENESS

Let's imagine we've identified a hazard in our patient monitoring software: there's a possibility that a critical alert won't appear on the main screen because the system incorrectly prioritizes other, less urgent information. We assessed the risk and found that the probability was low, but the severity was high because an unseen alert could lead to delayed intervention with serious consequences for the patient. Now we have to decide what to do with this information. We could redesign the interface so that critical alerts always appear in a dedicated area that cannot be obscured by other notifications. We could add a second notification channel, such as a text message, to the responsible physician's cell phone. We could include an audit trail that logs every alert generated and every view so that we can investigate whether alerts are being missed in practice. Alternatively, we could simply include a warning in the user manual stating that the physician must periodically review all the alerts. Which of these options we choose is not trivial: it determines the product's actual safety level and the effort we will have to devote throughout the product lifecycle. ISO 14971 provides us with a framework for making these decisions in a systematic and justified manner, not an arbitrary one.

Risk control is the process by which specific measures are implemented to reduce identified risks to levels acceptable to patients and users. The standard establishes a clear hierarchy of priorities for these measures, reflecting decades of painful experience in the safety of medical

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

devices and other critical systems.

- Inherent safety by design: modifying the fundamental design of the product to eliminate the hazard or drastically reduce the probability of its occurrence.
- Protective measures within the device itself: physical or logical barriers, alarms, and redundancies that mitigate risk without modifying the basic design.
- Safety information: warnings in the manual, user training, and contraindications in the documentation (only when the previous measures are insufficient).

This hierarchy is neither arbitrary nor capricious: it responds to the fundamental principle that it is far more reliable for the product to be inherently safe by design than to rely on the user behaving in a certain way, reading the instructions, or remembering their training in moments of stress.

4.3.1 THE HIERARCHY OF CONTROLS: FROM SAFE DESIGN TO INFORMATION

Inherent safety by design is always the preferred option when it is technically feasible. If we can design the software so that certain errors are impossible to make, or so that any failure automatically leads to a safe state without human intervention, we will have robustly eliminated the risk at its root. For example, if our software controls an intravenous infusion pump, we can limit the maximum administered dose via independent hardware so that not even a catastrophic programming error can cause a lethal overdose. If our software displays critical laboratory results, we can design the interface so that out-of-range values always appear in a distinctive color and size that make them unmistakable, regardless of the user's display preferences.

These measures are the most effective because they do not depend on the user doing something correctly in a moment of haste or fatigue or on external conditions being favorable. The product is safe by design, not by accident or due to ideal operator behavior.

However, the reality of engineering tells us that it is not always possible to eliminate all risks by design without destroying the product's useful functionality. Sometimes current technical limitations, prohibitive costs, or the very nature of the medical problem impose certain residual risks that must be managed in other ways. In such cases, protective measures come into play as a second line of defense. These measures do not eliminate the source of the danger, but they reduce the likelihood of it causing actual harm or mitigate the consequences if it does occur. For example, a monitoring system may include a persistent audible alarm that sounds if a critical alert has not been acknowledged within a certain time, forcing the user's attention. This alarm does not prevent the alert from going unnoticed initially, but it drastically reduces the likelihood that it will go unnoticed permanently.

Protective measures are less effective than inherent safety because they add complexity and additional points of failure, but they are still preferable to relying exclusively on user information.

At the bottom of the hierarchy, and only when the previous measures have exhausted their usefulness lies safety information: warnings in the manual, detailed instructions for use, user training programs, and contraindication labels. These measures are the least reliable of all because they depend on a chain of uncertain events: the user reads the information, understands it correctly, remembers it at the right moment, and follows it despite the pressures of the clinical environment. The accumulated experience shows that this occurs less frequently than we would ideally like. Manuals are lost or become obsolete, warnings are ignored out of habit, and training is forgotten over time or not applied in stressful situations. That is why the standard insists that these measures should be used only when it is not possible to effectively apply the previous measures and never as a substitute for a safe design. A warning that states "Caution: this software may cause serious errors" is not an acceptable control measure for a high-severity risk; it is an admission of technical inadequacy and an abdication of responsibility.

For every control measure implemented, regardless of its level in the hierarchy, it is necessary to verify that it actually works as expected and that it reduces the risk as predicted in the analysis. This verification can take different forms depending on the specific nature of the measure and the product context.

For architectural design measures, this may involve formal peer code reviews, specific boundary case testing, or mathematical analyses confirming that the implementation faithfully follows the design specifications. Protective measures such as alarms or redundancies may involve exhaustive functional testing to confirm that the alarm sounds when it should and does not sound when it should not, or that the system behaves as expected under simulated failure conditions. For informational measures, this may involve reviews of the readability and comprehensibility of

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

manuals or empirical studies with representative users demonstrating that they understand and recall critical information. In all cases, verification must be properly documented, and the results must be retained as tangible evidence that the measure is effective in practice, not just in the designer's theory.

4.3.2 VERIFICATION UNDER REAL-WORLD CONDITIONS AND RESIDUAL RISK ASSESSMENT

A crucial aspect of verification that distinguishes mature teams from novices is that it must be conducted under conditions that adequately represent real-world use, not just the idealized environment of the development lab. It is not enough to test that the alarm sounds in a quiet room; one must ensure that it remains audible and distinguishable in a noisy hospital environment with multiple other devices emitting similar sounds. It is not enough for the manual to be wellwritten according to linguistic criteria; one must verify that real users, with their actual level of training and under time pressure, understand it and consult it when they have critical questions. Verification under ideal conditions can give a false sense of security that is worse than ignorance because it generates unfounded confidence. The only verification that truly matters is one that honestly reflects operational reality, with all its imperfections. That is why the involvement of experienced clinicians in this phase is so valuable because they know the actual conditions of use and can detect problems that engineers, in their controlled laboratory, would never anticipate or be able to reproduce.

Once the control measures have been implemented and verified, the process requires a reassessment of the residual risk—that is, the risk that remains after all selected control measures are applied. This residual risk must be compared with the acceptability criteria previously defined by the organization in an explicit and justified manner. If the residual risk is acceptable according to those criteria, the process can proceed to the next phase. If it is not, the cycle must be repeated, including new additional control measures, new verification of their effectiveness, and new assessment of the resulting risk. After several iterations of this cycle, the residual risk may remain unacceptable despite all reasonable efforts. In that case, the manufacturer faces a difficult and high-stakes decision: it may accept the residual risk if it considers that the product's clinical benefits far outweigh the risks, or it may decide not to market the product because it cannot guarantee its adequate safety.

This decision is not purely technical but deeply ethical and businessrelated and must be made at the highest level of the organization with full awareness of the implications, carefully documenting the entire reasoning.

Benefit-risk assessment is a central concept in this phase of the process, recognizing that no medical product is completely safe in an absolute sense; all carry some level of residual risk. The fundamental question is whether the benefits the product provides to patients' health outweigh those risks and whether the risks are reasonable compared with existing alternatives for treating the same condition. Software that helps diagnose a life-threatening disease in its early stages may be associated with a higher risk than software that simply automates administrative tasks with low clinical impact is. Software that offers significant advantages in efficacy or safety over existing treatments may present risks that would be unacceptable if there were equally effective but safer alternatives. This assessment must be explicit, fully documented, and based on scientific evidence whenever possible. It is not a decision that can be taken lightly or delegated to lower levels of the organization.

For development teams, implementing control measures is often the most rewarding and tangible part of the risk management process. It is where abstract thinking becomes a concrete action, where ideas materialize into code and design, where product safety is visibly improved. However, it is also where the natural tensions of product development arise because control measures often come with associated costs: they add complexity to the system, can negatively impact performance or usability, and sometimes conflict with other desirable functional requirements. An overly sensitive alarm system may end up being ignored by users because of alert fatigue. Security redundancy can slow the system down to the point of impracticality. A conservative safety constraint can reduce useful functionality to the point of making the product uncompetitive. Balancing these conflicting factors is part of the art of engineering, and there are no magic formulas that apply to all cases. The standard requires that decisions be explicit, justified with sound arguments, and documented for review.

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

4.3.3 CONTINUOUS LEARNING, MONITORING, AND DOCUMENTATION

Verifying the effectiveness of control measures is also a valuable opportunity for organizational learning if the team is willing to listen to what the results reveal. When a measure does not work as expected in testing or when testing reveals unanticipated problems in the original design, that information is invaluable even if it is uncomfortable. This may indicate that our initial understanding of the risk was incorrect or incomplete, that the design has deeper flaws than we detected, or that the actual conditions of use differ from what we assumed in our hypotheses. Instead of hiding these findings, downplaying them, or attributing them to testing errors, mature teams celebrate them as opportunities for genuine improvement. Every problem detected during the verification and validation phase is a problem that will not reach the patient under real-world conditions. Every flaw found in time, even if it requires rework, is a life potentially saved or an injury prevented.

For clinicians involved in these development processes, the control and verification phase is where they can contribute their most practical and valuable perspective. They can help design control measures that are compatible with the actual workflow of clinical units, that do not add unnecessary administrative burdens and that are intuitive for users even in stressful situations. They can actively participate in verification testing, simulate real clinical situations and evaluate whether the measures function as expected in realistic contexts. They can review safety information from their perspective as end users, ensuring that it is clear, understandable, relevant, and memorable. Their involvement in this phase is not mere formality but a guarantee that control measures will be effective in actual clinical practice, not just on paper as envisioned by engineers.

One aspect that is often underestimated or treated as secondary is the need to maintain control measures over time as the software evolves. Medical software is not static like a traditional physical device; it is periodically updated, modified to correct errors or add features, and adapted to new technological or clinical environments. Every change, no matter how small it may seem, can affect existing control measures, inadvertently disabling them, reducing their effectiveness, or creating new, unforeseen interactions. That is why verification is not a one-time event completed before commercialization but rather a continuous process repeated throughout the entire product lifecycle. Every time the software is modified, it is necessary to reassess whether the control measures remain effective in the new context and, if necessary, update them or add new mitigating measures. This continuous monitoring is especially critical in Class C software, where risks are higher and the consequences of a failure can be devastating.

Risk control documentation must reflect this entire process in a comprehensive and transparent manner. It must be clear which specific measures were implemented for each identified risk, why those measures were chosen over other alternatives considered, how their effectiveness was verified using which methods and results, and what residual risks remain after all the measures have been applied. This documentation is not meaningless bureaucratic red tape but tangible evidence that the process was carried out correctly and with due diligence. In the event of a subsequent incident, it will be the first thing that regulators, investigators, and potentially the courts will examine to determine whether the manufacturer acted responsibly. If the documentation is deficient, incomplete, or contradictory, it is presumed that the process was as well. If it is complete, rigorous, and consistent, it will demonstrate that the manufacturer has fulfilled its duty of care.

Risk control is not an exercise in impossible perfection but rather in realistic continuous improvement. No medical product is completely safe in an absolute sense, no development process is infallible, and no verification is 100% exhaustive. What the standard requires is not utopian perfection but a systematic and documented effort to achieve the highest level of safety reasonably possible given the state of the art. That “reasonably possible” is important because it recognizes that there are technical, economic, and practical limits that cannot be ignored without destroying the product’s viability. What the standard does not tolerate under any circumstances is negligence, improvisation, or the passive acceptance of avoidable risks for the sake of convenience or economy.

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

4.4 HOW 14971 AND 62304 RELATE

> "Separating them is like separating the map from the territory: the map without the territory is a theoretical exercise; the territory without the map is a labyrinth with no exit."

Let's imagine we're building a house. On the one hand, we have an architect who focuses on the structure, the materials, and the layout of the spaces. On the other hand, we have a safety engineer who analyzes the risks: fires, floods, and landslides. Both work separately, with different plans, with hardly any coordination between them. The architect designs a beautiful spiral staircase; the engineer points out that in the event of a fire, it would be difficult to evacuate via it. The architect designs large windows; the engineer warns that they could break easily and cause injuries. If they do not work together from the start, the result will be a house full of conflicts between design and safety, or worse yet, an unsafe house because the engineer's warnings came too late to modify it without prohibitive costs. This same dynamic occurs in medical software development when ISO 14971 (risk management) and IEC 62304 (software lifecycle) are applied as parallel and disconnected processes, each in its own silo, without the integration that both standards actually presuppose.

The relationship between the two standards is, in reality, one of deep interdependence that is not always recognized at first glance. IEC 62304 establishes the activities that must be carried out throughout the software lifecycle, but many of those activities are specifically aimed at managing the risks identified according to the methodology of ISO 14971. ISO 14971 provides a conceptual framework for identifying and evaluating risks, but the control measures it proposes must be implemented through the specific development activities defined by IEC 62304. Separating them is similar to separating the map from the territory: a map without territory is a theoretical exercise with no practical application; territory without a map is a labyrinth with no exit where one walks blindly. Understanding how to apply them synergistically is essential for efficiently building safe medical software, without duplicating efforts or leaving dangerous gaps between the two approaches.

The first fundamental connection between the two standards is established in the project planning phase. IEC 62304 requires a software development plan that includes, among other essential elements, the identification of risks and the risk management activities to be carried out. This means that the risk management plan derived from ISO 14971 should not be a separate document, isolated in a quality folder, but should be organically integrated into the overall development project plan. It must explicitly specify how hazards are identified in each phase, how associated risks are estimated, how technical control measures are implemented, and how their actual effectiveness is verified. It must also clearly define the risk acceptability criteria, which will guide difficult decisions throughout development.

The second connection occurs during the requirements analysis phase, where the abstraction of risk is transformed into concrete specifications. ISO 14971 requires that safety requirements be derived directly from risk analysis, creating a necessary bridge between the two worlds. This means that the hazards identified in the analysis must be translated into specific, measurable, and verifiable requirements that the software must meet to effectively mitigate those hazards. For example, if the hazard has been identified such that a critical alert regarding a change in vital signs might go unnoticed by the clinician, a functional requirement must be generated specifying that critical alerts must be displayed in a reserved and unmistakable area of the screen or that they must be accompanied by a distinctive audible signal that cannot be confused with other alarms. These safety requirements become part of the set of requirements that IEC 62304 requires to be documented and managed with traceability. Bidirectional traceability between hazards and requirements is essential: Each hazard must be linked to the specific requirements that control it, and each safety requirement must be explicitly justified by one or more identified hazards, avoiding "just-in-case" requirements not grounded in actual risks.

The third connection lies in the software's architectural design, where technical decisions have safety implications. IEC 62304 requires that the software architecture takes into account the identified safety requirements and that the mapping of these requirements to the various system components be documented. ISO 14971, for its part, requires that control measures be implemented following the hierarchy of controls, ideally through inherent safety by architectural design. The integration of both perspectives means that the architecture is designed with risks in mind from the very first draft, not as a late-stage review. For example, if an identified risk is the potential corruption of critical data due to transmission failures, the architecture may include storage redundancy and cryptographic integrity checks. If another risk is potential interference

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

between modules of different criticality levels, the architecture may ensure strict isolation through process separation or virtual machines. Architectural decisions are not merely technical but also direct responses to identified risks and must be documented as such.

The fourth connection manifests itself in code implementation, where risks determine the intensity of scrutiny. IEC 62304 requires that the code be written following practices that minimize errors, such as structured coding standards and peer reviews. ISO 14971 complements this by requiring that specific control measures be implemented correctly and that their actual effectiveness be verified, not just their existence. In practice, this means that parts of the code implementing control measures critical to high-severity risks must receive special and disproportionate attention: more thorough reviews by multiple reviewers, more rigorous unit tests with near-100% coverage, and more detailed documentation explaining the reasoning. Risk analysis helps identify which parts of the code are most sensitive from a safety perspective, allowing the team to focus its limited efforts where they are most needed and avoiding the mistake of treating all code with the same intensity. Not all codes deserve the same level of scrutiny; risk rationally determines priorities.

The fifth connection occurs in the verification phase, where testing becomes evidence of safety. IEC 62304 requires exhaustive testing to demonstrate that the software meets its specified requirements. ISO 14971 requires that the effectiveness of the implemented risk control measures be specifically verified. Integrating both requirements means designing tests so that they not only verify that the software does what it is supposed to under normal conditions but also confirm that risks have been adequately mitigated under adverse or failure conditions. For example, if a control measure consists of an alarm that must sound under certain clinical risk conditions, the tests must include scenarios that specifically trigger that alarm and verify that it actually sounds, that it is audible in the anticipated noisy environment, that it does not trigger falsely or under normal conditions and that it cannot be easily ignored. Risk verification is not a separate, additional activity but rather another dimension of overall verification that adds depth and realism to testing.

The sixth connection lies in change management, where risk dynamics become evident. IEC 62304 requires a formal change control process that assesses the impact of any proposed modification to the software. ISO 14971 complements this by requiring that risks affected by the change be specifically re-evaluated. Whenever a software modification is proposed, one must systematically ask the following: Does this change introduce new hazards that did not exist before? Does it alter the probability or severity of existing hazards, improving or worsening them? Does it affect the effectiveness of previously implemented control measures, weakening or nullifying them? The answers to these questions determine whether the change can be implemented directly with confidence, whether it requires new additional control measures, or whether it necessitates a complete risk reassessment that may alter the software's classification. Integrated change management is the only way to ensure that the software's risk profile remains up-to-date and known throughout its entire lifecycle, preventing well-intentioned modifications from introducing unnoticed hazards.

The seventh connection manifests in the documentation, where consistency facilitates understanding and auditing. Both IEC 62304 and ISO 14971 require detailed documentation but not separate and redundant documentation. Ideally, risk documentation should be integrated into general development documentation, creating a coherent narrative of the process. The design history file (DHF) must contain the complete risk analysis, explicit traceability between hazards and safety requirements, the results of the verification of control measures, and the reassessments performed following changes. This integration facilitates regulatory audits and, above all, makes risk information accessible and relevant to the entire team in their daily work. When a developer reviews the requirements for a component, they should also see the risks that the component helps mitigate. When a tester designs test cases, they should know which specific risks they are verifying. Risk information is not just for the historical record; it is a working tool for the team.

Once we have integrated risk management throughout the software lifecycle, how can we ensure that the final product is usable in the real clinical environment, where the human factor can be both the last line of defense and the primary source of errors?

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

REFERENCES

ISO 14971:2019 - Medical devices — Application of risk management to medical devices [Internet]. [cited May 13, 2026]. Available from: https://www.iso.org/standard/72704.html
Failure Mode and Effects Analysis - What is FMEA [Internet]. [cited May 13, 2026]. Available at: https://www.cegid.com/ib/es/blog/que-es-amfe-fmea/

https://doi.org/10.62486/978-9915-9876-8-2.Ch04

Chapter 5 / Capítulo 5

Check for updates

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch05
Pages: 79-94

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 5. IEC 62366: Human Factors and Usability

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "No matter how perfect an algorithm may be, no matter how rigorously it has been developed, if the physician using it cannot understand it, if the patient does not know how to interpret it, if in an emergency the interface becomes an obstacle, then all the effort will have been in vain. Technology does not exist in a vacuum: it exists to be used by people, and it is those people who, ultimately, determine its success or failure."

Imagine an operating room in the middle of a complex procedure. The surgeon needs quick access to the patient's information, but the monitor screen is cluttered with irrelevant data. He tries to adjust the settings, but the menus are confusing, and the buttons are too small for his gloves. He asks the nurse for help, but she cannot find the right option either. Seconds tick by—seconds that are precious in surgery. Finally, someone manages to access the information, but the delay has created tension and, worse still, has broken the team's concentration. This unfortunately common scenario illustrates an uncomfortable truth: technically perfect software can be clinically useless—or even dangerous—if it does not consider the people who will be using it. That is exactly what usability engineering and the standard that governs it, IEC 62366, are all about.

IEC 62366 is the international standard that establishes the requirements for the usability engineering process applied to medical devices. Its full title is "Medical devices—application of usability engineering to medical devices," and its objective is to ensure that products are safe and effective when used by the people for whom they are intended in the environments where they are actually used.(1) Unlike other standards that focus on the internal workings of software, 62366 looks outward, toward the interaction between the product and its users. In addition, it does so with a deep conviction that usage errors are not the user's fault but rather the fault of the design and that preventing those errors is the manufacturer's responsibility.

In this chapter, we explore the fundamentals of usability engineering as applied to medical software. We will see how to integrate it from the earliest stages of design, how to identify usage-related risks, how to evaluate the interface with real users, and how to document the entire process. However, we start with the most important thing: understanding that usability is not an aesthetic embellishment or a superficial layer added at the end but a fundamental dimension of safety. Software that is difficult to use is not just a nuisance; it is a risk to the patient.

5.1 USABILITY ENGINEERING FROM THE FIRST SKETCH

> "Usability is not a superficial layer, but a deep-seated property that emerges from design decisions made throughout the entire process."

There is a widespread misconception in the world of software development, especially among those coming from unregulated environments, who view usability as something added at the end of the project, when the functionality is already implemented and all that remains is to "polish" the presentation. A graphical interface designer is hired and asked to "beautify" the application with attractive colors and modern fonts, and it is expected that with a few cosmetic tweaks, the product will become intuitive and userfriendly. This approach, which is already questionable in any type of commercial software, is simply dangerous and potentially lethal in the medical field. Because usability is not a superficial coat of paint but a deep, structural property that emerges from the design decisions made throughout the entire development process. If it is not considered from the very beginning—when requirements are being defined and the system architecture is being outlined—the final result is very likely to be a product that, although it functions correctly from a technical perspective, is difficult or impossible to use safely in actual clinical practice.

Usability engineering, as rigorously defined by IEC 62366, is a systematic and planned process that must begin in the earliest stages of development, long before the first line of code is written. It begins with a thorough identification of the product's users: not only medical specialists but also nurses on different shifts, maintenance technicians, patients who may interact with the system, and family members who sometimes act as informal caregivers. Each group has different demographic characteristics, distinct clinical needs, widely varying levels of technological training, and specific usage contexts that must be deeply understood. Software intended for use by medical specialists in a controlled hospital setting can afford a certain level of complexity and information density; software intended for use by elderly patients in their private homes must be extremely simple, intuitive, and error-tolerant. Getting to know the actual users, understanding their physical and cognitive limitations as well as their real capabilities, is the essential first step toward designing something that truly serves them in their living or working context.

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

5.1.1 CONTEXT OF USE, PROTOTYPES, AND EARLY ITERATION

The next methodological step is to gain a deep understanding of the actual context of use, which rarely aligns with designers' idealizations. Using software in a quiet primary care office, with ample time and no interruptions, is not the same as using it in a bustling intensive care unit, with alarms constantly ringing, distressed family members asking questions, and life-or-death decisions to be made in seconds. It is not the same to use it in a well-lit hospital with stable infrastructure, as it is in a private home with dim lighting, reflective screens, and an intermittent internet connection. It is not the same to use it hands-free and sitting comfortably, as it is while it is wearing latex gloves during a sterile procedure or in the middle of a 24-hour night shift when physical and mental fatigue clouds one's attention and decision-making.

Usability engineering requires designers to literally immerse themselves in these contexts, to observe without interruption, to ask questions with humility, and to understand the real and often adverse conditions in which the product will be used. Only then can they anticipate usability issues and design solutions that work in real clinical practice, not just in the development laboratory under ideal conditions.

With these collected ethnographic data, the actual interface design phase begins. However, the goal is not to design the polished final version directly but rather to quickly create prototypes, sketches, and mockups that can be evaluated with real users before investing scarce time and resources in full-scale development. A prototype can be as simple as drawings on paper that simulate screens or as elaborate as an interactive simulation that responds to user actions. What matters is not the degree of technical sophistication but that it allows ideas to be tested with real users quickly and cost-effectively before they commit to decisions that are difficult to reverse.

This iterative approach—involving rapid, low-cost trial and error—is a hallmark of usability engineering and stands in stark contrast to the traditional approach of designing everything in detail first and testing only at the end, when changes become prohibitively expensive. Testing early on, testing frequently with different users, and making adjustments on the fly on the basis of real evidence constitute the most efficient and reliable way to achieve a truly usable interface in the clinical setting.

The participation of real users in these prototype tests is absolutely irreplaceable, no matter how uncomfortable it may be for designers. Regardless of how hard design professionals try to put themselves in the shoes of a medical professional or patient, cultural, practical, or cognitive aspects escape them because of differences in training and context. Only by directly observing a real clinician interacting with a prototype do we discover that the button that seemed obvious and wellplaced was not so obvious to someone in a hurry that the screen flow that seemed logical from a technical perspective did not match the actual sequence of clinical work and that the medical terminology used by engineers was not exactly what clinicians routinely use in their daily practice.

These observations, although they sometimes humble our design assumptions, are pure gold for improving the product and must be systematically documented and used to iterate on the next prototype. Usability is not designed in an ivory tower of experts; it is built through continuous and respectful dialog with those who use the product in their real lives.

A crucial aspect of this process that distinguishes it from conventional interface design is the specific identification of risks related to the product's use. IEC 62366 formally requires a systematic analysis of potential usage errors and their possible consequences for the patient. For example, if a critical action button is too small on a touchscreen, there is a predictable risk that the user will accidentally press the wrong button in a stressful situation. If a clinical alert is not sufficiently visually and audibly prominent, there is a risk that it will go unnoticed in a noisy environment. If the sequence of steps to complete a task is confusing or unintuitive, there is a risk that the user will complete the steps in the wrong order or omit a critical one.

Each of these usability risks must be explicitly identified, assessed in terms of potential severity, and, if necessary, mitigated through improvements to the interface design. Usability in the medical context is not just a matter of convenience or user satisfaction but of patient safety.

5.1.2 HIERARCHIES OF CONTROL, USER DIVERSITY, AND DESIGN CULTURE

The hierarchy of control measures we saw in technical risk management also applies here to usage risks, with the same priority logic. The best way to prevent a usage error is to design the product so that the error is physically or logically impossible to make.

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

For example, if we do not want the user to enter an incorrect medication dose, we can limit the values allowed by the software to a clinically established safe range so that even if the user attempts to enter an out-of-range value by mistake, the system simply does not accept it and prompts for reconsideration. If we cannot make the error impossible for reasons of clinical flexibility, we can at least make it difficult to commit inadvertently, for example, by requiring conscious double confirmation for critical actions that cannot be easily undone. If we cannot make it difficult, we can at least make it visible and recoverable, for example, by clearly showing the consequences of the action before it is definitively confirmed and allowing it to be undone within a reasonable time window. Safety information, such as generic on-screen warnings or notes in the manual, is the last line of defense and should be used only when the previous design measures are insufficient because of technical or clinical limitations.

Documentation of the entire usability engineering process is another formal requirement of the standard that serves an important practical purpose. There must be a written record of exactly who the intended users are and their characteristics, what the detailed context of use is, which prototypes were developed in each iteration, which usability tests were conducted with whom and under what conditions, what specific usability issues were identified, and how they were resolved through design changes. This documentation is not meaningless bureaucratic red tape but tangible evidence that the process was carried out correctly and with due diligence. In the event of an incident related to the use of the product, it will be the first thing regulators and potentially the courts examine to determine whether the manufacturer acted responsibly.

If the documentation demonstrates a rigorous process, with early identification of usability issues and control measures implemented progressively, the manufacturer will be in a much stronger position than if everything was done haphazardly or without evidence.

For developers accustomed to working in other, less regulated fields, systematic usability engineering can represent a significant cultural shift that requires adaptation. It involves accepting that one's own design work will be observed, constructively criticized, and modified on the basis of the opinions of people who are not experts in technology or design. It involves setting aside pride in one's own solution when user testing shows that it does not work as expected. It involves understanding that the ultimate goal is not to demonstrate how clever or elegant one is as a designer but to build something that effectively serves others in the context of their needs. This intellectual humility is difficult for technical professionals proud of their abilities, but it is essential for those who wish to seriously dedicate themselves to medical software. Unlike in other fields of software development, an interface design error does not simply mean a bad review in the app store or a complaint on social media but potentially a harmed patient or a life lost.

For doctors and other healthcare professionals involved in these design processes, usability engineering is an invaluable opportunity to contribute their most valuable and contextualized knowledge: that of real-world, chaotic, and imperfect practice. They can pinpoint exactly why a certain screen flow does not match how they actually work in their department, why technical terminology is not what they typically use in their daily communication, or why a feature that seems useful in theory is rarely used in practice because of time constraints or prioritization. They can actively participate in prototype testing, candidly pointing out what they find confusing, what they consider unnecessary, and what they feel is missing for their work.

One aspect that is often underestimated, even by well-intentioned teams, is the actual diversity of users within a single clinical setting, which necessitates designing for the "reasonable worst-case scenario." A physician with twenty years of experience in the department is not the same as a resident who has just arrived and is still receiving the hang of the protocols. A day shift nurse, with the full support team available, is not the same as a night shift nurse, alone on the floor, with limited resources and accumulated fatigue. A young, tech-savvy patient is not the same as an older patient with vision problems, reduced manual dexterity, or a fear of technology. Usability engineering must explicitly consider this diversity and design for all of them, not just for the ideal, expert user under optimal conditions. A product that only works for the expert user under ideal conditions is a product that will fail in real-world practice because clinical practice is full of diverse users in diverse and often adverse conditions.

The rapid evolution of technology also poses fascinating yet complex new challenges for medical usability. Multitouch interfaces, voice commands, augmented reality, and wearable devices open up new possibilities for interaction but also introduce specific new risks that must be managed. A voice command misinterpreted because of ambient noise can have serious consequences if it activates a critical function without confirmation. A touch interface with buttons that are too small or too close together can lead to touch errors in stressful situations. An augmented reality system that overlays visual information in the surgeon's field of view can

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

distract them rather than help them if it is not carefully designed. Usability engineering must keep pace with these emerging technologies, assessing their specific risks of use and developing innovative methods to mitigate them without sacrificing their benefits. It is not a matter of conservatively rejecting the new out of fear of change but of incorporating it with scientific prudence and rigorously testing it before relying on it for patient care.

It is worth remembering that usability in the context of medical software is not an aesthetic luxury or a commercial gimmick but a fundamental aspect of patient safety. Software that is difficult to use, confusing, or frustrating is not just an annoyance for the professional; it is an active risk to the patient. When the user becomes frustrated and when they cannot quickly find what they are looking for, when they hesitate to interpret critical information or how to perform an urgent action, the probability of error increases exponentially. In medicine, errors have real and often irreversible human consequences. Usability engineering, rigorously applied from the very first sketch and maintained throughout the entire lifecycle, is the best tool available to prevent such usage errors. It does not absolutely guarantee that they will not occur, but it makes them far less likely. In a field where the probability of error directly translates into human lives affected, any reduction—no matter how small it may seem statistically—is worth every effort invested.

5.2 USAGE ERRORS VS. USER ERRORS

Imagine we are in an intensive care unit, in the middle of a hectic and noisy night. An experienced nurse, who has been on duty for 12 hours and is visibly fatigued, urgently needs to program an infusion pump to administer a critical vasoactive drug to stabilize an unstable patient. The pump's interface displays the options on a touchscreen, but the buttons are small and closely spaced, and the tactile feedback is imprecise. With fingers trembling from physical exhaustion and emotional pressure, the nurse accidentally pressed the button next to the one she actually intended, unwittingly selecting a dose ten times higher than what the doctor prescribed. The pump begins to infuse the medication at that dangerous rate, and it is not until several minutes later, when the patient begins to show alarming signs of severe hypotension, that someone realizes the error and reverses the situation. Whose fault is this incident that nearly cost the patient their life? The nurse's, for not being careful enough at a critical moment? The manufacturer, for designing an interface that facilitates and almost invites errors under stressful conditions? The hospital, for having such long shifts that inevitably lead to chronic staff fatigue? The answer to this question is not merely academic or philosophical: it determines how the incident will be formally investigated, what corrective measures will be implemented to prevent recurrences, and above all, how future patients will be protected from similar situations.

The distinction between usage error and user error is among the most important and simultaneously most misunderstood concepts in all usability engineering applied to medical contexts. A user error is fundamentally attributed to a lack of attention, specific knowledge, or technical skill on the part of the individual using the product at that moment. It is viewed as a personal failure, a deviation from the expected ideal behavior. A usage error, on the other hand, is caused, facilitated, or amplified by characteristics of the product's design itself, by interface decisions that did not adequately consider human limitations.

The difference is subtle on the surface but crucial in its practical implications because it determines where we focus our intervention to prevent future incidents. If we consider what happened to be an isolated user error, the solution will be to better train staff through additional courses or to discipline the nurse as a deterrent or simply to verbally remind her to be more careful in the future. If we consider it a systematic usage error, the solution will be to redesign the interface so that the error is physically more difficult or even impossible to make, eliminating the condition that causes it. The first approach barely reduces the likelihood of it occurring again to another user in another stressful situation; the second eliminates the possibility of that specific error at its root for all future users.

5.2.1 DESIGN RESPONSIBILITY AND TASK-CAPABILITY MISMATCH

IEC 62366 takes a clear and explicit stance regarding this distinction: usage errors are the responsibility of the medical device manufacturer, not the user. The standard formally requires that the usability engineering process proactively identify potential errors users might make given the proposed design, assess the potential consequences of those errors for the patient, and

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

take design measures to prevent them or at least mitigate their effects. This does not mean, of course, that users are considered infallible or that manufacturers must absolutely guarantee that no errors will ever be made. This means that the design must be inherently tolerant of human error, recognizing from the outset that the people who will use the product become physically and mentally fatigued, are distracted by multiple stimuli, make mistakes under pressure, and that the product must be explicitly prepared for these inevitable situations.

Good medical design does not assume that the user will be perfect, attentive, and expert; it assumes that the user will be human with all their limitations and designs accordingly to protect the patient from those limitations.

To systematically distinguish between usage errors and user errors in incident analysis, human factor experts use the technical concept of a "mismatch" between the cognitive and physical demands of the task and the actual capabilities of the typical user under real-world working conditions. When a task consistently demands more than a typical user can provide under normal conditions of use in the clinical environment, the resulting error is by definition a usage error, not an error of the individual. For example, if a software interface requires a complex sequence of steps that are only used in rare emergency situations, it is predictable that in a real crisis situation, the user, no matter how competent, will forget some critical step because of stress. This is not a user error due to carelessness; it is a design error for failing to provide mnemonic support or simplify the workflow.

If an alarm is so frequent that users end up ignoring it out of habit and therefore fail to respond to a truly critical alarm, that is not user negligence for failing to pay attention; it is a design flaw for failing to properly calibrate the system's sensitivity. The burden of responsibility should not fall on the user to compensate for the product's shortcomings; it is the product that must adapt to human capabilities.

Fatigue is another critical factor that turns seemingly individual errors into predictable systemic design failures. A product that functions acceptably with rested users under laboratory conditions can become dangerously unsafe with fatigued users on night shifts. Given that chronic fatigue is endemic in modern healthcare settings—with shifts of twelve hours or more, nights without adequate sleep, and high emotional and physical workloads—designers must assume as a design condition that their users will frequently be fatigued and explicitly design accordingly.

This means simpler interfaces with fewer simultaneous options, mandatory confirmations for critical irreversible actions, and interface elements large enough and high-contrast enough so that they do not require exquisite visual attention and workflows that minimize the cognitive load during times of stress. A design that works only with perfectly rested and attentive users is a design that will inevitably fail in real clinical practice, where fatigue is the norm, not the exception.

The user's expected prior knowledge is another determining factor in this classification. A physician who is an expert in their clinical specialty may have no significant knowledge of computing or touch interfaces. An elderly patient may have no medical or technological knowledge whatsoever. Assuming that the user possesses certain technical knowledge that they do not actually have is a predictable recipe for user error. That is why the design must be rigorously based on the actual, documented profile of the intended users, not on an idealized profile imagined by engineers. It must use language they can understand without a dictionary, follow workflows that feel natural given their work habits, and provide contextual help when necessary without being condescending.

User training can complement good design, but it can never completely replace an intuitive and self-explanatory design. If a medical product requires a training course lasting several hours to use it without making serious errors, the fundamental problem is likely not a lack of user training but rather the unnecessary complexity of the design that does not respect human cognitive limitations.

5.2.2 LEGAL IMPLICATIONS, LEARNING FROM ERRORS, AND FORESEEABLE USE

The distinction between an error of use and a user error has significant legal and regulatory implications that manufacturers must understand. In many developed legal systems, if an incident is considered an isolated user error, legal liability falls primarily on the healthcare professional or the employing institution. If it is considered a systematic error of product use—liability may fall on the manufacturer for a design defect. That is why some manufacturers, in a short-sighted and ethically questionable manner, resist acknowledging that recurring errors with their products are design flaws, preferring instead attributing them to individual user inexperience or exceptional

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

circumstances. However, this defensive attitude, in addition to being ethically reprehensible because it puts patients at risk, is commercially short-sighted. If the design is not fundamentally improved, errors will continue to occur with different users in different places, and sooner or later, some independent researchers or regulators will statistically demonstrate that the error pattern is systematic and that the root cause lies in the product, not in the people who use it. At that point, the manufacturer's legal and reputational liability will multiply.

For development teams, the distinction between usage errors and user errors must be actively present in every phase of the design, becoming a standard lens for analysis. In formal testing with representative users, when someone makes an unexpected error, the question the team should ask is not "Why is this user so clumsy or distracted?" but "What is it about our current design that led to this predictable error?" This simple shift in perspective—from judging the user to analyzing the system—transforms usability testing from a superficial validation exercise into a genuine source of deep learning.

Every error identified during testing is a valuable opportunity to improve the design before it causes real harm, not a personal failing to be hidden or downplayed. Teams that consistently adopt this mindset of systemic analysis develop products that are inherently safer because they have been shaped and refined by the real errors of real users under controlled conditions, not by the theories of isolated designers.

A related and equally important concept is that of "reasonably foreseeable abnormal use," which the standard explicitly incorporates. The standard recognizes that real users sometimes use medical products in ways not explicitly anticipated by the manufacturer in its manuals. They may skip mandatory steps when in a hurry, use the product for purposes slightly different from those authorized, combine it with other devices in unauthorized but practical ways, or find undocumented shortcuts that save time. These behaviors, while not ideally anticipated, are statistically foreseeable given human nature and the pressures of the clinical environment, and the manufacturer must actively consider them in its analysis of risks of use. If it is foreseeable that a user, frequently in a hurry, will attempt to speed up a process by skipping safety confirmations, the design must technically prevent that from being possible, not merely warn against it. If it is foreseeable that a patient at home will manipulate the device in unauthorized ways out of ignorance or desperation, the design must be robust against such foreseeable manipulation. Foreseeable abnormal use is not a legal excuse for the manufacturer; it is part of the context of use that must be considered and managed as a risk.

Documentation of usage errors identified during development and the measures taken to prevent them is a mandatory part of the usability dossier required by IEC 62366. It must include a comprehensive list of potential errors identified through systematic analysis, their estimated severity for the patient, their probability of occurrence under real-world conditions, and the design control measures implemented for each. It must also include the results of empirical user testing, showing which errors actually occurred during the test sessions and how they were resolved through design iterations. This documentation is valuable not only for the regulators who will audit the product but also for the development team itself because it constitutes an organizational record of the decisions made and the lessons learned in the process. In long-term projects spanning years, with inevitable staff changes, this documented record prevents the same design errors from being repeated repeatedly with each new generation of developers.

For physicians and other healthcare professionals collaborating on these design projects, understanding the distinction between usage errors and user errors helps them participate more effectively and constructively in usability testing and the early identification of problems. Drawing on their accumulated experience, they can identify which errors are likely to occur in real-world practice even if they do not occur during controlled laboratory testing. From their clinical perspective, they can help distinguish between what is an occasional error attributable to a specific user and specific circumstances and what is a systematic pattern attributable to design flaws that will recur. Their perspective is essential for usability analysis to be realistic and not limited to what happens in the laboratory under artificial conditions. A physician who has spent years on night shifts can anticipate usage errors that a young designer would never imagine.

One aspect that often generates technical and ethical controversy is the dynamic balance between functional flexibility and patient safety. A highly restrictive design, which prevents any deviation from the intended workflow, can be very safe but very frustrating for expert users who need efficiency and adaptability in unforeseen situations. By allowing too many options, a highly flexible design can be efficient and adaptable but also prone to usage errors. There is no single universal answer; it depends on the specific clinical context, the intended user type, and the criticality of the tasks being performed. What the standard requires is that this tension be explicitly

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

addressed in the design process, that decisions be informed by evidence and that the reasons for those decisions be documented for future reference. Usability is not about maximizing a single variable but about finding the right balance for each specific usage situation.

Usage errors are not limited to those made by the clinical end-user or patient. They can also be made—with equally serious consequences—by those who install the software, maintain the hardware, or even clean the product in the hospital setting. An installation technician who misconfigures software because the instructions are confusing or incomplete; a maintenance professional who inadvertently disables a safety alarm because they do not know how to temporarily silence it correctly; and a cleaning assistant who wets a sensitive device because it is not clearly marked as vulnerable to water. All of them are users in the broadest sense of the term, and their errors must also be systematically considered in usability analysis and risk management. Patient safety is a chain of responsibilities that extends throughout the entire product lifecycle, and it is only as strong as its weakest link, which can be at any point in that chain.

5.3 EVALUATION WHILE WEARING GLOVES, SLEEPDEPRIVED, AND WITH INTERRUPTIONS

Usability evaluation in controlled laboratory settings produces clean, orderly, and deceptively optimistic results. When a designer observes a physician interacting with a prototype in a quiet room, free from auditory distractions, with ample time and the user well-rested and attentive, they inevitably obtain an idealized picture of human behavior—which is rarely replicated in reality. That picture is useful for identifying obvious, serious design problems, but it does not reveal how the product will behave under the real-world conditions in which it will be used—which are exactly the opposite: chaotic, noisy, interrupted, and exhausting. Real-world clinical practice is characterized by constant interruptions that fragment attention, accumulated fatigue that clouds judgment, physical limitations such as latex gloves or poor nighttime lighting, and constant time pressure that leads to cognitive shortcuts. Evaluating under these adverse conditions is not just one methodological option among many; it is an absolute necessity for patient safety.

IEC 62366 explicitly requires that usability evaluation be conducted under conditions that adequately represent the intended real-world usage environment, not just under ideal laboratory conditions. This means that formal laboratory testing with rested, attentive, and motivated users is not sufficient. Test scenarios must be deliberately designed to incorporate the stressors typical of the clinical environment: constant background noise, scheduled interruptions, forced multitasking, induced or natural fatigue, and real physical limitations. Only in this way can we discover how the product will actually behave when it is in the hands of those who truly need it in their daily work. A product that works perfectly in the laboratory but consistently fails in clinical practice is a product that has been poorly evaluated by design, no matter how rigorous the formal testing may have appeared to be.

5.3.1 PHYSICAL LIMITATIONS AND FATIGUE: THE BODY IN THE CLINICAL SETTING

Evaluation while wearing gloves is one of the most concrete, revealing, and frequently overlooked aspects of real-world clinical usability. In many hospital settings, professionals are required to wear latex or nitrile gloves, which significantly reduce tactile sensitivity and hinder the precise manipulation of small interfaces. A touch interface that works perfectly with a bare, dry finger can become unmanageable or dangerous with wet gloves or standard-thickness gloves. Buttons that are too small on the screen, sliders that require pinpoint precision, and complex multitouch gestures can become frequent sources of error when the user does not feel the contact well or when the glove rubs against the surface. Systematic evaluation with gloves is not a methodological luxury; it is a necessary condition for any product intended for use in environments where protecting staff from infection is mandatory and nonnegotiable.

Fatigue is another critical factor that is rarely genuinely incorporated into conventional usability evaluations, even though its impact is devastating. Healthcare professionals work long shifts of twelve hours or more, often at night, with accumulated hours of wakefulness that objectively affect their cognitive and motor abilities. A fatigued person has slower reaction times, makes more errors of omission because of carelessness, tends to rely on cognitive automatisms rather than carefully processing new information, and has reduced divided attention capacity. A product that requires constant sustained attention, presents information in a complex or dense manner, or demands critical decisions at moments of peak fatigue will be particularly prone to

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

usage errors when used under real-world night shift conditions. Testing with genuinely fatigued users, even if logistically difficult to organize, provides vital information that cannot be obtained through any other form of evaluation. Some teams even induce controlled fatigue during test sessions to simulate these effects.

5.3.2 INTERRUPTIONS, MULTITASKING, AND TIME PRESSURES

Interruptions are the absolute norm in contemporary clinical practice, not the occasional exception that designers would prefer to ignore. Doctors may be interrupted dozens of times during a consultation or procedure by calls, alarms, or questions from colleagues or family members. A nurse may have to handle multiple tasks simultaneously in different rooms, switching from one to another without being able to complete any of them from start to finish because of the urgency of the demands. A software product that is not explicitly designed to handle interruptions—one that loses its task state when the user is briefly away or that forces users to restart processes from the beginning without saving partial progress—becomes a source of intense frustration and systematic error. Usability evaluation must include scenarios with scheduled interruptions, where the user has to resume incomplete tasks after a distraction, remember exactly where they left off, and retrieve information they may have lost from working memory. Only then can the true robustness of the design be evaluated in the face of fragmented clinical reality.

Multitasking is a natural and constant extension of interruptions in healthcare work. Professionals rarely perform a single task from start to finish without deviations; rather, they simultaneously manage a set of tasks that overlap in time, switching between them according to the changing priorities of the moment. A product that assumes an ideal linear workflow, with sequential steps that must be completed strictly in order without deviations, clashes violently with this cognitive and operational reality. The evaluation must include scenarios where the user must explicitly manage multiple concurrent tasks, switch between them frequently, and maintain mental coherence for each without becoming confused. This reveals whether the design supports the inherent cognitive complexity of real-world practice or, conversely, adds an additional burden that increases the risk of error due to attentional overload.

Compared with relaxed laboratory conditions, real-world time pressure is another factor that profoundly distorts human behavior. When time is short because of clinical urgency, users skip verification steps, ignore information they consider secondary, and make quick decisions on the basis of recognition heuristics rather than systematic analysis. A product that appears safe and welldesigned under calm laboratory conditions can become dangerously unsafe when used under real-world time pressure. Safety confirmations that, in theory, should prevent critical errors may be accepted mechanically without actually reading the content, alarms may be silenced quickly without processing their meaning,and double-checking procedures may be skipped because of overconfidence in the technology.

The evaluation must include scenarios with realistic and demanding time limits, forcing users to prioritize quickly and take cognitive shortcuts. This reveals which parts of the design are robust under time pressure and which become sources of risk when the user cannot devote the attention the designer assumes.

5.3.3 PHYSICAL ENVIRONMENT, HUMAN DIVERSITY, AND INVESTMENT IN SAFETY

Poor lighting in patient rooms at night to avoid disturbing rest, intense ambient noise from multiple devices that masks important alarms, limited physical space around the bed that hinders equipment handling, and extreme temperatures in certain environments such as operating rooms or burn units. A product that has not been systematically evaluated under these adverse conditions may fail precisely when its proper function is most needed. The evaluation must reproduce, as far as possible, these real-world environmental conditions or at least explicitly consider how each affects the specific use of the product. An audible alarm that is clearly audible in a quiet laboratory may be completely inaudible in a noisy ICU; a display that is legible under controlled ambient lighting may be illegible in direct sunlight or under glare from surgical lamps.

The actual physical diversity of users is another factor often overlooked in standard evaluations, limiting the generalizability of the results. Not all users have the same visual acuity, the same fine motor skills, the same hearing ability, or the same grip strength. Healthcare professionals, like any human population, include people with visual impairments who wear glasses or contact lenses,

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

people with essential tremors or mild Parkinson's disease, and people with hearing problems caused by occupational factors or aging. A product that assumes the "standard" sensory and motor abilities of a young, healthy user effectively excludes a significant portion of its potential users and, worse still from a safety perspective, can lead to systematic errors when those users with limitations are forced to use it without adaptations. Evaluation must deliberately include users with diverse functional abilities or at least explicitly consider, through task analysis, how common limitations in the population affect the safe use of the product.

Cultural and linguistic diversity also matters significantly in globalized and multicultural healthcare settings. Iconographic symbols that are intuitive in one culture may be confusing or even offensive in another. Colors that have clear and consistent meanings in one cultural context may have different or contradictory meanings in another. Interface metaphors that work perfectly in one language may not translate well or may create semantically false friends in another. A product intended for use in multicultural settings—as is often the case in modern healthcare—must be evaluated with users from diverse cultural and linguistic backgrounds. This does not necessarily mean designing radically different versions for each culture, but it does mean ensuring that fundamental navigation and safety elements are universally understandable and that there are no cultural biases in the design that could lead to critical misinterpretation.

Specific evaluation with patients, not just with healthcare professionals, adds another layer of complexity and necessity. Patients who use medical devices at home have very different characteristics from trained professionals do. They may have cognitive limitations associated with their underlying condition, age-related vision or hearing problems, or motor difficulties due to neurological conditions. They may be anxious about their diagnosis, frightened by the procedure, or suffer from pain that distracts their attention. They may have no technical or medical training whatsoever and rely blindly on what the device tells them. A product that is easy for a trained professional to use may be completely incomprehensible or dangerous for a vulnerable patient.

For development teams, organizing evaluations under these realistic conditions is undoubtedly complex and costly and consumes valuable resources. It requires recruiting users with the appropriate characteristics who are willing to participate under adverse conditions, replicating physical environments that simulate clinical reality with a certain degree of fidelity, and designing test scenarios that convincingly incorporate relevant stressors. However, the alternative is worse in every way: launching a product onto the market that has not been tested under real-world conditions and discovering, through clinical incidents resulting in patient harm, that it fails precisely when its reliability is most needed. The economic and reputational cost of a comprehensive realistic evaluation is always lower than the human and legal cost of a subsequent clinical failure. Investing in realistic evaluations from the design stage is, ultimately, investing in patient safety and business sustainability.

The documentation of these evaluations must reflect not only the quantitative results but also the specific conditions under which they were conducted, with a level of detail that allows for an assessment of the ecological validity of the findings. A report that generally states "it was tested with ten physicians and there were no significant errors" is insufficient and potentially misleading if it does not explicitly specify whether those physicians were well-rested or fatigued, whether they were wearing gloves or not, whether there were scheduled interruptions during the tasks, or whether they were under actual time pressure. To assess their predictive validity for real-world use, competent regulators, and above all the development teams themselves who must learn from experience, need to know the exact conditions under which those results were obtained.

With respect to the physicians and healthcare professionals who participate in these realistic evaluations, the experience can be profoundly revealed at both the personal and professional levels. By seeing themselves make usage errors under time pressure, with induced fatigue, or with constant interruptions, they understand firsthand that these errors are not personal moral failures of carelessness but are predictable consequences of a design that does not adequately account for real-world working conditions. This experiential understanding is valuable not only for improving the specific product they are evaluating but also for changing the organizational culture of individual blame that often prevails in healthcare institutions when incidents occur. When an error is understood as a system or product design problem, the sought-after solution is not to blame and punish the person who acted but to improve the system so that the error is less likely to occur in the future for everyone.

The direct relationship between realistic usability evaluation and patient safety is undeniable and quantifiable. The more closely evaluation conditions resemble actual clinical use conditions, the more likely design problems are to be detected before they cause real harm to people. An

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

error that occurs during a controlled evaluation is a valuable opportunity for design improvement; an error that occurs in actual clinical practice is an adverse event with potentially serious consequences for health and life. Investing resources in comprehensive, realistic evaluations during development is, therefore, a direct investment in preventive safety. It is not an unnecessary operational expense; it is one of the most cost-effective investments a serious medical software manufacturer can make because it prevents costs associated with incidents, product recalls, litigation, and, above all, harm to people who trusted the product.

5.4 COGNITIVE LOAD AS A DESIGN METRIC

Imagine that a primary care physician must enter a patient's data into an electronic health record system during a fifteen-minute consultation. The screen displays dozens of input fields, some mandatory and marked in red, others optional and of dubious utility, spread across multiple tabs that must be navigated sequentially. The doctor must simultaneously recall what relevant information was requested during the clinical interview, search for it in their hastily written handwritten notes, identify which specific field each piece of data goes into, and meticulously ensure that they do not skip any mandatory steps that would block the entry. Moreover, the patient waits seated, starting at the doctor's back, the clock ticks relentlessly toward the end of the appointment, and the waiting list in the hallway grows with other impatient patients. At the end of a twelve-hour shift, the doctor has spent hours on this bureaucratic task and leaves work mentally exhausted, with the frustrating feeling of having spent more time struggling with the computer than actually listening to and caring for their patients. This experience, which is unfortunately common in modern healthcare systems, is a tangible manifestation of a concept that medical software designers should know and deeply respect: cognitive load.

Cognitive load refers to the amount of mental effort required to perform a specific task using a given interface design. It is not an abstract or metaphorical concept but a physiological reality backed by neuroscience: our brain has a limited capacity to process information simultaneously, especially when it must integrate new information with prior knowledge under pressure. When that capacity is exceeded, cognitive performance decreases, errors of omission and commission increase exponentially, and mental fatigue builds up rapidly.

In clinical settings, where decisions have serious consequences for people's lives and interruptions are constant, managing a professional's cognitive load is not a secondary matter of comfort or job satisfaction but a fundamental aspect of patient safety. A medical product that imposes an excessive cognitive load on its users is not merely annoying or inefficient; it is actively dangerous because it predisposes users to errors at critical moments.

5.4.1 THE THREE TYPES OF LOAD AND THE PRINCIPLES OF REDUCTION

The theory of cognitive load, developed by the educational psychologist John Sweller and subsequently applied to interface design, distinguishes three fundamental types of load that designers must understand and manage. Intrinsic load is that which is inevitably imposed by the complex task itself—that, by its nature, requires mental processing and cannot be eliminated without changing the task. A diagnosis of a rare disease with contradictory symptoms has a high intrinsic load because it requires the integration of a great deal of contradictory medical information; we cannot reduce this load without artificially simplifying the diagnosis. The relevant or germane load is that imposed by the way the task is presented but which helps to perform it better or learn from it. A well-designed flowchart can increase the relevant load because its elements must be processed visually, but in return, it greatly facilitates understanding of the diagnostic algorithm. The irrelevant or extraneous load is imposed by the design without providing any functional benefit, such as having to search for critical data in a deep and confusing menu or remembering a numerical code that should be visible on the screen.

The goal of good medical software design is to drastically minimize extraneous load, optimize relevant load to facilitate learning and accuracy, and scrupulously respect the limits of the intrinsic load without adding unnecessary weight.

For medical software designers, understanding and applying cognitive load theory involves continually asking themselves throughout the design process: What information does the user have to remember at this exact moment in the task? What critical information do they need to keep in mind simultaneously to make a safe decision? What visual, or functional, distractions are

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

we imposing on them that add no value? Are we making efficient use of their limited processing capacity, or are we squandering it on mechanical tasks with no clinical value? These questions, applied systematically in every design review, reveal opportunities for substantial improvement that often go unnoticed in traditional functional evaluations, which verify only whether the system "works" technically.

One of the fundamental principles for reducing the cognitive load in medical interfaces is recognition over recall. It is neuroscientifically much easier and faster for the human brain to recognize something it is already presented with than to actively recall it from working memory. That is why interfaces that explicitly display available options are cognitively superior to those that force the user to type commands from memory or recall arbitrary numerical codes.

A physician should not have to memorize the ICD-10 codes for the most common diagnoses in their specialty; they should be able to select them from a list wellorganized by clinical categories with a predictive search. An elderly patient should not have to remember exactly when they took their last dose of medication; the app should clearly display this information on the main screen. This principle, applied consistently throughout the design, drastically reduces the cognitive load and associated memory errors.

Another essential principle is structural and visual consistency. When similar elements behave similarly throughout the system, the user can efficiently transfer their learned knowledge from one part to another without having to relearn new rules for each context. Buttons that activate similar functions should have consistent colors and be in similar positions on the screen. Workflows for analogous clinical tasks should follow analogous navigation structures. Consistency reduces the cognitive load because the brain can automatically apply learned patterns rather than consciously processing each situation as entirely new. In contrast, arbitrary inconsistency forces the user to remain constantly alert to exceptions, which quickly leads to fatigue and increases the likelihood of errors due to incorrect automatism.

5.4.2 VISUAL HIERARCHY, FRAGMENTATION, AND APPROPRIATE METAPHORS

Visual hierarchy is another powerful tool for managing cognitive load in environments where attention is a scarce resource. Not all presented information has the same clinical importance for patient safety and therefore should not compete for the professional's attention with the same visual emphasis. Data critical to immediate safety, such as laboratory values outside dangerous ranges, should stand out visually through size, color, and position, capturing the user's attention even under conditions of fatigue or partial distraction. Secondary supporting data should be available but should not compete for primary attention when more subtle visual coding is used. Data truly irrelevant to the current task should be omitted from the main view or relegated to deep levels of the interface accessible only on demand. A well-designed visual hierarchy guides the user's gaze toward what is important in a preattentive manner, reducing conscious search effort and the risk of overlooking critical information due to information overload.

Intelligent fragmentation of information is another crucial aspect of cognitive load management. Presenting all available information about a patient on a single overwhelming screen overwhelms the user and forces them to perform manual filtering that the design should have done for them through prioritization. Dividing the information into successive screens, each with a clear purpose and a manageable amount of data relevant to that specific purpose, reduces the cognitive load through simplification. However, this principle must be applied with care and moderation: too many successive screens can be just as bad as a single cluttered screen because they force constant navigation and active recall of what was seen on previous screens, increasing the working memory load. The optimal balance lies in grouping information logically, following the natural flow of the clinical task, and always providing a visible navigation "map" that allows the user to know exactly where they are in the process and where they are going.

The use of familiar and consistent metaphors also reduces cognitive load by enabling the transfer of prior knowledge. When a digital interface behaves in a way analogous to something the user already knows from the physical world or other systems, they can apply that existing knowledge without having to learn entirely new associations from scratch. Icons that represent actions through universally recognizable images, approval workflows that mimic traditional paper-based processes, and terms that use the everyday language of the medical profession—all of these facilitate the transfer of prior knowledge and accelerate learning. However, this principle must be applied with critical care: metaphors that do not align well with the software's actual functionality can be more confusing than helpful. A digital trash can that does not allow you

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

to recover deleted items, a shopping cart that does not display the running total—these are metaphors that mislead and frustrate more than they help.

Immediate and clear feedback is essential for reducing the cognitive load of uncertainty. When the user performs an action on the system, the system should immediately confirm that the action has been received and display the result. Without this feedback, the user wastes valuable cognitive resources, wondering if they did the right thing, if they should repeat the action, or if the system has frozen or is still processing. This active uncertainty affects working memory, whichshould be devoted to the primary clinical task. A simple progress animation, a brief confirmation message, or a visible status change in the interacting element are sufficient to close the action-feedback loop and relieve the user of that burden of uncertainty.

5.4.3 ERROR PREVENTION, MEASUREMENT, AND SYSTEMIC SUSTAINABILITY

Error prevention through proactive design is among the most effective ways to reduce a system's overall cognitive load. When the system technically prevents the user from making certain common mistakes, it spares them from having to constantly be on guard against those possibilities, freeing up mental resources for other tasks. Input fields that only accept numbers within a valid range, buttons that are visually disabled when prerequisites are not met, and explicit confirmations for critical irreversible actions—all of these act as a safety net that allows the user to focus on what truly matters clinically. However, this principle must be carefully balanced: too many paternalistic restrictions can frustrate and slow things down, especially for expert users who need efficiency and flexibility in unforeseen situations. The design must strike the right balance where error prevention does not become an insurmountable barrier to the necessary clinical efficiency.

For development teams, incorporating cognitive load as an explicit design metric requires specific evaluation methods that go beyond functionality. There are validated standardized questionnaires, such as the NASA-TLX (task load index), that allow users to subjectively assess their perceived cognitive load across different dimensions: mental demand, physical demand, time demand, overall effort, frustration, and perceived performance.(2) There are ethnographic observation techniques that identify moments of uncertainty through unexpected long pauses, facial expressions of frustration, or frequent consultations with colleagues.

There are even objective physiological measures, such as pupil dilation or heart rate variability, which can indicate high levels of cognitive effort in real time during interactions. None of these techniques are perfect on their own, but when combined, they provide a rich, multidimensional picture of how a specific design affects users' mental load in practice.

The greater the cognitive load imposed by the design, the greater the statistical probability of human error due to attentional overload. That is why agencies such as the FDA and the EMA are paying increasing attention to this aspect in required usability evaluations. IEC 62366, "(3)", although it does not explicitly mention the term "cognitive load" in its text, requires a usability engineering process that, when properly applied with attention to these principles, inevitably leads to its reduction. Manufacturers who deliberately ignore this aspect expose themselves not only to less competitive products in the market but also to clinical incidents that could have been avoided with a design more respectful of the limits of the human mind.

For physicians and healthcare professionals who use these products daily, excessive cognitive load manifests tangibly in the form of accumulated mental fatigue, chronic frustration, and, in tragic instances, clinical errors with consequences. When a system forces them to remember too much arbitrary information, to search for critical information poorly organized in deep structures, or to navigate mazes of confusing screens, they are paying a cognitive price they should not have to pay—one that distracts them from their true role of care. That price translates into less quality time for patients, more accumulated work-related stress, and a higher risk of errors during times of peak pressure. Healthcare professionals should be the first to systematically demand that the systems they use respect the limits of their cognitive capacity, and they should actively participate in evaluations that measure this specific aspect by contributing their real-world usage experience.

Finally, excessive cognitive load is not merely an individual problem for the professional experiencing it but also a systemic problem that affects the sustainability of the entire healthcare system. A technological system that chronically imposes a high cognitive load on its users not only affects their immediate performance but also actively contributes to professional burnout, burnout syndrome, and eventually the decision to leave the healthcare profession due to exhaustion. In times of healthcare staffing crises in many developed countries, where the demand for care is growing and the supply of professionals is stagnating, designing technological tools that respect

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

and care for the minds of those who care for others is not just a matter of product quality but of the sustainability of the public health system. Every conscious effort to reduce unnecessary cognitive load is an investment in the mental health of professionals and, through their well-being and continued presence in the profession, in the health of all potential patients.

Once we have learned to design with users, their limitations, and the real-world context of use in mind, how do we integrate all this knowledge into a systematic process that can be audited and continuously improved?

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

REFERENCES

What Is IEC 62366? Usability Engineering Guide [Internet]. [cited May 13, 2026]. Available from: https://www.jamasoftware.com/requirements-management-guide/medical-devices/iec-62366/
NASA Task Load Index (TLX) - NASA [Internet]. [cited May 13, 2026]. Available from: https://www.nasa.gov/human-systems-integration-division/nasa-task-load-index-tlx/
Applying Human Factors and Usability Engineering to Medical Devices | FDA [Internet]. [cited May 13, 2026]. Available at: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/applying-human-factors-and-usability-engineering-medical-devices

https://doi.org/10.62486/978-9915-9876-8-2.Ch05

Chapter 6 / Capítulo 6

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch06
Pages: 95-110

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 6. Clinical Terminologies and Languages

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Data without context is just a number; data with the wrong meaning *are* worse than no data at all. When a laboratory sends a result and the system interprets it as something it is not, when one medical record mentions 'fever' and another 'pyrexia,' when an algorithm interprets 'heart attack' where the doctor wrote 'suspected heart attack,' communication breaks down, and with it the possibility of providing safe and coordinated care."

Let's imagine for a moment that two doctors from different countries are trying to discuss a patient. One speaks Spanish, and the other speaks Mandarin. Neither understands the other's language, but both need to share critical information to save a life. The situation is desperate and almost comical if it is not so serious. Now, let's imagine that this same situation occurs daily, not between people but between computer systems. A laboratory sends a result labeled "glucose" to an electronic health record system that expects "GLU." A pharmacy records a prescription for "paracetamol" that the dispensing system interprets as "acetaminophen." An artificial intelligence algorithm trained on data from a Boston hospital receives data from a Madrid hospital and does not understand a thing because the coding systems are different. This is not a hypothetical scenario; it is the everyday reality of healthcare interoperability, and behind each of these miscommunications are patients receiving subpar care, doctors wasting time, and missed opportunities for improvement.

The solution to this problem is not technical but semantic. It is not enough for systems to be able to exchange data; they must exchange meaning. They must speak the same language or at least have a common dictionary that allows them to translate from one to the other. This is exactly what clinical terminologies are about: bringing order to the chaos of language, ensuring that when one system says "hypertension," all the others understand exactly the same thing. It may seem like a minor detail, a matter of vocabulary of little importance, but it is actually one of the pillars upon which digital health is built. Without common terminology, there is no real interoperability; without interoperability, there is no integrated healthcare system; and without integration, technology fails to deliver on its promise of improving care.

In this chapter, we explore the fascinating world of clinical terminologies. We will see why it is so important that data mean the same thing across all systems, what problems arise when it does not, and how the major international terminologies address these challenges. The goal is not to turn the reader into an expert coder but to provide the necessary awareness to understand why this aspect—often overlooked in development projects—is actually critical for safety and quality. Because medical software that does not speak the same language as the rest of the system is not a solution, it is an additional problem.

6.1 WHY DATA MUST MEAN THE SAME THING ON BOTH SIDES

There is a natural tendency, especially among those who do not work directly with clinical information, to believe that data are objective and universal, that a number is simply a number and a word is a word and that their meaning is self-evident without the need for interpretation. This naive belief, which is already problematic in everyday life, becomes directly dangerous to patient safety in the healthcare setting. Clinical data are not abstract entities with universal and immutable meaning but rather complex human constructs laden with context, cultural nuances, implicit assumptions, and coding decisions.

When an experienced physician writes "fever" in an electronic health record, they are not simply recording an objective, neutral fact but actively interpreting a multifaceted clinical reality: they have previously measured a specific body temperature with a thermometer, considered the patient's full context—including their age and baseline conditions—decided that this elevated temperature warrants being classified as "fever" rather than "hyperthermia" or simply "elevated temperature," and they choose that specific term among other possible alternatives. When a different doctor reads that same record days later, they make a similar but not identical interpretation on the basis of their own clinical experience, their particular medical training, and the context of reading. This process of encoding and decoding, which works reasonably well among humans owing to shared medical education, a common clinical context, and the ability to ask for clarification, breaks down critically when automated computer systems intervene.

For a computer system lacking true semantic understanding, "fever" is merely a string of ASCII characters with no inherent meaning. It does not automatically associate a numerical range of body temperatures; it does not conceptually distinguish between a mild fever of 37.5°C and a high fever of 40°C; it does not know that it may be due to a bacterial infection or environmental

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

heatstroke; it cannot infer the therapeutic implications of that condition. If a different system receives that same text "fever" via data exchange, it stores it as-is in its database but cannot do anything intelligent or automated with that information: it cannot add it to comparable epidemiological statistics, it cannot generate alerts on the basis of temperature thresholds, and it cannot automatically correlate it with other clinical data from the patient to detect patterns.

For computer systems to truly do something useful with clinical information beyond simple storage, the data must be encoded in a standardized manner using formal medical terminologies that assign precise, structured, and shared meanings to each clinical term.

6.1.1 THE FIVE FUNDAMENTAL REASONS FOR SEMANTIC INTEROPERABILITY

The first fundamental reason why data must mean exactly the same thing across all systems that exchange information is direct patient safety. A classic and tragically common example is documented drug allergies. If a primary care system records that the patient is allergic to "sulfamides" using that colloquial term and another hospital system, upon receiving that information via electronic exchange, interprets it as the specific chemical group "sulfonamides", or vice versa, the receiving system may fail to recognize through automatic matching that both terms conceptually refer to the same allergy and, therefore, may not generate the corresponding critical alert when a physician inadvertently prescribes a medication from that group. The result can be a severe, potentially anaphylactic allergic reaction that could have been completely avoided with standardized coding that eliminates terminological ambiguity.

Real patients do not receive medical care at a single, isolated healthcare facility but rather necessarily move between primary care, specialist clinics, emergency departments, acute hospitalization, posthospital rehabilitation, and often between different autonomous communities or even countries with distinct healthcare systems. For relevant clinical information to follow them reliably—such that the night shift emergency physician knows what treatments they have recently received and so that a specialist in a different city is aware of the results of previous diagnostic tests performed at another facility—computer systems must be able to "understand" one another semantically.

When population-based epidemiological studies are being conducted, when it is necessary to know precisely how many patients have a specific disease in a region or when the comparative effectiveness of different treatments is being evaluated through analyses of large databases, clinical data must be semantically comparable and aggregatable. If every hospital or system codes diagnoses in its own particular way, using its own nonstandardized local abbreviations and varying levels of granularity in the description, the results of any analysis will be inconsistent and biased, and the conclusions will potentially be erroneous or dangerous. Large healthcare databases that promise to revolutionize research through big data analysis are useless or even counterproductive if the data they contain do not speak the same precise semantic language. It is like trying to perform rigorous statistics by mixing responses in different natural languages without a translator or a common dictionary.

Doctors currently spend a large amount of their working time entering data into computer systems, and an equally considerable amount of interpreting data are entered by other colleagues who use different formats or terminologies. If the systems spoke the same standardized semantic language, much of that double-entry and reinterpretation work could be automated or eliminated. A laboratory result correctly coded with standard terminology can automatically populate timeseries charts, generate smart alerts when it falls outside safe ranges, and be incorporated into discharge reports without the need for manual rewriting. A precisely coded medication prescription can be automatically checked against known drug interactions, maximum recommended doses, and specific contraindications for that patient. However, for this intelligent automation to be possible, the data must be semantically interpretable by machines, not merely readable by humans who spend time decoding it.

AI models are primarily trained on historical data, and the semantic quality of that data directly determines the quality, reliability, and safety of the resulting models. If the training data are inconsistent, with ambiguous labels and variable coding that conceptually represents different things, the algorithms will learn erroneous patterns, artifacts, or—worse still—systematic biases that do not correspond to clinical reality. An algorithm trained to predict early sepsis that does not semantically distinguish between "fever" as a subjectively reported symptom and "elevated body temperature" measured objectively is useless or dangerous. An algorithm that learns from data where "myocardial infarction" means different things in different hospitals depending on their

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

local coding criteria will be unpredictable in its behavior. Rigorous semantic standardization is an absolute prerequisite for reliable and ethically acceptable medical artificial intelligence.

6.1.2 EXAMPLES OF AMBIGUITY AND TERMINOLOGIES AS A SOLUTION

To understand the practical magnitude of the problem of lack of standardization, it is helpful to examine real and common examples of terminological ambiguity in everyday clinical practice. A single disease can have multiple colloquial and formal names: "hypertension," "HTN," "high blood pressure," "high blood pressure," and "systemic hypertension." A single drug may have different names depending on the country or tradition: "paracetamol" in Europe, "acetaminophen" in the Americas, and "APAP" in some technical contexts. The same surgical procedure can be described with different levels of clinical detail: "heart surgery" (very generic), "coronary bypass" (more specific), and "coronary artery myocardial revascularization" (technically precise). When computer systems handle these natural variations in medical language without a formal mechanism to semantically unify them, information becomes irreparably fragmented and loses its added value for coordinated patient care.

The practical consequences of this lack of semantic standardization are evident daily in hospitals and healthcare centers around the world. Physicians who lack important test results because the originating laboratory used a code different from what the receiving system expected. Nurses who administer the wrong medication because the electronic prescription used a brand name they did not recognize or that the system did not correctly link to the active ingredient. Researchers who spend months on tedious work "cleaning" data and manually correcting inconsistencies before they can actually begin an epidemiological study. Patients who have to repeat invasive diagnostic tests because the results from other centers cannot be automatically interpreted by the current system. All this massive waste of professional time, healthcare resources, and opportunities for improvement has a single, identifiable root cause: we have not yet managed to obtain healthcare IT systems to speak the same precise semantic language.

The solution to this problem necessarily involves adopting standardized medical terminologies, rigorously developed by international organizations with the participation of clinical experts from around the world. These formal terminologies are not simple flat lists of approved terms but complex ontological structures that organize medical knowledge systematically, with logical hierarchical relationships (hypernymy and hyponymy), documented synonyms, precise definitions of each concept, and unique, unambiguous codes for each clinical entity. When a computer system uses these standard terminologies, each piece of clinical data is associated not only with the readable text that the physician sees on the screen but also with a formal code that machines can interpret unambiguously in any other system that shares the same terminology. This code allows data to travel between heterogeneous systems while preserving its essential clinical meaning intact.

For medical software developers, understanding the critical importance of standardized terminologies is essential from the initial architectural design phase. This is not a minor technical detail that can be postponed for later, but a fundamental decision that determines the product's future interoperability, its ability to integrate with other systems in the healthcare ecosystem, and the long-term utility of the generated data for research and quality improvement. Choosing one standard terminology over another—or worse, ignoring this aspect and using only unstructured free text—determines whether the product will be an isolated island of information or a connected node in the healthcare network. Deliberately ignoring this aspect means building a product that, no matter how technically sound it may be in its isolated functions, will exist disconnected from the broader healthcare ecosystem—like an island unable to communicate efficiently with the mainland.

The massive transition to semantically standardized systems is neither easy nor quick. It requires significant financial investment, extensive training of clinical and technical staff, profound changes to established workflows, and institutional coordination that often extends beyond the boundaries of a single organization. However, it is an investment with a guaranteed return in multiple dimensions. Every time clinical data can be reused without needing to be manually rewritten, every time a safety alert works correctly because the system semantically understood the documented allergy, every time a research study can begin without months of prior data cleansing, part of that initial investment is recouped. Above all, every time a patient receives safer, more coordinated care based on complete information because their data travel with them while retaining their meaning, the ultimate purpose of this entire collective effort is fulfilled: to truly

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

improve the health of the people who trust the healthcare system.

6.2 SNOMED-CT: THE SEMANTIC BACKBONE

Let us imagine that we want to build a universal library containing all of humanity's accumulated medical knowledge. Not just any traditional library but also one where every book, every article, and every piece of information is cataloged with pinpoint precision; where every medical concept has a specific place within an organized system; and where the logical relationships between ideas are clearly defined and can be navigated automatically. A library in which "pneumonia" was searched not only revealed all the documents mentioning that disease but also automatically displayed its hierarchical relationship to "respiratory infection," its associated clinical manifestations such as "fever" and "productive cough," its standard treatments such as "antibiotics," and any other relevant medical concept semantically connected. This ideal library of medical knowledge exists not as a physical building with bookshelves but as a computable clinical terminology called SNOMED-CT, and it is likely the most ambitious, comprehensive, and sophisticated tool ever created by humanity to organize medical knowledge in such a way that machines can process it intelligently and humans can benefit from that organization.

SNOMED-CT, which stands for Systematized Nomenclature of Medicine Clinical Terms, is a comprehensive, multilingual, open-domain clinical terminology that covers most of the semantic knowledge necessary for modern healthcare practice.(1) Developed and rigorously maintained by SNOMED International, a nonprofit organization that brings together dozens of member countries and thousands of clinical experts, this terminology currently contains more than 350,000 unique and active concepts, organized into logical hierarchies and linked to one another through dozens of types of formal semantic relationships.

It is not simply an extensive list of approved medical terms, as a traditional dictionary might be, but a complex ontological semantic network that faithfully reflects the deep structure of medical knowledge: how diseases relate to their symptoms and signs, diagnostic and therapeutic procedures to their indications and contraindications, drugs with their mechanisms of action and adverse effects, and laboratory findings with the pathological conditions they suggest.

6.2.1 UNIQUE CONCEPTS, MULTIPLE DESCRIPTIONS, AND HIERARCHICAL STRUCTURE

What makes SNOMED-CT truly special is not only its impressive size and coverage but also its fundamental conceptual approach that clearly distinguishes between meaning and its linguistic expression. Each entry in SNOMED-CT is not merely a term or a word but also an abstract medical concept with a unique and established clinical meaning. That abstract concept is assigned a unique 18-digit numerical code that identifies it unambiguously and uniquely worldwide, regardless of language or context. However, the same concept may have multiple associated linguistic descriptions that express it in different ways: the preferred term in British English, the preferred term in American English, valid alternative synonyms, common clinical abbreviations, and official translations into dozens of national languages.

Thus, when a computer system sends a SNOMED-CT code, it actually sends a precise clinical meaning, not simply a word in a particular language. The receiving system can display the same meaning to the clinician in their preferred local language, using the term most familiar in that cultural context, but the key point is that both systems know with certainty that they are referring to exactly the same clinical entity, not ambiguous interpretations.

The hierarchical structure of SNOMED-CT allows for systematic navigation from the most general to the most specific, or vice versa, in a logical and computable manner. All the concepts are organized into 19 main hierarchies on the basis of "is-a" taxonomic relationships. For example, the specific concept "bacterial pneumonia caused by *Streptococcus pneumoniae*" is a more specific type of "bacterial pneumonia," which in turn is a type of "pneumonia" (inflammation of the lung parenchyma), which is a type of "infectious disease of the lower respiratory tract," which is a type of "infectious disease," which is ultimately a type of "disease" or pathological condition.

This formal hierarchical structure allows computer systems to automatically make useful logical inferences: if a patient has a specific bacterial pneumonia documented, the system can automatically deduce that they have pneumonia in general, that they have an infectious disease and that they have a disease requiring treatment. This is extraordinarily useful for information retrieval in searches ("find all patients with any type of respiratory infection"), for epidemiological

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

data aggregation ("how many cases of infectious disease did we have this month"), and for generating rule-based clinical alerts ("alert if the patient has an infection and has not been prescribed antibiotic therapy").

However, SNOMED-CT goes far beyond simple taxonomic classification hierarchies. It also includes other types of semantic relationships that connect concepts from different hierarchies to one another, creating a rich and multidimensional network of medical knowledge. For example, the concept "pneumonia" can be semantically related to the concept "fever" via a formal "has associated clinical finding" relationship and to the concept "amoxicillin" via a "has drug treatment of choice" relationship. These attributive relationships allow for the representation of causal and associative medical knowledge in a much richer and more useful way than a simple classification taxonomy.

A system that uses SNOMED-CT in a sophisticated manner can answer complex clinical questions that span multiple domains: "Find all patients with an infectious disease of the respiratory tract presenting with high fever who have been treated with beta-lactam antibiotics in the past 30 days." Without a terminology featuring these formal semantic relationships, answering that question would require imprecise text searches in free-text notes or multiple separate queries to noninteroperable databases.

6.2.2 TECHNICAL INTEGRATION, CONTINUOUS UPDATES, AND DATA QUALITY

For medical software developers, SNOMED-CT offers an invaluable resource that avoids reinventing the semantic wheel. Instead of having to create and maintain their own lists of medical terms—with all the clinical consensus work and potential for systematic error that entails—they can use this terminology, which has already been developed, validated, and is continuously maintained by international experts across multiple specialties. This not only saves months or years of development work but also ensures that their products will be semantically interoperable with other systems that also use SNOMED-CT correctly. A code sent from their application will be understood unambiguously by any other system that adheres to the standard, regardless of the country of origin, the local implementation language, or the specific software vendor.

The technical integration of SNOMED-CT into a medical software application is not trivial given the complexity of the terminology, but there are tools and standards that considerably facilitate the task. There are open-source and commercial software libraries available for virtually all modern programming languages (Java, Python, C#, and JavaScript) that allow you to query the terminology locally, validate entered codes, programmatically navigate hierarchies, and retrieve descriptions in different languages on the basis of the user's context. There are also specialized terminology servers that expose standardized APIs (such as FHIR Terminology Services) so that applications can query SNOMED-CT remotely via the web without needing to incorporate the entire knowledge base into their local installation. The choice between one architectural option or another depends on the specific needs of each project: applications that must operate in environments without an internet connection (such as in rural areas or during emergency deployments) may require a complete local copy of the terminology; applications that are always connected can benefit from centralized services that update automatically without user intervention.

For physicians and clinical professionals involved in designing applications that use SNOMED-CT, understanding the basic characteristics of this terminology helps them grasp why computer systems sometimes offer coding options that seem limited or why they do not immediately find terms they consider obvious and common. Formal terminology is not a perfect reflection of colloquial medical natural language but rather a standardized artificial construct with its own logical rules, naming conventions, and coverage limitations. It may be that the specific term the clinician routinely uses in their practice is not exactly the preferred term in SNOMED-CT or that the clinical concept they wish to express precisely does not have an exact entry in the terminology and they must settle for a more general, approximate concept. Understanding these reasonable limitations helps in having realistic expectations about what the technology can currently offer and in collaborating constructively with developers to find the best available coding solution.

An important point for developers to understand is that SNOMED-CT is neither static nor immutable. The terminology is continuously updated through biannual cycles to incorporate new emerging medical knowledge, correct detected errors, improve the coverage of underrepresented specialties, and refine existing semantic relationships. A new official international version is published every six months, and national member countries may also create local extensions to

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

address specific needs of their particular healthcare system (such as specific funding procedures or locally available medications). This means that developers must stay up to date with these updates, implementing system update processes to use the latest versions when appropriate and ensuring backward compatibility with previous versions when required by regulatory requirements.

The relationship between SNOMED-CT and other specialized medical terminologies is also relevant for realistic implementation. SNOMED-CT does not aim to be the only term used in the healthcare ecosystem, nor does it seek to replace all others; rather, it is intended to serve as the central semantic backbone that connects other, more domain-specific terminologies. For example, for medications, another term called RxNorm (in the U.S.) or ATC (internationally) is typically preferred; for clinical laboratory tests, LOINC is the dominant standard; for surgical procedures in billing contexts, CPT or the ICD-10-PCS are preferred.(2) SNOMED-CT can conceptually act as a reference map that relates and translates between all of them through formal mappings, allowing a concept originally expressed in a specific terminology to be automatically translated into the others depending on the context of use. This capacity for terminological integration is essential in a real healthcare ecosystem where multiple historical and specialized standards necessarily coexist.

For development teams, an important architectural decision is determining what level of granularity and specificity to use from SNOMED-CT for different use cases. The terminology allows coding with varying degrees of clinical specificity, ranging from very general to extremely detailed and specific. For example, one can simply code “diabetes mellitus” (general), or one can specify precisely “type 2 diabetes mellitus with stage 3 renal complications” (very specific). The choice of the appropriate level of detail depends critically on the subsequent use of the clinical data. For an electronic health record that must support precise clinical decisions and longitudinal follow-up, specific details are important and valuable. For an administrative billing system based on diagnosis-related groups, the general category may suffice.

A terminology, no matter how comprehensive and sophisticated, does not save lives on its own. What truly saves lives is the ability to share accurate information among professionals, to make clinical decisions informed by complete data and to coordinate care seamlessly across multiple professionals and levels of care. SNOMED-CT is the semantic infrastructure that makes all of this possible in today’s digital world, but the true value lies in what developers, clinicians, and organizations build upon it. Developers who understand this in depth do not view the terminology as an imposed technical burden but rather as the fundamental semantic infrastructure upon which they can build innovative applications that truly make a difference in patients’ lives.

6.3 LOINC FOR LABORATORY AND RXNORM/ATC FOR MEDICATIONS

Imagine a patient who is receiving a blood test at a clinical laboratory in their city. The results revealed a glucose concentration of 110 mg/dL, which the doctor interpreted in context. That simple number travels from the automated analyzer to the laboratory management system, and from there, it must be transmitted to the electronic health record so that their primary care physician will later consult to make treatment decisions. However, for that number to be truly clinically useful, it is not enough for it to arrive correctly over the network; the receiving system must know with certainty that “110” refers specifically to blood glucose, not total cholesterol or creatinine; that it is expressed in mg/dL, not in mmol/L, which would require conversion; that it is a venous blood sample obtained by venipuncture, not a fingerstick capillary sample; and that it is a measurement taken under fasting conditions, not postprandially after a meal. All this critical contextual information, which is obvious to laboratory professionals because of their training, is completely inaccessible to the receiving computer system if it is not properly encoded using agreed-upon standards. This is where two specialized terminologies come into play, together with SNOMED-CT as the backbone, to complete the fundamental semantic ecosystem of modern digital health: LOINC for clinical laboratory tests and RxNorm or ATC for medications.

LOINC, which stands for Logical Observation Identifiers Names and Codes, is the international reference terminology for unambiguously identifying laboratory tests and other structured clinical observations.(3) Developed and rigorously maintained by the Regenstrief Institute at Indiana University, LOINC provides a unique alphanumeric code for each type of test or observation, with sufficient semantic granularity to distinguish all variants that may be clinically relevant in real-world practice. Blood glucose is not the same as 24-hour urine glucose, nor is a fasting measurement the same as a postprandial oral glucose tolerance test, nor is a serum

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

measurement the same as an EDTA plasma measurement or a capillary blood measurement. LOINC allows for the systematic coding of all these contextually relevant variants in a precise and standardized manner, ensuring that when a laboratory sends a result coded with LOINC, the recipient can interpret it correctly without ambiguity regarding exactly what was measured and under what preanalytical conditions.

6.3.1 THE STRUCTURE OF LOINC AND ITS PRACTICAL IMPLEMENTATION

The logical structure of the LOINC is based on six main axes that, when systematically combined, uniquely and unambiguously define each clinical or laboratory observation. The component specifies which substance or analyte is measured (such as glucose, creatinine, or hemoglobin). The property indicates which aspect of the component is quantified (mass concentration, molar concentration, cell count, partial pressure). The time frame specifies whether the measurement is a single point in time or integrated over a time interval. The system or sample type identifies the biological material (venous blood, serum, citrated plasma, urine, and cerebrospinal fluid). The analytical method is specified when clinically relevant for interpretation (enzymatic method, immunoturbidimetry, chromatography). Finally, the scale indicates the type of result (numerical quantitative, ordinal qualitative, descriptive nominal, or free narrative). This systematic approach to semantic decomposition allows LOINC to currently contain more than 90,000 standardized terms,(3) covering not only conventional laboratory tests in biochemistry and hematology but also structured clinical observations, coded vital signs, functional assessment scales, structured imaging results, and many other types of clinical data requiring precise identification.

For clinical laboratories, adopting LOINC as an internal coding standard is a fundamental step toward true semantic interoperability. When a laboratory sends results to other systems using standardized LOINC codes, any recipient that understands the standard will be able to interpret them correctly and automatically, regardless of how the sending laboratory internally refers to those tests in its own legacy system or commercial catalog. This is especially critical when patients move between different care centers throughout their lives or when data from multiple sources are aggregated for epidemiological research or population-based quality management. Without LOINC, each laboratory inevitably uses its own local internal codes, and semantic integration becomes a logistical nightmare of adhoc manual mappings that consumes valuable professional time, is prone to interpretation errors, and quickly becomes outdated when catalogs change.

For software developers working with laboratory data in clinical applications, LOINC offers a powerful way to standardize information from multiple heterogeneous sources. A medical record application that receives results from dozens of different laboratories can, if all correctly use LOINC, process them in a completely uniform manner internally, applying the same clinical interpretation rules, the same population-adjusted normal reference ranges, and the same critical value alerts. Without this prior terminological standardization, the application would have to know and keep updated the proprietary internal codes of each connected laboratory and apply potentially different interpretation rules for each one—which is technically possible but extremely complex, costly to maintain, and prone to inconsistencies that compromise patient safety.

The implementation of LOINC in a healthcare organization involves not only a technical code-mapping project but also a disciplined exercise in standardizing clinical processes and quality management. It compels laboratories to precisely and explicitly define which tests they actually perform and how they name them internally, to resolve accumulated historical ambiguities and to align with internationally agreed-upon practices. This standardization process, although costly in terms of time and resources initially, has substantial and long-term benefits in terms of data quality, interanalyst consistency, and operational efficiency. A laboratory that knows exactly which tests it offers, uses which methodology, and how it is named in the international standard, is better positioned to rationally manage its service catalog, adequately train its technical staff, and communicate transparently with prescribing clinicians.

6.3.2 RXNORM: DRUG STANDARDIZATION WITH FLEXIBLE GRANULARITY

Let us now turn to the domain of medications, which is, if anything, even more semantically complex than the laboratory domain because of the multiplicity of dimensions involved. A single drug may have multiple commercial formulations with different dosages, multiple active

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

ingredients in fixed combinations, multiple trade names protected by patents in different countries, and multiple routes of administration and dosage forms. To address this inherent complexity, there are two main reference terminologies, with complementary approaches that address different usage needs: RxNorm, developed and maintained by the U.S. National Library of Medicine (NLM), and the ATC (Anatomical Therapeutic Chemical) classification, maintained by the World Health Organization (WHO).(4)

RxNorm is a terminology system that standardizes the names of medications in the U.S. market, providing stable, unique identifiers for each relevant pharmacological concept. What makes RxNorm particularly powerful is that it explicitly distinguishes between different levels of semantic abstraction required for various clinical uses: from the pure active ingredient (such as "atorvastatin" as a chemical substance) through combinations of active ingredients (such as "atorvastatin + amlodipine" as a conceptual product) to specific commercial formulations with defined dosages and dosage forms (such as "atorvastatin 20 mg coated tablets"). Each of these levels of abstraction is assigned its own unique identifier (RXCUI), and RxNorm formally maintains the semantic relationships between them so that one can systematically navigate from the generic to the specific and vice versa according to the needs of the use case.

This multilevel granularity structure is extremely useful for various clinical applications. An electronic prescribing system can operate comfortably at the generic active ingredient level, allowing the physician to prescribe "atorvastatin" without having to specify the brand name or exact formulation, leaving that decision to the dispensing pharmacist. A hospital pharmacy dispensing system may require a specific formulation level to know exactly which physical product to retrieve from the warehouse. A drug interaction alert system can function efficiently at the ingredient level, regardless of the specific commercial formulation the patient is taking. RxNorm allows all these different systems to speak the same semantic language, each operating at the level of detail it needs for its specific function, while maintaining consistency among them.

6.3.3 ATC: THERAPEUTIC CLASSIFICATION FOR POPULATION ANALYSIS

ATC classification, on the other hand, has a different and complementary conceptual approach. It does not focus primarily on drug names or their commercial presentations but rather on their therapeutic classification on the basis of the organ or biological system on which they primarily act and their specific pharmacological properties. The ATC structure () assigns each active ingredient a five-level alphanumeric code on the basis of its primary therapeutic indication. For example, all drugs for cardiovascular diseases begin with the letter C; within this group, C10 specifically refers to lipid-lowering agents; C10AA refers to statins as a pharmacological group; and C10AA05 specifically refers to atorvastatin as an individual active ingredient.

ATC is primarily used for epidemiological studies of drug utilization, analyses of population-based prescribing patterns, pharmaceutical formulary management, and postmarketing safety surveillance. It allows drugs to be grouped by functional therapeutic categories, facilitating aggregate analyses that would not be possible using the individual names of thousands of different commercial formulations. For example, to study total antibiotic consumption in a population over a season, ATC allows for the easy grouping of all drugs in group J01 (antibacterials for systemic use), regardless of their specific active ingredient or commercial formulation. For pharmacovigilance studies and drug utilization analyses at the public health level, this therapeutic grouping capability is absolutely essential.

For medical software developers, the choice between RxNorm and ATC (or the decision to use both simultaneously) depends critically on the application's specific use case and the geographic context of implementation. A clinical electronic prescribing application will likely need the detailed granularity of RxNorm (or its national equivalent) to handle formulations, dosages, and routes of administration accurately. A public health application focused on analyzing aggregated population data may prefer the therapeutic perspective of ATCs for trend analysis and benchmarking. In many cases, the ideal approach is to use both terminologies in a complementary manner, maintaining mapping relationships between the codes of one and the other in the master database. A specific medication has an RxNorm code that uniquely identifies it on the market, as well as an ATC code that classifies it therapeutically, and both pieces of information are useful in different application contexts.

The integration of these specialized terminologies with SNOMED-CT as the backbone completes the interoperable semantic ecosystem. SNOMED-CT provides a general reference framework, with concepts for diseases, symptoms, procedures, laboratory findings and medications

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

at a clinical abstraction level. LOINC and RxNorm/ATC provide the specific technical details within their respective specialty domains. The formal relationships between them are defined in official reference maps maintained by the standards organizations so that one can navigate semantically from one terminology system to another as needed. For example, a SNOMED-CT concept for “type 2 diabetes mellitus” may be formally related to the specific LOINC code for “glycated hemoglobin” as a recommended follow-up test and to the ATC codes corresponding to oral antidiabetics as first-line treatment. This rich network of semantic relationships between terminologies is the technical foundation upon which true semantic interoperability in the digital health ecosystem is built.

For software development teams, working effectively with these terminologies requires familiarity with their internal logical structures and with the technological tools available for their practical management. There are standard web APIs for querying LOINC and obtaining its full descriptions, RESTful web services for accessing RxNorm and its -level relationships, and open-source software libraries for handling ATC codes in local applications. There are also specialized semantic mapping services that help convert proprietary local codes from legacy systems to these international standards, facilitating gradual technological migration. The initial investment in learning to use these tools and integrating them architecturally pays off quickly in terms of the quality of the clinical data generated, ease of integration with other systems, and the capacity for subsequent population analysis.

A critical practical aspect of managing these terminologies is version control and the management of terminological changes. All these terminologies are periodically updated with new versions: LOINC has three releases per year (January, June, and September); RxNorm is updated monthly with new commercial products; and ATC is revised annually by the WHO. Systems that use them must have technical mechanisms and organizational processes to incorporate these updates without losing backward compatibility with historical data coded using previous versions. This is especially critical when longitudinal patient data spanning years are handled: a code that was valid and standard five years ago may have been refined, split, or even declared obsolete in more recent versions, and the system must be technically capable of interpreting it correctly within its historical context. Managing terminological change over time is a complex process that requires advanced architectural planning and appropriate versioning tools.

For physicians and clinical professionals who use applications based on these terminologies, the user experience should be completely transparent and should not increase their workload. They should not have to memorize or know the LOINC codes for the laboratory tests they routinely order or the ATC codes for the medications they prescribe daily. The system should provide them with intuitive search interfaces using common names in their language, with intelligent autocompletion, and handle the translation of those textual searches into the corresponding standard codes internally. The underlying terminological complexity must be carefully hidden beneath a layer of well-designed usability that respects natural clinical workflows. When this is effectively achieved, clinicians reap all the benefits of semantic standardization (interoperable data, smart alerts, quality analysis) without having to pay the price of learning a new artificial technical language.

The final quality of clinical data coded with these terminologies depends, as always in information systems, on those who enter it at the point of care. A physician who prescribes “paracetamol 500 mg tablets” is generating information that can be automatically coded by the system if it is welldesigned with synonym dictionaries. However, if the physician uses nonstandardized local trade names or enters dosages in inconsistent formats (sometimes “500 mg,” other times “0.5 g,” and still others “two tablets”), automatic coding becomes more difficult and prone to errors. That is why it is so important for systems to actively guide data entry through interfaces that suggest standardized options, validate semantic consistency in real time, and provide immediate feedback when they detect something inconsistent or uncodable.

Behind every LOINC code lies years of consensus-building work by laboratory experts from around the world to define exactly what a specific test means and how it differs from similar ones. All this international collaborative work—invisible to most clinical professionals who simply use the resulting systems—is the fundamental semantic infrastructure upon which all modern digital health is built. Without it, clinical data would be unintelligible noise; with it, it becomes actionable information for providing better and safer patient care. In addition, that, at the end of the day, is the only thing that truly matters in medicine.

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

6.4 MAPPINGS AND TRANSFORMATIONS BETWEEN TERMINOLOGIES

Imagine a chronic patient arriving at an emergency department in a city other than their own, far from their usual primary care physician. They bring with them a recent discharge summary from their home hospital, which documents that they have "type 2 diabetes mellitus with stage 3 renal complications" and are on active treatment with "metformin 850 mg tablets every 12 hours." The emergency physician urgently needs to incorporate this critical information into the receiving hospital's electronic health record to make safe management decisions, but this hospital's computer system uses SNOMED-CT for diagnosis coding and ATC for medication coding, whereas the referring hospital uses ICD-10-CM for billing diagnoses and its own nonstandardized internal code for medication prescriptions. The patient's information is clinically invaluable for continuity of care, but it is expressed in a terminological "language" that the receiving system cannot understand directly without translation. Someone—ideally a high-quality automated process—must semantically translate these concepts between systems. This is precisely what mappings between terminologies are all about: establishing formal semantic bridges that allow clinical information to travel from one system to another while preserving its essential meaning for decision-making.

A terminological mapping is, in essence, a declared correspondence between concepts in different medical coding systems.

It can be a simple one-to-one relationship, where a concept in one terminology exactly and unambiguously corresponds to a concept in another; a one-to-many relationship, where a general concept in one terminology corresponds to several more specific concepts in another, requiring a decision on which one to choose; a many-to-one relationship, where several specific concepts in the source terminology are necessarily grouped under a more general one in the target owing to a lack of granularity; and even more complex relationships may exist, with partial overlaps in meaning, approximate or "sufficiently similar" equivalences, or even a total lack of correspondence when a concept simply does not exist in the target terminology because it is too specific or novel.

Managing this inherent complexity is among the most fascinating and critical technical, semantic, and epistemological challenges of modern digital health engineering.

6.4.1 THE THREE REASONS FOR THE NEED FOR MAPPING

The practical need for terminological mappings arises from several fundamental reasons that coexist in the real-world healthcare ecosystem. The first is historical and legacy-related: different national and regional healthcare systems have adopted different standard terminologies over the decades on the basis of their legislative and technological contexts and now need to interoperate without replacing their entire legacy of historical data. A hospital may have been using ICD-9-CM for decades to code discharge diagnoses, and migrating its entire historical archive of millions of episodes to SNOMED-CT would be a titanic data engineering task with prohibitively high costs. Well-designed mappings allow new systems based on modern terminologies to understand and interpret old information coded with previous standards and enable historical systems to communicate bidirectionally with new systems without needing to replace the entire existing infrastructure all at once, facilitating gradual technological transitions.

The second reason is functional and relates to use cases: different medical terminologies inherently serve different clinical and administrative purposes, and a single information system may need to use several of them simultaneously for different functions. A single patient diagnosis may need to be coded in SNOMED-CT with high clinical granularity for detailed electronic health records, in ICD-10-CM with billing codes for the hospital discharge coding system, and in a specific simplified classification for a national epidemiological registry of a particular disease. Instead of forcing the clinician to manually enter the same diagnostic concept multiple times in different codes for different systems—which would be inefficient and prone to inconsistencies—automatic mappings allow systems to automatically translate the concept into the other representations needed for various downstream uses once it has been entered and validated in primary terminology.

The third reason is global interoperability and cross-border care: When clinical information is exchanged between countries with different terminological traditions or between systems that have adopted different national standards for historical reasons, mapping is the only technically viable way to ensure that data are mutually understood without critical semantic loss. A European

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

patient receiving emergency care in the United States, an international multicenter research study involving hospitals on different continents with different coding standards and an international drug safety alert requiring rapid and accurate data sharing among regulatory agencies in different countries—all these critical scenarios depend fundamentally on the existence of reliable, validated, and maintained mappings between the terminologies used in each jurisdiction.

The most important, reliable, and widely used mappings internationally are those officially developed and maintained by the organizations responsible for the source and target terminologies themselves. SNOMED International, for example, maintains and publishes updated official mappings between SNOMED-CT and the ICD-10 for both the detailed clinical version and the simplified mortality statistics version. The U.S. National Library of Medicine (NLM) maintains official mappings between RxNorm and various national drug terminologies used in other European and Asian countries. These official mappings are the result of meticulous, methodologically rigorous work by clinical experts and terminologists who analyze concepts by concept, determine the most appropriate semantic correspondence in each case, explicitly document ambiguous cases or instances of imperfect correspondence, and validate the results with real-world test data.

6.4.2 LOCAL MAPPINGS, GRANULARITY MANAGEMENT, AND DIRECTIONALITY

However, official international mappings, although of high quality, do not cover all the practical needs of real-world healthcare systems. Often, individual healthcare organizations need to create and maintain their own “local” mappings to connect their proprietary legacy systems with modern international standards or to adapt generic international standards to their specific local clinical practice needs. A hospital may have its own internal coding system for hundreds of laboratory tests, developed organically over decades of service evolution, and needs to systematically map it to standard LOINC to exchange results with other centers in the regional network. These local mappings are as important operationally as official international mappings are and require equally rigorous methodologies to ensure their semantic quality and ongoing maintenance.

Creating high-quality terminological mapping is not a trivial task that can be fully automated without expert supervision. It requires in-depth knowledge of both terminologies involved in their entirety, a contextual understanding of the specific clinical environment in which the mapped codes will be used, and a documented systematic methodology for resolving ambiguous or conflicting cases. Multidisciplinary teams that develop high-quality mappings typically follow a formalized process that includes a preliminary comparative analysis of the structures of both terminologies, automated identification of obvious correspondences on the basis of term similarity, manual review of doubtful or conflicting cases by clinical experts in the relevant specialty, empirical validation of the resulting mapping using real test data, and comprehensive documentation of the decisions made and cases of imperfect correspondence. It is a technically complex, semantically meticulous task that requires constant updating as the source and target terminologies evolve with new versions.

A crucial aspect of mapping design is the explicit management of differences in granularity between terminologies. When one term is systematically more detailed than another in a specific domain, one-to-many relationships may arise that require algorithmic or human decisions on how to proceed. For example, SNOMED-CT has very specific concepts for different types of diabetes mellitus and their specific complications, while an older classification, such as ICD-9, may have only “diabetes mellitus” as a general aggregate category. In these cases, of granularity asymmetry, a strategic decision must be made: either map everything to the general level, thereby losing valuable detailed information, or create contextual rules that allow for preserving detail whenever possible through dual coding. There is no single universal answer; it depends on the specific use case and the criticality of the detail for the clinical or administrative decision that will be supported by the data.

The directionality of the mapping also matters significantly and must be explicitly documented. A mapping that works well from Terminology A to Terminology B may not work equally well from B to A, especially when there are asymmetric relationships in terms of granularity or specialization. That is why good professional mappings are usually bidirectional when possible or at least clearly document the limitations and information loss in each translation direction. A system that receives data coded in a source terminology and needs to translate it into a target terminology must explicitly know whether the correspondence is exact (equivalence), approximate

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

(similarity), or whether there is simply no valid correspondence for that specific concept.

6.4.3 TECHNICAL IMPLEMENTATION, UNCERTAINTY MANAGEMENT, AND QUALITY

For medical software developers, working with terminology mappings in their applications involves architecturally incorporating the ability to perform robust automatic semantic translations. This can be implemented in various technical ways depending on the context: using remote terminological web services that offer updated official mappings (such as those from the NLM or SNOMED International), incorporating local mapping tables into system databases for quick lookups without network dependency, or implementing more complex transformation engines that apply algorithmic inference rules for cases not directly mapped. The architectural choice depends on the volume of data to be processed, the required frequency of real-time or batch translations, and the need to maintain automatic synchronization with periodic updates to the source and target terminologies.

A significant and often underestimated technical challenge is the explicit management of the uncertainty inherent in mappings. Not all terminological correspondences are perfect or exact; some are reasonable approximations, others are partial with a loss of nuance, and others simply do not exist for very specific or new concepts. Information systems that use mappings must be designed to handle this uncertainty transparently, for example, by flagging automatically translated data with metadata such as "automatically mapped—review recommended" so that the clinical user knows it may not be exact or by implementing mandatory human review workflows for cases flagged as questionable by the mapping algorithm. Explicit transparency regarding the quality and reliability of mappings is essential to maintain professional trust in the translated data and to enable the detection of translation errors before they cause harm.

The quality of mappings in production can be measured and monitored using objective metrics such as accuracy (the percentage of machine translations that are correct according to expert validation) and coverage (the percentage of source concepts that can be translated to the target without unacceptable loss of information). Teams maintaining operational mappings must continuously monitor these metrics, investigate errors detected through auditing, and iteratively improve their maps. It is a task of continuous maintenance throughout the system's lifecycle, not a one-time implementation project with a defined end.

For physicians and clinical professionals who use systems based on terminology mapping, the ideal user experience should be completely seamless within their normal workflow. They should not have to consciously worry about whether the diagnosis they enter into SNOMED-CT is correctly translated to the ICD-10 for billing; the system should do this automatically and display both codes for verification if necessary. However, they should be aware, as part of their professional digital literacy, that such automatic translation exists and that it may have semantic limitations in complex cases. When a system displays clinical information from another healthcare facility, automatically translated from another terminology, they should know that they are viewing a semantic interpretation, not necessarily the exact original record from the clinician who treated the patient. This critical awareness helps you maintain an appropriately skeptical attitude and detect potential translation errors that could affect patient safety.

A particularly complex special case of mapping is the transformation between different versions of the same terminology over time. When SNOMED-CT releases a new version every six months (), some concepts may change codes because of refinement, disappear because of obsolescence, or split into several more specific concepts because of changes in the ontological model. Systems that handle longitudinal patient data accumulated over years require mappings between versions to correctly interpret older codes within the context of current terminology. These "medical record" mappings are provided by the organizations responsible for the terminologies, but their technical implementation requires special care to avoid the loss of valuable historical information or the introduction of anachronistic errors in retrospective interpretation.

The relationship between the quality of the mappings and the quality of the resulting clinical data is direct and critical. An incorrect mapping, especially if it is systematic and undetected, can turn clinical data that are perfectly valid in the source terminology into erroneous or misleading information in the target terminology, with potentially serious consequences for patient care decisions. That is why mappings must be treated with the same rigor in quality and security management as any other critical component of the medical information system. They must be developed by multidisciplinary teams of experts, empirically validated with real test data, subjected to formal version control, periodically audited by independent third parties, and

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

thoroughly documented. Clinicians' confidence in automatically translated data must be as high as their confidence in the original manually entered data, and this can be achieved only through rigorous quality processes.

As terminologies mature and become internationally standardized and as new systems predominantly adopt common standards such as SNOMED-CT and LOINC, the acute need for complex mappings may gradually diminish, but it will never disappear completely. There will always be legacy systems that must be kept operational for years; there will always be legitimate local needs for specific coding; and there will always be new emerging domains of medical knowledge (such as genomics or personalized medicine) that will require new specialized terminologies, which will then need to be mapped to existing standards. Mappings are, and will remain for the foreseeable future, an essential and permanently necessary part of the semantic infrastructure of global digital health. Software developers, data engineers, and clinicians who deeply understand their importance, learn to design them rigorously, and develop the skills to work with them critically are better prepared to build information systems that can truly communicate with one another securely, overcoming the barriers of medical language and allowing clinical information to flow seamlessly to where it is most needed: at the patient's bedside, at the moment of critical clinical decision-making.

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

REFERENCES

What is SNOMED CT? · NLM Customer Support Center [Internet]. [cited 2026 May 13]. Available from: https://support.nlm.nih.gov/kbArticle/?pn=KA-04795
Fung KW, Xu J, Brear H, Lane A, Lau M, Wong A, et al. Promoting interoperability between SNOMED CT and ICD-11: lessons learned from the pilot project mapping between SNOMED CT and the ICD-11 Foundation. J Am Med Inform Assoc [Internet]. August 1, 2024 [cited May 13, 2026];31(8):1631. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC11258399/
Health Data Standards and Terminologies: A Tutorial [Internet]. [cited 2026-05-13]. Available from: https://www.nlm.nih.gov/oet/ed/healthdatastandards/02-530.html
RxNorm Overview [Internet]. [cited May 13, 2026]. Available from: https://www.nlm.nih.gov/research/umls/rxnorm/overview.html

https://doi.org/10.62486/978-9915-9876-8-2.Ch06

Chapter 7 / Capítulo 7

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch07
Pages: 111-132

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 7. Interoperability Standards

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Interoperability is neither a technical luxury nor a distant goal: it is the minimum requirement for 21st-century medicine to be practiced reliably. When systems cannot communicate with one another, the one who pays the price is not the IT specialist who failed to make them talk, but the patient whose medical history is fragmented, the physician who makes decisions with incomplete information, and the healthcare system that multiplies unnecessary tests."

Imagine a patient visiting a cardiology clinic. The cardiologist needs to know the results of the latest lab tests, discharge summaries from previous hospitalizations, the current medications prescribed by the patient's primary care physician, and imaging tests performed at another facility. In an ideal world, all this information would be available on the computer screen and integrated in a coherent, up-to-date, and reliable manner. In the real world, what usually happens is very different: the cardiologist has to ask the patient to bring the reports, call the health center, or ask the patient to repeat the tests because the previous results are not accessible. Behind this everyday scenario lies a problem that is not medical but technical, and it has a name: lack of interoperability.

Interoperability is the ability of different information systems to communicate, exchange data, and use exchanged information. In the healthcare sector, this ability is not an option but a clinical necessity. Patients are not treated in a single location by a single doctor using a single system. They move between primary and specialty care, between emergency rooms and hospitalization, between public and private facilities, between autonomous communities, and sometimes between countries. For their information to follow them, for every professional treating them to have access to the relevant data, the systems must be able to understand one another. It is not a matter of administrative efficiency but of clinical safety.

In this chapter, we explore the main standards that enable interoperability in healthcare. We start with the FHIR, the most modern and promising standard, which is revolutionizing the way healthcare applications exchange information. We will also look at HL7 v2, the legacy standard that still dominates many hospital systems, and DICOM, the standard that enables the exchange of medical images. Finally, we explore SMART on the FHIR, an extension that enables the development of applications that integrate directly into electronic health records systems. This is not an exhaustive technical guide but rather an overview designed to help clinicians and developers understand the possibilities and limitations of each standard and make informed decisions about when and how to use them.

7.1 FHIR: PHILOSOPHY, RESOURCES, AND USE CASES

At the end of the last decade, the world of healthcare interoperability found itself in a frustrating paradox. Technical standards existed—some with decades of history and maturation—but they were complex to implement, rigid in their architecture, and difficult for modern developers to adopt, given their familiarity with elegant REST APIs, agile data exchange in JSON, and rapid iterative development cycles. Integration projects between healthcare systems inevitably dragged on for months or years, required highly specialized and expensive consulting experts, and often resulted in highly customized solutions that were neither reusable nor maintainable. Against this backdrop of widespread frustration, a group of visionaries within HL7 International—the organization historically responsible for healthcare interoperability standards—decided that it was time to start over with a radically different approach. Not simply to create yet another technical standard among many but to rethink from scratch what the exchange of health information should look like in the modern era of the internet and mobile computing. The result of that effort was FHIR, and it literally changed the game for healthcare interoperability.(1)

FHIR, which stands for fast healthcare interoperability resources, is an open standard for the exchange of healthcare information that combines the best of previous semantic standards with modern web technologies that developers already know and master. Its name says it all: it is fast because it uses stateless REST APIs and lightweight exchange formats such as JSON; it is specifically for healthcare because it is designed from the ground up for the clinical domain with all its complexities; and it is fundamentally based on resources, which are the fundamental building blocks upon which the entire interoperability architecture is built. Since its official launch, the FHIR has been widely adopted by leading commercial electronic health record vendors (Epic, Cerner, and Meditech), by government health agencies around the world, and by thousands of independent developers who have found in it a finally simple and accessible way to make

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

healthcare systems talk to each other.(1)

7.1.1 THE THREE FUNDAMENTAL PRINCIPLES OF THE FHIR

FHIR's design philosophy is based on several fundamental principles that clearly distinguish it from previous interoperability standards. The first principle is intentional simplicity: The FHIR is explicitly designed to be easy to understand and implement, even for software developers without specific prior experience in healthcare interoperability. It uses technological concepts familiar to any modern web developer: RESTful architecture with standard HTTP verbs (GET, POST, PUT, and DELETE), resource exchange in JSON or XML depending on preference, authentication via OAuth 2.0, and interactive online documentation. This technological familiarity drastically reduces the historically steep learning curve and makes integration between systems much faster to implement and less costly to maintain.

The second principle is architectural modularity. The FHIR is not a rigid, indivisible monolith but rather a set of more than 150 independent and well-defined resources that can be used separately or combined according to the specific needs of the use case. A resource can conceptually represent a patient, an individual clinical observation, a medication prescription, a diagnostic condition, a performed procedure, or any of the many types defined in the standard. Each resource has a well-defined data structure, with mandatory fields for identification and optional fields for semantic richness, and can be formally extended via profiles to adapt to specific local needs without breaking core compatibility with other FHIR systems.

The third principle is evolution based on the best practices of the modern web. FHIR explicitly adopts the same software engineering practices that have made the global web successful and scalable: semantic versioning of specifications (major.minor.patch), guaranteed backward compatibility within major versions, interactive documentation with executable examples, and an active global community that openly shares reference implementations, national profiles, and lessons learned. This means that, unlike previous standards that require costly, disruptive updates every few years, the FHIR can evolve gradually and continuously, adding new resources and functionalities without intentionally breaking what is already working in production.

Resources are the architectural heart of the FHIR and its fundamental unit of exchange. A resource is a structured, typed representation of a clinical, administrative, or logistical concept in the healthcare domain that is explicitly designed to be exchanged between heterogeneous systems. Each resource has a globally unique identifier (ID) and a set of typed data elements (such as name, date, numeric value, terminology code) and may contain formal references to other related resources. For example, an "observation" resource representing a systolic and diastolic blood pressure measurement may reference the "patient" resource representing the patient on whom the measurement was taken, the "encounter" resource representing the clinical encounter during which the measurement was taken, and the "practitioner" resource representing the healthcare professional who performed or validated the measurement. This network of references between resources allows the complete clinical context to be computationally reconstructed from individual exchanged resources.

7.1.2 THE RICHNESS OF RESOURCES AND THE CONCEPT OF A PROFILE

The list of standard FHIR resources covers virtually the entire semantic domain of modern healthcare practice. There are fundamental resources for demographic and organizational data (Patient, Practitioner, PractitionerRole, Organization, Location), for clinical process management (Encounter, EpisodeOfCare, Appointment, Schedule, Slot), for recording diagnoses and conditions (Condition, AllergyIntolerance, FamilyMemberHistory, ClinicalImpression), for documenting procedures and interventions (Procedure, MedicationAdministration, Immunization, CarePlan), for observations and results (Observation, DiagnosticReport, ImagingStudy, Media, Specimen), for medication management (Medication, MedicationRequest, MedicationDispense, MedicationStatement), and many others for administrative, financial, and follow-up functions. This broad and deep semantic coverage enables the FHIR to be used in virtually any conceivable interoperability scenario, ranging from simple document exchange between organizations to real-time integration of IoT medical devices, including structured access to data for clinical research or artificial intelligence.(1)

One of the most important and powerful innovations of the FHIR is the formal concept of

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

a “profile.” Since the base standard must be sufficiently flexible and generic to adapt to different national contexts and specialties, the base resources have many optional elements and possible extensions. A profile is a formal restriction of those base resources for a specific use, defining which elements are mandatory in that context, which terminology vocabularies must be used for coding, which local extensions are necessary, and which value restrictions apply. For example, a Spanish national profile may define how patient identification numbers (DNI, NIE, health card) must be specifically represented in the patient resource. A profile for a specific clinical trial may define which outcome observations must be collected and in what standardized format. Profiles allow the FHIR to be simultaneously a globally interoperable standard and a tool locally adapted to specific needs without fragmentation.

7.1.3 TRANSFORMATIVE USE CASES AND THE FUTURE OF FHIR

The practical use cases for FHIRs are as varied and numerous as the global healthcare practice itself is. One of the most common and visibly transformative is access to patient data via consumer mobile apps. A patient can use an app on their smartphone to securely view their recent lab results, current active medications, upcoming scheduled appointments, or even download their complete medical record, provided that their hospital or insurance provider’s system exposes that data through an authenticated and authorized FHIR API. This same API can be used by a primary care physician to access patient data from their own certified application or by an authorized academic researcher to extract aggregated, anonymized data for a population-based epidemiological study.

Another high-impact use case is the seamless integration of medical devices and health wearables. A continuous glucose monitor, a connected digital blood pressure monitor, a smart pacemaker, or an IoT bathroom scale can automatically send their readings to the patient’s electronic health record using standard FHIR resources (typically Observation). This eliminates the need for patients or healthcare professionals to manually enter data, reducing transcription errors, improving the quality of information with data of greater temporal granularity, and facilitating the remote monitoring of chronic patients. The accelerating trend toward connected medicine and hospitals without walls, with devices that are increasingly smart, ubiquitous, and affordable, makes this use case increasingly strategic for future care models.

Clinical research is another area where the FHIR has a significant effect on data efficiency and quality. Traditional clinical trials require the collection of large amounts of structured data from participants, and this was traditionally done using paper-based data collection notebooks or separate electronic data capture (EDC) systems disconnected from actual electronic health records. With FHIR, it is technically possible to automatically extract relevant data from the participant’s electronic health record, drastically reducing the burden of double data entry, minimizing transcription errors, and improving data quality and accuracy since it comes directly from the routine clinical record. Initiatives such as the U.S. ONC’s Sync for Science project are exploring precisely this transformative possibility, allowing patients to authorize the sharing of their structured FHIR data with researchers in a simple, secure, and ethically supervised manner.

Continuity of care across levels of care is perhaps the most classic use case, yet one that remains unsatisfactorily resolved in many systems. When a patient is referred from primary to specialty care or when they are discharged from the hospital to their home, it is crucial that relevant clinical information travels with them in a structured and actionable format. FHIRs allow this information to be packaged into a “DocumentReference” or a “Bundle” containing all relevant resources: coded diagnoses, active medications with dosages, documented allergies, tests performed with results, and a discharge care plan. This “ package can be transmitted electronically to the receiving system, which can incorporate it into its own repository while maintaining the structure, clinical meaning, and traceability of the data.

For software developers, working with the FHIR is now much more accessible and productive than working with previous standards. There are mature and well-documented client libraries for virtually all modern programming languages (Java, C#, Python, JavaScript/TypeScript, and Swift), open-source FHIR servers (HAPI FHIR, Smile CDR, and Microsoft FHIR Server) that can be deployed locally for development and testing, and an active global community on forums and GitHub that shares experiences, resolves technical questions, and contributes improvements. The official HL7 FHIR documentation is clear, wellorganized by resources and operations, and includes interactive, executable examples that allow you to test the APIs directly from your browser without the need to install any local software. This technological accessibility has been key to the rapid adoption of

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

the FHIR, especially among innovative startups and small developers who previously could not afford the cost of entry into the closed world of healthcare interoperability.

For physicians and clinical professionals involved in FHIR system development or implementation projects, understanding the basic principles of this technology helps them grasp the new possibilities it opens up for improving clinical practice. They can envision and propose clinical applications that previously seemed impossible or prohibitively expensive: an intelligent prescription assistant that automatically checks for drug interactions with the patient's current medications in real time via FHIR; a predictive alert system that notifies staff when a high-risk patient has missed scheduled follow-up appointments; and an integrated visualization tool that aggregates data from multiple sources (hospital, laboratory, pharmacy, devices) into a single longitudinal monitoring screen. All of these applications are technically possible today with FHIRs, and many of them are already in use at leading hospitals around the world.

Information security is a fundamental and nonoptional aspect of FHIRs. Since the APIs expose highly sensitive clinical data protected by strict regulations (HIPAA, GDPR), the standard incorporates robust authentication and authorization mechanisms based on industry standards such as OAuth 2.0 for authorization and OpenID Connect for identity authentication. Patients can granularly control which third-party applications access which categories of their data and for how long, revoking access whenever they wish. FHIR systems can comprehensively audit every data access to detect potential misuse or intrusions. This security layer, combined with TLS encryption in transit and at rest, makes FHIR technically suitable even for the most demanding and heavily regulated healthcare environments.

The future of the FHIR as the dominant standard is promising and increasingly wellestablished. The leading electronic health record systems in the global market (Epic, Cerner, Meditech, Allscripts) already offer certified FHIR APIs in their products, and many countries have formally adopted it as a mandatory national standard for interoperability. The U.S. Office of the National Coordinator for Health Information Technology (ONC) has mandated the use of the FHIR for the certification of new health systems, which has dramatically accelerated its adoption in the North American market. In Europe, the European Health Data Space (EHDS) is actively promoting the use of the FHIR for the cross-border exchange of patient summaries and research health data. All signs point to the FHIR becoming the de facto universal common language of digital health in the coming years—r the technical Esperanto that will allow systems worldwide to communicate effectively with one another.

It is important to remember, realistically, that the FHIR is not a magic solution that automatically resolves all historical interoperability problems in healthcare. Careful and expert work remains absolutely necessary in architectural design, and appropriate semantic profiles for each context, quality data governance, and organizational change management for adoption must be defined. However, the FHIR has drastically lowered the technical barrier to entry, ultimately making interoperability accessible to a much broader and more diverse community of developers, entrepreneurs, and clinical innovators. In addition, in a world where technology is advancing ever faster and the need for coordinated care is becoming increasingly urgent, it is likely its most valuable and lasting contribution to improving global health.

7.2 HL7 V2: THE LEGACY THAT WON'T GO AWAY

In any hospital we visit today, we'll find a reality that surprises developers new to the healthcare sector. Systems that have been in operation for twenty or thirty years, built on technologies that are archaic by today's standards, remain the beating heart of daily operations. Their communication interfaces are based on a standard born in the 1990s, when the internet was not yet the global infrastructure we know today. Every time someone proposes modernizing it, the same inevitable dilemma arises: replacing everything would be economically prohibitive and clinically risky, but keeping it means living with limitations that hinder innovation. This hospital is not an exception but the absolute norm in healthcare across half the world.

In terms of all the odds, HL7 v2 remains the dominant standard for systems integration in hospital settings. Everything indicates that it will continue to be so for many years to come, not for lack of modern alternatives, but owing to the inertia of thousands of critical implementations that no one can afford to shut down. Understanding why this standard persists and understanding how to coexist with it intelligently has become an essential skill for any developer aspiring to work in real-world healthcare environments. Ignoring its existence is equivalent to attempting to build bridges without knowing the ground on which the foundations rest.

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

7.2.1 THE ORIGINS OF A PRAGMATIC STANDARD

HL7 v2 was born in the late 1980s and early 1990s, when connectivity was a luxury rather than a commodity. The internet did not yet exist as a global infrastructure, and hospitals needed to exchange information between systems that rarely shared the same manufacturer. The solution proposed by Health Level Seven International was surprisingly simple: plain text messages, machine-readable but also interpretable by humans with some training, based on simple delimiters such as the pipe (|) and the caret (^). These messages could be transmitted over basic protocols such as TCP/IP or even serial connections, adapting to the technical limitations of the time without the need for sophisticated infrastructure.(2)

The design philosophy reflected the needs of an era where bandwidth efficiency was crucial and processing power was limited. What might seem like a cryptic and unnecessarily complex monstrosity to a modern developer, accustomed to REST APIs with JSON and OAuth authentication, represented an elegant and relatively simple solution to implement for the systems of that era. It requires no heavy libraries, no proprietary binary formats, and no complicated external dependencies. A program written in COBOL could generate these messages just as easily as one written in C or in the early fourth-generation languages.

The structure of an HL7 v2 message follows conventions that, once understood, are logical although verbose. It typically begins with an MSH (message header) segment that serves as the message's cover letter, indicating who is sending it, who should receive it, what type of message it contains, and when it was generated. This header is followed by other specialized segments depending on the purpose of the communication: PID segments for patient demographics, PV1 for information about the care episode, OBR for diagnostic test orders, and OBX for the results themselves. Each segment consists of fields separated by pipes, and each field can be broken down into subfields using circumflex accents.

The visual result is a long and seemingly cryptic text string, where the information appears densely packed without the whitespace or indentation that make other formats readable. However, this apparent chaos follows strict rules and is perfectly interpretable by any system that knows the grammar. A laboratory results message may contain dozens of OBX segments, each with multiple fields describing everything from the test code to the numerical value obtained, including units of measurement, reference ranges, and technician's notes. The density of information is remarkable, even if human readability is not desired.

What made HL7 v2 a successful standard—and what explains its surprising longevity—was precisely its combination of simplicity and flexibility. Unlike more rigid standards that attempted to anticipate and standardize every possible scenario from the outset, HL7 v2 adopted a pragmatic and modest approach. It provided a robust basic structure but allowed organizations to adapt it to their local needs through extensions, optional fields, and bilateral implementation agreements. This flexibility made it relatively easy to implement even for manufacturers with limited resources.

The ability to adapt to the enormous diversity of systems and processes existing in hospitals around the world was decisive. Each facility had its own unique legacy workflows and local coding systems. A standard that was too rigid would have failed in the face of this variability, but HL7 v2 managed to become the de facto standard for integration because it knew how to yield to local realities without compromising global interoperability. As a result, during the 1990s and 2000s, it became the lingua franca of communication between hospital systems—the dialect that everyone understood, even though each spoke it with their own accent.

7.2.2 CURRENT REALITY: PERSISTENCE AND DOMINANCE

Today, decades after its initial creation, HL7 v2 maintains a dominant position that baffles those who predicted its demise on multiple occasions. Laboratory systems continue to send results to electronic health records via HL7 v2 messages, exactly as they did in the 1990s. Hospital pharmacy systems receive electronic prescriptions in the same format. Admission systems report admissions, discharges, and transfers using the same syntax as always. Approximately ninety percent of U.S. hospitals use HL7 v2 for some of their critical integrations, and the situation in Europe, Latin America, and other regions does not differ substantially.

This persistence is not due to a lack of modern alternatives. The FHIR, the most recent standard from the same organization, HL7 International, offers technically superior capabilities in almost every respect. However, the economic and operational realities of hospitals create formidable inertia. Replacing a laboratory system that works, is validated, is thoroughly familiar

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

to staff, and is integrated with dozens of other systems via HL7 v2 represents a multimillion-dollar cost and unacceptable operational risk. Even quite rightly, hospital executives wonder why they should risk the continuity of care to obtain technical benefits that often do not translate into perceptible clinical improvements.

For developers new to the healthcare field, working with HL7 v2 can be a deeply bewildering experience. Messages are difficult for a human to read without specific tools; they lack the visual hierarchical structure offered by formats such as XML or JSON, and the available documentation can be overwhelming because of its length and technical density. Debugging tools are limited compared with modern development ecosystems, and the testing cycle often requires complex environments that simulate the hospital setting. However, it is an absolutely necessary skill because most real-world integrations in hospitals involve HL7 v2 at some point in the process.

Ignoring this standard is tantamount to condemning oneself to work in a technological fantasy world where only modern, well-designed systems exist when the reality of healthcare is radically different. A developer who proposes a solution based solely on modern REST APIs, without considering how it will integrate with the legacy systems that handle eighty percent of clinical data, is building on quicksand. Hospitals are not software labs where systems can be rebooted every year; they are living organisms where continuity is a priority and where every change must be clinically justified.

One of the most important features of HL7 v2, and one of its greatest sources of operational complexity, is the multitude of versions coexisting in the ecosystem. There are HL7 v2.1, v2.2, v2.3, v2.3.1, v2.4, v2.5, v2.5.1, v2.6, v2.7, and intermediate versions that add specific functionalities. Each version incorporates new segments, new fields within existing segments, and new functionalities that reflect the evolution of clinical practice. However, the most problematic issue is not the existence of multiple versions but rather that not all systems implement them consistently or completely.

Often, two systems that supposedly speak the same version of HL7 v2 have different interpretations of how certain optional fields should be used or of which values are valid for specific codes. This variability means that every integration between two specific systems is a custom engineering project, requiring detailed analysis of each system's particular specifications and exhaustive testing to ensure that communication is semantically correct. It is not enough for the message to be syntactically valid; it must convey the correct clinical meaning between systems that may have radically different internal data models.

7.2.3 COEXISTENCE STRATEGIES AND COMPLEXITY MANAGEMENT

To manage this inherent complexity, most medium and large hospitals have adopted an architecture based on integration engines, also known as interface engines. These specialized products act as intelligent intermediaries between different systems, receiving messages in one format, translating them into another when necessary, routing them to the correct recipients according to configurable rules, and providing monitoring and debugging tools that compensate for the limitations of the original systems. A typical integration engine can receive an HL7 v2 message from a laboratory, convert its local coding to a hospital standard, enrich it with additional information from other systems, and deliver it to the electronic health record in the exact format it expects.

These engines are, in reality, the invisible glue that holds together the ecosystem of heterogeneous systems in a modern hospital. Without them, each system would have to know the specific details of every other system with which it communicates, creating a network of dependencies that would be impossible to manage. With a centralized integration engine, each system only needs to know how to communicate with the engine, and it is the engine that handles the complexities of translation and routing. The most established products on the market, such as Mirth Connect (now NextGen Connect), Rhapsody, or Intersystems Ensemble, have perfected these capabilities over decades.

The relationship between HL7 v2 and more modern standards such as the FHIR is not, as is sometimes portrayed, one of confrontation or imminent replacement. The FHIR is not going to "kill" HL7 v2, at least not within the foreseeable timeframe of the next decade. What is happening in practice is a structured coexistence where each standard occupies its functional niche. Hospital systems are exposing FHIR APIs for new developments, mobile applications, and patient portals, while their critical internal integrations and connections to legacy systems continue to use HL7 v2 exactly as before.

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

Integration engines are evolving into bilingual bridges, adding native capabilities to translate between both worlds. In this way, a modern FHIR-based application can communicate seamlessly with a 1990s-era lab system that only understands HL7 v2, without either end needing to know the other exists. This translation architecture is the practical reality of contemporary healthcare interoperability and will likely remain so for many years, given the pace of hospital system upgrades.

For developers building new applications for the healthcare market, a sensible architectural strategy is to use the FHIR to interface with the outside world and with modern systems but to be prepared to integrate with systems that speak only HL7 v2 when necessary. This involves understanding the basics of the legacy standard, knowing how to read and interpret messages manually when automated tools fail, and understanding how integration engines work to collaborate effectively with hospital infrastructure teams. It also means being aware of the inherent security limitations.

HL7 v2 was designed at a time when cybersecurity was not the central concern it is today. The original standard lacks a robust security model comparable to that of FHIRs and does not support modern token-based authentication, and messages often travel in plain text unless additional encryption measures are taken at the transport layer. Integrating legacy systems requires special attention to security and privacy and implementing encrypted tunnels, VPNs, or network segmentation to compensate for the application protocol's shortcomings. Clinical data are sensitive and contain regulated information, and we cannot allow them to travel across hospital networks without adequate protection.

For healthcare professionals involved in development projects, understanding that this legacy world exists helps them maintain realistic expectations regarding the timelines and complexity of integration. When a developer explains that accessing lab data will take weeks of work because they must understand exactly how the current system sends it, they are not making excuses or exaggerating difficulties but rather accurately describing the technical reality. Legacy systems lack well-documented APIs, do not uniformly follow standards, and often require reverse engineering to determine what information the messages they generate actually contain.

One frequently overlooked aspect is that HL7 v2, despite its age, remains the subject of active maintenance and controlled evolution. HL7 International regularly publishes new versions that incorporate requirements from contemporary clinical practice, although the changes are necessarily incremental and maintain backward compatibility so as not to break existing implementations. There are also profiles and implementation guides that help reduce problematic variability between systems. In the United States, the IHE (Integrating the Healthcare Enterprise) initiative defines specific usage profiles for HL7 v2 in specific scenarios such as the exchange of laboratory reports or the notification of hospital discharge.

These profiles facilitate integration by reducing the available options and establishing common conventions on how to use optional fields, which encodings to employ, and how to handle exceptional scenarios. When two systems follow the same IHE profile, the likelihood of seamless interoperability increases significantly. However, the adoption of these profiles is not universal, and many legacy systems were developed before they existed or implemented them only partially. The quality of HL7 v2 implementations is, consequently, extremely variable in the market.

Some systems, especially the most modern ones or those developed by large corporations, follow the specifications of the letter and generate clean, well-structured, and predictable messages. Others, particularly those developed in-house by hospitals or by small local vendors, have highly proprietary implementations, with fields used in undocumented ways, local encodings without standard equivalents, and erratic behaviors that defy all logic. Part of the daily integration work consists precisely of addressing this variability, meticulously documenting the peculiarities of each system and developing adapters that normalize messages to a common internal format.

Monitoring HL7 v2 communications is another critical aspect that we cannot underestimate. In a hospital environment where information must flow continuously to support clinical decisions, messages can be lost because of network errors, corrupted by encoding issues, or arriving out of order because of variable latencies. Systems can go down because of maintenance or failures, resulting in unprocessed messages in overflowing queues. This is why it is essential to have tools that allow for real-time visualization of message flow, detection of statistical anomalies, alerts for interruptions, and reprocessing of messages that were not delivered correctly.

Integration engines typically include basic monitoring capabilities, but there are also specialized tools that provide comprehensive visibility to the entire communication ecosystem, allowing technical teams to identify bottlenecks, analyze traffic patterns, and anticipate problems

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

before they impact clinical operations. The availability of these monitoring systems is just as important as the integration capability itself because in a hospital, a silent interface can mean that a physician is not receiving critical lab results.

Looking ahead, HL7 v2 will likely continue to exist for decades, simply because the cumulative cost of replacing all systems that use it is economically and operationally prohibitive. However, its role will gradually and predictably shift. From being the primary standard for new integrations, it will become the de facto standard for legacy systems, while new applications will adopt FHIR almost exclusively. Integration engines will become increasingly sophisticated universal translators, allowing both worlds to coexist without apparent friction for clinical users.

Developers who can navigate this technical duality—who understand both the grammar of HL7 v2 and the principles of FHIR—will be the most valuable professionals in a job market that desperately needs to build robust bridges between the past and the future of digital health. It is not about choosing a side in a standards war but about mastering the entire ecosystem to be able to offer solutions that work in the real world, not just in development labs.

7.3 DICOM: MEDICAL IMAGES AND BEYOND

"A medical image without context is like a diagnosis without a medical history: technically present, but clinically useless."

Imagine a radiologist sitting at his workstation, focused on interpreting a chest CT scan. The image he sees on the screen is far from a simple digital photograph like the ones we take with our phones. It is a complex web of multidimensional information: precise spatial coordinates on three axes, quantified X-ray attenuation values measured in Hounsfield units, time sequences in the case of a dynamic contrast-enhanced study, and an invisible yet crucial layer of metadata describing the patient, the equipment used, the acquisition protocol followed, and the specific technical conditions. This seemingly simple image lies behind a sophisticated standard that enables the CT scanner, the mass storage system, the specialized workstation, and the electronic health record to share not only the pixels but also all the context necessary to interpret them correctly from a medical standpoint. This standard is called DICOM, and it constitutes, after HL7 v2, the most widespread and fundamental protocol in the global healthcare ecosystem.

DICOM, an acronym for Digital Imaging and Communications in Medicine, represents the de facto international standard for comprehensive management, long-term storage, specialized printing, and securtion of medical images in clinical settings.(3) Developed and actively maintained by a joint committee that includes the Radiological Society of North America and the American College of Radiology, DICOM transcends the mere definition of a static file format. It is, in fact, a complete and living protocol that rigorously defines how imaging devices must communicate with one another, how images must be structured and stored to ensure their temporal integrity, how associated metadata must be organized to preserve the clinical context, and how the semantic consistency of the information must be ensured throughout its entire lifecycle. Any medical imaging device that aspires to interoperability—whether a simple digital X-ray, a sophisticated multislice computed tomography scanner, a high-field magnetic resonance imaging system, a Doppler ultrasound, or a digital mammography system—inevitably speaks the DICOM language.

7.3.1 THE PHILOSOPHY OF IMAGE-DATA INTEGRATION

What makes DICOM truly special and what radically sets it apart from conventional image formats such as JPEG or PNG is its ability to integrate both the image itself and all the contextual data necessary for its valid medical interpretation into a single standard container. A DICOM file contains not only the pixel matrix representing the visualized anatomy but also an extensive, structured header with complete patient demographic information, comprehensive technical data on the acquisition process, specific details of the equipment used—including calibrations and hardware parameters—the applied algorithmic reconstruction settings, and even semantic relationships with other studies of the same patient or from the same session. This intrinsic integration is not a technical luxury but an absolute necessity for safe and effective clinical practice.

A medical image stripped of its technical and clinical context is practically useless and even potentially dangerous if misinterpreted. Knowing that a certain pulmonary opacity appears in a CT image has no real diagnostic value if we do not know whether the study was performed

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

with or without intravenous contrast, at what specific phase of the respiratory cycle the slice was acquired to assess mobility, which reconstruction algorithm and which display window were used, or whether the patient was in the supine or lateral decubitus position. DICOM ensures that all this critical information travels inseparably attached to the image so that any system receiving it can accurately reconstruct the acquisition conditions and present it to the specialist with all the information necessary for an informed interpretation.

The internal structure of a DICOM file is organized hierarchically into functional modules that group semantically related attributes.

Its modules are as follows:

- Patient identification module: name, unique identifiers, date of birth, sex, and other essential demographic data.
- Study module: unique study identifier, date and time of acquisition, type of test requested, and referring physician.
- Series module: specific technical conditions for acquiring a related group of images.
- Image module: pixels and their technical encoding characteristics.

This modular organization is not arbitrary; it allows different specialized applications to efficiently access the information they need without having to process and parse the entire file.

An advanced visualization system can prioritize loading the pixels and the image module to quickly display the anatomy to the radiologist, while a file management system can index and search exclusively within the identifying metadata without needing to open and decode the full images, which are large. This functional separation optimizes performance in environments where millions of studies are managed and where different users have distinct access needs. The clinician needs to view the image; the system administrator needs to manage the metadata for archiving and retrieval.

One of the most fundamental and characteristic concepts in the DICOM universe is that of acquisition "modality" or "medium." Each type of medical imaging equipment is assigned its own modality, identified by a standardized three-letter code that allows for the immediate identification of the technical nature of the study. CR identifies traditional computed radiography, DX refers to modern direct digital radiography, CT designates computed tomography, MR corresponds to nuclear magnetic resonance, US identifies ultrasound or sonography, NM refers to nuclear medicine, and so forth for dozens of specialized techniques. This coding is not merely administrative; each modality has its own defined mandatory and optional specific attributes, reflecting the physical and technical characteristics inherent to each imaging method.

For example, a computed tomography image necessarily includes detailed information on the dose of ionizing radiation administered to the patient, expressed in multiple standardized metrics for dosimetric monitoring, whereas a magnetic resonance image includes complex technical parameters such as the pulse sequence used, repetition and echo times, magnetic field strength, and the characteristics of the receiving coils. This modality-specific semantic richness is precisely what makes DICOM so powerful for specialized clinical practice, but it also explains its considerable technical complexity and the learning curve it presents for developers new to the field.

7.3.2 SERVICE ARCHITECTURE AND PACS ECOSYSTEM

Communication between DICOM devices in a hospital network does not occur via simple file transfer but rather through a sophisticated system of standardized services that define which operations a system can perform on medical objects managed by another. The most fundamental and widely used services are the storage service, which allows an acquisition device to send images to a file server; the query service, which allows a viewer or workstation to ask what studies are available for a specific patient; and the retrieval service, through which the viewer requests and obtains the specific images from a particular study for display. These three services form the basic triangle of DICOM communication.

Beyond these essential services, the standard defines advanced capabilities for printing on film or high-quality diagnostic paper, for managing worklists——that allow radiology technicians to view pending requests directly on the device console, for performing complex procedures involving multiple steps, and for real-time communication of events and alerts between systems. This architecture of modular, well-defined services allows equipment from different manufacturers, designed in different decades and installed in different countries with varying regulations, to interoperate transparently and predictably, provided that both correctly implement the DICOM

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

conformance profiles.

The central system and nerve center of any modern medical imaging infrastructure is the PACS, an acronym for the Picture Archiving and Communication System. A PACS is an integrated software-hardware platform that receives, temporarily and permanently stores, efficiently distributes, and enables the visualization of medical images exclusively in DICOM format.(4) When a CT scanner completes the acquisition of a study, it automatically sends it to the institutional PACS using the DICOM storage service. From the PACS, any authorized viewing station on the hospital network—or even at secure remote locations—can browse the catalog of available studies and retrieve the specific images needed for diagnostic or clinical work.

PACS also assumes the critical responsibility of long-term storage management, implementing sophisticated data migration policies that automatically move images between high-speed, high-cost disks for frequently accessed recent studies and more economical but slower cold storage systems for older studies that are rarely consulted but must be retained for legal and clinical reasons for years or decades. Without a functional and properly scaled PACS, managing the massive volume of imaging data generated daily in a modern hospital would simply be impossible, and digital radiology as we know it could not exist.

The seamless integration of medical images with electronic health records is another crucial aspect of contemporary healthcare practice. Radiologists do not work in an isolated vacuum; they must have access to the patient's complete medical history, including known allergies—particularly to iodinated contrast agents—previous diagnoses that provide context for current findings and the results of other diagnostic tests. Conversely, clinicians who order imaging tests need to view radiological results coherently integrated into the patient's overall context, not in a separate, disconnected application. For this reason, modern health information systems integrate the DICOM viewer directly into the electronic health record interface.

On the same workspace screen, the physician can simultaneously view the structured radiology report and the images supporting it, navigate through the study's three-dimensional slices, perform basic measurements, and compare the results with those of previous studies of the same patient. This seemingly simple integration requires complex technical orchestration where the PACS and the EHR communicate continuously, typically using HL7 v2 or FHIR for the exchange of demographic and admission data and DICOM for the management of the images themselves. The user perceives a unified experience, but behind the scenes, there are two distinct technological worlds that must be perfectly synchronized.

7.3.3 MODERNIZATION: DICOMWEB AND ARTIFICIAL INTELLIGENCE

One of the most recent, promising, and transformative developments in the DICOM world is the DICOMweb specification ((5)), which represents a modern extension of the traditional standard () designed specifically for the web and cloud era. DICOMweb allows access to medical images and their associated metadata using the same RESTful APIs and exchange formats—such as JSON—that dominate contemporary application development. Traditionally, programmatic access to DICOM images requires specialized low-level libraries, deep knowledge of the original binary protocol, and considerable implementation complexity that limits the ability to develop medical applications.

With DICOMweb, any web developer with experience in standard technologies can build medical applications that access MR or CT images using exactly the same techniques and design patterns they would use to access any other resource on the internet. This democratization of access to medical images is enabling an explosion of innovation, ranging from fully browser-based viewers that require no plugins to sophisticated artificial intelligence tools capable of analyzing millions of images at a massive scale on cloud infrastructures. Specific DICOMweb services such as WADO-RS for object retrieval, QIDO-RS for querying, and STOW-RS for storage map traditional DICOM capabilities to modern web paradigms.(5)

Artificial intelligence and deep learning are radically and rapidly transforming the world of medical imaging, and the DICOM standard is actively evolving to adapt to this new reality. Modern deep learning algorithms can detect potentially malignant pulmonary nodules with sensitivity comparable to or exceeding that of humans, quantify coronary stenosis in angiograms with millimeter precision, automatically segment brain tumors for surgical planning, and perform countless other analytical tasks that previously required exclusively the expert interpretation of a radiologist. These algorithms require massive access to images for training and operation, but they also need to return their results in a structured format that can be seamlessly integrated into

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

established clinical workflows.

DICOM is progressively incorporating new specialized data structures to represent algorithmically generated anatomical segmentations, structured annotations of findings, derived quantitative measurements, and inference results from artificial intelligence models. These specialized DICOM objects allow AI findings to be permanently stored alongside the original images, displayed as overlays for radiologists on their standard workstations, and included in the patient's legal medical record with full traceability. AI ceases to be an external black box and becomes a first-class citizen of the DICOM ecosystem.

For software developers working specifically with medical images, DICOM undoubtedly presents a steep learning curve, but it is remarkably welldocumented and supported by a rich ecosystem of open-source tools. There are mature and actively maintained libraries such as DCMTK for C++, fo-dicom for the .NET environment, or pydicom for Python, which greatly facilitate the reading, writing, modification, and manipulation of DICOM files in virtually any modern programming language. There are also complete, open-source DICOM servers such as Orthanc or DCM4CHEE that provide reference implementations of the full protocol.(6)

These servers allow developers to focus on the specific logic of their innovative applications without having to implement the complex state machine and DICOM communication protocol stack from scratch. The quality of the processed images and the absolute accuracy of the associated metadata are critical aspects where no compromises are allowed in DICOM. A seemingly minor error in the spatial orientation of the image, a metadata tag misinterpreted by software, or a loss of clinical information during a compression process can have serious diagnostic consequences and even put the patient's life at risk if they lead to misinterpretation.

For this reason, DICOM systems include robust integrity verification mechanisms using checksums, and exhaustive conformance testing is an absolutely essential and nonnegotiable part of deploying any medical imaging system. The IHE initiative, which we already know for its work with HL7, also defines specific integration profiles for the DICOM ecosystem that specify how systems must behave in specific clinical scenarios such as mammogram reading or worklist management and offers rigorous testing tools to verify that systems technically comply with the specifications.

For physicians who use these systems on a daily basis, DICOM is invisible yet absolutely essential, such as oxygen for breathing. When a radiologist opens a complex study and sees the multiple image series perfectly organized chronologically, with patient data correctly displayed in the headers and with window and level manipulation tools working instantly, they are seeing the tangible result of decades of international standardization work. When a clinician accesses their patient's images from medical records and can navigate smoothly through the thin slices of a three-dimensional CT scan, they are utilizing a complex technological infrastructure that is only possible because of the existence and maturity of DICOM.

The standard's operational transparency—the fact that it simply works without the user having to think about it—is perhaps the best proof of its success and maturity. The future of DICOM is intrinsically linked to the very future of medical imaging as a discipline. Emerging technologies such as dual-energy computed tomography for tissue characterization, functional magnetic resonance imaging for brain mapping, high-resolution molecular imaging, or hybrid techniques such as PET–CT for simultaneous metabolic–anatomical diagnosis all find their reflection and support in new, carefully designed extensions of the standard. Advanced artificial intelligence, immersive three-dimensional visualization, and augmented reality for surgical planning depend fundamentally on DICOM to access the data they need to process and enrich.

Far from being a mature standard in the sense of being static or complete, DICOM continues to actively evolve to adapt to the new clinical and technical needs that constantly arise. In conclusion, it is worth remembering that DICOM is not merely an abstract technical specification but also a fundamental enabler of modern medicine as we practice it. Without DICOM, comprehensive digital radiology could not have developed, teleradiology for specialized support in remote areas would be practically impossible, and integrated multimodal imaging would remain an unfulfilled dream.

Every time a patient does not undergo an unnecessary repeat test because their images from another facility are accessible and viewable owing to standardization, every time a specialist can urgently consult a study from home after hours to make a critical decision, every time an artificial intelligence algorithm helps detect cancer in its early stages when it is still curable, we are witnessing the tangible fruits of this standard that has made universal interoperability in medical imaging possible. DICOM is, in this profound sense, one of those invisible infrastructure technologies that make everything else possible—the very thing we take for granted until it fails.

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

7.4 SMART ON FHIR: APPS THAT INTEGRATE WITH EPIC AND CERNER

> "The best health app in the world is useless if the doctor has to step out of their workflow to use it; seamless integration is the true battleground of clinical innovation."

Imagine a primary care physician who uses an electronic health records system from one of the major global providers, such as Epic or Cerner, on a daily basis. Dozens of essential features coexist on their screen: appointment scheduling, electronic prescribing, viewing lab results, writing clinical reports, and communicating with other professionals. One day, during a medical conference, they discovered an application developed by an innovative external team that calculates their patients' cardiovascular risk using a validated, novel algorithm that considers genetic factors and emerging biomarkers. To use this potentially useful tool, the doctor would have to completely step out of their usual work environment, open a separate browser window, manually enter all the demographic and clinical data for the patient they are seeing, wait for the calculation, note the result mentally or on paper, and then return to their main system to continue the consultation. This process is tedious, prone to transcription errors, consumes valuable time that cannot be devoted to the patient, and brutally disrupts the cognitive workflow that the physician has developed over the years. Most likely, after trying it a couple of times with initial enthusiasm, the physician will abandon its use and return to traditional methods—not because the application is bad but because the friction of use outweighs its potential value.

This scenario, which has played out countless times over the past decade in the history of digital health, illustrates a fundamental problem that developers and entrepreneurs in the sector have ignored for far too long: there is no point in having the best medical apps, the most accurate algorithms, or the most beautiful interfaces if they do not integrate seamlessly and naturally into the environment where healthcare professionals already work and spend most of their day. The barrier to adoption lies not in the quality of the software but in its operational ubiquity. SMART on the FHIR was created precisely to solve this integration problem in an elegant, standardized, and technically robust way. SMART, which stands for Substitutable Medical Applications, Reusable Technologies, is an open standard initially developed by researchers at Harvard and Boston Children's Hospital that allows applications developed by independent third parties to integrate securely, contextually, and seamlessly within leading electronic health record systems.(7)

Combined with the FHIR for structured access to clinical data, SMART provides a comprehensive framework that allows any developer—from a two-person startup to a large technology corporation—to create applications that can run directly withi, Epic, Cerner, Meditech, or any other system supporting the standard, behaving as if they were native features of the underlying system itself. The clinician can launch the application from a simple button or context menu within their usual interface, and the application automatically receives the full context of the patient they are currently viewing, including essential demographic and clinical identifiers, without requiring the clinician to manually enter any data. The transition between the main system and the integrated application is so seamless that the user perceives a unified experience, not two separate systems that require a mental shift in context.

7.4.1 THE FOUNDATIONAL PRINCIPLES OF THE ECOSYSTEM

The design philosophy of SMART on FHIRs is built on several key principles that together define a vision of open architecture for digital health. The first fundamental principle is substitutability: applications are explicitly designed to be replaceable by others that perform similar or equivalent functions, without the need to modify the underlying EHR system or reconfigure complex point-to-point integrations. This fosters genuine competition on the basis of clinical quality and usability and accelerates innovation, as EHR vendors can offer their customers a broad and diverse ecosystem of specialized applications without having to bear the cost and risk of developing them all in-house. A hospital can switch its cardiovascular risk calculation application to an improved one without having to touch its underlying Epic or Cerner infrastructure.

The second principle is cross-platform reuse: a single application, developed once according to SMART on FHIR standards, can run seamlessly across different EHR systems, provided that they correctly implement the standard. This drastically reduces development costs by eliminating the need to create platform-specific versions and allows the best applications—regardless of their geographic or corporate origin—to reach more clinical users worldwide. A Spanish developer can create a tool that works perfectly in U.S. hospitals using Epic or French hospitals using Cerner

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

without modifying a single line of code for each market.

The third principle is security by design: access to sensitive patient data is strictly controlled through modern authorization mechanisms based on OAuth2, derived from best practices in the web industry, and the patient can have transparent visibility into which applications are accessing their medical information and for what specific purpose. This security approach is not an afterthought but is built into the architecture from the ground up, recognizing that trust is the most valuable asset in health data exchange.

The technical architecture of the SMART on the FHIR combines several specialized components that work in coordination to deliver this integrated experience. At the heart of the system is the electronic health record system's FHIR server, which exposes structured clinical data via a modern RESTful API, accessible through standard HTTP calls. In addition to this FHIR database, a dedicated SMART authorization service is implemented that centrally manages which applications can access which data, under what contextual conditions, and for how long. The applications themselves are standard web applications, typically developed using technologies such as HTML5, and modern frameworks, which run in the user's browser or in a secure sandboxed container within the EHR system.

When the physician decides to launch an application from their usual interface, an OAuth2 authorization flow is automatically initiated, which, following authentication and with the user's explicit consent, grants the application a temporary access token limited specifically to the current context: the particular patient the physician is seeing at that moment, for the duration of the consultation or the configured session time. This token does not allow the application to access other patients or unrelated historical data, thereby limiting the risk in the event of a compromise. Once authorized, the application can make calls to the FHIR API to obtain the necessary data, process it according to its specific logic, and present results or collect additional input from the physician, all within the same window or panel integrated into the main interface.

One of the most important and distinctive concepts in SMART on FHIR is that of "launch" or "launch context". When an application is launched from the host EHR system, it receives not only the technical access token but also a rich set of information about the current operational context: which specific patient the clinician is currently viewing, the authenticated user's identifier and professional role, the specific visit or care episode they are in, and even the location or department. This contextual information allows the application to configure itself automatically and intelligently, immediately displaying the correct patient data, applying known user preferences, and positioning itself at the appropriate point in the clinical workflow without requiring the physician to make any manual selections or enter identifiers.

The result is a remarkably seamless user experience, where the external application feels like a natural and organic extension of the core system, not an external intrusion. The clinician does not experience the sensation of "switching programs" that generate so much resistance to the adoption of new tools; they simply access additional functionality that magically appears contextualized with the information they are already viewing. This architectural transparency is precisely what makes the clinical adoption of external innovations viable in an environment where the clinician's time and attention are scarce and precious resources.

7.4.2 THE APPLICATION ECOSYSTEM AND THE DEVELOPER EXPERIENCE

The practical and successful use cases of SMART on FHIRs are extraordinarily diverse and have grown exponentially since the standard matured. There are sophisticated clinical decision support applications that calculate specific disease risks, suggest treatments on the basis of updated clinical practice guidelines, or provide real-time alerts about potentially dangerous drug interactions that the base system does not detect. There are specialized applications for the comprehensive management of chronic diseases that help monitor the longitudinal progression of patients with diabetes or hypertension, visualizing trends in clinical values and suggesting preventive interventions based on predictive algorithms. There are applications aimed at clinical research that allow for the identification and recruitment of patients potentially eligible for clinical trials directly from the daily clinical setting without interrupting the flow of care.

Applications for patient education are also proliferating, automatically generating materials tailored to a patient's specific condition, reading comprehension level, and language preferences or advanced visualization tools that transform complex laboratory or imaging data into intuitive graphical representations for the clinic. The list is practically endless and grows every day as more developers discover the platform's possibilities and the relative ease with which they can bring

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

their innovations into the real clinical setting, without the commercial and technical barriers that previously impeded market access.

For software developers, SMART on the FHIR drastically reduces the historical barriers to entry into the digital health market, which for years was dominated by a few large vendors with closed APIs and opaque integration processes. Before this standard existed, creating an application that integrated reasonably well with Epic or Cerner required complex commercial negotiations that could last years, costly specific partnership agreements, and painful technical adaptations to each vendor's proprietary APIs and incomplete documentation. Each integration was a custom consulting project, not a scalable product. With SMART on the FHIR, any developer with standard modern web development skills can create an application that works on any system supporting the standard, using tools and frameworks they already know.

Detailed implementation guides are publicly available at no cost; there are open-source libraries in multiple languages that facilitate OAuth2 authentication and access to FHIR data, and there are public test environments such as the SMART Launcher where developers can validate their applications with realistic synthetic data before attempting to deploy them in hospital production environments. This democratization of access to the ecosystem has sparked an explosion of innovation from diverse sources: agile startups, hospital IT departments, academic research groups, and even individual developers with specific ideas for particular clinical niches.

Security and privacy are absolutely central to the design of the SMART on the FHIR, not secondary considerations. Applications never have direct access to the underlying electronic health record database, not even in read-only mode; they can only access data through the controlled FHIR API under the constant supervision of the SMART authorization service. Permissions are granted via scope profiles that define, with granular detail, which FHIR resources the application can read or write and in which specific patient or user contexts. The patient can be transparently informed about which applications have access to their data and for what stated purposes and can revoke that access at any time through consent management interfaces.

All access by applications is automatically recorded in immutable audit logs that enable the detection of inappropriate use, anomalous patterns, or violations of institutional policies. This security model, derived from and adapted to best practices in the modern web industry and standards such as OAuth2 and OpenID Connect, provides an adequate and auditable level of protection for sensitive health data, complying with regulations such as HIPAA in the United States or GDPR in Europe, although it is always the responsibility of the end implementer to ensure specific compliance with each jurisdiction.

For large EHR systems such as Epic and Cerner, adopting and promoting SMART on FHIR has been a strategic decision that balances competition and collaboration. It allows them to offer their institutional clients a much richer, more diverse, and up-to-date ecosystem of applications than they could develop internally with their own resources, without having to assume the cost of development, maintenance, and legal liability for each individual application. It also allows them to maintain their position as a central and indispensable platform within the hospital's architecture, since the applications run within their technological environment and rely fundamentally on their APIs to function, thus creating a symbiotic relationship where the success of external applications reinforces the value of the base system.

Major vendors have created their own certified app stores, such as Epic's App Orchard or Cerner's Code, where customers can discover, evaluate, and install SMART on FHIR applications that have passed security reviews, technical compliance checks, and basic clinical validation. These stores operate similarly to smartphone app stores but with more rigorous validation standards given the risks inherent in medical applications. For developers, having a presence in these stores represents a massive and highly credible distribution channel.

7.4.3 ARTIFICIAL INTELLIGENCE AND THE FUTURE OF CLINICAL INTEGRATION

The relationship between SMART on FHIRs and artificial intelligence applied to healthcare is particularly promising and is generating some of the most sophisticated applications in the ecosystem. Machine learning and deep learning models can be easily deployed as SMTs in FHIR applications that, launched directly from the electronic health records, analyze structured patient data in real time and return probabilistic predictions, management recommendations, or risk alerts that are immediately integrated into the clinical decision-making workflow. A radiologist could use an application that employs computer vision to detect and quantify suspicious pulmonary nodules while viewing CT images, receiving an overlaid annotation without leaving

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

their usual viewer.

A cardiologist could use an application that estimates the risk of major cardiovascular events over the next five years on the basis of the patient's complete history, recent biomarkers, and lifestyle factors, presenting the result as an interpretable probability with highlighted contributing factors. A primary care physician could use an application that suggests specific follow-up tests according to the most up-to-date clinical guidelines, automatically checking for contraindications and prioritizing recommendations on the basis of the patient's profile. All of this occurs without the professional having to leave their usual work environment, with the patient's context already loaded and available, reducing the friction of adopting these emerging technologies to a minimum.

For physicians using these applications in their daily practice, the ideal experience should be completely transparent and technologically invisible. A discreet yet accessible button on the main screen, a context menu that appears at the right moment, an additional tab in the patient interface—and the application is there, offering its specific functionality without perceptible friction or steep learning curves. What happens behind this seemingly simple interface—the complex dance of OAuth2 authorizations, calls to REST APIs, secure data transfer, and algorithmic processing—must remain completely invisible to the clinician. When a SMART on an FHIR application works well, the clinician does not even consciously perceive it as something separate or external; it is simply another feature of their system, as natural as any other feature they use routinely.

This is the design ideal to which SMART on FHIRs aspires: that technological integration be so natural and seamless that it disappears from the user's awareness, and the only thing that remains is tangible clinical utility and improved efficiency or quality of care. The technology becomes transparent, and what remains is pure medical value.

Application governance and curation are critical aspects that hospitals and healthcare systems must carefully consider before opening their doors to this ecosystem. Not all SMTs in FHIR applications are equally reliable, clinically validated, or secure, and the use of an open standard does not guarantee their quality, efficacy, or safety for patients. Hospitals need to establish clear institutional processes to evaluate applications before allowing their productive use, verifying who developed them and their reputation in the market, what specific data they access and whether that access is proportionate to the stated purpose, what guarantees of technical support and updates they offer, and how security vulnerabilities discovered later are managed.

Some electronic health record (EHR) system vendors offer certification programs that help hospitals select prevalidated, trusted applications, but the ultimate responsibility for patient safety and regulatory compliance always rests with the institution authorizing the deployment. Governance is not an obstacle to innovation but rather a necessary framework for ensuring that innovation is sustainable and safe.

The evolutionary future of SMART on FHIRs is intrinsically linked to the future of FHIRs as a data standard and to that of EHR systems as open platforms. As more vendors—including those outside the U.S. market dominated by Epic and Cerner—adopt the standard and as more advanced functionalities are exposed through increasingly rich FHIR APIs, the application ecosystem will continue to grow in complexity and sophistication. We are likely to see the emergence of entire categories of applications that we cannot even fully imagine today, leveraging the convergence of massive amounts of clinical data, advanced artificial intelligence, applied genomics, and integrated user experience. SMART on the FHIR is not just a technical specification; it is a vision of how innovation in digital health can be democratized and distributed, allowing the world's best developers, regardless of their location or size, to contribute to improving global healthcare.

SMART on the FHIR, while powerful, does not magically solve all healthcare integration problems. It remains absolutely necessary for EHR systems to expose the right, high-quality data through their APIs; for FHIR profiles to be welldefined and consistently implemented; for authorization services to function correctly without creating bottlenecks; and for the applications themselves to be clinically valid, usable, and maintained. However, SMART on FHIR provides the essential architectural framework that makes everything else possible—the technical and semantic scaffolding upon which to build an open, competitive, and secure innovation ecosystem. In a world where medical technology is advancing ever faster and where the need for specialized solutions exceeds the capacity of any individual provider, having a standard that allows new solutions to be seamlessly and securely integrated with existing ones is likely the most valuable architectural contribution we can make to the future of digital health.

If SMART on the FHIR has succeeded in opening up electronic health record systems to external innovation, how should we redefine models of clinical validation and legal liability when

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

these third- r applications make decisions that directly affect patient safety, and who bears the blame when an integrated algorithm makes a diagnostic error within an established provider's environment?

7.5 APIS FOR WEARABLE DEVICES AND CONNECTED DEVICES

> "The watch on the patient's wrist generates more data in a single day than their entire previous medical history; the challenge is not a lack of information, but transforming that torrent into actionable medical knowledge."

Imagine a patient with chronic heart failure who has just been discharged from the hospital after being treated for acute decompensation. On their wrists, they now wear a state-of-the-art smartwatch that continuously monitors not only their heart rate and physical activity level but also, using advanced bioimpedance and signal analysis algorithms, estimates subtle changes in their total body fluid volume—one of the earliest indicators of cardiac deterioration. In the bathroom of their home, a connected smart scale automatically records their body weight every morning before breakfast, transmitting the readings to the cloud. On their night stand, an automatic wrist blood pressure monitor takes scheduled readings of blood pressure and resting heart rate. All this information, which is continuously generated in the patient's natural environment, could be extraordinarily valuable to their primary care physician and cardiologist, allowing them to detect early signs of cardiac decompensation days before the patient notices subjective symptoms and thus avoiding costly and potentially dangerous hospital readmission. However, for this vision of predictive medicine to become a reality, the data generated by these heterogeneous devices must traverse multiple technological layers, ultimately reaching the institutional electronic health record system, integrate coherently with the rest of the patient's information, and be presented in a way that allows the physician to interpret it quickly and act on it with confidence. This is exactly what wearables and connected device APIs are about: building technical and semantic bridges between the consumer ecosystem of personal devices and professional healthcare infrastructure.

The massive explosion of connected devices in the health and wellness sector represents one of the most transformative technological phenomena of the past decade in medicine. It is estimated that hundreds of millions of active wearables are already in use globally, ranging from multigenerational smartwatches to adhesive biosensor skin patches, continuous ambulatory glucose monitors, smart inhalers with adherence sensors, and long-term implantable cardiac monitoring devices. All of these factors generate a continuous, voluminous, and multidimensional flow of data that, when properly processed, contextualized, and integrated, can revolutionize population-based preventive medicine, the proactive management of complex chronic diseases, and the conduct of large-scale clinical research using real-world data. However, this tsunami of personal data also poses formidable challenges regarding technical interoperability, information security, clinical validation of algorithms, and ethical privacy management on a scale that traditional healthcare systems have never faced before.

7.5.1 THE TOWER OF BABEL OF MANUFACTURERS AND THE EMERGENCE OF STANDARDS

The first unavoidable issue that arises when seriously addressing the integration of wearables is the overwhelming diversity of manufacturers, hardware architectures, and proprietary communication protocols. Through its Fitbit and Android ecosystem, Samsung, Garmin, Withings, Oura, and dozens of other specialized manufacturers all have their own devices with specific sensors, their own companion mobile apps with distinctive user interfaces, and their own idiosyncratic ways of exposing data to external developers. For a developer aiming to build a clinical application that universally collects information from all of them, the initial task can seem daunting, requiring the mastery of multiple SDKs, different authentication models, and the mapping of incompatible data formats. Fortunately, in recent years, standardization initiatives and intermediary platforms have emerged that progressively facilitate this integration, reducing the fragmentation that threatens to paralyze the ecosystem.

The most prominent platforms in this space are Apple's HealthKit, which provides a centralized repository of health data on iOS devices, and Google Fit, which offers similar functionality for the Android ecosystem. These operating system APIs allow authorized applications to read and write a wide variety of standardized data types, such as step count, resting and active heart rate, sleep patterns and quality, body weight, blood glucose levels, blood pressure, oxygen saturation,

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

and many other biometric parameters. These APIs follow modern authorization models based on granular permissions, where the user maintains explicit control over which specific apps can access which categories of sensitive data and for how long and can revoke these permissions at any time from a centralized privacy interface.

The next critical step in the integration chain is to transport these data from consumer manufacturers' applications to institutional electronic health records systems. This is where the FHIR standard, which we have explored in the previous sections, plays an absolutely central role as the lingua franca of digital health. An intermediary application can read data from the wearable via the manufacturer's API or the operating system's health platform, semantically transform it into standardized FHIR resources (observation for point-in-time or time-series biometric readings, questionnaire response for health questionnaire responses, device for information on the hardware used, etc.), and finally send it to the hospital's FHIR server using precisely the SMART on the FHIR APIs we developed in the previous section. The architectural result is that data generated by the patient in the privacy of their home during their daily activities are integrated into their official electronic health records in the same structured and secure manner as laboratory results processed at the hospital or imaging tests performed in radiology.

However, the technical integration does not end there, and in fact, that is only the beginning of the potential value. Once the data from wearables are structured within the electronic health record, they can be leveraged for multiple sophisticated clinical purposes. It can feed remote monitoring dashboards that the physician systematically reviews before a scheduled visit, providing a longitudinal context that enriches the in-person consultation. It can generate intelligent automatic alerts when analysis algorithms detect abnormal patterns or concerning trends, such as sudden weight gain in a cardiac patient, suggesting fluid retention. It can be incorporated into machine learning predictive models that dynamically stratify patient risk and prioritize proactive care. They can even be used to adjust pharmacological treatments remotely—through automated titration protocols—without the need for the patient to physically attend a consultation. The ultimate value of wearable data does not lie in the bits and bytes themselves but in the preventive and personalized clinical actions they enable.

7.5.2 REGULATED MEDICAL DEVICES AND DATA QUALITY

One of the most advanced and clinically mature use cases involves connected medical devices that, unlike general-purpose wearables, already incorporate capabilities for direct and secure communication with professional healthcare systems, designed from the outset for clinical integration. Modern pacemakers with telemetry capabilities can send detailed automatic reports to cardiologists when their internal algorithms detect significant arrhythmias or changes in pacing parameters. Next-generation continuous glucose monitors can share real-time readings not only with the patient but also with the hospital's multidisciplinary diabetes team, enabling immediate insulin adjustments. Smart inhalers with usage sensors can accurately record when, where and how the patient uses them, helping to identify patterns of environmental triggers for asthma attacks and assess actual treatment adherence. These devices are not mass-market wearables but are regulated medical devices with CE marking or FDA clearance, and their APIs are specifically designed to meet the safety, reliability, and traceability requirements of professional clinical settings.

Information security and patient privacy are, as in the entire digital health ecosystem, absolutely critical and nonnegotiable aspects. Even if it comes from consumer devices, the data generated by wearables are unquestionably sensitive personal health data and are therefore subject to the same strict regulatory protections as any other traditional clinical data. This means that all communications between devices, intermediary applications, and hospital systems must be encrypted using robust protocols such as TLS; access must be rigorously authenticated through mechanisms such as OAuth2 and managed via role-based authorization; and there must be immutable audit logs that allow tracking of who accessed which specific data and at what time. It also means that the patient must maintain full control over their data and be able to make informed decisions about what to share, with which institutions or applications, for how long, and for what specific purposes. Wearable manufacturers' platforms typically provide user interfaces for patients to manage these permissions in a granular manner, but the ultimate responsibility for technically respecting these consent decisions lies with the applications that consume them and the systems that store them.

The clinical quality and metrological reliability of wearable data is another aspect that

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

deserves critical attention and is often underestimated by technology enthusiasts. Not all devices available on the market are equally accurate, validated, or robust, and real-world patient usage conditions can significantly affect the quality of measurements. A smartwatch that measures heart rate using photoplethysmography can produce erroneous readings or artifacts if it is not properly positioned on the wrist, if the patient moves vigorously during the measurement, or if the skin is damp. An activity tracker might count movements that are not actually walking—such as arm swinging while driving—as steps. Clinical systems that receive and interpret these data must be explicitly aware of these technical limitations and present them to healthcare professionals with the appropriate caveats and context to enable informed critical interpretation.

A physician who sees an anomalous reading on a telemonitoring dashboard must be able to easily access the relevant technical context: is this an isolated single value or a sustained trend over several days? Was the device recently calibrated according to the manufacturer's specifications? Are there environmental or usage factors that could explain an apparently erroneous reading? What is the sensor's stated accuracy for the observed range of values? Without this contextualization, the risk of overdiagnosis—of false alarms that cause patient anxiety and strain the system—or of underdiagnosis due to widespread mistrust is considerable.

The integration of data from wearables also poses significant technical challenges in terms of volume, velocity, and variety that traditional health information systems were not designed to handle. A patient with an ambulatory continuous glucose monitor can easily generate hundreds of individual readings per day, each with multiple associated data dimensions. Multiplied by thousands of patients being monitored simultaneously in a region or healthcare system, the cumulative volume of data can reach large data magnitudes that challenge conventional clinical database architectures.

Systems must be capable of ingesting, storing, and processing this massive volume without degrading the operational performance of critical clinical applications, and they must present the resulting information in an aggregated and summarized form that is practically useful to clinicians, who obviously cannot manually review hundreds of individual readings per patient during a fifteen-minute consultation. This requires sophisticated capabilities for temporal aggregation, trend visualization, intelligent alert thresholding, and analysis algorithms that proactively highlight what is relevant and suppress noise.

7.5.3 THE FUTURE OF CONNECTED, PATIENT-CENTERED MEDICINE

For software developers working specifically with wearables and connected devices, the variability of APIs and formats remains a constant operational challenge that requires flexible and well-designed software architectures. Each manufacturer maintains its own way of semantically representing data, its own authentication and authorization mechanisms, its own rate-limiting and access quota restrictions, and its own evolving privacy policies. Fortunately, specialized libraries and aggregation services are emerging in the market that abstract much of this underlying complexity, providing a unified and standardized interface for multiple devices and manufacturers. Industry-level standardization initiatives are also emerging, such as those promoted by the Personal Connected Health Alliance, which seek to define common formats and interoperable protocols for wearable data, facilitating their seamless integration with institutional healthcare systems and reducing the development burden for integrators.

The synergistic relationship between wearables and telemedicine is particularly relevant and transformative for the future of care models. Compared with traditional sporadic in-person visits, connected devices effectively function as the physician's extended eyes, ears, and senses in the patient's home, within their natural living environment, providing continuous and contextualized information that enables much closer, more frequent, and personalized clinical monitoring. This is particularly valuable for patients with complex chronic conditions requiring intensive monitoring; for frail elderly individuals living alone, early detection of functional decline can prevent institutionalization; for patients in remote rural areas with limited geographic access to healthcare facilities; and for populations with reduced mobility. The intelligent combination of wearables, synchronous and asynchronous telemedicine, and clinical decision algorithms promises a more proactive, preventive, personalized, and equitable healthcare system.

For patients who generate these data, the ability to actively share them with their doctors can be both an empowering opportunity and a legitimate source of anxiety and concern. On the one hand, it allows them to feel more involved and engaged in their own healthcare, better monitored and safer, and more connected to their care team between in-person visits. On the

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

other hand, it can raise valid concerns about the privacy of their most intimate data, about who else might access it without their knowledge, about how it will be used in the long term, and about the risks of discrimination or stigmatization if certain sensitive data are leaked or used out of context. It is the ethical and legal responsibility of developers and healthcare systems to design user experiences that fully respect patient autonomy, are radically transparent about the specific use of data, and offer clear, simple, and effective controls to manage sharing permissions at any time.

The future of wearables and connected health devices is extraordinarily promising and is accelerating continuously. Devices will become progressively more metrologically accurate, smaller and more discreet, and more autonomous in terms of battery life and local processing capacity. They will measure an increasing number of physiological parameters with greater clinical reliability over longer periods without the need for recharging or recalibration. They will integrate more naturally and invisibly with smart clothing, with everyday objects in the connected home, and even with biocompatible subcutaneous implants. They will generate volumes of personal data that seem unimaginable today but that will be the essential raw material for truly personalized, predictive, and preventive medicine. For developers who know how to navigate this complex ecosystem of heterogeneous APIs, emerging standards, healthcare regulations, and ethical considerations, the opportunities for positive clinical impact are immense and are only just beginning to be explored.

Technology, however sophisticated it may be, remains merely a means to a higher end. The ultimate goal of integrating wearables and connected devices with healthcare systems is not simply to accumulate more data for their own sake but also to empower professionals to make better clinical decisions and patients to better manage their health. Developers building these integrations must always maintain an obsessive focus on tangible clinical value—on how the information flowing from patients' devices can concretely help professionals provide better, more personalized, and more proactive care. That is, and must remain, the ethical and practical compass guiding all innovation efforts in this space.

If devices in patients' homes generate potentially critical clinical data that flows continuously into hospital systems, how can we redefine the boundaries of medical liability and professional–patient relationship models when monitoring is constant and when intervention can be remote and even automated?

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

REFERENCES

Ayaz M, Pasha MF, Alzahrani MY, Budiarto R, Stiawan D. The Fast Health Interoperability Resources (FHIR) Standard: Systematic Literature Review of Implementations, Applications, Challenges and Opportunities. JMIR Med Informatics [Internet]. July 1, 2021 [cited May 13, 2026];9(7):e21929. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC8367140/

Simborg DW. Reflections on the history of interoperability in hospitals. J Am Med Inform Assoc [Internet]. December 1, 2023 [cited May 13, 2026];30(12):2083. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC10654835/

Bidgood WD, Horii SC, Prior FW, Van Syckle DE. Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging. J Am Med Informatics Assoc [Internet]. 1997 [cited 2026 May 13];4(3):199. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC61235/

Andriole KP. Picture archiving and communication systems: past, present, and future. J Med Imaging [Internet]. December 28, 2023 [cited May 13, 2026];10(6):061405. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC10754358/

Genereaux BW, Dennison DK, Ho K, Horn R, Silver EL, O'Donnell K, et al. DICOMweb™ : Background and Application of the Web Standard for Medical Imaging. J Digit Imaging [Internet]. June 1, 2018 [cited May 13, 2026];31(3):321. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC5959831/

DCMTK - dicom.offis.de [Internet]. [cited 2026-05-13]. Available from: https://dicom.offis.de/dcmtk.php.en

Mandel JC, Kreda DA, Mandl KD, Kohane IS, Ramoni RB. SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. J Am Med Inform Assoc [Internet]. September 1, 2016 [cited May 13, 2026];23(5):899-908. Available at: https://pubmed.ncbi.nlm.nih.gov/26911829/

https://doi.org/10.62486/978-9915-9876-8-2.Ch07

Chapter 8 / Capítulo 8

Check for updates

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch08
Pages: 133-150

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 8. Architecture in the Healthcare Environment

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Software architecture is not an abstract design exercise but rather the structure that determines how a system will respond when it is needed most. In a hospital, where every second counts and every failure has consequences, choosing the right architecture is not just another technical decision: it is a clinical decision with a capital C."

Imagine a hospital information system that manages appointments, medical records, prescriptions, lab results, images, billing, and dozens of other features. For years, this system has operated as a monolith: a single application that does it all, with a central database, and a development team that maintains the whole system. One day, someone proposes changing how appointments are managed, adding artificial intelligence features to optimize the schedule. However, making that change requires modifying the entire monolith, with the risk of affecting other parts of the system. The team hesitates, deadlines are pushed back, and the improvement never materializes. This scenario, which is common in many hospitals, illustrates the fundamental dilemma of software architecture in healthcare settings: How do we balance the need for stability and reliability with the need to evolve and innovate?

Software architecture is the discipline concerned with the structure of systems—how their components are organized, how they relate to one another, and how they evolve over time. In the healthcare sector, architectural decisions have implications that go beyond the technical: they affect the ability to integrate with other systems, the ease of incorporating new features, resilience against failures, and ultimately, patient safety. That is why it is so important for both developers and clinicians to understand the fundamentals of architecture—not to make technical decisions but to participate in the conversations that will determine what the systems they use every day will look like.

In this chapter, we explore the key architectural decisions faced by teams building healthcare software. We begin with the classic debate between monoliths and microservices, analyzing what makes sense in a hospital setting. We will continue with the choice between cloud computing and edge computing, considering the implications for latency, privacy, and processing. We address the particularities of clinical databases, which must handle sensitive information with very demanding availability and traceability requirements. Finally, we will reflect on how to design systems with interoperability in mind from the start rather than as a later addition. This is not an exhaustive technical guide but rather an overview that allows readers to understand the options and their implications.

8.1 MONOLITH VS. MICROSERVICES: WHAT MAKES SENSE HERE

There is a recurring and sometimes heated debate in the world of software development that pits two radically different architectural philosophies against each other, each with its own traditions, tools, and battlegrounds. On the one hand, monoliths: single applications that concentrate all functionality into a single executable block, with a single central database, and a unified, sequential development, testing, and deployment process. On the other hand, microservices are sets of small, independent, and autonomous applications, each with its own dedicated database, that communicate with one another via network APIs and that can be developed, deployed, and scaled separately, even by different teams. Each approach has its own staunch advocates who proclaim its absolute virtues and its detractors who point out its fatal flaws, and the choice between one or the other can profoundly shape a system's entire future development, its operating costs, and its capacity for evolution. In the healthcare sector, however, this choice takes on additional nuances of criticality, regulation, and operational context that should be analyzed carefully and without dogmatism imported from other, less regulated sectors.

Monoliths have a long and venerable tradition in hospital software, and this is neither by chance nor due to technological conservatism. The large electronic health record systems that dominate the global market, such as Epic, Cerner, and Meditech, are essentially monolithic architectures: massive, mature applications that integrate hundreds of features into a single, coherent technological environment. This approach offers clear operational advantages that explain its enduring popularity. The first is systemic consistency: since all code and data reside within the same logical system, the end-user experience is uniform and predictable, data are centralized without synchronization issues, and functionalities are integrated naturally through internal program calls rather than through complex network orchestrations. The physician navigates a familiar environment where the prescription, laboratory results, diagnostic imaging, and discharge report share the same interface, the same database, and the same security model.

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

The second advantage is relative operational simplicity: managing a single monolithic system, even a large one, is conceptually simpler than managing dozens of independent services that interact in complex ways. The backups are from a single database, disaster recovery plans focus on a single point, and performance monitoring has a single target. The third advantage, crucial in clinical settings, is latency performance: internal communications between modules within a monolith are carried out via calls to functions in shared memory, which are orders of magnitude faster than HTTP calls to remote services—a factor that can be decisive in situations where the speed of information access is critical for making urgent medical decisions.

However, monoliths also carry significant disadvantages that have become more evident and costly over time and as expectations have evolved. The first is structural rigidity: modifying any part of the system, even for localized improvement, requires touching the whole, recompiling, thoroughly testing the entire system, and deploying in bulk, with the inherent risk of seemingly unrelated functionalities being affected. This makes innovation slow, costly, and feared because the cost of an error in production is systemic. The second drawback is granular scalability: if a specific feature, such as querying lab results during an epidemic outbreak, requires more computational resources, the entire monolith must be vertically scaled, even if the rest of the modules do not need it, which is economically inefficient. The third is limited resilience: a catastrophic failure anywhere in the code, such as a memory error or an unhandled exception, can reduce the entire monolith, affecting all functionalities simultaneously and leaving the hospital without an information system. In a healthcare environment where 24/7 availability is critical for patient safety, this is particularly problematic and feared.

8.1.1 THE PROMOTION OF MICROSERVICES AND THEIR RISKS

Microservices emerged precisely in the broader technology industry to address these structural limitations of large, mature monoliths. Instead of a single massive application, many small, specialized applications are built, each responsible for a specific business function: patient demographic management, appointment scheduling, electronic prescribing, results viewing, billing, etc. Each microservice has its own database optimized for its use case, its own potentially autonomous development team, and its own independent deployment lifecycle. They communicate with each other via well-defined APIs, typically using HTTP REST and formats such as JSON or Protobuf, over an internal network or even the internet. This architecture offers significant advantages in terms of flexibility for evolution, independent scalability, and resilience to partial failures.

Flexibility is perhaps the advantage most valued by agile development teams. With microservices, a component can be modified, refactored, or even completely replaced without affecting the rest of the ecosystem, provided that the API contract is respected. A hospital's team might decide to modernize its appointment management system by hiring a specialized team to develop a new microservice from scratch using current technologies, while the rest of the system continues to operate stably with legacy components. This allows for continuous, gradual evolution without the need for costly and risky full-system migrations. It also facilitates the adoption of new technologies without the accumulation of technical debt: a microservice can be implemented in a modern language such as Go or Kotlin or a framework such as Node.js—different from the rest of the ecosystem—if that provides productivity or performance benefits for that specific use case.

Independent scalability is another key operational advantage. If a specific feature—such as the bulk querying of lab results during a pandemic where thousands of tests are performed daily—experiences extraordinary spikes in demand, additional instances of that specific microservice can be automatically deployed using orchestrators such as Kubernetes, without the need to vertically scale the entire monolithic system. This allows for a much more efficient and cost-effective use of infrastructure resources, especially in public cloud environments where one pays strictly for consumption, and facilitates adaptation to unpredictable future growth, since each component can scale horizontally according to its own demand curves.

Resilience is perhaps the most significant advantage from a purely clinical and patient safety perspective. In a well-designed microservice architecture following patterns such as bulkheads and circuit breakers, the catastrophic failure of one component should not propagate or affect the other services. If the scheduled appointment management microservice decreases because of a coding error, patients can still be seen in the emergency department, and the emergency admission service continues to function perfectly because they are independent services. If the electronic prescription service experiences a temporary outage, laboratory results

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

remain available for review, and imaging reports can still be viewed. This ability to isolate failures and limit the "surface area" of an incident is crucial in an environment where the continuous availability of certain functionalities can literally be a matter of life and death. Naturally, there are functionalities that maintain critical dependencies on one another, and a failure in a truly central service can have serious systemic consequences, but a good microservice architecture intentionally minimizes those hard dependencies through decoupling patterns.

However, microservices are not a universal magic solution, and their own significant drawbacks are often underestimated in the initial adoption phase. The main problem is explosive operational complexity. Managing dozens or even hundreds of independent services in production, each with its own database, its own environment configurations, its own potentially asynchronous deployment cycles, and its network interdependencies, is orders of magnitude more complex than managing a monolith, even a large one. Sophisticated and mature tools are required for container orchestration, dynamic service discovery, centralized distributed monitoring, aggregated log management, and handling of cascading failures. A platform team with advanced skills in these areas is needed, along with a reliability engineering culture that many traditional healthcare organizations simply do not possess. For small organizations or those with limited IT resources, this complexity can be prohibitive and create more problems than solutions.

Another inherent drawback is the introduced network latency. Communications between microservices, even on a fast internal network, occur via network protocols that introduce millisecond-level delays that do not exist in internal process calls within a monolith. In most administrative use cases, these delays are perfectly tolerable, but in real-time critical applications—such as continuous patient monitoring in intensive care units where responses are required in fractions of a second—they can be problematic or even unacceptable. Data consistency must also be considered: Since each microservice has its own isolated database, maintaining transactional consistency between them requires the implementation of additional complex patterns, such as distributed sagas or eventual consistency with trade-offs, which add considerable complexity to the design and are difficult to debug when they fail.

8.1.2 CONTEXTUAL DECISIONS AND HYBRID ARCHITECTURES

In the specific context of the healthcare environment, which architectural approach makes the most sense? The honest answer, as is often the case in mature software engineering, is that it depends critically on multiple contextual factors. It depends on the size and complexity of the organization, the available technical and human resources, the criticality and diversity of the applications to be supported, the required pace of innovation, applicable regulatory constraints, and many other organizational and cultural factors. What is absolutely clear is that there is no single, universally optimal solution and that the decision must be made after a careful, multidisciplinary analysis of the specific circumstances, not by following technological fads or vendor pressures.

For relatively small information systems, such as those of a medium-sized regional hospital, a specialized clinic, or a primary care center, a well-designed monolith remains perfectly adequate and is often preferable. Operational simplicity, user consistency, and the reduced need for massive scalability make the disadvantages of monoliths less relevant, while the complexity of managing microservices would be an unnecessary burden. Furthermore, limited IT resources in these environments make the cognitive overhead of microservices difficult to bear without risk. In these cases, the sensible choice is to opt for a monolith, but one designed internally in a modular fashion, with well-defined interfaces between its logical components, following clean architecture principles, to facilitate a potential future migration to microservices should the organization's needs change radically.

For very large and complex systems, such as those of large university hospitals with multiple campuses, integrated regional healthcare networks, or national digital health platforms, microservices can offer significant strategic advantages that justify the investment in complexity. The ability to evolve components independently, to scale selectively according to actual load requirements and to isolate failures to maintain continuity of care amply justifies the investment in platform capabilities and specialized talent. However, it is crucial to address this complexity systematically and progressively, investing seriously in the necessary orchestration, observability, and security tools, and designing the architecture carefully from the outset using proven patterns. Chaotic or rushed migration to microservices, if poorly executed, can be worse operationally than a well-managed, mature monolith.

Pragmatic hybrid approaches may also be suitable for many intermediate scenarios. For

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

example, maintaining a stable monolithic core containing the most critical, mature, and stable functionalities of the clinical business and implementing peripheral functionalities—those that change frequently, experience variable load spikes, or require different specific technologies—as independent microservices. Alternatively, using a "minimodules" or macroservice architecture within a logical monolith, which allows for some fault isolation and independent deployments, facilitates a potential full migration to microservices if conditions warrant it in the future. The important thing is not to dogmatize or idealize any architecture but to evaluate each specific case using technical and clinical criteria and to make the decision that best serves the present and foreseeable future needs of the organization and its patients.

For clinicians and healthcare professionals involved in these architectural debates, understanding the practical implications of technology choice helps them better advocate for their operational and patient needs. If rapid innovation and experimentation with new features are strategically important for the hospital, they can support architectural choices that facilitate agile evolution with sound arguments. If, on the other hand, operational stability, predictable performance, and maximum availability are critical for patient safety, they can tip the balance toward more conservative and proven options. Your expert voice in the clinical domain is absolutely essential to ensure that technical decisions are not solely based on abstract software development criteria or industry trends but also on real clinical needs, existing workflows, and potential risks to patient care.

One aspect that is often overlooked or underestimated in the simplistic monolith versus microservice debate is data governance and privacy. In a traditional monolith, the central database makes it easier to maintain absolute transactional consistency and enforce uniform access control, auditing, and privacy policies at a single point. In a microservice architecture with distributed and potentially heterogeneous databases, governance becomes inherently more complex and requires explicit coordination. It is essential to ensure that each microservice consistently implements the same organizational policies for protecting sensitive data, that identifiable data are protected across all repositories and that a consistent, global access log is maintained for regulatory auditing. This requires the implementation of an additional layer of security infrastructure and distributed policy tools, such as service mesh sidecars, which add complexity but are necessary for compliance.

It shows that many organizations that initially embraced microservices with enthusiasm, drawn by the promises of agility, end up facing unexpected problems of operational complexity, rising infrastructure costs, and difficulties in finding qualified talent. Some consciously choose to return to more monolithic approaches or, more frequently, to intermediate solutions that have come to be called "modulitos": a pragmatic solution that seeks to capture the best of both worlds, maintaining operational simplicity where possible and flexibility where necessary. The general lesson is that there are no universal architectural magic formulas and that technical humility, adaptation to the specific circumstances of each organization, and iterative evolution are more important than blindly following the architectural fads of the moment.

Software architecture is a means to construct information systems that effectively help clinicians care for their patients better and managers operate efficiently. A well-designed, documented, tested, and operated monolith that runs reliably 24 hours a day and allows for the necessary business evolution is infinitely better than a complex and fragile microservice architecture that no one in the organization fully understands, that generates constant incidents and that consumes the entire IT budget on platform maintenance. Conversely, a well-implemented microservice architecture with trained teams, appropriate tools, and mature processes—one that enables rapid innovation in response to emerging clinical needs and elegantly isolates failures to maintain continuity of care—is better than a rigid, obsolete monolith that prevents any improvement and must be restarted weekly. What truly matters is maintaining an obsessive focus on tangible clinical value and choosing or evolving the architecture that best serves that purpose in each specific context of time, place, and organization.

8.2 CLOUD VS. EDGE: LATENCY, PRIVACY, AND PROCESSING

Imagine an intensive care unit where dozens of critically ill patients are simultaneously connected to sophisticated multiparameter monitors that continuously record, several times per second, their heart rate, invasive blood pressure, pulmonary oxygen saturation, core temperature, and dozens of other essential vital signs. At every moment, these monitors generate a torrent of data that must be immediately analyzed by algorithms to detect arrhythmias, trends in

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

hemodynamic deterioration, and patterns of incipient sepsis to trigger smart alarms when necessary and alert nursing staff before the patient goes into cardiac arrest. If that critical data had to travel first through the hospital network, then to the internet, be processed on a remote server in the public cloud, wait its turn in a shared processing queue, and finally send an alert back to the bedside monitor, the cumulative delay could be several seconds or even tens of seconds under network congestion conditions. In a sudden cardiac arrest situation, where every second without intervention drastically reduces the chances of intact brain survival, that delay is absolutely unacceptable and potentially lethal.

On the other hand, let us now imagine an ambitious multicenter epidemiological study that needs to analyze health trends in millions of patients over several decades, identify risk factors for rare diseases, evaluate the population-level effectiveness of large-scale health interventions, or train predictive artificial intelligence models with massive datasets. In that case, cloud computing, with its virtually unlimited capacity for distributed storage, massively parallel computing, and access to specialized resources such as high-density GPUs, is the only technically viable and economically reasonable option. These two extreme but realistic scenarios illustrate the fundamental and seemingly contradictory dilemma between cloud computing and edge computing in the modern healthcare environment and the need for an architecture that does not choose between one or the other but rather combines both paradigms according to the specific clinical context.

Cloud computing consists fundamentally of processing and storing data in remote, massively scalable data centers that are accessible via the internet through standardized APIs. (1) The advantages of this model are evident and have driven its widespread adoption across all sectors: virtually unlimited horizontal scalability that automatically adapts to demand, massive processing capacity for intensive workloads, ubiquitous access from anywhere with connectivity, pay-as-you-go economic models that avoid initial capital investments in proprietary infrastructure, and the ability to delegate infrastructure management to specialized providers. Major cloud providers such as Amazon Web Services, Microsoft Azure, and the Google Cloud Platform offer specific services designed for the healthcare sector, with regulatory compliance certifications such as HIPAA BAA or GDPR, and specialized tools for the secure processing of structured and unstructured clinical data. For applications in epidemiological research, population-based public health, healthcare administrative management, or retrospective analysis of large cohorts, the cloud is the natural and almost unavoidable choice.

Edge computing, on the other hand, represents the opposite yet complementary architectural philosophy: it involves processing data physically close to where it is generated, at the “edge” of the network, before it travels to the central cloud.(2) This processing can occur on the medical device itself, at a local aggregation gateway, or on a hospital server located within the organization’s perimeter. The main and most frequently cited advantage is extremely low latency: since the data do not have to be sent through multiple network hops to the internet and back, processing and response are practically instantaneous, on the order of milliseconds. There are also critical advantages in terms of privacy and data sovereignty: the most sensitive health information can be processed and stored locally without ever leaving the hospital’s controlled and regulated environment, reducing the surface area exposed to external security breaches. Finally, there are advantages in terms of operational autonomy: the system can continue to function critically even if the internet connection fails completely, maintaining continuity of care in emergency situations. For real-time physiological monitoring applications, control of active medical devices such as infusion pumps, or situations where external connectivity is not guaranteed for geographical or security reasons, edge computing is technically irreplaceable.

8.2.1 CRITICAL FACTORS IN ARCHITECTURAL DECISION-MAKING

Network and processing latency is perhaps the most critical—and often decisive—factor in choosing between the cloud and the edge for a specific use case. In strict real-time applications, such as continuous patient monitoring in intensive care units, closed-loop control of smart infusion pumps that automatically adjust doses on the basis of the patient’s response, or computer-assisted guidance for complex surgical procedures, delays of even a few hundred milliseconds can be unacceptable and can compromise patient safety. Transmission to the cloud inevitably introduces variable and hard-to-guarantee latencies, depending on the momentary quality of the connection, the geographical distance to the data center, the load from other users on the shared service, and internet congestion. In these mission-critical cases, edge processing is the only viable

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

option from a medical systems engineering standpoint. However, not all healthcare applications require strict real-time performance; for many routine clinical operations, such as reviewing stable patients' medical records, generating administrative reports, or viewing nonurgent laboratory results, latencies of one or several seconds are perfectly tolerable and do not impact the quality of care.

Data privacy and sovereignty are key factors that heavily influence architectural decisions, especially in jurisdictions with strict regulations such as the General Data Protection Regulation (GDPR) and the European General Data Protection Regulation (GDPR). Healthcare data are considered a special category of highly sensitive personal data, and sending them to the public cloud inherently involves trusting that the cloud provider will adequately protect them, comply with applicable regulations, and not use them for unauthorized secondary purposes. Major cloud providers offer robust contractual guarantees, with certifications such as the HIPAA in the United States and the ISO 27001 and GDPR compliance mechanisms in Europe, including standard contractual clauses and certifications of responsible conduct. However, for some organizations—especially those in environments with very strict sector-specific regulations, those handling particularly vulnerable patients such as VIPs or sensitive personnel, or simply due to a conservative organizational culture—the option of keeping sensitive data within the hospital's physical and logical perimeter may be preferable or even mandatory. Edge computing allows sensitive data to be processed locally, sending only information that has been previously anonymized, statistically aggregated, or stripped of identifiers to the public cloud when necessary for backup or population analysis functions.

Raw processing capacity is another technical factor to be carefully considered in the design process. The cloud offers virtually unlimited and elastic computing power and is capable of running complex artificial intelligence algorithms that require high-density GPUs, massive distributed processing clusters, or object storage at the petabyte scale. In contrast, the edge has resources inherently limited by the size, weight, power consumption, and cost constraints of local devices: a bedside monitor or a home gateway cannot run deep learning models with millions of parameters or store massive databases. However, this technological gap is rapidly narrowing: new processors specifically designed for artificial intelligence at the edge, such as neural processing units (NPUs), allow increasingly complex models to be run directly on medical devices, and advanced techniques such as model compression, quantization, or federated learning enable processing to be intelligently distributed between the edge and the cloud according to the capabilities of each level. The technological trend is toward a fluid continuum, where decisions about where to process each specific task are made dynamically and adaptively on the basis of the latency, privacy, and capacity requirements at any given moment.

Network connectivity is oftenunderestimated but is a critical factor in planning resilient healthcare architectures. In hospital settings, even in advanced urban centers, high-speed, low-latency internet connectivity is not guaranteed at all times. There may be planned outages due to infrastructure maintenance, technical issues with the service provider, or even sophisticated cyberattacks that leave the hospital without external access during critical hours. Medical applications that are truly critical to patient safety must be able to continue operating at full capacity under these conditions of network isolation, which architecturally requires substantial local processing and storage capabilities at the edge. A patient monitoring system that relies exclusively on the cloud to generate vital sign alarms can fail catastrophically precisely when it is needed most—during a cyberattack that paralyzes external connectivity. That is why it is important to design medical systems with an intelligent degradation mode: systems that operate with reduced but essential functionality when the connection fails, that store data locally in a queue and that automatically synchronize when connectivity is restored without loss of critical information.

The massive volume of data generated by modern medical devices is another factor tipping the balance toward hybrid architectures. Contemporary medical devices generate enormous amounts of data: high-resolution images from advanced modalities, video sequences of endoscopic or surgical procedures, and continuous high-frequency recordings of physiological signals such as ECG or EEG. Sending all these raw data to the cloud can be prohibitively expensive in terms of network bandwidth, generate unacceptable latencies, and require massive cloud storage with high recurring costs. The edge enables preprocessing, intelligent filtering, lossless compression, or local analysis to extract only the relevant and condensed information, sending only the processed results, detected anomalies, or statistically aggregated data to the cloud. For example, a smart surveillance camera monitoring a patient at risk of falling can locally analyze video images using computer vision and send to the cloud only the short clips where it detects

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

anomalous movement or a fall, rather than continuously streaming hours of video without events. This edge filtering approach drastically reduces network bandwidth and cloud storage requirements while maintaining essential clinical functionality.

8.2.2 FOG ARCHITECTURE AND CLOUD-EDGE SYMBIOSIS

The intelligent combination of cloud and edge, technically known as fog computing, is rapidly gaining ground in sophisticated healthcare environments as the dominant architectural choice. In this hierarchical model, first-tier edge devices perform immediate, critical, low-latency processing; second-tier local gateways aggregate data from multiple neighboring devices, perform intermediate trend analysis, and coordinate communication with higher tiers; and the central cloud handles long-term persistent storage, massive population analysis, AI model training, and global coordination across multiple hospitals or regions. This well-defined layered architecture allows each technological level to be optimized for its specific function, elegantly balancing the conflicting requirements of minimal latency, maximum privacy, and unlimited processing capacity.

A concrete, realistic example can help us better understand this practical combination. Imagine a comprehensive remote home monitoring program for patients with established chronic heart failure. At the patient's home, a purpose-built edge gateway continuously receives data from a connected smart scale, an automatic wrist blood pressure monitor, and an activity and heart rate monitor. This gateway locally analyzes trends in weight, blood pressure, and heart rate variability using programmed clinical rules, and if it detects patterns indicative of incipient cardiac decompensation, it immediately sends a high-priority alert to the patient's phone and simultaneously to their primary care center—all without leaving the local environment and in a matter of fractions of a second. Daily, during times of lower network load, the gateway sends a structured summary of all the day's data to the regional healthcare cloud for automatic incorporation into the patient's electronic health record. Monthly, the cloud runs a sophisticated artificial intelligence algorithm that analyzes the longitudinal progression of all patients in the program, identifying subtle predictive patterns that enable proactive improvements to alert protocols for the entire cohort. Each level of the architecture does what it does best, in the most appropriate place, with the most efficient resources.

For developers of modern healthcare software, the decision between the cloud and edge is no longer an exclusive binary choice but rather involves a rich continuum of architectural options that must be evaluated on a case-by-case basis. Each specific functionality, data flow, and clinical use case must be analyzed to determine its requirements in terms of maximum acceptable latency, required privacy level, connectivity availability, and necessary processing capacity. On the basis of this multidimensional analysis, the most appropriate architecture is designed, which in practice almost always combines both approaches in varying proportions. Modern development tools greatly facilitate this hybridization, with platforms such as Kubernetes that allow the same code to be deployed across different edge or cloud environments and data synchronization frameworks that manage consistency across layers transparently for the application developer.

For clinicians and healthcare professionals involved in these technology projects, understanding the practical implications of choosing between cloud and edge helps them better assess technical decisions that often seem opaque or arbitrary to them. When a developer explains that a certain critical alert functionality cannot be implemented purely in the cloud because of insurmountable network latency issues or that certain particularly sensitive genomic data cannot physically leave the hospital for ethical and regulatory privacy reasons, the clinician can understand the real technical limitations and collaborate constructively in seeking viable architectural alternatives. It can also help prioritize intelligently among multiple competing needs: which clinical features are truly critical in strict real time versus which can tolerate a few seconds of delay? Which data are so sensitive that they must always remain under absolute local control? Their expert perspective on the clinical domain is absolutely essential for making balanced architectural decisions that do not sacrifice patient safety on the altar of technical efficiency.

Cybersecurity is a cross-cutting issue that critically affects both cloud and edge deployments, albeit with different risk profiles. In the cloud, security responsibility is shared according to the shared responsibility model: the cloud provider secures the underlying infrastructure, the hypervisor, and physical storage; the healthcare organization secures its own applications, configurations, data, and access policies. At the edge, the organization has full control and absolute sovereignty over the infrastructure, but it also assumes full responsibility for implementing,

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

maintaining, and updating all appropriate security measures. In both cases, strong encryption of data in transit between layers and at rest in storage is needed, along with identity- and role-based access control, comprehensive auditing of access logs, and proactive protection against external threats through intrusion detection systems. The choice between the cloud and edge should not be based on simplistic assumptions about which is inherently more secure but rather on a detailed and up-to-date analysis of the specific security capabilities of each option and the protection needs of each type of data.

The total economic cost of ownership is another factor that must be considered comprehensively and over the long term. The cloud offers an operational pay-as-you-go model that can be highly efficient for variable and unpredictable workloads but can prove surprisingly expensive for continuous massive volumes of data or nonstop intensive processing. The edge requires an initial capital investment in proprietary hardware, physical maintenance, and cyclical renewal, but variable operational costs are lower once the investment is amortized. A truly informed cost analysis must consider not only visible direct expenses but also the opportunity cost of being unable to innovate quickly because of architectural rigidity or the potentially catastrophic cost of a security failure or data breach that exposes the organization to millions of regulatory fines and irreparable reputational damage. It is a complex, dynamic calculation specific to each organization that must be reviewed periodically.

The technological future points unequivocally toward an increasingly sophisticated and transparent integration of the cloud and edge, with distributed artificial intelligence that continuously learns from the aggregated data of millions of patients in the central cloud but executes inferences and predictions locally in real time on the patient's edge devices. Machine learning models are trained centrally using large volumes of historical data in the cloud, achieving high diagnostic accuracy, but are then deployed locally optimized for immediate use at the point of care. Model updates are distributed continuously and securely, enhancing device intelligence without the need to replace physical hardware. This evolutionary symbiosis between the cloud and edge is likely to be the dominant architecture in digital health in the near future and is already emerging in the most advanced systems.

8.3 CLINICAL DATABASES: CHARACTERISTICS AND SPECIFIC FEATURES

Let us imagine for a moment the operational database of a medium-sized university hospital, a complex organization that handles hundreds of thousands of clinical episodes annually. It simultaneously houses the demographic data of hundreds of thousands of unique patients accumulated over decades; the detailed records of millions of outpatient consultations and emergency room visits; the results of tens of millions of laboratory tests of all kinds—from simple blood tests to complex genetic studies; structured reports and images from hundreds of thousands of diagnostic imaging studies; prescriptions for countless drug treatments with their dosages, routes of administration, and durations; and narrative clinical notes from generations of physicians documenting their reasoning, observations, and decisions. This database is not merely large in terms of storage volume; it is profoundly complex in its semantic structure, radically heterogeneous in its data types, and above all, it is absolutely critical to the continuity of healthcare operations. Clinical decisions that directly affect people's health—and sometimes their lives—depend on their continuous, proper functioning. If the database decreases because of a technical failure, hospital operations are ground to a halt within minutes, and patients in the emergency room cannot be properly cared for. If data are lost because of corruption or human error, the accumulated clinical information is irreparably destroyed, and the continuity of care is disrupted. If it is slow or unresponsive, doctors become frustrated, patients wait unnecessarily, and the quality of care visibly deteriorates. Designing, implementing, and managing a clinical database of this nature is not a trivial technical task; it requires a deep understanding of its specific characteristics and respecting them in every architectural decision.

The first distinctive and fundamental characteristic of clinical databases is the deeply semantic nature of the data they store, which transcends mere numerical or textual representation. They are not simply isolated numbers and texts but medical information with a wealth of context and exceptional meaning. A laboratory result, for example, is not just an abstract numerical value; it is a value with specific units of measurement that must be interpreted correctly, with normal ranges that depend critically on the patient's age, sex, and sometimes ethnicity, with clinical interpretations that vary according to the physiological and pathological context in which it was obtained. A narrative clinical note is not just a block of free-form text; it is a linguistic construct

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

containing complex implicit information, the physician's diagnostic reasoning, specific treatment plans, and sometimes uncertainties expressed in natural language that require intelligent processing to be exploited. A medical image is not just a large binary file; it is a structured study comprising multiple series acquired at different times or under different conditions, with technical metadata essential for its correct interpretation and with structured annotations that pinpoint specific findings. Clinical databases must be capable of representing, preserving, and enabling the retrieval of this multidimensional semantic richness in a way that allows it to be efficiently processed by both automated systems and healthcare professionals.

An absolutely nonnegotiable characteristic is the extreme sensitivity of the stored data from an ethical, legal, and social perspective. Clinical information is considered the most personal and sensitive category of personal data in virtually every jurisdiction worldwide, and its unauthorized disclosure—whether accidental or malicious—can have devastating and lasting consequences for affected patients, such as employment discrimination due to preexisting health conditions, social stigmatization due to mental or infectious diseases, family problems due to the disclosure of hereditary genetic information, or even risks to physical safety in the case of public figures.

For this reason, clinical databases must inherently and mandatorily incorporate robust, multidimensional security measures: strong encryption of data at rest to protect against physical theft of storage media, granular access control based on roles and clinical contexts to ensure that only legitimate authorized personnel access specific information necessary for their role, detailed and immutable auditing of all read and write operations to enable subsequent investigations, and technical mechanisms for anonymization and pseudonymization when data must be used for secondary purposes such as research or healthcare planning. This is not merely a technical implementation issue but also a legal, ethical, and social trust imperative. Noncompliance with applicable data protection regulations, such as the European GDPR or the U.S. HIPAA, can result in millions of dollars in fines for the institution, but worse still from a sustainability perspective, the loss of patient trust can be irreparable and destroy the social legitimacy of the healthcare system.

The urgent need for continuous, uninterrupted availability—the famous "five nines" or 99.999% uptime, which is actually insufficient for clinical criticality—is another key characteristic. A modern hospital never closes, not even for scheduled database maintenance. Emergencies continue to arrive at any hour of the day or night, operating rooms continue to perform scheduled and emergency surgeries, and intensive care units continue to monitor critically ill patients 24 hours a day, 365 days a year, without pause. The clinical database must be available and operational at all times, with fault-tolerant architectures that ensure that even in the face of serious technical incidents, service is maintained or restored in seconds, not hours. This inevitably requires high-availability architectures with active redundancy, continuous real-time backups, thoroughly tested disaster recovery plans, and proactive monitoring systems that are alert to any anomaly before it becomes a catastrophic failure. The cost of a prolonged outage is measured not only in lost administrative work hours or uninvoiced revenue but also potentially in human lives lost because of delayed care or a lack of critical information.

In a clinical database, information must be accurate at its source, complete in its recording, and reliable in its retrieval. An error in clinical data—such as a miscoded allergy, a lab result attributed to the wrong patient due to an identification error, or an incorrectly recorded medication dose—can directly lead to a serious diagnostic error, potentially lethal mistreatment, or avoidable harm to the patient. For this reason, clinical databases incorporate strict input validation mechanisms, referential integrity constraints that maintain consistency between related tables, and immutable audit logs that allow tracking of any changes made—when, by whom, and why. They must also elegantly handle complex situations such as correcting errors detected after the fact: if erroneous data are identified and corrected, it is not enough to simply overwrite it and lose the trail; the record of the original erroneous value, the correction applied, and the documented reason for the correction must be maintained because this history may be relevant in a legal context of a claim or a retrospective clinical investigation.

In medicine, knowing precisely who did what, exactly when, and for what documented reason is essential for safety, quality, and accountability—this is traceability. A drug prescription must be irrevocably linked to the identified prescribing physician, the patient receiving it, the exact time of the order, and the clinical indication that justified it. A change to a primary diagnosis must leave an immutable record of who made the change, with what reasoned clinical justification, and what the previous value was. Access to sensitive data regarding a famous patient or politician must be permanently logged for potential audits of legitimacy. Clinical databases must support this comprehensive level of traceability, which implies not only storing the current valid data but also maintaining the complete history of temporal changes, the operational metadata for each

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

transaction, and the explicit relationships between data and identified individuals.

A typical modern clinical database simultaneously stores strictly structured data—such as numerical laboratory values and their metadata—semistructured data—such as medical reports with predefined fields but variable narrative content—and completely unstructured data—such as free-form clinical notes in natural language, narrative discharge summaries, or voice recordings. It also stores massive binary objects such as DICOM radiological images, continuous physiological monitoring signals, video files of endoscopic procedures, or PDF documents of signed informed consent forms. Managing this heterogeneity of formats, sizes, and access patterns efficiently requires different types of physical and logical storage, as well as different indexing, compression, and query strategies optimized for each case. Traditional relational databases are necessarily complemented by NoSQL-type document stores, distributed file systems for large objects, and specialized databases for time series of physiological signals, all of which are integrated under an abstraction layer that presents a unified view to the user and applications.

Queries to clinical databases can vary extraordinarily in their patterns. A doctor in the emergency room may need to retrieve the complete medical history of an unknown patient who arrives unconscious in less than a second to make immediate, life- y decisions. An epidemiological researcher may run a complex analytical query that scans and aggregates millions of historical records over several minutes for a cohort study. An automated clinical alerting system may need to continuously monitor new data arriving from hundreds of patients simultaneously in real time to detect patterns of deterioration. The database must be capable of handling this variety of concurrent workloads without degrading the performance perceived by any user, which requires a careful, multifaceted design of optimized indices, horizontal and vertical partitioning strategies, and intelligent cache layers that anticipate frequent access patterns.

Clinical databases do not exist or operate in a technological vacuum; they must continuously exchange information with other systems within the same hospital, with external healthcare facilities for continuity of care, directly with patients via health portals, and with healthcare providers in the ecosystem. This involves natively supporting exchange standards such as HL7 v2 for messaging, FHIR for modern APIs, or DICOM for imaging; exposing well-documented programming interfaces for data querying and updating; and maintaining eventual consistency when the same information is replicated or referenced across multiple heterogeneous systems. Interoperability is not an optional technical add-on that can be postponed but a fundamental functional requirement for the operational reality of modern digital health.

Healthcare systems undergo organizational change, medical knowledge advances and alters what is considered relevant to records, and information needs evolve with clinical practice and regulation. A clinical database must be capable of adapting to these inevitable changes without losing or invalidating previously stored historical information, which holds legal, clinical, and research value. When a new data field is introduced because of a change in clinical practice, older records must remain accessible and interpretable. When a standard coding scheme is modified because of an update in a medical classification, historical data coded with previous versions must be correctly interpreted within their temporal context. This requires the implementation of strategies for managing flexible database schemas, data versioning mechanisms, medical vocabularies, and sometimes complex migrations that preserve historical semantics.

The volumes of healthcare data generated are growing steadily and rapidly: more diagnostic tests per capita due to defensive medicine and increased use of technology, images with higher spatial and temporal resolution due to technological advances, more connected devices generating continuous monitoring streams, and the emergence of massive amounts of genomic data that multiply the size of the record per patient by orders of magnitude. A clinical database must be capable of scaling elastically to handle this continuous growth, both in terms of raw storage capacity and query processing capacity, without service interruptions or disruptive architectural redesigns. This may involve implementing horizontal sharding, geographically distributed database architectures, or hybrid cloud solutions that allow for on-demand expansion of storage and computing resources without massive upfront capital investment.

For software developers working specifically with clinical databases, these ten characteristics translate into concrete and complex technical design decisions. The choice of the underlying database management system is crucial and often involves polyglot solutions: mature relational databases such as PostgreSQL, Oracle, or SQL Server remain the dominant choice for critical transactional structured data but are necessarily complemented by NoSQL databases such as MongoDB for semistructured documents, distributed object file systems for massive images and videos, and specialized timeseries databases such as InfluxDB for continuous physiological signals. The current architectural trend is unequivocally toward heterogeneous yet integrated

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

polyglot databases, where each data type is stored in the management system best suited to its access patterns, and a semantic integration layer provides the application layer with a unified and coherent view of the entire patient.

Data schema modeling is another critical aspect that requires deep medical expertise in addition to technical skills. The entity-relationship model of a clinical database must accurately reflect the complexity of the real medical domain: patients who have multiple episodes of care throughout their lives, each episode with multiple clinical observations of different types, each observation with multiple components and values, and complex relationships between entities that must be maintained with strict referential integrity. Unique patient identifiers, managing duplicates caused by registration errors, and resolving identities when the same patient appears with different data in different systems are complex algorithmic and organizational problems that require specific and often localized solutions. Security is a cross-cutting layer that affects all aspects of design and implementation. Access control must be granular down to the field level in some cases, allowing for the definition of which specific users can view which sensitive data, in which justified clinical contexts, and for what documented purposes. Encryption must be applied both in transit between system components and at rest on storage media, using robust algorithms and proper key management. Auditing must log all accesses and significant modifications with sufficient forensic granularity to enable subsequent investigations of incidents or claims. This entire security framework must be implemented without excessively degrading the performance experienced by clinical users and without unduly complicating the development of new features.

Disaster recovery is another critical aspect that cannot be left to chance or improvisation. Backups must be frequent, automatic, regularly verified, and stored in secure physical locations that are geographically separate from the main system to protect against natural disasters or major incidents. They must be periodically tested through recovery drills to ensure that they are truly recoverable within defined target times, not just theoretically. There must be a detailed business continuity plan specifying exactly how service will be restored in the event of a major catastrophic failure, with assigned roles and responsibilities, maximum recovery times contractually agreed upon with management, and procedures for communicating with stakeholders.

For clinicians and healthcare professionals who use these databases daily without consciously seeing them, understanding their fundamental characteristics helps them maintain realistic expectations about the system's behavior and collaborate constructively with technical teams. When a system takes time to respond to a complex query, it may legitimately be due to the semantic complexity of the search and not necessarily because the system is technically deficient. When historical data are not immediately available following a system migration, it may be due to necessary consistency and audit mechanisms, not an operational failure. When the system requests additional information or reudentication to access certain sensitive data, it is for the protection of patient privacy and regulatory compliance, not due to unnecessary bureaucracy. This mutual understanding improves the collaborative relationship between clinical professionals and technical teams and reduces the mutual frustration that often arises from a lack of understanding of the other side's limitations and needs.

The future of clinical databases points toward an increasingly deep integration of diverse and massive data sources, including comprehensive genomic data from next-generation sequencing, continuous real-time data from connected devices, and social and environmental data that contextualize patient health. It also points toward the widespread use of artificial intelligence to automatically extract knowledge from data, with machine learning models trained directly on historical clinical databases to predict risks and suggest decisions. It points toward greater active involvement of patients themselves, who will be able to access their complete data, granularly control its secondary use, and contribute their own observations and data generated by wearables. These emerging developments pose new technical challenges of scalability and complexity, as well as new ethical dilemmas regarding privacy and consent, which the databases of the future will need to proactively address.

Clinical databases are not merely a technical repository of digital information but also the collective institutional and social memory of the healthcare system. They accumulate the knowledge generated by generations of medical and nursing professionals, the experiences of millions of patients who have entrusted their health to the system, and the lessons learned from countless individual clinical cases that have been meticulously documented. Preserving institutional memory, keeping it accessible for current and future clinical practice, and using it responsibly to improve care and research is an enormous responsibility shared by the developers who build these technical infrastructures, the administrators who operate them, and the clinicians who feed them with data and consult them for decision-making. Doing so well, with the highest

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

technical standards and the deepest ethical sensitivity, is among the most important and lasting contributions that can be made to people's health and the progress of medicine.

8.4 DESIGN FOR INTEROPERABILITY, NOT ISOLATION

> "A perfect but isolated medical system is like a brilliant doctor who speaks only a language that no one else understands: its potential value is lost in the lack of communication."

Let us imagine that a software development team is tasked with building a new specialized application for the comprehensive management of patients with diabetes mellitus. They meticulously analyze the functional requirements with endocrinologists, design a technically elegant architecture, implement all the requested features with high-quality code, and deliver the final product within the agreed-upon timeline and budget. The hospital installs it on its infrastructure, the doctors begin using it with initial enthusiasm, and everything seems to work perfectly from an isolated technical perspective. A few months later, however, someone in quality assurance realizes that the diabetes application cannot automatically share structured data with the hospital's main electronic health record system, which operates as a standalone technological island. Doctors are forced to manually enter the same information twice: once in the specialized diabetes app for specific disease monitoring and once in the general medical record for a comprehensive view of the patient. Patient data remain fragmented across incompatible systems; automatic drug interaction alerts do not work because the diabetes app lacks visibility into the full range of medications the patient receives from other specialties, and the glycemic control reports generated by the app cannot be automatically incorporated into the patient's unified medical record for future reference. An app, technically perfect in its isolated design, is clinically an operational failure that generates frustration, transcription errors, and risks to patient safety. This story, which repeats itself with endless variations in hospitals around the world every year, painfully illustrates what inevitably happens when medical software is designed for isolation rather than explicitly designed for interoperability from the very first line of code.

Designing for interoperability means assuming from the very start of the project—even before writing the first technical requirement—that the system we are building will never exist in a technological vacuum but will necessarily be part of a broader ecosystem of other heterogeneous systems, with which it will need to exchange information in a fluid, bidirectional, and reliable manner to create real clinical value. It is not a superficial feature that can be added at the end of development, like a makeshift glue that binds systems that were not conceptually designed to understand one another. It is a fundamental property of software architecture, which must be present and actively considered in every design decision, from the choice of patient identification schemes to the definition of public APIs, including the internal data model and cross-cutting security mechanisms. Ignoring interoperability from the earliest stages of design is to condemn the product to live in isolation forever, and clinical users suffer the operational consequences of that lack of communication.

8.4.1 FUNDAMENTAL ARCHITECTURAL DECISIONS

The first design decision that critically and permanently affects the system's interoperability is the choice of identifiers for all relevant entities. Every patient, healthcare professional, clinical care episode, and data resource requires a unique identifier that can be shared, referenced, and resolved across different systems in an unambiguous manner. If each system uses its own internally generated identifiers without considering the ecosystem context, the subsequent mapping between them becomes an integration engineering nightmare that consumes resources and generates errors. Ideally, standardized national or international identifiers should be used whenever possible, such as the national medical record number, the licensed healthcare professional identifier, or episode codes following administrative standards. Alternatively, at the very least, an internal identification system should be designed that enables unambiguous exchange through identity resolution. Interoperable identifiers must be persistent over time (they do not change when the patient changes address or insurer), globally unique within the ecosystem's namespace (no two different patients have the same identifier), and resolvable via directory services (there must be a standard technical way to obtain further information from them).

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

The second critical architectural decision is the data model and the terminologies used to represent clinical information. A system consciously designed for interoperability cannot afford to invent its own idiosyncratic way of representing universal medical concepts, creating incompatible semantic silos. must use, from the design phase, widely accepted and internationally validated terminological standards: HL7 FHIR for the structured exchange of clinical resources, SNOMED-CT for the semantically precise coding of clinical concepts, LOINC for the unambiguous identification of laboratory tests and observations, RxNorm or ATC for drug coding, and DICOM for medical image management. Using these standards from the outset of development ensures that other systems in the ecosystem can unambiguously understand the data our system produces and that we can correctly interpret the data we receive from external sources. Adapting a legacy system that used local proprietary terminologies to these international standards is technically possible but far more costly, complex, and prone to semantic mapping errors than incorporating them from the initial design.

The third decision is the controlled exposure of well-designed public APIs. A truly interoperable system is not an opaque black box; it is a system that exposes well-documented, stable, and versioned programming interfaces that allow other authorized systems to access its essential functionalities and data in a controlled manner. These APIs must follow modern web industry standards (RESTful architecture, JSON data formats, OAuth2 for authorization), be thoroughly documented with real-world use cases, and remain stable over time while adhering to semantic versioning rules that allow for evolution without breaking existing clients. They must be proactively designed with real integration use cases in mind that other systems will have, not just the immediate internal needs of the system being built. An API that exposes only the minimum the system needs to function internally—and not what other systems legitimately need to integrate with it—is an insufficient API that perpetuates isolation.

The fourth decision concerns the cross-system management of user identity and access control. Effective interoperability necessarily implies that authenticated users from other systems can access our data in an authorized manner and that our users can access data from other systems without having to memorize multiple credentials. This requires robust authentication and authorization mechanisms that operate in a federated manner across organizational and technological boundaries. Modern standards such as OAuth2 for delegated authorization and OpenID Connect for federated authentication, combined with healthcare-specific profiles such as SMART on FHIR, provide the complete technical framework to implement this securely and in a standardized manner. However, it must be designed and integrated from the very beginning of the security architecture, not as a retrofitted add-on, because it affects all layers of the system and all access control decisions.

The fifth decision concerns the flexible granularity of the exchanged data. A system designed for true interoperability must be capable of exchanging information at different levels of aggregation depending on the context of use and applicable privacy restrictions. Sometimes the most granular data possible are needed: the individual point-in-time value of a capillary blood glucose reading with its exact timestamp. Other times, an aggregated statistical summary suffices: the average blood glucose levels from the past week or the percentage of values within the target range. The design of the API and the data model must explicitly allow for both query possibilities and must include descriptive metadata that clearly indicate the level of aggregation applied and the context in which the data were generated. This is especially important for the protection of privacy: sometimes it is necessary to share aggregated population data for research or planning without revealing identifiable information about specific patients, and the system must be able to generate and serve these aggregates reliably.

The sixth decision concerns the explicit management of the synchronization and consistency of distributed data. When multiple heterogeneous systems share and replicate information about the same patients, the complex technical problem of maintaining consistency over time inevitably arises. If one system updates a clinical data point, how do the other systems maintaining related copies determine? What happens if two different systems update the same data point simultaneously with different values? Who holds the authority for the definitive “truth” for each data type? There are no universal answers valid for all cases; they depend critically on the clinical context and the specific use case. However, mechanisms must be consciously designed to address these questions: event systems that notify interested subscribers of relevant changes, APIs that allow explicit retrieval of the most recent available version, and documented conflict resolution policies when inconsistencies occur. Evenual consistency is often sufficient and acceptable for many noncritical clinical use cases but is not always acceptable, especially for those involving immediate patient safety.

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

The seventh decision concerns the complete traceability of data provenance and transformation. When data travel between multiple systems in an interoperable ecosystem, it is essential to maintain an immutable record of its original source and all the transformations it undergoes along the way. Does these laboratory result data come directly from the laboratory's automated analyzer, or was it manually entered by a technician? Has it been technically validated and signed off by a responsible clinician? Has it been subsequently modified by any normalization or correction algorithm? Complete traceability allows clinicians to critically evaluate the reliability and context of the information before making medical decisions on the basis of it. It is also absolutely essential for regulatory audits and for forensic investigations of security or quality incidents. Designing for traceability involves including metadata on provenance, authorship, and transformation in all data exchanges, not just in the data at rest.

The eighth decision involves robust and transparent management of communication errors. In an interoperable distributed system, communication failures are statistically inevitable and must be anticipated architecturally. The network may fail intermittently, an external system may be temporarily down for maintenance, or a message may become corrupted in transit because of line noise. The system design must include explicit and robust error-handling mechanisms: automatic retries with exponential backoff to avoid saturation, persistent message queues that survive system reboots, clear notifications to human operators when nonrecoverable failures are detected, and above all, the ability for end-user clinicians to explicitly know when the information they are viewing on screen may be out of date, incomplete, or potentially inconsistent because of synchronization issues. Operational transparency regarding the actual state of interoperability is essential for maintaining clinical trust in the system.

8.4.2 DEVELOPMENT CULTURE AND SHARED RESPONSIBILITY

For software developers working in the healthcare sector, explicitly designing for interoperability requires a profound shift in mindset compared with other, less-regulated industries. It is no longer just about building a system that functions correctly in isolation but about building a system that functions well as a cooperative member of a broader and more complex ecosystem. This requires investing time in deeply understanding industry standards, actively participating in professional communities where these standards are discussed and evolved, and remaining abreast of regulatory and technological developments. It also requires a certain degree of technical humility: recognizing that our system, however brilliant it may be, is not the center of the hospital's technological universe but rather just another node in a complex network of interdependent systems, each with its own history, limitations, and value. In addition, professional generosity is needed: thoroughly documenting APIs and data models to make life easier for other developers who will need to integrate with us, offering proactive technical support to integration teams, and contributing constructively to the continuous improvement of shared standards through feedback on real-world experience.

For clinicians and healthcare managers involved in information system procurement or development projects, designing for interoperability means making it a nonnegotiable requirement that the systems purchased or commissioned explicitly comply with recognized interoperability standards. It means systematically asking, before signing any procurement contract: Can this system automatically share structured data with our primary electronic health records? What specific terminology and exchange standards does it use? How is the unique patient identity managed across systems? Does it offer documented and stable APIs for integration? This means never settling for technically brilliant solutions that operate in isolation, no matter how good they may be in their specific functional domain, because an isolated solution is not truly a solution for the patient; it is a future problem of information fragmentation and security risk.

The history of health informatics is unfortunately full of examples of systems that were consciously designed for isolation—sometimes due to a vendor-lock-in business strategy, sometimes due to ignorance of the implications, and sometimes due to a rush to deliver. Clinical laboratory systems that generate results that cannot automatically flow into the electronic health record. Hospital pharmacy systems that lack real-time visibility of allergies and contraindications are recorded in the emergency department. Chronic patient monitoring applications that generate valuable remote monitoring data that no one else on the care team can view or use. Each of these systems, considered individually within its silo, may be technically excellent and fully meet the functional requirements for which it was designed. However, together, in the real patient care ecosystem, they form an archipelago of isolated technological islands where information is

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

critically lost, dangerously duplicated, and silently corrupted, and those who pay the ultimate price are the patients who receive fragmented, discontinuous, and potentially unsafe care.

Designing for interoperability is neither technically free nor trivial. It requires investing additional time in the initial design, sustained effort during implementation, and maintaining up-to-date specialized knowledge within the team. However, it is an investment with a guaranteed long-term return. An interoperable system can integrate seamlessly into the hospital's existing technological ecosystem; it can be expanded with new functionalities in the future without disrupting what already works, and it can eventually be replaced by a better system without losing accumulated data or operations. An isolated system, on the other hand, is a long-term trap: once it is adopted and filled with critical data, it is very difficult and costly to move away from it without losing valuable information or undergoing traumatic migration that paralyzes clinical activity.

The global regulatory trend in digital health is unequivocally toward a growing demand for interoperability as a market requirement. In the United States, the ONC (Office of the National Coordinator for Health Information Technology) information rule now requires certified electronic health record providers to expose standard FHIR APIs for data access. In Europe, the European Health Data Space is aggressively driving the adoption of common standards for cross-border interoperability. In many countries, public procurement of healthcare systems already explicitly requires compliance with proven interoperability profiles as a bidding requirement. Designing for interoperability is not just an optional technical best practice; it is a growing regulatory requirement for legal competition in the digital health market.

Once we have designed systems that can interoperate and share information seamlessly, how do we protect all that sensitive information from the growing cybersecurity threats that specifically target the healthcare sector, and how do we balance the openness necessary for interoperability with the protection essential for privacy?

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

REFERENCES

Mell P, Grance T. The NIST Definition of Cloud Computing. September 28, 2011 [cited May 13, 2026]; Available from: https://csrc.nist.gov/pubs/sp/800/145/final
What Is Edge Computing? | IBM [Internet]. [cited May 13, 2026]. Available from: https://www.ibm.com/think/topics/edge-computing

https://doi.org/10.62486/978-9915-9876-8-2.Ch08

Chapter 9 / Capítulo 9

Check for updates

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch09
Pages: 151-166

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 9. Clinical Cybersecurity

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "IT security in a hospital is not just another technical problem: it is an extension of clinical safety. A security breach that renders a medical record inaccessible, alters a patient's data, or allows unauthorized access to sensitive information is not an IT incident: it is a clinical incident with potentially serious consequences for those who trusted the system to protect them."

Imagine that a hospital suffers a ransomware attack. Attackers have managed to encrypt servers containing medical records, laboratory results, diagnostic images, and prescription systems. Doctors cannot access their patients' information. Nurses do not know what medication has been prescribed. The emergency room has to operate using paper and memory, with the risk of errors that this entails. Scheduled surgeries are canceled. Chronic patients lost their follow-up care. This scenario, which a decade ago appeared to be science fiction, has played out dozens of times in hospitals around the world. Cybersecurity has ceased to be a peripheral concern of IT departments and has become a central issue of clinical safety.

The healthcare sector has become one of the prime targets for cyberattackers. Hospitals store extremely valuable data, are willing to pay ransoms to regain access to their systems, and struggle to implement robust security measures because of the complexity of their environments and the age of much of their equipment. Added to this is the unique reality that healthcare systems cannot afford to be down: while a store can close for a day because of a cyberattack, an emergency room cannot. This asymmetry makes hospitals prime targets.

In this chapter, we explore the fundamental aspects of cybersecurity in healthcare environments. We begin with the regulatory framework that protects sensitive data: the HIPAA in the United States, the GDPR in Europe, and local regulations in other countries. We will continue by analyzing why hospitals have become such an attractive and vulnerable attack vector. We address the principles of security by design, which should guide the development of any healthcare system. Finally, we look at the specific mechanisms of authentication, authorization, and auditing that allow us to control who accesses what information and leaves a trail of that activity. The goal is not to turn the reader into a security expert but to equip them with the awareness necessary to understand that security is everyone's responsibility.

9.1 SENSITIVE DATA: HIPAA, GDPR, AND LOCAL REGULATIONS

There is a natural tendency, especially among software developers accustomed to working with commercial or financial data, to believe that health data are essentially similar to any other personal data and are just slightly more sensitive and therefore require slightly more care. This perception is dangerously naive and underestimates the unique nature of health information. A person's medical information is likely the most intimate, personal, and potentially harmful type of data about them. It reveals not only diagnosed medical conditions but also private lifestyle habits, complex family relationships, hereditary genetic predispositions, and profound aspects of personality and psychological functioning that not even the person's closest friends are aware of. A detailed psychiatric history documenting episodes of major depression or anxiety disorders, a diagnosis of HIV infection, a record of therapeutic abortion, a failed assisted fertility treatment, and a sexually transmitted disease: all of these are clinical data that, if disclosed without explicit consent or used for unauthorized purposes, can cause immense, lasting, and sometimes irreparable harm to the affected person, ranging from explicit or veiled workplace discrimination to severe social stigmatization, devastating family conflicts, and profound psychological damage resulting from the violation of the most intimate privacy.

For this fundamental reason of human dignity and social justice, over the course of decades, democratic societies have developed specific, complex, and constantly evolving legal frameworks to rigorously protect health information. The best known internationally, especially in the English-speaking world, is the Health Insurance Portability and Accountability Act (HIPAA), the U.S. Health Insurance Portability and Accountability Act, passed in 1996 and subsequently amended. The HIPAA establishes a comprehensive set of mandatory requirements to protect protected health information (PHI),(1) including detailed privacy rules that strictly limit permitted uses and authorized disclosures of data; technical security rules that require specific administrative, physical, and technical safeguards; and breach notification rules that require covered organizations to promptly notify affected patients and regulatory authorities when a security incident that compromises data occurs. For more than two decades, the HIPAA has been the global benchmark for health data protection, and many countries have adopted structural elements to develop their own national legislation tailored to their context.

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

In Europe, the General Data Protection Regulation (GDPR) of 2016 established a standard of protection that is even more demanding and comprehensive than the HIPAA in many respects. (2) The GDPR explicitly considers health data, broadly defined, as a "special category" of personal data subject to enhanced protection and additional restrictions. This means that, by legal default, these data cannot be processed without first obtaining explicit, free, specific, informed, and unambiguous consent from the data subject, except for very limited and strictly interpreted exceptions such as the provision of direct healthcare to the data subject themselves or scientific research with prior ethical approval from a competent committee. The GDPR also introduces new and powerful rights for European citizens, such as the right to be forgotten (to have their data erased when it is no longer necessary), the right to data portability (to receive their data in a structured format and transfer it to other controllers), and the right to be informed without an undue delay of security breaches affecting their data, within a maximum of 72 hours of detection. Penalties for noncompliance with the GDPR can reach 20 million euros or 4% of the company's annual global turnover, which has effectively focused the attention of healthcare organizations across Europe on strict compliance.

9.1.1 THE GLOBAL REGULATORY LANDSCAPE AND UNIVERSAL PRINCIPLES

In Latin America, the regulatory landscape varies significantly across countries but shows a clear and growing trend toward progressive harmonization with international protection standards. Argentina was a pioneer in the region with its Personal Data Protection Law No. 25,326 of 2000, which explicitly includes health data as sensitive data subject to enhanced protection. Brazil passed its General Data Protection Law (LGPD) in 2018, structurally inspired by the European GDPR, which also grants special protection and requires specific consent for health data.(3) Mexico has the Federal Law on the Protection of Personal Data Held by Private Parties, supplemented by various state laws that harmonize the framework. Specific legal frameworks for Colombia, Chile, Peru, Uruguay, and other Latin American countries have been developed, with varying degrees of maturity, technical requirements, and capacity for effective enforcement.(4) For healthcare software developers working on transnational projects or within the region, having a detailed understanding of the specific regulations currently in force and under development in each country of operation is absolutely essential, as differences can be significant in areas such as notification deadlines, the legal grounds permitted for processing, or the applicable penalties.

In Spain, the applicable regulations combine the European framework of the GDPR with the Organic Law on Data Protection and Guarantee of Digital Rights (LOPDGDD) 3/2018,(5), which elaborates on specific aspects of the GDPR within the Spanish context and adds relevant provisions such as enhanced protection of health data in the workplace or insurance sectors, where abuses have historically occurred. It also explicitly establishes the obligation to conduct data protection impact assessments (DPIAs) when data processing may pose a high risk to individuals' rights and freedoms, which is practically always the case in healthcare projects involving large-scale health data or emerging technologies such as artificial intelligence. The Spanish Data Protection Agency (AEPD) has issued specific guidelines for the healthcare sector that should be understood and followed, including technical recommendations on anonymization, pseudonymization, and security.

Beyond specific implementation differences across jurisdictions, all these global regulations share common fundamental principles that healthcare software developers must deeply internalize and apply in their daily work. The principles are as follows:

Legitimacy through consent: Given their sensitive nature, health data cannot be used for any purpose without an appropriate legal basis, with the patient's explicit consent being the general rule. This consent must be truly informed (the patient must genuinely understand what their data will be used for and what that entails), specific to each purpose (generic, open-ended consent allowing for any future use is not valid), and revocable at any time (the patient may change their mind and withdraw their consent, with effects for the future but not retroactively). Implementing this in software requires designing robust technical mechanisms to capture consent in an auditable manner, storing it securely and in a retrievable format, strictly respecting it in all processing operations, and allowing for its effective revocation while managing the consequences.

Data minimization: Only data strictly necessary for the specific, stated purpose should be collected, stored, and processed—nothing more. This entails designing systems that do not request or store more information than is essential for the clinical or administrative function they

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

perform and that implement automatic processes for deletion or anonymization when the data are no longer necessary for that purpose. A system that accumulates patient data indefinitely "just in case" it might be useful in the future, or out of mere technological conservatism, is flagrantly violating this principle and accumulates unnecessary risks.

Technical and organizational security: Data must be protected with specific, proportionate, and demonstrably adequate measures commensurate with the assessed level of risk. This necessarily includes strong encryption of data in transit between systems and at rest in storage, strict access control based on the principle of least privilege, continuous auditing of all significant operations, and the existence of documented security incident response plans. Regulations generally do not prescribe specific universal technical measures but rather require a documented risk analysis to determine the appropriate measures in each specific case, which implies responsibility and technical judgment on the part of the implementers.

Active transparency toward patients: citizens have the right to know, in a clear and accessible manner, what data are held about them, for what specific purposes it is used, who has accessed it and when, and with which third parties it has been shared. This means that health information systems must be technically capable of providing this information in a way that is understandable to nontechnical users and of efficiently handling requests from patients to exercise their rights of access, rectification, erasure, and portability.

Proactive and demonstrable accountability: It is not enough to passively comply with the rules; organizations must actively and documentarily demonstrate that they are complying. This involves maintaining detailed records of processing activities, documenting privacy-related design decisions, conducting periodic impact assessments, and always being prepared to demonstrate to regulators that appropriate safeguards have been implemented. This principle of accountability is central to the GDPR and is changing the culture of healthcare data management.

9.1.2 TECHNICAL CHALLENGES AND SHARED RESPONSIBILITY

For software developers working on healthcare applications, translating these abstract principles into concrete code and architectural decisions is neither trivial nor free of dilemmas. Implementing consent robustly means designing not only user interfaces to capture it but also data structures to store it in an immutable and queryable way, access control mechanisms that dynamically respect the consent status in every operation, and processes to manage revocation along with its implications for ongoing processing. Implementing data minimization means carefully considering which fields are truly necessary when designing database schemas, establishing retention policies with automatic deletion, and resisting the temptation to hoard data "just in case." Implementing transparency means building patient interfaces that allow them to view and download their data in an understandable way and logging systems that record all access operations with the detail necessary to respond to information requests.

A particularly complex aspect in practice is the granular management of consent in environments where data must be legitimately shared among multiple organizations for comprehensive patient care. A patient may consent to their data being used for scientific research, but only for certain specific types of studies (for example, cancer research but not psychiatric research), and only by certain institutions of recognized prestige and only for a specified period of time. Managing these finely granular consents and ensuring that they are technically respected at every node in an interoperability network is a major technical and organizational challenge that is driving innovation in standards. Technical specifications and emerging technologies specific to this problem are emerging, such as HL7 FHIR consent profiles that allow consent to be represented as an automatically processable structured resource, but their widespread implementation is still in its infancy and requires further development.

Another critical area where developers bear direct responsibility is the reporting of security breaches. When a security incident occurs that affects patients' personal data, regulations require notifying those affected and regulatory authorities within very short timeframes (72 hours in the case of the GDPR from detection, with potential penalties for delays). This means that organizations must have the technical and organizational capabilities to quickly detect security incidents through proactive monitoring, technically investigate their exact scope to determine whether they constitute a reportable breach, and efficiently communicate with those affected using clear and useful information. This is not something that can be improvised when an incident occurs; plans, procedures, communication templates, and response teams must be prepared in advance and regularly tested through drills.

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

The tense relationship between data protection and scientific research is particularly delicate and requires balance. Accumulated health data serve as a goldmine for medical and epidemiological research, and its secondary use can generate knowledge that benefits society as a whole and future generations of patients. However, this use must be carefully balanced against the right of current patients to control their most intimate information. Regulations establish legitimate avenues for using data for research purposes without explicit individual consent under certain strict conditions (when obtaining consent is impractical because of the size of the cohort, when there is a demonstrated relevant public interest and when effective anonymization is guaranteed). However, these avenues are subject to strict formal requirements for ethical approval, security, and transparency, and their practical application always requires specialized legal advice and case-by-case evaluation.

For clinicians and healthcare professionals involved in projects to develop or acquire information systems, a thorough understanding of these regulations helps them be aware of their own legal and ethical responsibilities. When a physician accesses a patient's electronic health record, they must do so solely in the context of providing direct care to that patient, never out of personal curiosity or to spy on acquaintances. When a researcher requests data for a study, they must ensure that they have all the necessary ethical and legal authorizations before accessing any data. When a manager decides to share data with another institution for patient care, they must verify that appropriate confidentiality agreements and legal bases are in place. Data protection is not just a matter for specialized lawyers or security engineers; it is a shared responsibility of everyone who handles health information in any way, from the physician to the administrative staff to the developer.

These regulations are constantly and rapidly evolving, and developers must proactively stay up to date through ongoing training. The GDPR is subject to periodic revisions and interpretive guidelines that clarify its application to new cases. In the United States, the HIPAA is updated through additional rules that adapt its requirements to emerging technologies. In Europe, the European Health Data Space (EHDS) is being actively developed, which will create a detailed, specific regulatory framework for the cross-border exchange of health data for care and research purposes. In many countries, sector-specific laws are emerging for technologically complex fields such as genomics, telemedicine, or medical artificial intelligence algorithms. Staying up to date in this changing environment is an ongoing professional challenge, but it is absolutely essential for building and operating systems that remain legal, ethical, and socially acceptable throughout their lifecycle.

Health data protection laws exist because there have been serious historical abuses because health information was systematically used to discriminate in employment, to socially stigmatize, and to harm vulnerable people. Complying with the letter of the law is the ethical minimum we can do as professionals; what is desirable and aspirational is to go beyond the legal minimum, building systems that respect patients' privacy and autonomy not because a regulatory standard requires it but because we recognize that it is the morally correct and professionally ethical thing to do. Patients entrust us—as developers, clinicians, and institutions—with their most intimate and vulnerable information; they deserve for us to treat it with the utmost respect, care, and protection possible, today and in the future.

9.2 THE HOSPITAL AS A VECTOR OF ATTACK

Let's imagine a modern, mid-sized hospital in a European city. Hundreds of devices connected to the institutional network coexist daily within its facilities: desktop computers in outpatient clinics, robust servers in the data center, multiparameter monitors in intensive care units, networked programmable infusion pumps, MRI machines with complex control systems, PACS image storage systems, smartphones for medical and nursing staff, tablets for bedside nursing management, and a long list of other medical and administrative technologies. Many of these critical devices were originally designed without considering cybersecurity as a design priority; they run obsolete operating systems that no longer receive security patches and cannot be updated regularly because of manufacturer restrictions or regulatory certifications. The communication network connecting them has grown organically over decades, with successive patches, improvised additions, and unplanned expansions that have created a tangled web of technical and logical dependencies that is difficult to fully audit. Healthcare staff, legitimately overburdened with clinical and administrative work, cannot always follow best practices in terms of IT security, and cyberattackers are well aware of this and systematically exploit it. Far from

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

being the impregnable digital fortress that patients might imagine, the hospital has become a vulnerable, attractive, and highly profitable target for organized cybercrime.

The **healthcare sector** presents a unique and dangerous combination of structural factors that make it particularly attractive and profitable for all kinds of cybercriminals, from organized ransomware groups to sophisticated state-sponsored actors. The first factor is the high intrinsic value of medical data in black markets. Complete medical records, with all their personal, clinical, and financial information, are sold on underground forums at prices far higher than those of simple financial data such as credit card numbers. With a stolen credit card number, a criminal can make a few fraudulent purchases before the victim detects it and the bank cancels the card, limiting the damage and the criminal's profit. With a complete medical record, however, one can construct a solid false identity for health or life insurance fraud, obtain illegally prescribed controlled substances, commit tax or employment fraud, or even blackmail the affected person over stigmatizing conditions. Health data do not expire nor can they be "changed" by the victim: once it is circulating on the dark web, it can be exploited by multiple actors for years without the victim's knowledge or ability to mitigate damage.

Another factor that makes the healthcare sector irresistible to attackers is the extreme willingness of hospitals to pay for ransoms when they are compromised. Owing to the critical nature of their mission, hospitals simply cannot afford to be offline for extended periods without potentially lethal consequences for patients. When a ransomware attack encrypts critical systems for medical records, surgery scheduling, or lab results, hospital management faces a terrible, high-stakes ethical decision: pay the ransom demanded in cryptocurrency and regain access quickly or refuse on principle and face the consequences of days or even weeks without essential IT systems for patient care. Historically, many institutions have chosen to pay, even though they know this fuels the cybercrime business and attracts more attackers to the sector, and even paying does not technically guarantee that the data will be recovered or that it will not be published anyway. The pressure to provide care is so great that many organizations succumb to extortion.

A medium-sized hospital can also easily have hundreds of different software applications, from dozens of different vendors, with different security architectures, different update and patching cycles, different licensing and support models, and different contractual parties. Managing this technological heterogeneity from a security perspective is enormously complex and costly, and any weak link in this long chain can be exploited by attackers as an initial entry point. An MRI machine purchased fifteen years ago, running embedded Windows XP and no longer receiving security updates from the manufacturer but that remains connected to the hospital network because it needs to share images with the PACS, can be the unnoticed entry point that compromises the entire corporate system.

Many certified medical devices have regulated lifespans of ten, fifteen, or even twenty years and were designed at a time when cybersecurity was neither a design priority nor a regulatory requirement. They often run outdated versions of Windows, Linux, or proprietary operating systems that no longer receive security support from the original software manufacturers. Medical device manufacturers do not always provide security updates throughout the entire lifespan, and hospitals cannot afford or clinically justify replacing equipment that functions perfectly from a medical standpoint solely for cybersecurity reasons. The result is a fleet of inherently vulnerable connected medical devices that attackers know well, actively map, and routinely exploit.

The prevailing organizational culture in many hospitals, where cybersecurity is often perceived as a bureaucratic obstacle to urgent clinical work, rather than as essential protection for patients. Complex password policies are ignored or shared with "streamline" access; privileged access is not revoked in a timely manner when staff leave or change roles; new devices are connected to the network without going through established security checks; and USB storage devices are exchanged among staff without precautions or scanning. This culture—understandable in organizations where the stated top priority is caring for sick patients and where time is literally life—creates systemic vulnerabilities that attackers exploit on a massive scale through social engineering. Changing this culture requires a sustained effort in awareness-raising, ongoing training, and, crucially, designing security systems that are as unobtrusive as possible in critical clinical workflows.

9.2.1 THE EXPANDING ATTACK SURFACE AND CLINICAL CONSEQUENCES

Telemedicine for remote consultations, connected medical devices for home monitoring, remote access for professionals from their homes, and mandatory integration with other centers

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

to ensure continuity of care—all these legitimate and desirable trends exponentially increase the hospital's total attack surface. Every new authorized external connection is a potential entry point for attackers. These connections are often established under the pressure of immediate clinical need, without adequate prior analysis of the security risks they introduce, simply because the demand for care is urgently needed. The result is an increasingly porous and und hospital network, with multiple weak points that attackers can systematically explore and exploit.

In far too many institutions, all devices logically share the same flat network such that an attacker who initially gains entry through a weak point can move laterally without significant obstacles until it reaches the most life-critical systems; this stems from the frequent lack of adequate network segmentation in many hospitals. A human resources administration computer infected by a phishing email can serve as a bridge to eventually reach chemotherapy infusion pumps or ICU vital sign monitoring. Network segmentation—physically and logically separating life-critical devices from administrative systems and those with public or internet access—is a basic security measure that many hospitals have historically failed to implement because of cost, complexity, or lack of awareness.

Another limiting factor is the chronic shortage of qualified cybersecurity professionals in the sector. The global market for cybersecurity experts is extremely competitive, and public or nonprofit hospitals generally cannot offer the salaries and benefits of banks, technology consulting firms, or large software companies. Hospital IT teams are often overwhelmed in keeping legacy systems running, with no time or resources to devote to proactive security, threat analysis, or responding to sophisticated incidents. Partially outsourcing security functions to specialized firms can help, but it introduces its own trust risks and does not completely replace the need for in-house staff with deep knowledge of the specific clinical environment.

The consequences of a successful cyberattack on a hospital go far beyond direct economic damage or institutional reputation. First and foremost are the human lives that could be lost or harmed. In 2017, the WannaCry ransomware attack, which exploited a known window vulnerability, paralyzed dozens of National Health Service hospitals in the United Kingdom, canceling thousands of scheduled medical appointments, elective surgeries, and urgent diagnostic procedures. In 2020, a ransomware attack on a university hospital in Düsseldorf, Germany, forced the transfer of a critically ill emergency room patient to another facility more than 30 kilometers away; the patient died during the transfer, and German authorities formally investigated whether the delay caused by the cyberattack directly contributed to her death, which would have potentially constituted the first documented ransomware-related homicide. In 2021, a massive attack on the Irish healthcare system caused systems to be paralyzed for weeks, the cancellation of thousands of appointments, and the permanent loss of years of clinical data from patients who had not been properly backed up. Behind every aggregate statistic are individual patients who suffer real consequences from the disruption of their medical care.

Cyberattacks affect not only the availability of computer systems but also, more insidiously, the integrity of stored clinical data. A sophisticated attacker who gains privileged access could silently modify a patient's laboratory results, change a critical dosage in an electronic prescription, or subtly alter a diagnostic image, causing medical harm that is difficult to detect at the time and potentially lethal. If robust data integrity auditing mechanisms are not in place, it may never be known with certainty whether a clinical error was truly a human error or was maliciously induced by an attacker. This type of data integrit t attack—more sophisticated and stealthier than visible ransomware—is also potentially more dangerous because it directly undermines the fundamental trust in clinical information upon which all medical decisions are based.

9.2.2 COMPREHENSIVE RESPONSE: TECHNICAL, ORGANIZATIONAL, AND CULTURAL

An effective response to this growing and evolving threat cannot be solely technical, nor can it be limited to simply purchasing more firewalls or antivirus software. It must be comprehensive, organizational, cultural, and, ultimately, clinically oriented. Hospital cybersecurity leaders must sit at the same coordination table as clinical and nursing leaders do, mutually understanding that decisions regarding IT security have direct and sometimes critical implications for the ability to care for patients, and vice versa. Clinicians and nursing staff must proactively understand that security measures are not unnecessary bureaucratic hurdles imposed by IT professionals out of touch with reality but rather essential protection for their patients' safety, analogous to hand hygiene or medication verification. Hospital executive leadership must allocate adequate financial and human resources to cybersecurity, explicitly recognizing that it is as important to

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

modern patient safety as traditional clinical safety or the prevention of nosocomial infections.

Staff awareness and ongoing training are key components of any defense strategy. A single careless click on a link in a sophisticated phishing email can initially compromise the entire corporate system. A weak or shared password can be the gateway for privilege escalation. A USB drive found in the parking lot and curiously plugged into a workstation could contain malware specifically designed for the healthcare sector. Training all hospital staff—from medical specialists and nursing staff to administrative personnel, lab technicians, and cleaning and maintenance staff—in basic cybersecurity hygiene practices is an investment with a guaranteed return on security. This training cannot be a one-time event at the start of employment; it must be ongoing, adapted to emerging threats identified in the environment, and periodically evaluated through phishing drills or incident response exercises.

Active collaboration among hospitals and between hospitals and regulatory and law enforcement authorities is also essential for effective collective defense. Cyber attackers share information about new vulnerabilities, successful attack tactics, and vulnerable targets in clandestine forums; legitimate defenders must do the same in authorized and trusted forums. Sharing technical indicators of compromise (IOCs), issuing timely alerts about newly detected phishing campaigns, and reporting incidents so that other hospitals can proactively protect themselves are practices that strengthen security across the entire healthcare sector. In many developed countries, healthcare-specific information-sharing and threat-analysis centers (ISACs) already exist that facilitate this necessary collaboration.

For medical software developers who create products deployed in these vulnerable hospital environments, understanding that the hospital is an implicit attack vector means designing with security as an absolute priority from the earliest stages of development, not as a late-stage add-on. In the design of medical software, one cannot assume that the hospital environment where the product will be deployed will be secure or benign; one must explicitly design with the assumption that the environment will be hostile, that there will be active attackers and that the software must withstand them. Medical data must be encrypted at all times, communications between components must be mutually authenticated, access must be thoroughly audited, and the software must be architecturally updatable because threats are constantly evolving and vulnerabilities are inevitably discovered after initial deployment.

For clinicians who use these systems daily, understanding this reality of persistent threats means accepting that cybersecurity is an inherent part of their modern professional responsibility. Reporting suspected security incidents, following established access policies, and actively participating in security training are not optional activities that can be ignored because of a lack of time. They are as much a part of their fiduciary responsibility to patients as they are to wash hands between patients, verify the patient's identity before administering medication, or maintain the confidentiality of information during conversations. Because a cybersecurity breach can have clinical consequences as serious as a traditional medical error, even if the pathways to harm are different.

The future of hospital cybersecurity inevitably hinges on the widespread adoption of "zero-trust" architectures, in which no device, no person, and no network connection is considered trustworthy by default without continuous verification. It involves comprehensive network segmentation, widespread encryption of data in transit and at rest, mandatory multifactor authentication for all privileged access, and continuous, automated monitoring of behavioral anomalies on the network. Effective public–private collaboration among hospitals, device manufacturers, software developers, and authorities is needed. It involves the development and adoption of specific, rigorous security standards for connected medical devices. Above all and ultimately, it involves widespread cultural recognition that modern cybersecurity is indistinguishable from traditional clinical safety and that protecting IT systems directly protects patients who trust us with their care.

9.3 SECURITY BY DESIGN, NOT BY PATCH

There is a way of building software—unfortunately still prevalent in many commercial development environments—that could be colloquially described as "functionality first, security second, if there's time and budget left." In this superficial architectural approach, developers focus obsessively on quickly implementing the functional features the user has requested, on making the software "work" for normal use cases, and on meeting contractual delivery deadlines. Security is considered retrospectively at the end of the project as a superficial layer added on top of the

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

finished product: configuring a network firewall here, enabling some basic encryption there, and implementing rudimentary access controls in the user interface. This approach produces systems that are inherently insecure by design because security is not integrated into their fundamental structure but rather slapped on with digital duct tape as an afterthought. When a security vulnerability is inevitably discovered during operation, a patch is hastily applied; when another vulnerability appears, yet another patch is added. Over time, the system becomes a tower of accumulated technical patches, difficult to maintain operationally, impossible to fully audit, and surprisingly easy for a determined attacker to penetrate. This is patch-based security, and it is exactly what we must not do, under any circumstances—ly in healthcare environments where software security can literally be a matter of life and death for patients.

The mature and professional alternative is security by design: an architectural approach where security is considered from the very start of the project, as a nonnegotiable functional requirement, on the same level of importance as clinical functionality or system performance. In this approach, all architectural decisions are made with explicit consideration of their potential security implications. Threat models are systematically analyzed before the first line of production code is written. Security testing is continuously integrated into the development process, not as an isolated final milestone. Security is not something hastily added at the end but rather something that permeates the entire digital product, from its architectural foundations to its end-user interface, as an emergent property of sound design. This approach, while requiring a greater initial investment of time and expertise, produces systems that are inherently more secure, more resilient to attacks, and surprisingly easier and cheaper to maintain in the long term because it avoids the technical debt of accumulated patches.

9.3.1 FUNDAMENTAL PRINCIPLES OF SECURITY BY DESIGN

Early and systematic identification of potential threats: Before beginning to build any component, the team must rigorously ask itself, Who might have the motivation and interest to attack this specific system? What stored or processed data are most valuable on the black market or for competitive intelligence? What technical and human entry points exist in the proposed architecture? What specific types of harm could a successful attacker cause to patients, the institution, or public trust? These questions are not abstract academic exercises; they must be answered as concretely and realistically as possible, and the documented answers must directly guide subsequent technical design decisions. A well-developed threat model, following methodologies such as the STRIDE or PASTA, identifies the most relevant risks and allows security measures to be prioritized on the basis of their probability and potential impact.

Layered defense in depth:It is not enough to rely on a single layer of security; no matter how sophisticated it may be; one must professionally assume that any individual layer may eventually fail and have additional layers of protection that kick in when that happens. An attacker who manages to bypass the network perimeter firewall should immediately encounter strong multifactor authentication requirements. If authentication is compromised through social engineering, there should be granular access controls based on the principle of least privilege. If access controls fail because of a configuration error, there should be a layer of comprehensive auditing to detect and respond to unauthorized access. Each additional layer of defense exponentially increases the difficulty for the attacker and provides multiple opportunities for early detection and timely response.

The principle of least privilege applied systematically: every human user, every software process, and every system component should be assigned only the permissions strictly necessary to perform its legitimate function, and absolutely nothing more. A primary care physician needs access to the records of their own assigned patients but not to those of all patients in the hospital. An application for viewing lab results should not have permissions to modify them. A batch process that runs in the background to generate statistics should not have network access unless explicitly needed. Applying this principle consistently drastically reduces the total attack surface and limits the potential damage that a single compromised component can cause because its capabilities are restricted by design.

Security by default in configuration: the initial default configuration of any system should be as secure as possible, not the most open, permissive, or convenient for the administrator. If the system has configuration options that are inherently insecure or risky, they should be disabled by default until an experienced administrator explicitly enables them, with full, documented knowledge of the risks being assumed. This prevents systems that are hastily deployed in hospital

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

environments from being exposed because of carelessness or ignorance. It also greatly facilitates long-term security management because the secure configuration is factory default and does not require specialized intervention to enable it.

Separation of critical responsibilities: particularly sensitive functions should not rest with a single person or a single unsupervised technical component. The professional who authorizes privileged access should not be the same one who subsequently audits that access. The developer who writes the code should not be the only one who tests it from a security perspective. The systems administrator who manages the infrastructure should not be the same person who defines organizational security policies. This separation of duties hinders internal fraud and undetected abuses of power and provides independent cross-checks that increase the system's overall reliability.

Proactive minimization of the attack surface: Every implemented feature, every exposed interface, every open network port, and every service running continuously constitute a potential entry point for a creative attacker. For this reason, anything not strictly necessary for clinical operations must be removed or disabled. If an advanced feature is used by only a few specialized users, it should be disabled by default and require explicit activation. If a network service is not missioncritical, it should not run at all. If a remote administration interface is not routinely used, it should be closed and only opened temporarily when necessary. A minimalist, clean system is inherently easier to secure than one overloaded with features that no one uses but that leave doors open.

Secure and controlled error and exception handling: When a system experiences a failure, its error messages should not reveal technical information that could be useful to an attacker in refining their strategy. Error messages visible to the end user should be generic and noninformative regarding the internal structure, while full technical details should be directed solely to internal audit logs, which are protected and analyzable. A message that simply says "incorrect credentials" is secure because it does not distinguish whether the user exists or not; one that says "user does not exist" reveals valuable information about which identifiers are valid in the system. Errors must always be handled via secure code paths that do not leak sensitive information, and robust mechanisms must be designed for the secure recovery of a consistent state.

Secure and verifiable updatability: No software system is perfect, nor will it remain free of vulnerabilities forever; new weaknesses will always be discovered after initial deployment. For this reason, it is absolutely essential that the system be architecturally designed to be securely updatable (), without introducing new vulnerabilities into the update mechanism itself, without interrupting critical clinical service more than strictly necessary, and with cryptographic guarantees that the update is authentic and intact and has not been tampered with in transit by an attacker. Remote update mechanisms must be protected by robust digital signatures and must allow for gradual canary deployments that minimize the impact of potential issues not detected during testing.

Resilience to attacks and failures: Medical systems must be explicitly designed to continue operating—albeit in a degraded but safe manner—even when under active attack or after suffering partial compromises. This requires redundant architectures, isolation of critical components, and rapid, automated recovery capabilities. A system that goes completely down in the face of a distributed denial-of-service (DDoS) attack is not a secure system; it is a fragile system that has failed in its design. Security is not just about preventing successful attacks but also about withstanding them when they occur, degrading them gracefully, and recovering them quickly without loss of data or essential functionality.

Operational transparency and complete auditability: A truly secure system must leave detailed, immutable traces that allow one to retrospectively determine what occurred, who performed which action, and exactly when. These audit logs must be technically protected against subsequent tampering and must be analyzable proactively to detect anomalous behavior patterns before serious damage occurs. However, auditing is not just a forensic tool for use after an incident; it must also be used proactively in real time to identify suspicious behavior, unusual access, or patterns suggesting partial compromise of the system.

9.3.2 THE CULTURE OF SECURE DEVELOPMENT AND SHARED RESPONSIBILITY

For software developers working in the healthcare sector, consistently applying these ten principles in their daily work requires a profound and sustained shift in mindset. Security is not

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

something delegated to a specialized team of "security experts" who review the code at the end; it is the shared responsibility of all members of the development team. Every line of code must be written with active consideration of how it could be exploited by a resourceful and motivated attacker. Every architectural design decision must be explicitly evaluated from the perspective of its security implications. This does not mean being paranoid or paralyzed by risk analysis; it means being professionally rigorous. In other traditional engineering disciplines, safety is a natural and inseparable part of the work: a civil engineer does not design a public bridge without thoroughly considering its structural resistance to earthquakes, extreme winds, or unforeseen overloads. In medical software engineering, it should be exactly the same.

For clinicians and healthcare professionals involved in information system development or procurement projects, understanding these fundamental principles of security by design helps them evaluate and support the technical decisions of development teams, even when these involve apparent operational drawbacks. When a developer explains that a requested feature cannot be implemented quickly but insecurely because of security principles, they are not making technical excuses or being stubborn; they are professionally applying secure design principles that protect patients. When the system requires additional multifactor authentication to access particularly sensitive data, it is not an unnecessary bureaucratic hassle; it is an extra layer of defense that can save lives. Understanding these fundamentals improves interdisciplinary collaboration and reduces mutual frustration between clinicians and technical staff.

Security by design is neither an optional luxury nor a market differentiator; it is an absolute necessity in an environment where cyberattacks are becoming increasingly sophisticated and frequent and where the consequences of a security breach can be literally fatal for individual patients. Hospitals and healthcare systems cannot afford systems that are "secure" only until a determined attacker puts them to the test; they need systems that are architecturally designed to withstand attacks, to detect intrusions early, to respond effectively, and to recover quickly. This is achieved only if security is present, considered, and prioritized from the very first design sketch, in every line of code written, in every architectural decision made, and in every code review performed. It is more costly in terms of time and expertise at the beginning of the lifecycle but is infinitely less expensive than the catastrophic consequences of a security breach in production that affects real patients.

9.4 AUTHENTICATION, AUTHORIZATION, AND ACCESS AUDITING

> "At the heart of healthcare IT security lie three eternal questions: Who are you? What can you do? And what have you done? The correct answers to these questions separate patient care from the chaos of intrusion."

Let us imagine a completely ordinary scenario in any modern hospital during a typical work morning. An internal medicine physician accesses the electronic health record system to urgently check the laboratory test results of a patient who has just arrived at the clinic with acute symptoms. A ward nurse carefully records the administration of intravenous medication in the electronic care summary of a postsurgical patient. An admission clerk updates the contact information for a patient whose address has recently changed. A medical researcher requests formal access to anonymized data from a patient cohort for an epidemiological study approved by the ethics committee. A chronic care patient checks the results of their latest follow-up blood test from home via the hospital's portal. Behind each of these seemingly simple and routine actions lies three fundamental security questions that any robust healthcare information system must be able to answer accurately and reliably: Who exactly is the person or system attempting to access the data? What specific permissions do they have to perform the requested action on that specific data? Once they have accessed the data and performed operations, what permanent and immutable record remains of what they have done? These three universal questions correspond to the three inseparable pillars of access security in any sensitive information system: authentication, authorization, and auditing.

Authentication is the technical process of reliably verifying the identity of who or what is attempting to access the system. It is the concrete answer to the fundamental question, "Who are you truly?" In the healthcare setting, where identity confusion can have serious clinical consequences, this process must be particularly robust and multifactorial because the correct identification of users is critical to ensuring that sensitive information reaches precisely those who are legitimately authorized to access it and absolutely not those who are not.

Authentication methods have evolved significantly from simple, memorizable passwords—

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

which have proven insufficient against multiple attack vectors—toward more secure and resilient multifactor authentication (MFA) schemes. Multifactor authentication combines something the legitimate user knows (traditionally a complex password) with something they physically possess (a registered mobile phone, an access smart card, a hardware token) or something they inherently are (a unique fingerprint, a facial recognition pattern, voice geometry). This combination of factors is rapidly becoming the de facto standard for accessing critical systems in hospital settings because the compromise of a single factor does not allow unauthorized access.

Authentication is not only necessary for human users who interact directly with the system. Automated systems that communicate with each other over networks—such as when a clinical laboratory sends results electronically to the central medical record system—also need to authenticate each other to establish trust. Both systems must be cryptographically certain that they are actually communicating with the entity they believe it to be and not with an imposter or an intercepted system. This is achieved through X.509 digital certificates issued by trusted certification authorities, shared and periodically rotated API secret keys, or other robust machine-to-machine authentication mechanisms. In a modern hospital environment with dozens or even hundreds of interconnected specialized systems, managing these digital identities in a consistent, scalable, and secure manner is an operational challenge that requires mature, specific infrastructure, such as centralized directory services (LDAP, Active Directory) or modern federated identity management (IAM) platforms.

9.4.1 GRANULAR AUTHORIZATION AND SPECIAL CONTEXTS

Authorization is the second essential pillar, which comes into play once the requester's identity has been successfully verified through authentication. It answers the critical question: "What specifically are you allowed to do in this system with these data?" Not all authenticated users have—or should they have—the same access and modification permissions. A treating physician can view and modify the medical records of patients under direct care but should not be able to change their own system credentials or access human resource data. A nurse can record vital signs and medication administration but should not be able to modify primary medical diagnoses. A patient can view their own personal data and clinical results, but not those of other patients or the physician's confidential notes. A researcher can access aggregated and anonymized data for their approved study but not individually identifiable information. Authorization rigorously defines these access rules on the basis of organizational policies, legal regulations, and professional ethics and enforces them technically in an automated and unavoidable manner.

The most common and traditional model for implementing authorization in information systems is role-based access control (RBAC).(6) In this classic model, individual users are assigned to predefined functional roles (doctor, nurse, administrative staff, patient, and researcher), and specific permissions for accessing resources are assigned collectively to these roles rather than to individuals. When an authenticated user attempts to perform an action on a resource, the system checks whether any of the roles assigned to that user explicitly have permission for that action in that context. This model is relatively simple to manage at scale and easy to understand conceptually, but it can be overly rigid and lacking in granularity in complex clinical environments where real-world situations define simple categorizations.

Owing to this limitation, more flexible and context-aware authorization models are emerging and maturing, such as attribute-based access control (ABAC), which considers multiple factors simultaneously when authorization decisions are being made: the user's nominal role, the operational context (time of day, physical location, device used), the type and sensitivity of the requested data, and whether or not there is an established therapeutic relationship with the patient whose data are being accessed.(7)

In the specific healthcare setting, authorization presents unique characteristics that make it particularly complex to design and implement correctly. One of these characteristics is authorization by temporary delegation: a physician may explicitly authorize a specific nurse to perform certain advanced tasks on their behalf during their absence, such as modifying medication doses within an established protocol—a delegation that must be revocable and auditable. Another critical feature is authorization for medical emergencies: in extreme clinical situations where delay can cost lives, it may be necessary to temporarily allow access that would not normally be authorized for that professional in that context while always leaving a clear, auditable record to ensure that there was no abuse. Another is granular authorization at the field level: it is not enough to state generically that a physician "can view records"; it must be technically specified that they can view

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

the records of their assigned patients but not those of patients from other departments and that within each record, they can view certain clinical data but not others that are particularly sensitive (for example, they can view general medical diagnoses but not detailed notes on psychiatry or mental health, which require specific additional authorization).

Auditing is the third fundamental pillar, and it answers the retrospective yet crucial question: "What exactly did you do with the system, and when?" Every access to sensitive identifiable data, every modification of a clinical record, every failed authentication attempt, and every privileged administrative operation must be permanently recorded in audit logs that allow for the subsequent reconstruction of events with forensic precision. These audit logs must include sufficient and detailed information, such as who accessed the data (authenticated identity), what specific data (resource identifier), from which network location and device (IP address, hostname), exactly when (precise timestamp with time zone), with what result (success, failure, denial), and what specific operation was attempted or performed. The logs must be technically protected against subsequent tampering so that not even system administrators with elevated privileges can modify or delete them without leaving an immutable trace of such an attempt. They must be efficiently analyzed, preferably with automated correlation and anomaly detection tools that identify suspicious patterns without continuous human intervention.

9.4.2 THE TRIAD IN ACTION: INTEGRATION AND CONTINUOUS MONITORING

Auditing is not merely a forensic tool for use after a security incident occurs; it also serves an important preventive deterrent function. Knowing that all access is permanently logged and reviewable deters potential nosy or malicious insiders from unjustifiably accessing data that are not concerned with them. In many audited hospitals, cases have sadly been detected among healthcare professionals who improperly access the medical records of famous patients, personal acquaintances, or even ex-partners, simply out of morbid curiosity—serious privacy violations that auditing allows them to be detected, documented, and subject to disciplinary action.(8) Auditing also enables the technical investigation of confirmed security incidents, precisely identifying how an external attacker gained entry, which compromised the credentials they used, and what specific data they may have viewed, copied, or modified during their intrusion—information essential for regulatory reporting and mitigation.

Detailed audit logs in a hospital of this size generate massive volumes of data that defy manual analysis. A large hospital can easily generate millions of audit events per day, each with multiple metadata fields. Managing this massive amount of information requires specialized security information and event management (SIEM) systems—(9)—that centralize logs from multiple heterogeneous sources, normalize them into standard formats, and apply sophisticated temporal and contextual correlation rules to detect emerging threats. These SIEM systems are capable of identifying complex patterns that would be impossible for a human operator to detect manually: a specific user accessing an unusually high number of patient records in a short period of time, access from a geographic location unusual for that regular user, multiple consecutive failed authentication attempts followed by a suspicious success, or the correlation of seemingly innocuous events across different systems that together suggest a coordinated attack pattern.

The functional relationships among authentication, authorization, and auditing are intrinsically close and must be designed and implemented in an integrated manner from the architecture level, not as separate silos. Authentication provides the verified identity that authorization uses as input for its permission decision, and auditing comprehensively records both the results of authentication processes (successes and failures) and the authorization decisions applied, as well as the specific actions subsequently performed on the resources. A failure or weakness in any of these three links potentially compromises the integrity of the entire security system. For this reason, it is absolutely critical that medical software developers understand in depth how these three pillars are related to one another and implement them coherently and consistently in their applications, using frameworks and libraries that facilitate this proper integration.

For healthcare software developers, robustly implementing these three fundamental mechanisms requires selecting and properly configuring mature technologies and proven standards. For authentication, widely adopted standards such as OAuth2 for delegated authorization, OpenID Connect for token-based federated authentication, and SAML for integration with enterprise legacy systems exist, which enable secure integration with centralized corporate identity providers. For authorization, there are standardized policy languages such as

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

XACML (eXtensible Access Control Markup Language) or, more recently, JSON-based policies (JSON Web Tokens with authorization claims) that allow for the definition of complex, contextual, and automatically evaluable access rules. With respect to auditing, standardized formats such as Syslog or CEF (Common Event Format) facilitate integration with corporate SIEM systems.(10) However, technology is only an enabler; what matters most is designing the right security policies from the outset on the basis of a careful and documented analysis of actual clinical requirements, applicable legal obligations, and patients' ethical expectations.

For clinicians and healthcare professionals who use these systems daily, understanding these security mechanisms helps them grasp why the system sometimes requires them to authenticate themselves with a second factor or why, under certain circumstances, they cannot access certain data even though they legitimately believe they should be able to. It also helps them be aware that all their access actions are permanently logged and that misguided curiosity or negligence in safeguarding their credentials can have serious disciplinary consequences, in addition to ethical violations. It provides them with the basic technical vocabulary needed to participate constructively in organizational discussions about access policies, contributing their invaluable expert knowledge about which accesses are truly legitimate and necessary in each specific clinical context.

A particularly delicate aspect of design in the healthcare domain is access management in acute medical emergencies, where security must not hinder the speed of clinical response. When a patient arrives at the emergency room in critical condition and is unconscious, the on-call physician needs immediate access to their prior medical history even if they are not the patient's regular physician or have an established therapeutic relationship. Information systems must incorporate technical "break the glass" mechanisms that allow for these exceptional urgent accesses but with compensatory controls to prevent systematic abuse. Typically, these emergency mechanisms require healthcare professionals to provide a documented clinical justification at the time, generate immediate automatic alerts to security and privacy supervisors, and leave a detailed record of the entire transaction that must be reviewed later by an audit committee. A delicate and necessary balance between the urgent clinical need to save lives and the fundamental protection of patient privacy is needed.

The correct integration of these security mechanisms with the patient's real identity is another critical technical challenge. When a patient accesses their own personal data through a hospital's web portal or mobile app, they need to authenticate themselves securely and verifiably, but it is also absolutely essential to ensure, both technically and organizationally, that they see only their own complete data—not that of another patient, which might appear due to a configuration error or software bug. This requires unequivocally and verifiably linking the patient's digital identity (their portal account) to their unique clinical identifier in the medical records database and ensuring through rigorous testing that this link is correct and persistent. Errors at this integration point can lead to serious and widespread privacy breaches, such as a patient mistakenly viewing another patient's complete medical record, with devastating legal and reputational consequences for the institution.

The future of authentication, authorization, and auditing in the healthcare sector points toward greater technical and contextual sophistication. Continuous, passive biometric authentication, combined with user behavior analysis (behavioral biometrics), promises more robust identity verification that is simultaneously less intrusive to the clinical workflow. Dynamic authorization based on real-time operational context and policies evaluated using artificial intelligence will enable more granular, adaptive, and context-appropriate access control.

AI-powered auditing will detect anomalous behavior patterns that today go completely unnoticed amid the noise of millions of events. However, these promising technological advances must not make us forget the fundamental point that the security of access to healthcare information is, ultimately and beyond technology, a matter of human trust. Patients trust that their most intimate data are protected by competent and ethical professionals. Healthcare professionals trust that they can access the information they legitimately need when they need it to provide better care. Society's trust that the healthcare system as a whole treats personal health information with respect to, care, and protection it deserves because of its unique nature.

Once we have established robust mechanisms to control who accesses what information and to leave an immutable record of it, how do we transform all that accumulated information ethically and securely into useful medical knowledge through statistical analysis and artificial intelligence, without betraying the trust of the patients who entrusted us with their data?

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

REFERENCES

Introduction - Beyond the HIPAA Privacy Rule - NCBI Bookshelf [Internet]. Washington (DC): National Academies Press (US); 2009 [cited 2026 May 13]. Available from: https://www.ncbi.nlm.nih.gov/books/NBK9576/

GDPR vs HIPAA: Key Differences & Compliance 2026 [Internet]. [cited May 13, 2026]. Available from: https://www.atlassystems.com/blog/gdpr-vs-hipaa

Protection of Sensitive Data: What Companies Need to Know – Nunes & Asociados Study [Internet]. [cited May 13, 2026]. Available at: https://estudionunes.com.ar/proteccion-de-datos-sensibles-lo-que-las-empresas-deben-saber/

Personal Data Protection Law: Key Points for Businesses [Internet]. [cited May 13, 2026]. Available at: https://www.innovaciondigital360.com/cyber-security/proteccion-de-datos-personales-en-argentina-claves-principales/#Ley_N°_25326_de_Argentina

Regulations on Personal Data: Ministry of Finance [Internet]. [Cited May 13, 2026]. Available at: https://www.hacienda.gob.es/es-ES/El Ministerio/Paginas/DPD/Normativa_PD.aspx

Coyne EJ, Weil TR. ABAC and RBAC: Scalable, flexible, and auditable access management. IT Prof. 2013;15(3):14-6.

Hu VC, Ferraiolo D, Kuhn R, Schnitzer A, Sandlin K, Miller R, et al. Guide to Attribute-Based Access Control (ABAC) Definition and Considerations. January 2014 [cited May 13, 2026]; Available from: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf

UCLA Hospitals Receives $865K HIPAA Fine for Failing to Protect Celebrity Medical Records [Internet]. [cited May 13, 2026]. Available from: https://www.hipaajournal.com/ucla-hospitals-receives-865k-hipaa-fine-failing-protect-celebrity-medical-records/

Security information and event management (SIEM) systems | Internal Revenue Service [Internet]. [cited May 13, 2026]. Available from: https://www.irs.gov/privacy-disclosure/security-information-and-event-management-siem-systems

Ferraiolo D, Chandramouli R, Kuhn R, Hu V. Extensible access control markup language (XACML) and next generation access control (NGAC). ABAC 2016 - Proc 2016 ACM Int Work Attrib Based Access Control colocated with CODASPY 2016 [Internet]. March 11, 2016 [cited May 13, 2026];13-24. Available at:/doi/pdf/10.1145/2875491.2875496?download=true

https://doi.org/10.62486/978-9915-9876-8-2.Ch09

Chapter 10 / Capítulo 10

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch10
Pages: 167-182

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 10. Construction and Validation of Clinical Datasets

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "An algorithm is only as good as the data it is trained on. If the data *are biased, the algorithm will learn biases. If the data are erroneous, the algorithm will learn errors. If the data reflect* only a part of reality, the algorithm will be blind to everything else. That is why, before we talk about artificial intelligence, neural networks, or deep learning, we must address the fundamentals: how we build the datasets that will power these systems."

Imagine that we want to construct an algorithm that can detect pneumonia in chest X-rays. We need thousands of images, each labeled as "pneumonia" or "not pneumonia." However, who assigns those labels? A radiologist with twenty years of experience? A first-year resident? A medical student? What if two radiologists disagreed on an image? What is the truth? What if the image shows an ambiguous finding—one that could be pneumonia or something else? These questions, which may seem minor, are actually at the heart of the problem. The quality of the algorithm depends on the answers we provide, and patient safety depends on the quality of the algorithm.

Building clinical datasets is among the most underappreciated yet most critical aspects of developing artificial intelligence in healthcare. While technical aspects (which neural network architecture to use and which optimization algorithm) receive disproportionate attention, the dirty work of collecting, cleaning, and labeling data is often relegated to the background. Researchers compete to publish models with better metrics but rarely ask how those metrics were obtained, what the quality of the data was, or what biases might be hidden. This imbalance between attention to the model and attention to the data are dangerous because a brilliant model trained on mediocre data will produce mediocre results, and no one will notice until it is too late.

In this chapter, we explore the fundamental aspects of building and validating clinical datasets. We will begin with the clinician's role in data annotation, a role that goes far beyond simply applying labels. We will continue with the challenge of interrater disagreement and how to measure it using the kappa coefficient. We address biases in data collection, which may be present even before labeling begins. Finally, we discuss sample size and statistical power, which are essential concepts for determining how much data we need for our model to be reliable. The goal is not to turn the reader into a machine learning expert but to equip them with the questions they should ask when faced with any clinical dataset, and the criteria for evaluating whether the data they are asked to work with are trustworthy.

10.1 WHO ANNOTATES AND HOW: THE ROLE OF THE CLINICIAN

There is a persistent fantasy in the world of medical artificial intelligence that is both appealing and dangerous. In this fantasy, a team of engineers receives hundreds of thousands of X-ray images and processes them through a crowdsourcing platform where medical students label them in a matter of weeks, and from this effort emerges a dataset considered "ground truth." The numbers are impressive: one hundred thousand annotated cases, fifteen thousand hours of work, coverage of twelve different pathologies. With these data, a deep neural network is trained that achieves 94% accuracy in internal validation. An article is published in a high-impact journal, funding is secured for a startup, and the algorithm is prepared for deployment in hospitals. However, when it finally faces real patients, something goes wrong. The system confuses atypical pneumonias with heart diseases, ignores subtle findings that any resident would detect, and generates predictions that senior radiologists consider absurd. What happened? In the rush to scale up, quantity was confused with quality, speed with wisdom, and the opinion of those just beginning their training with the judgment of those who had been interpreting images for decades.

The reality of clinical annotation is vastly different from this industrialized fantasy. It begins with a question that seems simple but rarely is as follows: What exactly are we trying to capture? When we say we want to detect "pneumonia" on a chest X-ray, we are using a term that, in medicine, encompasses a wide range of conditions. There is typical bacterial pneumonia with its classic lobar consolidation but also viral pneumonia with diffuse interstitial infiltrates, pneumonia in immunocompromised patients who do not mount a visible inflammatory response, and evolving pneumonia that shows changing patterns. There are microbiologically confirmed pneumonias and others that are diagnoses of exclusion. Some images show unequivocal findings, while others are ambiguous even to experts. Before any student touches an image, before a single line of code is written, a group of experienced clinicians must sit down to define with surgical precision what constitutes each category, what criteria differentiate one from another, and how to handle cases

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

that do not fit neatly into any box.

10.1.1 ART OF DEFINING THE INDEFINABLE: PROTOCOLS THAT SAVE ALGORITHMS

The construction of a valid annotation protocol is perhaps the most underestimated task in the entire development of clinical artificial intelligence. A good protocol is not simply a list of categories; it is a living document that anticipates the complexity of real-world medical practice and provides clear guidelines for navigating that complexity. It must include precise operational definitions, representative examples of each category, explicit exclusion criteria, and rules for handling ambiguity. When two expert radiologists disagree on an image, both may be correct because the image is inherently ambiguous. A smart protocol does not force false certainty but rather establishes procedures to document that ambiguity and, when appropriate, allows for multiple labels or categories of uncertainty.

The process of developing these protocols must be iterative and collaborative. You cannot write a protocol in a two-hour meeting and expect it to work. Pilot tests with small case sets, analysis of interrater variability, refinement of definitions, and further testing are needed. Clinicians involved in this phase must be true experts, not only in their medical specialty but also in the ability to articulate their clinical reasoning. Many excellent physicians struggle to articulate why they see something in an image; their knowledge is tactile, intuitive, and developed through years of practice. The challenge is to translate that tacit knowledge into explicit rules that others can follow without losing the essence of what makes an expert an expert.

The handling of ambiguous cases is particularly critical. In real clinical practice, a physician may request additional information: How has the patient been doing? Do they have a fever? What do the blood tests show? In the context of artificial intelligence annotation, this additional information may or may not be available. If it is available, it improves the quality of the annotation but introduces a bias: the algorithm will learn to make a diagnosis on the basis of information that will not be available at the time of the actual prediction. If it is not available, the annotations may be less accurate but more realistic from a deployment standpoint. This decision is not technical but strategic and must be made consciously with a full understanding of the implications.

Quality controls during the annotation process are just as important as final checks are. It is not enough to review 10% of the cases at the end of the project; mechanisms are needed to detect deviations in real time. This may include inserting gold cases whose correct label is known, monitoring the annotation speed (too fast often indicates inattention), and statistically analyzing the label distribution (abrupt changes may indicate that an annotator has misinterpreted the instructions). When problems are detected, correction should be immediate and formative and not punitive. The goal is not to punish errors but to improve the process continuously.

10.1.2 THE EXPERTISE PYRAMID: BUILDING SUSTAINABLE ANNOTATION TEAMS

Choosing who performs the annotations is a balancing act between the ideal and the feasible. In a perfect world, every image, every record, every signal would be reviewed by the top expert available—someone with decades of experience and international recognition in their field. However, these experts are scarce and expensive and have clinical responsibilities that they cannot abandon. A senior radiologist may take fifteen minutes to analyze a complex CT scan; multiplied by thousands of cases, the project becomes economically and temporally unfeasible. The solution is not to sacrifice quality but to design annotation architectures that distribute expertise intelligently.

An effective model is the three-tier pyramid. At the base, annotators with basic medical training but no specialization (advanced medical students, specifically trained imaging technicians) handle clear and routine cases, which represent the majority of the volume. At the intermediate level, residents or junior specialists review cases in which the base-level annotators have flagged as questionable and conduct quality checks on the basis-level work. At the top, senior experts focus exclusively on the most complex cases, on resolving discrepancies among lower-level raters, and on defining and reviewing protocols. This structure allows experts' time to be used where it truly adds value, while the volume is handled by more abundant resources.

Supervision and mentoring are essential components of this pyramid. Entry-level annotators should not work in isolation; they must have access to consultations with higher-level staff, ongoing training sessions, and regular feedback on their performance. An annotator who

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

receives only written instructions and never interacts with an expert will develop bad habits that will become ingrained over time. In contrast, an annotator who is part of a team where they can ask questions, discuss difficult cases, and see how experts reason will continuously improve their skills. This investment in human talent development pays dividends in data quality.

Documenting who annotated what and with what level of confidence is crucial for subsequent data science. Not all labels in a dataset are equally reliable, and that's fine as long as we know it. A record indicating that a certain annotation was made by a first-year resident and marked as "uncertain" is more valuable than a label forced to be definitive. Machine learning algorithms can incorporate this uncertainty information, giving more weight to high-confidence annotations and treating uncertain annotations in a special way. Furthermore, this traceability allows for retrospective audits: if we discover that a specific annotator made systematic errors, we can identify and correct those cases without invalidating the entire dataset.

Annotator well-being is an often-overlooked variable that directly affects quality. Medical annotation is intellectually demanding and emotionally taxing work. Reviewing hundreds of images of severe pathology, reading reports on terminally ill patients, and making decisions that will affect the ability of algorithms to diagnose real people can lead to fatigue and stress. Protocols must include regular breaks, task rotation to avoid monotony, and strict limits on the number of cases per day. An exhausted annotator makes mistakes that they would not make when rested, and those mistakes multiply by the thousands in a large dataset. Investing in humane working conditions is not only ethical; it is pragmatic from the perspective of data quality.

10.1.3 BETWEEN LABORATORY PERFECTION AND THE NOISE OF REALITY: ANNOTATION CONTEXT BIASES

A fundamental paradox of annotation for medical artificial intelligence is that the very act of annotating creates an artificial context that can distort the data. When a radiologist knows they are annotating images for a research project—without the pressure of a hectic shift, without the responsibility of a diagnosis that will determine a real patient's treatment—their performance changes. They may be more meticulous, spend more time on each image, and look for findings that they might overlook in routine clinical practice. The result is annotations that represent an idealized medicine, a version of clinical practice that exists only in research laboratories. When the algorithm trained on these "superperfect" data is confronted with the reality of an overcrowded emergency department, where images are of lower quality, patients have multiple overlapping conditions, and time is scarce, the mismatch can be stark.

To mitigate this context bias, some projects have explored retrospective annotation based on real-world practice. Instead of asking doctors to annotate specifically for the project, they use the diagnoses that clinicians recorded in their daily work at the time of patient care. This approach has the virtue of capturing medicine as it is actually practiced, with all its noise and imperfections. However, it also has serious limitations: clinical diagnoses may be incomplete, influenced by administrative factors (coding for billing), or simply erroneous. A diagnosis of "pneumonia" in a medical record may reflect an initial suspicion that was later ruled out or a diagnosis of exclusion that was never confirmed. Using these data without additional validation can introduce systematic noise into the dataset.

The most promising solution is prospective annotation integrated into the clinical workflow. In this model, annotation is not a separate activity but a natural component of patient care. When a radiologist interprets an image in their usual PACS system, their structured report automatically becomes the annotation for the dataset. Artificial intelligence tools are embedded within the clinical environment, not on external platforms. This approach requires a sophisticated technological infrastructure and changes to workflows, but it produces high-quality data that reflects real-world practice. Furthermore, it enables continuous updating: the algorithm can be retrained with fresh data that reflect changes in epidemiology, new imaging techniques, or evolving diagnostic standards.

The relationship between clinicians and developers must evolve into true collaborative cocreation. Too many projects treat physicians as mere data providers, paying them by the hour for data entry without involving them in study design, interpretation of results, or implementation decisions. This dynamic is unsustainable and counterproductive. Clinicians who feel part of the team—who understand why certain data are collected and how it will be used and who have a say in the project's direction—contribute not only data but also invaluable clinical insights. They can point out biases in the design, suggest relevant variables that engineers might not

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

have considered, and anticipate implementation issues that are evident only from a practical perspective.

Proper recognition of clinical contributions is both ethical and practical. In academia, clinicians involved in data annotation should be listed as coauthors of the resulting papers if their intellectual contribution is substantial, not merely acknowledged in the acknowledgments section. In commercial projects, they should receive fair compensation that reflects the value of their expertise, not crowdsourcing rates. In all cases, there must be transparency regarding how the data will be used, who will have access to it, and what safeguards exist to protect patient privacy. This collaborative ethic is not only the right thing to do; it builds the trust necessary for top clinicians to want to participate in future projects.

AI-assisted annotation represents a promising evolution that does not replace clinicians but rather amplifies their productivity. In this model, a preliminary algorithm proposes labels for clear-cut cases, and the clinician focuses on reviewing and correcting, especially in cases where the algorithm has low confidence. Over time, as the algorithm improves, the proportion of cases requiring human attention decreases, allowing the volume of annotation to scale without sacrificing quality. However, this approach requires caution: if the initial algorithm has systematic biases, these can be perpetuated if the clinician tends to accept the system's suggestions. The design of these interfaces must encourage critical scrutiny, not passive acceptance.

Clinical annotation is not a necessary evil that we will eventually automate entirely but a fundamental activity of 21st-century medicine. As artificial intelligence becomes more deeply integrated into healthcare, the ability to generate high-quality labeled data becomes a critical competency for healthcare systems. Investing in training professionals in these skills, developing rigorous methodologies, and creating infrastructures that facilitate annotation is important for the future of medicine itself. With the development and execution of algorithms, neural architectures evolve, but the need to connect human medical knowledge with the potential of computing remains. In that connection, in that bridge between the clinician's mind and the bits of the dataset lies the true frontier of medical artificial intelligence.

10.2 DISAGREEMENT AMONG ANNOTATORS AND THE KAPPA COEFFICIENT

Imagine you have gathered three experienced radiologists in a room with a hundred chest X-rays. Your goal is to create the definitive dataset to train a pneumonia detection algorithm. Doctors work in silence, marking their diagnoses with the confidence that comes from experience. When they finish, you collect the annotation sheets and compare the results with anticipation. Seventy images show perfect agreement among the three: they all see pneumonia, or they all see a healthy chest. However, in thirty images, something is off. Fifteen showed an uncomfortable pattern: two radiologists agreed on the diagnosis, but the third strongly disagreed. In the other fifteen, it's total chaos: each doctor sees something different, and there's no majority to resolve the discrepancy. Your initial instinct is frustration. How is it possible that trained experts disagree so much? Did you choose the participants poorly? Are the images of poor quality? However, as you delve deeper into the discordant cases, you discover something more interesting: the images with total disagreement show atypical infiltrates—patterns that could be viral pneumonia, cardiac edema, or even old scarring. The radiologists are not incompetent; they are honest about the inherent ambiguity of medicine. Disagreement is not a flaw in the process; it's a window into the complexity your algorithm needs to understand.

This scenario illustrates an uncomfortable truth that many medical AI projects prefer to ignore: variability among expert observers is not a methodological flaw to be eliminated but a fundamental feature of clinical practice. When we force artificial agreement, when we arbitrarily choose one opinion as "the truth" and discard the others, we are not purifying our data; we are impoverishing it. We are teaching our algorithms that the world is binary and welldefined; when in reality, it is often gradual and ambiguous. However, before we can incorporate this variability constructively, we need to measure it systematically. We need to know when the raters agree because the case is clear and when they disagree because the case is inherently difficult. For this task, statistics offers us elegant tools, with the kappa coefficient being the most widely used and, when interpreted correctly, one of the most revealing.

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

10.2.1 BEYOND THE PERCENTAGE: WHY SIMPLE AGREEMENT IS MISLEADING

The most common mistake when evaluating agreement among annotators is to blindly rely on the simple agreement percentage. If two doctors agree on eighty out of a hundred cases, we conclude that they have 80% agreement and feel satisfied. However, this metric can be deeply misleading, especially in medicine, where categories are often unbalanced. Imagine a dataset for breast cancer detection in mammograms, where only 5% of the images show actual malignancy. Two annotators who simply marked all the images as “benign” agreed in 95% of the cases, achieving an apparently excellent agreement percentage without having made any real diagnosis. They have agreed by taking the easy way out, not because they share medical judgment. This phenomenon, known as chance agreement or prevalence agreement, makes the simple percentage virtually useless for evaluating the quality of a medical dataset.

The kappa coefficient, developed by psychologist Jacob Cohen in 1960, was created precisely to correct this distortion.(1) In its most intuitive form, kappa compares the observed agreement among annotators with the agreement that would be expected by chance alone, given the distribution of the categories. If the annotators agree that more than chance would explain, kappa is positive. If they agree exactly as chance would predict, the kappa is zero. If they agree that less than chance would predict (which would indicate systematic disagreement), the kappa coefficient can be negative. This adjustment for chance makes kappa a much more conservative and realistic measure than the simple percentage. A kappa of 0.70, which indicates substantial agreement, may correspond to a percentage agreement of 90% or more in unbalanced categories but only 60% in balanced categories. Interpretation requires understanding this fundamental distinction.

Conventional kappa interpretation scales, although arbitrary, provide a common language for discussing the quality of agreement. Values less than 0.20 indicate poor agreement, which is barely better than chance. Between 0.21 and 0.40, the agreement is slight, suggesting that the raters share some criteria but with considerable noise. The 0.41–0.60 range indicates moderate agreement, which is acceptable for some applications but concerning for others. Between 0.61 and 0.80, we find substantial agreement, which is generally considered adequate for most clinical purposes. Above 0.81, the agreement is nearly perfect. These thresholds are not natural laws but practical conventions, and their application must be contextual. A kappa of 0.50 may be acceptable for a population-based screening task where absolute accuracy is less critical but unacceptable for an algorithm that will guide aggressive cancer treatments. What matters is not the number itself but the conversation this number enables regarding the quality of our data.

Calculating Cohen’s kappa for two raters is conceptually simple but revealing. First, we calculate the observed proportion of agreement (Po) simply by counting the cases where both the raters agree and dividing by the total. Afterward, we calculate the proportion of agreement expected by chance (Pe), which depends on the marginal frequencies: if rater A marks 70 positives and 30 negatives and rater B marks 60 positives and 40 negatives, chance predicts that they will agree on positives in 42% of the cases (0.70 × 0.60) and on negatives in 12% (0.30 × 0.40), for a total expected agreement of 54%. Kappa is then (Po - Pe)/(1 - Pe), which normalizes the observed agreement relative to the maximum possible agreement beyond chance. This formula proportionally penalizes agreement due to chance, making kappa zero when Po = Pe and one when the agreement is perfect.

When working with more than two raters, Fleiss’s kappa ((1)) extends this logic to multiple observers. Although the calculation is more complex, the interpretation remains intuitive: it measures how much beyond chance all the raters agree simultaneously. For ordinal categories where disagreements are not all equally serious (confusing “mild” with “moderate” is less problematic than confusing “mild” with “severe”), the weighted kappa is used. This variant assigns different weights to the various types of disagreement, better reflecting clinical reality where errors have different degrees of severity. Choosing the appropriate variant is not a minor technical detail; it determines which aspects of agreement we are measuring and, therefore, what conclusions we can draw about the quality of our data.

10.2.2 WHEN KAPPA WHISPERS: DIAGNOSING THE CAUSES OF DISAGREEMENT

A low kappa coefficient is similar to a fever in a patient: it is not the disease itself but a symptom that requires further investigation. The first cause we must consider is ambiguity in the definitions of the categories. If we ask raters to classify “pneumonia” without specifying

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

whether we include atypical, active, or resolved cases, each physician will apply their own clinical interpretation. One will be restrictive, requiring clear lobar consolidation; another will be broad, including subtle interstitial infiltrates. The discrepancies do not reflect incompetence but a lack of prior consensus on what we are measuring. The solution is not to dismiss the raters but to refine the protocol: consensus meetings where experts discuss borderline cases, written operational definitions with visual examples, and explicit rules for ambiguous situations. Often, we discover that what we thought was a simple category actually encompasses subtypes that we should distinguish.

The second common cause is the inherent difficulty of certain cases. Some medical images are objectively difficult to interpret, even for senior experts. A pulmonary nodule with poorly defined borders, pneumonia overlapping with cardiac edema, and a bone fracture in a suboptimal projection generate legitimate disagreement that reflects the epistemic uncertainty of the medicine itself. In these cases, forcing agreement by choosing the opinion of the majority or the most senior expert is methodologically dangerous. We teach the algorithm that there is a single truth where in reality, there is a spectrum of valid interpretations. The most sensible approach is to explicitly document these difficult cases and exclude them from training if the ambiguity is extreme or, better yet, design the algorithm so that it can express uncertainty when faced with similar situations. A model that says “I’m not sure; this case needs human review” is more valuable than one that blindly relies on an arbitrary label.

Over time, fatigue and performance degradation constitute a third major cause, especially in high-volume projects. Medical annotation is cognitively demanding work that requires sustained attention. After fifty cases, concentration wanes; after one hundred, errors become more frequent; after two hundred, the annotator may be operating on the autopilot, applying rules of thumb rather than clinical reasoning. Disagreement patterns can reveal this phenomenon: if we calculate the kappa by time blocks, we can detect whether agreement deteriorates as the day progresses. The solutions are prosaic but effective: mandatory breaks every hour, task rotation, strict limits on the number of cases per session, and interspersed quality checks that detect deviations in real time. A well-rested annotator produces higher-quality data than an exhausted annotator does, regardless of their level of expertise.

A lack of adequate training or experience represents a fourth source of variability. Not all physicians are equally prepared for all the annotation tasks. An excellent cardiologist may not be the best annotator for neuroradiology images; a first-year resident lacks the case history needed to recognize atypical patterns. When we mix annotators of different skill levels without stratification, the kappa reflects these differences in ability rather than the intrinsic difficulty of the cases. The solution lies in carefully curating the annotation team, assigning tasks according to expertise, and providing specific training before the project begins. Less experienced annotators can be valuable if their work is properly validated, but we must be transparent about their limitations in the final dataset.

Systematic biases among annotators can manifest as low Kappas with characteristic patterns. A radiologist trained at a conservative school may tend toward overdiagnosis, indicating cases as positive that others consider doubtful. Another, perhaps scarred by litigation experience, may be excessively restrictive. These biases are not random errors but consistent trends that can be detected by analyzing confusion matrices between pairs of annotators. When we identify these patterns, we have options: we can calibrate the annotators through joint discussion of discordant cases, adjust the instructions to correct identified trends, or even consciously incorporate these differences in judgment in s as a way to increase the robustness of the dataset. An algorithm trained with annotators from different “schools” of thought may generalize better than one trained with a single but potentially idiosyncratic criterion.

10.2.3 BEYOND KAPPA: INCORPORATING VARIABILITY INTO ALGORITHM DESIGN

The most important conceptual revolution in the analysis of disagreement among annotators is the shift from viewing it as a problem to accepting it as valuable information. Traditional machine learning approaches assume that there is a single “fundamental truth” for each case and that the algorithm’s goal is to learn to predict that truth. However, in medicine, this assumption is often false. When three expert radiologists give three different diagnoses for an ambiguous image, there is no single truth that two of them are ignoring; there is a clinical reality that is genuinely uncertain. Modern multiannoter learning methods incorporate this uncertainty directly into the model, producing more honest and clinically useful algorithms.

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

An elegant approach is to model annotator confidence. Instead of treating all the labels as equally valid, we simultaneously estimate the true (latent) label and the reliability of each annotator. If a particular radiologist shows high agreement with the consensus of their colleagues in clear-cut cases, their opinion is given more weight when they disagree in difficult cases. This approach, known as learning with multiple noisy annotators, not only improves the quality of training but also provides diagnostic information about the annotators themselves. We can identify who is an expert in which types of cases, who has correctable systematic biases, and who is simply unqualified for the task. This feedback improves both the current dataset and the design of future annotation projects.

Probability distribution as a learning objective represents another powerful innovation. Instead of training the algorithm to predict a single label (pneumonia yes/no), we train it to predict the distribution of responses from expert annotators. If, for a given image, 60% of the radiologists see pneumonia, 30% see atelectasis, and 10% see normal findings, that is, the pattern the algorithm learns to reproduce. When the algorithm encounters a new image and predicts a similarly dispersed distribution, it inherently signals that the case is ambiguous. This explicit uncertainty is clinical gold: it allows the system to request human review when confidence is low or to present the clinician not just with a label but with a spectrum of possible interpretations along with their probabilities.

Stratified kappa analysis by case subgroup reveals patterns that the overall value hides. We can calculate separate Kappas for early versus late-stage disease, for different radiological presentations, or for patients from different demographic groups. We often find that agreement is high in typical cases but breaks down in atypical cases or that certain disease subtypes generate more variability than others do. This information is invaluable for algorithm design: it tells us where we can rely on labels for standard supervised training and where we need alternative strategies such as semisupervised learning, anomaly detection, or simply excluding cases from training and later labeling them via expert consensus.

Kappa confidence intervals, often ignored, are crucial for proper interpretation. In small datasets, kappa can be highly unstable; a value of 0.50 might have a 95% confidence interval ranging from 0.20 to 0.80, making any firm interpretation impossible. The calculation of these intervals, using bootstrap methods or analytical approximations, should be standard practice in any data quality report. Furthermore, comparing Kappas across different conditions (is agreement with Protocol A significantly higher than with Protocol B?) requires appropriate statistical tests that account for sample variability. Numerical precision matters when decisions about which data to use for training models that will affect human lives depend on these comparisons.

Integrating the annotator as part of the system, rather than as an external source of truth, opens up fascinating possibilities for medical artificial intelligence in the future. Imagine systems where the algorithm and the clinician collaborate iteratively: the algorithm proposes preliminary labels, the clinician corrects and comments, the algorithm learns from these corrections, and the process continues. In this paradigm, disagreement is not a failure but the engine of learning. Kappa becomes a metric of evolution: as the system improves, agreement between the algorithm's prediction and the expert's review should increase, but agreement among human experts may remain stable or even decrease if the algorithm discovers patterns that humans are overlooking. This complex dynamic requires new metrics that capture the quality of human–machine interactions, not just static agreement.

10.3 BIASES IN DATA COLLECTION: HOW TO IDENTIFY THEM

In recent years, the word "bias" has proliferated in conversations about artificial intelligence with a frequency that risks emptying it of meaning. We accuse an algorithm of being biased if it discriminates by gender, another if it performs worse on minority populations, and a third if it reproduces historical stereotypes. However, we rarely stop precisely defining what constitutes bias in the specific context of computational medicine or how it manifests in the very fabric of datasets. Here, bias is not a political opinion or a personal preference disguised as code; it is a systematic distortion, a consistent bias in the way we collect, measure, label, or select data that causes our algorithms to learn patterns that do not exist in reality or that exist only in a conveniently limited version of it. A biased algorithm is not merely imperfect; it is dangerous precisely because its harm is not random but directional, most severely affecting those who already face barriers to accessing quality healthcare.

Identifying biases must come before any discussion of performance metrics, accuracy, or

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

ROC curves. It is tempting to start by measuring how well our model performs, but that question is meaningless if we do not first know what reality we are comparing it to. An algorithm that achieves 95% accuracy on a biased dataset is not successful; it is an accident waiting to happen. That is why, before training the first neural network, we must subject our data to forensic scrutiny that examines not only what it contains but also what it has excluded, how it was collected, and what invisibly embedded assumptions it carries with it. This chapter explores the most common biases in clinical data collection, how to detect them, and why mitigating them is an ethical responsibility of both developers and clinicians.

Selection bias is the most frequent and, in many ways, the most fundamental. This occurs when the data sample used to train the algorithm is not representative of the population to which it will subsequently be applied. This mismatch is not a minor statistical error; it is a structural flaw that invalidates the generalizability of the model. Imagine an algorithm for detecting diabetic retinopathy trained exclusively on fundus images captured in university hospitals in major cities in Northern Europe. The images are high-resolution and taken with standardized equipment from patients with regular access to eye exams. When this algorithm is deployed in rural health centers in sub-Saharan Africa, where equipment is different, the prevalence of comorbidities varies, and disease presentation patterns differ; thus, its performance decreases. This is not because the algorithm is inherently flawed but because it learned to recognize European retinopathy, not human retinopathy.

Identifying selection bias requires a comprehensive demographic mapping of our data. We must systematically ask ourselves which institutions do they come from? Which geographic regions do they represent? Which age groups, genders, ethnicities, socioeconomic levels, and health insurance statuses are included? More importantly, who is missing? Gaps are more difficult to detect than presences are, but they are equally informative. If our breast cancer dataset does not include women under 40 years of age, the algorithm will never learn the atypical patterns of cancer at younger ages. If there are no patients with disabilities for whom standard imaging is difficult, the system will fail. Representation is not just a matter of social equity; it is a technical requirement for the model's robustness.

Stratified performance analysis is the most powerful diagnostic tool for detecting hidden selection bias. Even if the overall accuracy seems acceptable, we must disaggregate by relevant subgroups: Does the algorithm perform the same in men and women? In young and elderly people? In different ethnic groups? In patients with and without multiple comorbidities? We often find that the model performs well in the demographic majority but systematically fails among minorities. These disparities are not minor flaws; they are manifestations of selection bias that must be documented and, where possible, corrected through targeted sampling strategies, reweighting techniques, or additional data collection in underrepresented populations.

Exclusion bias is a particularly pernicious variant of selection bias. It occurs when certain groups are systematically excluded from the data, whether due to explicit inclusion criteria or implicit barriers to access. Traditional clinical trials, which frequently exclude pregnant women, children, frail elderly individuals, and patients with multiple comorbidities, generate datasets that do not represent the reality of clinical practice. When we train algorithms with these exclusionary data, we create systems that know how to function in "clean" patients but are dangerously ignorant of the complexity that characterizes most of those seeking medical care. Identifying this requires critically examining inclusion criteria and asking: Who is left out, and why? Is this exclusion medically justified, or is it methodological convenience that we are transferring to the algorithm?

Measurement bias arises when the method of capturing variables is inconsistent across groups or contexts, introducing artificial differences that the algorithm learns as if they were biological. Consider the seemingly trivial example of blood pressure. If in some centers in our dataset it is measured with the patient sitting, in others lying down, and in others standing; if some use cuffs of inadequate size for large arms; if the prior rest time varies between 5 and 20 minutes; then the numerical values are not directly comparable. An algorithm that learns from these inconsistent measurements may associate differences in measurement technique with differences in cardiovascular risk, producing absurd recommendations such as considering it riskier to go to certain hospitals simply because measurements are taken differently there.

Detecting this bias requires an archaeological examination of data collection protocols. We must explore the instructions given to staff, the device operating manuals, and the equipment calibration records. It is laborious but essential: a single poorly calibrated thermometer can introduce a temperature bias that affects sepsis detection in an entire subgroup of patients. Temporal variations also matter: if a laboratory analyzer was used during the first year of the study

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

and a different analyzer with varying specifications during the second year, we artificially created an “improvement” or “worsening” that the algorithm would erroneously attribute to patients or treatments.

Labeling bias, closely related to the interrater agreement challenges we explored previously, occurs when the “ground truths” in our dataset are not as accurate as we assume. If labels always come from the same radiologists, the algorithm will learn their idiosyncratic biases as if they were objective reality. If a particular pathologist tends to overdiagnose certain lesions, the model will reproduce that tendency. If labels are automatically extracted from clinical reports using natural language processing, we inherit all the errors, ambiguities, and inconsistent styles of those who wrote those reports. Identification requires an audit of the labeling process: Who did the labeling? What was their training? Under what time pressure? Are there independent reference standards against which to validate?

Confounding bias represents perhaps the most subtle and dangerous form of distortion. It occurs when the algorithm learns spurious associations—correlations that do not reflect causal or clinically relevant relationships. The classic example, documented in the literature, is the pneumonia detection algorithm that learned to identify the disease not by lung patterns but by the presence of chest drains. At the training hospital, patients with severe pneumonia frequently receive chest tubes; the algorithm, which is efficient in its correlational learning, simply learned that a chest tube implies pneumonia. When applied in centers where chest tubes were less common, its actual detection ability plummeted, even though it maintained high “accuracy” in detecting plastic tubes.

Identifying these confounders requires interpretable feature engineering and in-depth clinical knowledge. We must examine what the algorithm is actually seeing: is it focusing on relevant anatomy or on technical artifacts? On pathological findings or on process markers (such as the time of day of admission, which may correlate with severity in triage systems)? Explainability techniques such as SHAP or LIME can reveal these spurious dependencies, showing that the model uses proxy variables instead of those that truly matter. Prevention requires careful design of the datasets: intentional variability in procedures across centers so that the algorithm cannot learn to rely on specific local practices.

Survival bias is particularly insidious in longitudinal studies where time plays a central role. This occurred when we included only those patients who survived to a certain point in follow-up, systematically ignoring those who died or were lost to follow-up earlier. A risk prediction algorithm for heart failure trained on data from patients who completed one year of follow-up was used. Those who died in the first few months—precisely those at highest risk—are absent from the dataset. The algorithm learns “low-risk” patterns that are actually “short-term survival”, and when applied to new patients, it systematically underestimates the danger of severe cases.

Detection requires analysis of loss rates during follow-up and comparison of characteristics between those who complete the study and those who are lost to follow-up. If patients who are older, have poorer renal function, or have lower treatment adherence disappear from the dataset at higher rates, we introduce a dynamic selection bias that distorts the entire risk distribution. Statistical methods for survival analysis, with their explicit handling of censoring, must be applied not only to the clinical endpoint but also to continued participation in the study itself. Furthermore, we must be transparent about these limitations: an algorithm trained on one-year-old survivors should not be applied to palliative care decisions for terminally ill patients.

Recall bias, also known as outcome-knowledge bias, particularly affects data collected retrospectively. When a researcher reviews medical records and understands the patient’s outcome, they unconsciously look more intents for certain factors in the cases than in the controls. A physician who extracted data from patients who developed sepsis may have recorded previous febrile episodes more diligently than patients who did not develop it did, not because they were absent from the latter group but because the physician did not look for them with the same attention. This bias is especially dangerous because it resides in the mind of the data collector, not in the data itself, making it virtually impossible to detect through subsequent statistical analysis.

The only effective defense is a prospective design with standardized protocols that are blinded to the outcome. When data are collected before the result is known and when staff follow strict checklists without interpretive flexibility, we eliminate the opportunity for memory to distort data collection. For unavoidable retrospective data, we must apply sensitivity adjustments: we assume that certain findings may be underreported in controls or use multiple independent reviewers who are blinded to one another. Transparency regarding the retrospective origin of the data is mandatory; presenting it as if it were prospective constitutes scientific misconduct.

Publication bias, while not directly affecting a specific dataset, distorts the knowledge

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

ecosystem upon which we build our projects. Studies with positive results, high-performing algorithms, and successful innovations are published more frequently and in higher-impact journals than failures and implementations that did not work. When we review the literature to decide which model architecture to use or which features to include, we are looking at a sample biased toward success. Algorithms that "did not work" in certain contexts, limitations discovered too late, and biases identified postimplementation remain in researchers' drawers or in the proceedings of minor conferences.

This distortion of available knowledge makes us unfoundedly optimistic. We believe that certain approaches are more robust than they actually are because we do not see the cases where they failed. Awareness of this bias should foster an attitude of healthy skepticism: valuing negative studies, actively seeking out reports of failure, and assuming that any published algorithm likely performed worse in unreported contexts. For developers, this means not blindly trusting high-impact papers but contacting authors for unpublished data, independently replicating studies, and being transparent ourselves about the limitations of our own systems.

10.3.1 TOWARD A PRACTICE OF CONSCIOUS BIAS COLLECTION

Identifying biases is not a one-time task that we complete before training; it is a continuous process that accompanies the entire algorithm lifecycle. Before we collect data, we must actively map out which populations might be excluded and why to design proactive inclusion strategies. During data collection, we monitor representativeness in real time, adjusting the sampling if we detect deviations. After training, we evaluate not only overall performance but also equity across subgroups, explicitly documenting where the algorithm fails. This continuous monitoring requires infrastructure: data pipelines that enable automated stratified analysis, equity dashboards that alert us to emerging disparities, and protocols for periodic auditing even after deployment.

For clinicians who participate in artificial intelligence projects, bias detection is perhaps their most valuable contribution. Their knowledge of real-world practice allows them to identify situations where data may not be representative: they know that Friday afternoon emergency room patients differ from Monday morning patients, that summer images show different patterns than winter images do and that certain groups avoid going to certain centers because of linguistic or cultural barriers. They can anticipate who will be harmed by a biased algorithm and advocate for their inclusion in evaluations. A clinician who points out a potential bias is not hindering progress; they are exercising their professional responsibility to protect patients.

Mitigating identified biases requires a diverse technical arsenal. Targeted stratified sampling can correct underrepresentation. However, no single technique is a panacea; each assumes that we know what biases exist, and all have trade-offs that must be evaluated.

Epistemic humility must be our compass. Recognizing that our data are always imperfect and that our algorithms will always be imperfect is not an excuse for complacency but a starting point for continuous improvement. Every identified bias is a learning opportunity. Every failure in specific subgroups is a call to action. The question is not whether our system is biased—it is—but whether we know its biases, whether we have documented them, whether we have implemented safeguards, and whether we are committed to reducing them iteratively. In that context, honesty about our limitations lies in the only solid foundation for building trustworthy medical artificial intelligence.

10.4 SAMPLE SIZE AND STATISTICAL POWER

> "The question is not how much data you can accumulate, but how much you need to learn without memorizing, to generalize without ghosts."

It is one of the most frequently asked questions at any initial medical AI project meeting and, curiously, one of the worst answered. The head of the cardiology department wanted to know how many ultrasounds his team must review. The director of the data unit calculates budgets on the basis of annotation hours. The principal investigator drafts ethical protocols with figures that later prove impossible to meet. All too often, the answer comes from intuition, availability ("we have access to five thousand historical cases"), or the simplistic mantra of "the more, the better." However, this approach is dangerously naive. An inadequate sample size—whether too large or too small—can ruin a project in ways that become evident only when it is too late: algorithms that memorize noise instead of patterns, that fail spectacularly in new patients, or that consume

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

scarce resources on unnecessary data collection. Understanding the fundamentals of sample size and statistical power is not a methodological luxury; it is a survival requirement for any serious project.

The complexity of this issue goes beyond simple formulas because it depends on multiple factors that interact in nonlinear ways. There are no universal magic numbers or even foolproof rules of thumb. What does exist are sound statistical principles, estimation methods adapted to the era of machine learning, and above all, the need to think critically about what we want our algorithm to learn and with what degree of confidence. This chapter explores these fundamentals, from classic empirical rules to modern sample efficiency techniques, with the aim of providing a framework for making informed decisions about how much data we actually need.

10.4.1 THE FOUR PILLARS THAT DETERMINE HOW MUCH IS ENOUGH

The first pillar is the intrinsic complexity of the problem we are trying to solve. Not all medical artificial intelligence tasks are equally dataintensive. A simple binary classification between normal chest images and those with evident lobar consolidation, where visual differences are dramatic, may require relatively few examples for an algorithm to learn the distinctive patterns. In contrast, predicting the response to a specific cancer treatment on the basis of hundreds of genomic, clinical, and immunohistochemical variables—where signals are subtle and nonlinear interactions abound—requires substantially larger volumes of data. The classic rule of thumb of ten events per predictor variable, originating from logistic regression statistics, provides a conservative starting point: if we include twenty variables in our model, we need at least two hundred events of the outcome we are predicting. However, this rule is a floor, not a ceiling, and for complex models, it may be insufficient.

The second pillar, closely related but distinct, is the prevalence of the event of interest. In medicine, we frequently encounter rare outcomes: serious complications occurring in 2% of cases and autoimmune diseases with an incidence of one in a thousand specific genetic mutations present in minute fractions of the population. A dataset of ten thousand patients may seem impressive, but if it contains only one hundred positive cases of the condition we want to detect, its usefulness for training is limited. Algorithms, such as medical students, need to see enough examples of the pathology to learn to recognize it. One hundred cases scattered among ten thousand controls may not provide the variety of presentations, degrees of severity, and subtypes that characterize the disease. The rule of ten events per variable then becomes a difficult goal: with one hundred positive cases, we can only include ten variables with reasonable statistical confidence, which may be insufficient to capture the complexity of the phenomenon.

Population heterogeneity constitutes the third determining pillar. A homogeneous dataset, where all patients are of similar ages, come from the same center, share limited comorbidities, and were treated with standardized protocols, may require fewer cases to characterize its patterns. However, these savings in sample size come at the cost of poor generalizability: the algorithm learns medicine from that specific context, not universal medicine. If our goal is to develop tools applicable to diverse populations, we need data that reflect that diversity, which multiplies the required sample size. An algorithm that works equally well in young adults and frail elderly individuals, in patients with and without renal failure, and across different ethnic groups with relevant genetic variations needs sufficient examples in each subgroup to learn their particularities. Stratifying sample size by relevant subpopulations is an essential but frequently overlooked practice.

The fourth pillar is the complexity of the algorithm we intend to train. Simple models, such as logistic regression with few variables or shallow decision trees, have relatively few parameters to tune and therefore require less data to estimate them reasonably. Deep neural networks, especially convolutional architectures for medical images or transformers for clinical text, can have millions of parameters. Each additional parameter is a degree of freedom that the model can use to adapt to the training data, including its specific noise. Training these models with small datasets invites overfitting: the algorithm memorizes the examples it has seen, including their idiosyncratic peculiarities, but fails to extract the generalizable patterns that would allow it to perform on new data. The paradox is that more powerful models, theoretically capable of capturing more complex relationships, are also more data-hungry and more prone to failure when that data is scarce.

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

10.4.2 STATISTICAL POWER: THE PROBABILITY OF NOT BEING MISLED

The concept of statistical power, although originating in the design of traditional clinical trials, extends remarkably well to the field of medical machine learning. In its classical formulation, power is the probability of detecting an effect of a certain magnitude if that effect actually exists, thereby avoiding Type II errors (false negatives). In the context of algorithm development, we can reinterpret power as our study's ability to distinguish a genuinely useful algorithm from one that has merely been lucky with the available data. A study with low power has a high probability of producing misleading conclusions: it may lead us to believe that an algorithm works when its good results are due to random fluctuation, or it may fail to detect that a promising algorithm actually has merit.

Power depends on four factors that we must explicitly consider: the size of the effect we expect to detect, the inherent variability of the data, the sample size, and the significance level we accept. In medicine, large and consistent effects are rare; most useful interventions have modest effects on outcomes that vary widely among individuals. This leads to sample size requirements that are larger than we would intuitively expect. An algorithm that reduces mortality by 20% in relative terms—from 10% to 8%—is clinically important, but detecting this difference with confidence requires hundreds or thousands of patients, not dozens.

For the developer of clinical algorithms, power analysis is not a preproject academic exercise but a tool for ongoing planning. Before we begin, we must estimate the sample size needed to detect a clinically relevant improvement over standard practice. During development, we must monitor whether we are achieving the necessary power for our comparisons. Finally, we must report the power achieved so that others can evaluate the robustness of our conclusions. Ignoring power is similar to building on invisible foundations: it may support the building, or it may collapse under the first independent test.

10.4.3 PRACTICAL METHODS FOR DETERMINING THE RIGHT SAMPLE SIZE

Learning curves are the most intuitive and visual empirical method for assessing whether we have enough data. The logic is elegant in its simplicity: we train the algorithm on increasing subsets of our data—first 10%, then 20%, 30%, and so forth—and plot how the validation performance evolves. Initially, with little data, the performance is low and variable: the algorithm has not seen enough examples to learn robust patterns. As we increase the sample size, the performance improves, and the variance between runs decreases. At some point, the curve stabilizes: adding more data yields minimal marginal improvements. This saturation point indicates that we have reached the effective sample size for our given problem and model.

Interpreting these curves requires caution. If the curve is still rising steeply when we exhaust our data, we need more. If it has stabilized but is at an insufficient performance level, the problem may be data quality, model complexity, or the inherent difficulty of the problem. If there is a large gap between training and validation performance, we are facing overfitting: the model has the ability to memorize but not to generalize, suggesting that we need more data or a simpler model. Learning curves are powerful diagnostic tools, but they require that we already have enough data to perform subsampling, which makes them more useful for evaluating whether we need more data than they need for initial planning.

Simulation-based power analysis offers an alternative when data are not yet available. In this approach, we generate synthetic data with known characteristics—prevalence, variable effects, and noise levels—that simulate our real-world medical problem. We train the algorithm with different simulated sample sizes and observe at what size we correctly recover the patterns we know exist. This method is computationally intensive and critically depends on our simulations accurately capturing reality, but it provides estimates tailored specifically to our algorithm and problem. It is especially valuable for novel architectures where traditional sample size formulas do not apply.

Stratified cross-validation is an essential tool not only for evaluating performance but also for determining the stability of our results with respect to sample size. If we observe high variability in performance across different validation folds, it is a sign that our sample is small and that the results are unstable. A robust model should behave consistently regardless of which specific subset of data is set aside for validation. Excessive variability suggests that we are in the data-scarce regime, where sampling noise dominates the true signal. In this regime, increasing the sample size will likely improve both the average performance and the reliability of our estimates.

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

For binary classification problems, sample size formulas that combine the factors of prevalence, acceptable margin of error, and desired confidence level exist. The use of the Fleiss formula for proportions, adapted to the context of algorithm validation, allows us to calculate how many positive cases we need to estimate sensitivity with a certain degree of precision and how many negative cases for specificity. For example, to estimate a sensitivity of 90% with a 5% margin of error and 95% confidence, we need approximately 138 positive cases. If the prevalence is 10%, this implies a total dataset of approximately 1,380 patients. These formulas provide concrete figures for planning, although they must be adjusted for the complexity of the model and the need for independent validation.

10.4.4 EFFICIENCY AND WISDOM: WHEN MORE IS NOT ALWAYS BETTER

The future of sample size in clinical artificial intelligence does not rely solely on the accumulation of more data but rather on the use of existing data more intelligently. Transfer learning represents a revolution in sample efficiency: instead of training a model from scratch with data specific to our problem, we start with a model pretrained on related tasks—such as general medical image recognition or clinical language processing—and adapt it to our specific task with relatively few examples. A vision model pretrained on millions of natural images can learn to detect pneumonia in X-rays with just hundreds of medical examples because it has already learned fundamental visual representations that need only fine-tuning. This transfer of knowledge across domains multiplies the value of every specific piece of data we collect.

Data augmentation techniques, applied with clinical judgment, can artificially expand our dataset by generating plausible variations in existing examples. In medical images, rotations, translations, scalings, and contrast adjustments within physiologically reasonable ranges can create “new” examples that train the model to be invariant to these transformations. In time series, small Gaussian perturbations that preserve the temporal structure can increase robustness. However, these techniques have limitations: they cannot create information that does not exist, and if applied without discretion, they can generate medically impossible examples that confuse the algorithm. Augmentation must be conservative and clinically validated, not mechanical.

Adaptive data collection designs allow us to adjust the sample size in real time as we learn. Instead of setting a number a priori that may be excessive or insufficient, we collect data in stages, continuously evaluating the achieved accuracy and the stability of the estimates. When the confidence intervals of our performance metrics narrow sufficiently, we can stop data collection, saving resources. If the accuracy is insufficient, we continue. This approach, inspired by adaptive clinical trials, requires a logistical infrastructure for continuous data collection and interim analysis but optimizes the balance between information and cost.

For clinicians involved in artificial intelligence projects, understanding these fundamentals empowers them to engage in realistic negotiations. When a team of engineers promises to develop an algorithm using 100 cases in three months, the informed clinician can point out that the statistical power will be insufficient to detect clinically relevant effects or that cross-validation will reveal unacceptable variability. They can advocate for adequate resources for data collection, which typically consumes 70% or more of the budget and time of a serious project. They can critically evaluate published results: an algorithm with an area under the curve of 0.95 developed using fifty patients should be met with professional skepticism, not naive admiration.

Quality can never be completely replaced by quantity, but an insufficient quantity invalidates quality. An algorithm trained on a small amount of data, no matter how carefully annotated, is like a doctor who has seen only ten patients in his life: he may have excellent theory, but his clinical judgment will lack the empirical basis necessary for confidence. The crucial difference is that the doctor, aware of their limited experience, will be cautious, while the algorithm, lacking self-awareness, will issue predictions with the same confidence as if it had been trained on millions of cases. Building systems that estimate and communicate their own uncertainty—that know when they are operating outside their statistical comfort zone—is among the most important ethical challenges of modern medical artificial intelligence.

If we have learned to properly scale our datasets and validate their internal quality, how do we ensure that our model maintains its performance when it leaves the laboratory and encounters real patients in different hospitals, with different equipment, and with different clinical practices?

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

REFERENCES

McHugh ML. Interrater reliability: the kappa statistic. Biochem Medica [Internet]. 2012 [cited May 14, 2026];22(3):276. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC3900052/

https://doi.org/10.62486/978-9915-9876-8-2.Ch10

Chapter 11 / Capítulo 11

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch11
Pages: 183-197

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 11. Model Development and Validation

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "A model that works on the data it was trained on is not a model that works; it is a model that has proven it can *remember*. The true test of its worth lies not in the past, but in the future: in its ability to correctly predict cases it has never seen before, in patients who were not in its development sample, in hospitals where it has never been tested."

The history of predictive model development is full of examples of algorithms that reported excellent metrics in papers but failed spectacularly in real-world practice. This is not because the researchers were dishonest but because they confused two radically different concepts: performance on training data and performance in the real world. A model can achieve 99% accuracy on the dataset where it was developed and is completely useless when faced with new patients. This gap between the lab and reality is among the greatest challenges of artificial intelligence in medicine, and it can be addressed only through rigorous validation.

Model validation is not a formality carried out at the end of the project but a continuous process that must accompany the entire development. It begins by dividing the data into training and test sets, continues with cross-validation to optimize hyperparameters, and culminates in external validation on independent populations. Each of these steps answers a different question: Can the model learn patterns from the data? Is its performance stable? Does it generalize to new contexts? Ignoring any of these questions is similar to building on sand.

In this chapter, we explore the different levels of validation for the predictive models. We will begin with the fundamental distinction between internal and external validation and why the former cannot replace the latter. We will continue with the concept of dataset shift, one of the main causes of failure in model deployment. We address the problem of generalization with real-world examples of algorithms that worked in one hospital and failed in another. Finally, we introduce clinical MLOps, the set of practices that enable the maintenance of models in production, monitoring their performance, and updating them when necessary. This is not an exhaustive technical guide but rather a reflection on what it truly means for a model to be reliable.

11.1 INTERNAL VS. EXTERNAL VALIDATION

A model that shines in your lab but fades in the real world is not artificial intelligence; it is a statistical mirage.

There is a silent hierarchy in the validation of predictive models that separates promise from reality, academic research from truly useful medicine. At its base lies internal validation, which is seemingly robust but potentially misleading. At its peak, almost inaccessible to many projects, stands for external validation, the only guarantee that an algorithm deserves clinical trust. Between the two, intermediate steps that few manage to complete. This pyramid is not a methodological whim; it is the accumulation of decades of bitter experience where brilliant algorithms failed spectacularly upon hitting the ground in real hospitals. For the developer who aspires to build tools that truly help patients, understanding this hierarchy is as fundamental as knowing neural network architectures. For clinicians evaluating whether to trust a new digital tool, demanding adherence is an exercise in professional responsibility.

The distinction between internal and external validation transcends statistical techniques to delve into the epistemology of computational medicine. What does it truly mean for an algorithm to "work"? That it correctly predicts on data from the same hospital, same equipment, same period, and same physicians who generated its training data? Or that it maintains its utility when it crosses institutional, temporal, and population boundaries? The answer determines whether we are building toy systems for academic publications or trustworthy medical infrastructure. This chapter explores each level of the validation pyramid, its promises, its limitations, and why skipping steps mean taking risks that can result in harm to patients.

11.1.1 INTERNAL VALIDATION: WHEN DATA LIES KINDLY

Internal validation constitutes the first mandatory filter in the development of any predictive model. Its logic is seemingly simple: we divide the available data into two portions, one to train the model and the other to evaluate it. If the model predicts well on data it did not see during training, we have demonstrated that it has not simply memorized the examples but has extracted generalizable patterns. However, this "generalization" is profoundly limited: it only guarantees that it works on data drawn from the same distribution as the training data. It is like proving that

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

a student has learned the lesson, not that they will be able to solve new problems in different situations.

Internal validation techniques have evolved to become more sophisticated and less naive. The simple training–test split, where we randomly set aside 20–30% of the data for evaluation, is the bare minimum. k-fold cross-validation increases rigor: we divide the data into k subsets, train them on k-1, and evaluate them on the remaining subset; this process is repeated k times so that each subset serves as a test set once. This provides more stable performance estimates and allows us to use all the data for training without sacrificing evaluation. Variants such as stratified cross-validation preserve the class proportions in each fold, which is crucial when outcomes are rare. Nested cross-validation adds an additional level for hyperparameter selection, preventing us from optimizing the model by indirectly using the test data.

However, all these sophisticated techniques share a fatal limitation: all the data come from the same universe. If we train on X-rays from the Central University Hospital and evaluate them using X-rays from the same hospital, which are taken with the same equipment and interpreted by the same radiologists, during the same period, we can measure the model's ability to adapt to the peculiarities of that specific context. It may have learned that a certain image contrast indicates pneumonia, when in reality it indicates that the technician at that hospital adjusts the equipment in a certain way. Internal validation, however technically rigorous it may be, is blind to these contextual biases.

Temporal validation represents an intermediate step that introduces an additional dimension of reality. It consists of training the model with data from a historical period and evaluating it with data from a later period. For example, patients were trained from 2015–2019, and patients were tested from 2020–2021. This approach is more demanding because it introduces the possibility of temporal changes, such as the evolution of clinical practice, new diagnostic protocols, changes in the patient population, or even disruptive events such as a pandemic. If the model maintains its performance over time, we gain confidence in its robustness. However, it remains valid within the same center, with the same teams and the same institutional biases. A model that survives temporal validation may still completely fail when transferred to another hospital.

11.1.2 EXTERNAL VALIDATION: CROSSING BOUNDARIES THAT MATTER

External validation is the gold standard for separating research toys from serious clinical tools. It involves evaluating the model on data from completely different centers, different institutions, different patient populations, different imaging equipment, different clinical protocols, and different medical practices. If the model maintains its performance in this foreign territory, we have genuine evidence that it has learned universal patterns of the disease, not local artifacts of the training context. External validation is the only way to prove that an algorithm generalizes in the medically relevant sense: that it can travel and remain useful.

The difference between performance in internal and external validation can be dramatic and devastating. The medical literature documents cases in which the areas under the curve of 0.90 or higher during development decreased to 0.60 or lower in the external validation. This drop is not an anomaly; it is the norm when models capture patterns specific to the training center. A COVID-19 detection algorithm for chest X-rays, trained at a hospital where critically ill patients were positioned a certain way on the stretcher, learned to detect the patient's orientation rather than viral pneumonia. It worked perfectly at that hospital but failed completely at others. Only external validation would have revealed this fragility before a potentially dangerous deployment.

Planning for external validation must begin before the first data point is collected. Ideally, a project sets aside a portion of the data for external validation—sourced from different centers—from the very start and keeps that "sanctuary" untouched until the end of development. This discipline avoids the temptation to "tap into" that data when performance in internal validation is mediocre or to adjust the model so that it performs better on the test dataset. It also requires establishing formal collaborations with other centers from the project's inception, not as an afterthought when reviewers demand external validation to accept the publication.

Geographically external validation, where the model is tested in different regions, adds another layer of rigor. An algorithm trained on data from Europe or North America may encounter populations with different disease prevalence rates, different age distributions, different ethnic groups with relevant genetic variations, and different lifestyles that alter clinical presentation. If the model survives, external validation across multiple continents provides evidence of robust generalizability. If it fails, it reveals important limitations that must be documented: the algorithm

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

may be useful in certain contexts but not in others—valuable information for its selective implementation.

Prospective external validation, in which the model is applied to new patients in real time and its predictions are compared with actual outcomes, represents the highest level of evidence. Unlike retrospective validation using historical data from other centers, prospective validation eliminates the risk that the model may have inadvertently been biased by characteristics of the validation data. It is the equivalent of a clinical trial in evidence-based medicine: the definitive test of utility. However, this approach is also the most expensive and time-consuming and requires clinical integration infrastructure, specific ethical approval, and patience to accumulate enough cases. Not all projects can reach this level, but all should aim for some form of external validation before considering their model ready for clinical use.

11.1.3 FROM THE PYRAMID TO PRACTICE: IMPLEMENTING RIGOROUS VALIDATION

For the developer of clinical algorithms, understanding this hierarchy transforms project planning from its very foundations. It is not enough to construct a brilliant model on local data; one must deliberately design the validation architecture. This includes decisions about which portion of the data to set aside for which level of validation, which centers to collaborate with, and how to ensure that the external data are truly independent (not just formally from another institution but collected by the same researchers using the same protocols). It also includes comprehensive documentation of the training characteristics: age, sex, comorbidities, equipment used, time period, and disease prevalence. This documentation allows others to assess whether their population is comparable and to expect similar performance or whether there are differences that suggest caution.

Transparency in validation reporting is an ethical and scientific imperative. Too many papers publish impressive internal validation metrics without clearly warning of their limitations, or mention “validation in an independent cohort” when it comes from the same hospital just six months later. The TRIPOD (Transparent Reporting of a Multivariate Prediction Model for Individual Prognosis or Diagnosis) guidelines provide frameworks for properly reporting the development and validation of predictive models, and their TRIPOD-AI extension adapts these recommendations to artificial intelligence. Following these guidelines is not bureaucracy; it is a service to the scientific community and to future patients who will be affected by decisions based on these models.

For evaluating clinicians, this hierarchy provides a framework for systematic scrutiny. When faced with any algorithm, the following questions should be answered: What level of validation has it achieved? Is it only internal? Temporal? External—across how many centers and how similar are those centers to my practice? The performance metrics must be interpreted in light of this validation: an AUC of 0.85 with multicenter external validation is more impressive than an AUC of 0.95 with only internal validation. Even with robust external validation, clinicians must remain vigilant: they should implement performance monitoring in their own practice, comparing model predictions with actual outcomes, because there can always be unanticipated differences between their context and the validation settings.

The relationship between validation levels is not one of substitution but of necessary complementarity. Internal validation is indispensable for iterative development: fine-tuning architectures, selecting features, and optimizing hyperparameters. Without it, we cannot construct models. External validation is indispensable for implementation: to demonstrate that the model serves a purpose beyond publishing articles. Without it, we should not use models. Both are part of a continuum that should be standard in serious clinical artificial intelligence. The fact that this is unfortunately not yet the case reflects the immaturity of the field, where the pressure to publish quickly and the lack of rigorous regulation allow insufficiently validated models to influence clinical decisions.

The future of validation lies in standardization, transparency, and continuity. Public model repositories, such as ModelHub or specific medical image repositories, allow researchers worldwide to evaluate published models on their own data and accumulate evidence of postpublication external validation. Continuous monitoring platforms in production, which track the performance of deployed models and alert when deviations (dataset shift) are detected, extend validation to real time. Collaborative validation networks, where multiple centers agree on common protocols to evaluate candidate models, distribute the workload and increase the generalizability of the evidence generated.

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

However, no infrastructure, guidelines, or platform can replace critical human judgment. A model validated externally in ten hospitals may fail in the eleventh if there is some unaccounted-for distinguishing feature: an unusual coding practice, an immigrant population with different genetics, and a recent change in equipment. Validation is not a one-time event but a continuous process. Every new patient, every new prediction, is an opportunity to assess whether the model remains reliable. In addition, when it ceases to be so, we must be prepared to update or withdraw it. Medicine based on artificial intelligence is no different from traditional evidence-based medicine: both require intellectual humility, healthy skepticism, and a willingness to change when the facts demand it. The difference is that algorithms, unlike drugs, can degrade silently as the world around them changes, without anyone noticing until the damage has already been done.

11.2 DATASET SHIFT: WHY YOUR MODEL FAILS AT THE NEXT HOSPITAL

> "A model trained in the past is a guest who has not been invited to the present; sooner or later, the present catches on."

A scene that has become painfully familiar in clinical informatics departments around the world. A team of researchers published a sepsis prediction algorithm with an AUC of 0.94 in a high-impact journal, which was validated at three teaching hospitals. The results generate excitement in the medical community. A community hospital, impressed by the evidence, decides to implement the system. In the first few months, the technical integration runs smoothly: alerts appear on the monitors, the doctors review them, and the workflow adapts. However, after six months, a nurse in the intensive care unit commented at a patient safety meeting that "that the sepsis algorithm is acting strange." Retrospective records confirm her suspicions: patients labeled as high-risk who progress without complications and low-risk patients who develop septic shock. The performance, which is now measured against real external validation criteria, has decreased from the published 0.94 to a barely acceptable 0.71. The model has not changed; the world around it has. This phenomenon, which accounts for more failures in the implementation of medical artificial intelligence than any other, has a technical name: dataset shift.

Data shift, or distribution shift, is the statistical manifestation of an uncomfortable truth that we developers prefer to ignore: predictive models are not universal artifacts such as a physical equation but historical creatures anchored in the time and place of their creation. When the statistical properties of the data change between training and deployment, the model becomes a stranger speaking an obsolete language, interpreting signals that no longer mean what they once did, and ignoring new signals it never learned to recognize. Understanding the mechanisms of dataset shift—its types, causes, and above all, warning signs—is an essential skill for any team aspiring to build medical systems that work not only in the laboratory but also in patients' real lives.

11.2.1 THE THREE FACES OF CHANGE: COVARIATE, PRIOR, AND CONCEPT SHIFT

Covariate shift is perhaps the most intuitive type of dataset shift. It occurs when the distribution of the input variables changes, but the fundamental relationship between those variables and the outcome they predict remains constant. An image of a sepsis detection model trained on hospitalized adults, where a heart rate of 120 beats per minute is considered significant tachycardia, is used. When the same model is applied to a pediatric unit, many healthy children have rates of 130–140, simply because of their different physiologies. The model, which applies the thresholds learned from adults, generates massive false alarms. The relationship between tachycardia and sepsis remains valid (septic children also experience increased heart rates), but what constitutes "tachycardia" has changed. The model has not learned this contextual invariance; it has learned absolute numbers that no longer apply.

This type of shift is particularly insidious because the model may continue to have seemingly reasonable metrics in terms of globally calculated sensitivity and specificity while systematically failing in specific subgroups. A cardiovascular risk prediction model trained on a population with a high incidence of obesity may underestimate risk in lean populations because it has learned associations between variables that depend on metabolic context. Detection requires stratified analysis: comparing variable distributions between training and deployment and assessing whether performance remains consistent across different value ranges.

The prior probability shift, or change in prevalence, represents the second fundamental type.

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

This occurs when the proportion of positive and negative cases changes between the training and application contexts, drastically altering the clinical significance of the predictions. A model trained to detect breast cancer in population-based screening, where the prevalence is 0.5%, may perform brilliantly in that context, with modest but acceptable positive predictive values. If the same model is applied in a high-risk clinic where the prevalence is 15%, without recalibration, it will generate an avalanche of false positives that will overwhelm the diagnostic system. The technical sensitivity and specificity may remain intact, but the positive pred e value will plummet, making each alert from the model statistically less credible.

This phenomenon explains why screening models rarely work directly in high-risk settings without adjustments. Probability recalibration, using techniques such as Platt scaling or isotonic regression on local data, can restore the model's utility. However, this recalibration requires labeled data from the new setting, which can be difficult to obtain quickly. Proactive planning includes designing models that report not only binary predictions but also well-calibrated probabilities and establishing automatic or semiautomatic recalibration procedures as part of deployment.

This conceptual shift is the most profound and dangerous of the three. It occurs when the relationship between predictors and the outcome itself changes, fundamentally invalidating the model's logic. The COVID-19 pandemic provided dramatic examples: pneumonia mortality prediction models developed in 2019 failed spectacularly in 2020,(1) not only because the virus was new but also because the relationships amongage, comorbidities, and outcomes differed. Treatments have evolved rapidly, changing the prognosis for similar conditions. ICU admission criteria changed, altering the severity of patients across different levels of care. The medical world changed more in months than it did in years, and models trained in the past became not only inaccurate but also potentially dangerous if used to guide decisions.

Detecting a concept shift is particularly challenging because it is not revealed simply by comparing variable distributions. It requires monitoring of actual predictive performance: continuously comparing predictions with observed outcomes. When the model systematically overestimates or underestimates risk in specific groups, when the calibration curves persistently deviate from the diagonal and when the performance metrics degrade monotonically over time, we are faced with evidence of a concept shift. The answer is not recalibration but rather retraining with new data or, in extreme cases, removing the model until it can be rebuilt with updated logic.

11.2.2 THE ORIGINS OF CHANGE: WHY THE WORLD REFUSES TO STAY THE SAME

The causes of dataset shift are multiple, intertwined, and often invisible until the damage has already occurred. Demographic differences between populations are the most obvious but not the only ones. Age, sex, ethnicity, socioeconomic status, and access to prior care all modify disease presentation and the distribution of normal values. A model trained on an insured urban population may fail in an uninsured rural population, not because of a technical flaw but because the epidemiology is different, the stages of presentation are different, and the untreated comorbidities are different.

Differences in clinical practice operate at subtle but crucial levels. Referral criteria determine who arrives at the hospital: A facility with strict triage in primary care will see more carefully selected patients than one with direct access. Treatment protocols alter outcomes: hospitals with an aggressive anticoagulation policy have different rates of embolism than those with a conservative policy do, altering the relationship between risk factors and events. Thresholds for performing diagnostic tests introduce detection biases: if troponin tests are performed only on patients with typical chest pain, the model learns that atypical chest pain implies low risk—a relationship that may not hold where broader screening is practiced.

Technical differences in data acquisition are particularly relevant for imaging and signal models. Equipment from different manufacturers, of different generations, or with different calibrations produces images that, although visually similar to the human eye, have distinct statistical characteristics that algorithms capture and erroneously associate with pathology. A computer vision model trained on images from a specific manufacturer may "learn" artifacts from that equipment as signs of disease, failing catastrophically with images from another manufacturer where those artifacts are absent or different.

Temporal changes affect even models that remain in the same hospital where they were trained. Clinical guidelines evolve, modifying diagnostic and treatment practices. New drugs alter the natural history of diseases. Epidemics and pandemics alter the prevalence and presentation of conditions. Changes in information systems, seemingly minor, can alter how data are recorded.

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

A model is, at best, a snapshot of a reality in motion; at worst, a snapshot of a reality that no longer exists.

For the developer, the dataset shift is not a remote contingency but a certainty that must be accounted for from the design phase. No model will remain indefinitely without degradation. The question is not whether it will happen but when, in what form, and how to detect it before it causes harm. This requires a continuous monitoring infrastructure, a comparison of the distributions of the input variables between training and production, an evaluation of the predictive performance in real time, and the establishment of alert thresholds for degradation. It also requires exhaustive documentation of training characteristics—not just size and metrics but also demographic distributions, clinical protocols, equipment used, and time period—everything that allows others to assess transferability.

11.2.3 DETECTING THE SHIFT: SURVEILLANCE SYSTEMS FOR MODELS IN PRODUCTION

Early detection of dataset shifts requires a monitoring architecture that goes beyond global performance metrics. Comparisons of distributions of predictor variables between the training dataset and current data in production using statistics such as the Kullback–Leibler divergence or Kolmogorov–Smirnov tests can reveal covariate shift before it degrades performance. Monitoring probability calibration, using monthly updated calibration plots, is used to detect prior probability shifts. A performance analysis stratified by relevant subgroups (age, sex, hospital unit, and shift) revealed degradations masked by stable overall averages.

Anomaly detection techniques in the feature space can identify cases that the model never encountered during training, signaling potential concept shift. When a patient presents a combination of variables that falls outside the training distribution, the model is operating in an uncharted territory, and its predictions should be treated with extreme caution or automatically rejected. These “prediction rejection” systems are essential safeguards in high-risk applications.

Prospective continuous validation, where a fraction of the model’s predictions are systematically compared with actual outcomes, provides definitive evidence of sustained performance. Unlike variable monitoring, which is an indirect proxy, continuous validation directly measures what matters: is the model still predicting correctly? This practice, which is standard in the tech industry for recommendation and systems, is still in its infancy in medicine but should be mandatory for models that guide important clinical decisions.

11.2.4 ADAPTING TO CHANGE: MITIGATION AND SURVIVAL STRATEGIES

Mitigating dataset shifts requires a technical and organizational arsenal. Periodically updating the model—retraining it with recent data that incorporate the changes that have occurred—is the most direct strategy. However, this update is not trivial: it requires new quality labels, validation of the updated model’s performance, and regulatory processes if the model is a certified medical device. The frequency of updates depends on the rate of change in the domain: models for stable conditions can be updated annually; models for rapidly evolving fields, such as oncology with new drugs, may require updates every six months or even quarterly.

Calibration adjustment allows models to be adapted to new prevalences without full retraining. If only the proportion of positive cases has changed, techniques such as Platt scaling or isotonic regression can restore probability calibration with relatively few labeled data points from the new context. This approach is faster and more cost-effective than full retraining, although it does not address covariate shift or concept shift.

Designing robust models from the outset, using invariant or adversarial learning techniques, can reduce susceptibility to certain types of shifts. If we know that age affects the normal values of a variable, we design the model to be invariant to age or to explicitly incorporate age as a moderating covariate. If we anticipate variation between imaging systems, we train the model using data from multiple manufacturers to force it to learn genuine pathological signals rather than specific artifacts. This design robustness comes at the cost of increased complexity and requires more diverse training data, but it pays dividends in terms of the model’s longevity.

For the implementing clinician, dataset shift is a reality with which they must live. When faced with an externally developed model, they must demand local validation in their population before placing full trust in it, even if it involves a small sample that at least calibrates expectations. Once implemented, they must maintain professional skepticism: predictions that do not align

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

with clinical judgment deserve attention, not automatic dismissal or blind acceptance. When the model and the physician disagree, both may be wrong, but the model lacks the ability to recognize its error; the physician does.

This also has implications for the scientific evidence we generate. Clinical trials of algorithms should be designed to detect heterogeneity of effects across centers, not just to average overall results. They should include sufficiently long follow-up periods to capture temporal degradation. They should be transparent about the characteristics of participating centers, allowing future implementers to judge transferability. Evidence regarding an algorithm's efficacy is a snapshot, not a certificate of perpetual validity; it must be dated and contextualized as such.

11.3 THE PROBLEM OF GENERALIZATION

Generalization is the unspoken promise we make every time we train an algorithm. We show it a thousand cases of a disease, and we expect it to recognize the thousand and first, the two thousandth, and the case that arrives tomorrow from a different hospital with a different team and a different history. It is the difference between memorizing the answers to an exam and understanding the subject matter and between imitating the form and capturing the essence. Unfortunately, most medical AI projects meet this demand. The history of this field is written with the skeletons of algorithms that published dazzling metrics in Nature or The Lancet, only to crumble when others tried to use them in the real world. These are not anomalies or frauds; they are lessons that the field is still learning to digest. This chapter examines real-world examples of generalization failure, unravels their mechanisms, and explores how we can build systems that not only work in the laboratory but are also trustworthy in patients' lives.

11.3.1 WHEN THE MODEL LEARNS THE HOSPITAL, NOT THE DISEASE

One of the most frequently cited—and therefore most instructive—cases is that of the pneumonia detection algorithm developed by researchers at Mount Sinai Hospital and the National Institutes of Health (NIH). Published in 2017 in PLOS Medicine, the model achieved an area under the curve (AUC) of 0.93 in the joint internal validation, a figure suggesting an almost-supernatural ability to identify the disease. However, when external researchers tested it on data from the Indiana University Network for Patient Care, the AUC plummeted to significantly lower values. The difference was not statistically significant; it was a revelation.(1)

Forensic analysis revealed that the algorithm had learned to detect not pulmonary pneumonia but the digital fingerprints of the hospital where it was trained: specific marks from the institution's X-ray equipment—the Konica Minolta devices in the inpatient units—generated images with an inverted color scheme compared to the Fujifilm units in the emergency department—standard patient positions on the stretchers—and most embarrassingly, the ability to distinguish with 100% accuracy between portable X-rays from the inpatient units and the Mount Sinai emergency department. The model was an expert on Mount Sinai Hospital, not on pneumonia. When images from other hospitals lacked these institutional signatures, the algorithm was blind.

This failure illustrates a fundamental mechanism: deep learning models are correlation detectors without causal understanding. They do not know what pneumonia is; they know which pixel patterns cooccur with the diagnosis in their training data. If those patterns include specific technical artifacts, differences in equipment manufacturers, positioning protocols, or even the more frequent presence of chest drains at certain centers, the model will happily incorporate them. The solution is not more manual feature engineering but deliberate training design: multicenter data from the outset, adversarial domain techniques that penalize the model for using center-specific information, and rigorous external validation that detects these dependencies before deployment.

11.3.2 THE FRAGILITY OF ARTIFICIAL SENSES: IMAGES, TEXT, AND SIGNALS

Medical image processing algorithms are particularly vulnerable to generalization failures because of their dependence on acquisition characteristics. Diabetic retinopathy detection systems provide a telling example. Multiple studies have shown that models trained on high-quality images captured with specialized cameras under optimal conditions failed spectacularly

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

when confronted with images from portable cameras in resource-limited settings. Differences in lighting, resolution, field of view, and capture angle were sufficient to cause identical lesions to go undetected.

This problem has profound implications for the promise of democratizing eye care through artificial intelligence. If models work only with high-end equipment available in urban centers of wealthy countries, they are not expanding access; they are replicating existing inequalities. The solution lies in intentional training with technological diversity: deliberately including low-quality images from older equipment under suboptimal conditions to force the model to learn pathological signals that persist through technical noise. It also requires specific validation in the intended implementation contexts, rather than assuming that performance on high-quality images will translate to other scenarios.

Clinical natural language processing faces analogous challenges of linguistic generalization. A model trained to identify heart failure in notes from one hospital may fail in another simply because doctors use different vocabularies. A facility where "exertional dyspnea" is the standard term versus one where "shortness of breath when walking" predominates; one where abbreviations are common versus one where words are written out in full; one where the style is telegraphic versus one where it is narrative. The model learns statistical associations between words and diagnoses, but these associations are institution-specific dialects.

Generalization in medical NLP requires normalization and robustness strategies. Contextualized embedding techniques that capture meaning beyond the exact form of words. Training is performed with data from multiple institutions to expose the model to linguistic variety. Attention mechanisms that identify which text fragments are truly informative versus which are institutional noise. Validation at centers with different documentation practices was performed to verify that the model understood the clinical content, not just the local style.

11.3.3 THE LITMUS TEST: COVID-19 AND GENERALIZATION DURING THE CRISIS

The COVID-19 pandemic provided the most instructive large-scale experiment on generalization in the history of medical artificial intelligence. In early 2020, dozens of research teams developed predictive models for diagnosis, prognosis, and resource management. The urgency justified the rush; the need for tools—any tools—seemed to take precedence over methodological rigor. A subsequent systematic review of 232 published models revealed that more than 70% had a high risk of bias and that most were never externally validated. Of those that were, many showed massive performance degradation.(2)

Mortality prediction models for COVID-19 illustrate this pattern. Developed using data from early outbreaks in China or Italy, where the pandemic struck specific populations with particular triage practices, these models failed when applied to other continents, other points in the pandemic, or other stages of understanding the disease. The variables that were predicted in March 2020—certain inflammatory markers and CT scan patterns—changed significantly as treatments, virus variants, and population immunity evolved. The models, anchored in the past, became obsolete oracles.

This massive failure was not the fault of artificial intelligence as a discipline but rather of its careless application. Given the rush to publish, the lack of external validation, the assumption that a model that works in Wuhan will work in New York, and the confusion between pandemic correlation and causality. The lessons are clear: in rapidly changing situations, temporal validation is as important as geographic validation; models must be designed for continuous updating, not for static deployment; and epistemic humility—recognizing what we do not know—is more valuable than algorithmic confidence.

11.3.4 BUILDING FOR THE REAL WORLD: PRINCIPLES OF ROBUST GENERALIZATION

Generalization is not a magical property that some models possess and others do not; it is the result of deliberate design decisions. The first principle is intentional diversity in training. A model trained on data from a single hospital will learn that hospital. A model trained on data from fifty hospitals across different regions, sizes, populations, and practices will be forced to learn what is common to all of them: the patterns of the disease itself. This diversity comes with logistical and financial costs, but it is an essential investment, not a luxury. Multicenter consortia, international collaborative networks, and diverse data repositories are critical infrastructures for

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

robust medical artificial intelligence.

The second principle focuses on causal and physiological relationships versus superficial statistical associations. Some relationships are universal: severe hypoxia can damage organs in any human. Others are context specific: the association between a genetic marker and drug response varies across populations. The incorporation of explicit medical knowledge into model design—architectures that reflect physiological understanding and variables that capture causal mechanisms—can improve generalization by anchoring the model in stable biological realities. Causal learning techniques, although still emerging, promise models that identify interventions invariant across contexts.

The third principle is technical robustness against measurement variations. Appropriate normalization of variables, data augmentation techniques that simulate plausible acquisition variations, and architectures that are inherently invariant to irrelevant transformations are needed. A vision model that learns to be insensitive to rotations, scales, and contrast changes within physiological ranges will be more robust than one that memorizes specific orientations. Data augmentation is not a trick for small datasets; it involves training in generalization through exposure to variation.

The fourth principle is that external validation should be the standard, not the exception. Every model must be tested on data that did not share the same origin as the training data: different hospitals, different populations, different teams, and different time periods. This validation must be transparent: publishing not only average metrics but also breakdowns by subgroup, analyses of specific failures, and explicit limitations. A model that performs well on average but fails in specific subgroups is not generalizable; it is potentially dangerously biased.

For the developer, these principles imply that the first model will never be the final one. This implies planning for external validation from the study design stage, establishing multicenter collaborations, and setting aside independent data. It implies humility regarding the model's limitations and transparency regarding its conditions of validity. This implies that impressive metrics are not confused during development with real clinical utility, which is demonstrated only through generalization.

For the clinician, these examples serve as a guide to professional caution. When faced with any new model, the following questions must be answered: In which populations was it validated? Are they comparable to my practice? Is there evidence of performance in different contexts? Are limitations and failures reported, not just average successes? If the answers are unsatisfactory, skepticism is a professional obligation. This is not because the models are inherently bad but because the evidence of their usefulness is insufficient. In addition, in medicine, acting without adequate evidence violates the fundamental principle of primum nonnocere.

11.4 CLINICAL MLOPS: TRAINING, DEPLOYMENT, AND MONITORING

The history of software has taught us, through decades of costly mistakes, that writing code is merely the beginning of a much longer odyssey. Maintenance consumes the lion's share of the budget, emerging updates disrupt plans, monitoring reveals problems that tests never captured, and the orderly retirement of obsolete systems demands discipline that is rarely rewarded. This reality, now widely accepted in traditional software engineering, is only just appearing on the horizon of medical artificial intelligence. From this realization arises MLOps, a set of practices that professionalize the entire lifecycle of predictive models, from conception to retirement. In medicine, where a silent failure can result in harm to patients, these practices are not a technical option but an ethical imperative. This chapter explores how to operationalize the promise of sustainable clinical models—ones that not only start out with good metrics but also age with dignity and utility.

11.4.1 BEYOND TRAINING: THE COMPLETE LIFE CYCLE

The life cycle of a clinical model begins long before the first line of code or the first gradient descent. It begins with a rigorous definition of the clinical problem: what medical decision do we want to improve? What information will be available at the time of the decision? What outcome is predictable and relevant? A poorly defined problem here—predicting "readmission" without specifying the time window or "improvement" without standardizing how it is measured—does everything that follows. This conceptual design phase, often undervalued in the face of the appeal

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

of modeling, determines whether the model will have real utility or merely publishable metrics.

Identifying data sources and obtaining ethical and legal permissions are parallel and often slow steps. In regulated environments, access to clinical data requires ethics committee approval, confidentiality agreements, and technical security guarantees that can take months. Patience here is not a virtue but a necessity: shortcuts in data governance can legally invalidate the entire subsequent project. The design of the validation study, including the reservation of external data from the outset, must be completed before touching the training data. The temptation to "see what we have" before planning is natural but dangerous: it contaminates the independence of the test sets and compromises the validity of the entire validation.

The training phase, although the most visible in the scientific literature, is in MLOpsand involves just one link in an iterative chain. Yes, the algorithms are explored, the hyperparameters are optimized, and the architectures are validated. However, in a mature MLOps environment, this phase is automated through continuous integration pipelines: the modeling code is versioned, experiments are tracked, and the best models are registered in central repositories along with all their metadata (hyperparameters, metrics, dependencies, and data used). Training is not a one-time event but a permanent capability, ready to be reactivated when new data or new insights demand it.

11.4.2 DEPLOYMENT AS A PROCESS, NOT AN EVENT

Deploying the model into clinical production is the moment of greatest tension. A system that works in the lab may fail for reasons that have nothing to do with the quality of the algorithm: unacceptable latency in generating predictions, incompatibility with legacy clinical information systems, user resistance to interfaces that disrupt their workflow, or simple integration errors in the data pipeline. That is why, in clinical MLOps, deployment is not a binary switch but a gradual ramp of exposure.

The canary deployment strategy, borrowed from traditional software engineering, adapts elegantly to medicine. First, the model is deployed in a test environment that replicates production but without real patients, verifying technical integration. Then, it is activated in "shadow" mode: it makes predictions in real time but without showing the predictions to clinicians, allowing a comparison of what it would have recommended with what actually happened. Next, it is exposed to a small group of volunteer users, gathering qualitative feedback on utility and usability. Finally, it is scaled to noncritical services, where the model assists but does not determine decisions, before it is considered for application in high-risk contexts. Each transition requires explicit approval gates, documentation of decisions, and the ability to immediately roll back if problems arise.

Deployment infrastructure in clinical environments presents specific challenges. Many hospitals operate legacy systems that were not designed to integrate machine learning models. The latency requirements are strict: a prediction that takes thirty seconds to arrive may be useless in an emergency. High availability is critical: if the prediction service decreases, does the clinical system have a reliable fallback? Traceability is mandatory: Each prediction must be logged with a timestamp, model version, input data, and result for subsequent auditing. MLOps platforms such as Kubeflow, MLflow, or specialized commercial solutions provide infrastructure to manage this complexity, but they require adaptation to specific healthcare workflows.

11.4.3 CONTINUOUS MONITORING: KEEPING AN EYE ON THE DEPLOYED MODEL

Monitoring is at the heart of clinical MLOps. A deployed model is not a finished product but an evolving sociotechnical system that requires multidimensional monitoring. Monitoring predictive performance is the most obvious aspect: do sensitivity and specificity maintain the levels validated initially? This requires a systematic comparison between predictions and actual outcomes, which entails an infrastructure to collect posthoc data. In some cases, this is immediate (did the patient develop sepsis within the next 24 hours?), in others it requires follow-up (did they survive at six months?), and in others, it is complex (did their quality of life improve?). Without continuous validation mechanisms, the model operates in the dark, without knowing whether it remains useful.

Monitoring the quality of the input data involves detecting dataset shifts before they degrade performance. Comparing variable distributions between training and current data using divergence statistics, hypothesis tests, or anomaly detection methods can generate early

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

warnings. If the average age of patients suddenly increases, if a laboratory changes equipment and the results range, or if a pandemic alters the mix of pathologies, the system must notify decisionmakers for evaluation.

Monitoring clinical use reveals whether the model is being adopted, ignored, or misused. Do physicians follow the algorithm's recommendations? Do they systematically override them in certain subgroups? Are there differences in adoption between departments or shifts? A model with excellent technical performance that clinicians do not use is a total failure; a model they use blindly without question is a potential danger. Usage data, which are collected in compliance with privacy regulations, provide insights into both perceived utility and the need for additional training.

The detection of deviations must be accompanied by response protocols. When performance falls below predefined thresholds or when significant changes in input distributions are detected, automated alerts are triggered. However, the response cannot be purely algorithmic: it requires human investigation of causes. Is it actual model degradation, or a change in data quality? Is it an evolution in clinical practice or a technical issue in the pipeline? Does it require retraining, recalibration, or temporary withdrawal? Documented runbooks, which establish procedures for different scenarios, are essential for a coordinated response between technical and clinical teams.

11.4.4 CONTROLLED EVOLUTION: UPDATES, GOVERNANCE, AND WITHDRAWAL

Model retraining is a natural response to model degradation, but it is not trivial in regulated environments. Each new version requires validation, documentation of changes, and potentially regulatory review if the model is a certified medical device. The frequency of updates depends on the rate of change in the domain: models for stable conditions can be updated annually; models for rapidly evolving areas, such as oncology with new drugs, may require semiannual or even quarterly cycles. MLOps pipelines automate technical retraining, but the decision to deploy a retrained model for production must be made by a human and documented.

Governance establishes who has authority for each decision in the lifecycle. Who approves the initial deployment? Who authorizes an update? Who decides on a withdrawal? In mature organizations, these roles are formally defined: clinical validation committees, quality managers, and patient safety teams. Documenting decisions are not bureaucracy but essential traceability: if it is later discovered that a model caused harm, we must be able to reconstruct who approved it, on the basis of what evidence, and who had the authority to stop it.

The orderly retirement of models is just as important as their deployment. When a model ceases to be useful—due to obsolescence, irreversible degradation, or because it has been replaced by a better alternative—it must be retired from production in a controlled manner. This includes notifying users, transitioning to alternative systems, archiving historical data according to retention policies, and drawing lessons learned for future projects. A model left in production, continuing to generate predictions that no one reviews, is a latent risk.

The relationship between MLOps and regulation is becoming increasingly close. IEC 62304, which we discussed in previous chapters on the medical software lifecycle, provides a framework adaptable to predictive models. The FDA's guidelines on medical artificial intelligence explicitly require postmarket monitoring processes and controlled updates. The European MDR Regulation, as interpreted for medical software, includes vigilance requirements that MLOpsoperationalize. The alignment between technical MLOps practices and regulatory compliance is not a coincidence but a convergence toward patient safety standards.

For the developer, clinical MLOpsinvolve expanding skills. Expertise in deep learning architectures is no longer sufficient; an understanding of cloud or on-premise infrastructure, data pipelines, continuous integration, and systems security is needed. Close collaboration with hospital IT teams is needed, as they are often unfamiliar with the specifics of model deployment. Communication with clinicians is necessary to understand workflow needs and design useful interfaces. Compared with a laboratory researcher, the profile of the clinical MLOps engineer is hybrid and closer to that of a mission-critical software engineer.

For clinicians, MLOps provides tangible assurances of professionalism. Knowing that the model they consult is monitored, that there are processes to detect degradation and that there are channels to report issues increases legitimate trust in the tool. However, it also assigns responsibility: clinicians are the system's human sensors, often the first to notice when predictions "don't feel right." They must have mechanisms to report these intuitions, which can then be confirmed or refuted through statistical analysis. Vigilance is a shared responsibility, not

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

one delegated entirely to algorithms.

Ultimately, clinical MLOps is not an end in itself but a means to a higher end: maintaining models that truly help clinicians make better decisions for patients. A model deployed without monitoring is risk in disguise. A model that is never updated is planned obsolescence. A model without clear governance is institutionalized uncertainty. MLOps provides the framework to manage these risks to keep models alive, useful, and worthy of the trust we place in them. In the future, where artificial intelligence will be as ubiquitous in medicine as the stethoscope, having these processes in place will not be a competitive advantage but a condition of ethical existence.

If keeping traditional models up to date is already complex, how do we prepare for the arrival of massive-scale artificial intelligence systems, such as large language models, which were not designed for medicine but will inevitably be integrated into it?

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

REFERENCES

Subasri V, Krishnan A, Kore A, Dhalla A, Pandya D, Wang B, et al. Detecting and Remediating Harmful Data Shifts for the Responsible Deployment of Clinical AI Models. JAMA Netw Open [Internet]. June 2, 2025 [cited May 14, 2026];8(6):e2513685-e2513685. Available from: https://jamanetwork.com/journals/jamanetworkopen/fullarticle/2834882

Wynants L, Van Calster B, Collins GS, Riley RD, Heinze G, Schuit E, et al. Prediction models for diagnosis and prognosis of COVID-19: systematic review and critical appraisal. BMJ [Internet]. April 7, 2020 [cited May 14, 2026];369. Available at: https://pubmed.ncbi.nlm.nih.gov/32265220/

https://doi.org/10.62486/978-9915-9876-8-2.Ch11

Chapter 12 / Capítulo 12

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch12
Pages: 198-210

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 12. Generative AI and Large Language Models

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> *"Generative* artificial intelligence is not just another tool in the digital health arsenal; it is a paradigm shift as profound as the introduction of the electronic health record. For the first time, we have machines that not only classify or predict but also create, write, summarize, and converse. In addition, for the first time, the risk is not just that they will make mistakes but that they will invent things with such conviction that no one notices the error."

Imagine a doctor who ends their day with twenty medical records to write, discharge reports to complete, and referral letters to send. They spend hours on this bureaucratic task—time they could be spending with their patients. A generative AI system could, in seconds, produce a draft of each document, extracting relevant information from scattered notes, structuring it appropriately, and drafting it in clear, professional language. The doctor would only have to review and sign. The promise is tempting: freeing up clinical time, reducing the administrative burden, and improving documentation. However, it also poses risks: what if the system invents data? What if it omits critical information? What if its language is inappropriate? Who is held accountable when an automatically generated report contains an error?

Generative artificial intelligence, particularly large language models such as the GPT, have burst into public scenes with overwhelming force. Their ability to produce coherent text, summarize documents, answer questions, and hold conversations has generated enormous expectations regarding their potential in medicine. However, it has also raised concerns about their limitations and risks. Unlike traditional predictive models, which produce a number or a classification, generative models produce text, and text is inherently more difficult to evaluate and validate. An error in a numerical prediction is easy to detect; a hallucination in a text can go unnoticed for months.

In this chapter, we explore the applications and risks of generative AI in the clinical setting. We will begin with one of the most promising applications: the automatic summarization of medical records, which can free up clinical time and improve documentation. We will continue with the automation of bureaucratic tasks, such as report writing and diagnostic coding. We address the most feared risk of these models: hallucinations—those moments when AI confidently invents information that does not exist. Finally, we discuss how to evaluate these models in healthcare settings, which is a first-order methodological challenge. This is not a technical guide for implementing these models but rather a reflection on what it means to introduce systems that create—rather than merely predict—at the heart of clinical practice.

12.1 AUTOMATIC SUMMARY OF MEDICAL RECORDS

There is a cruel paradox in modern medical practice. Professionals who entered the field driven by a desire to care for people spend increasing amounts of time caring for documents. A primary care physician in a developed healthcare system may spend two hours on every eight-hour shift typing notes, writing reports, filling out forms, and justifying decisions for future audits. After seeing thirty patients, a hospital specialist faces a mountain of paperwork before being able to leave. This administrative burden is not just a bureaucratic nuisance; it is a recognized factor in professional burnout, a source of errors by omission when fatigue overcomes attention, and an invisible barrier between the clinician and the patient. The promise that artificial intelligence systems can automatically generate summaries of medical records, freeing up hours for direct care, is one of the most enticing of the era of large language models. However, like any technological promise in medicine, its realization depends on navigating technical, clinical, and ethical complexities that initial enthusiasm tends to obscure.

The automatic summarization of medical records is not merely a matter of text compression. A contemporary medical record is an ecosystem of heterogeneous information: progress notes in natural language—sometimes telegraphic, sometimes narrative—laboratory results in structured tables, imaging reports with technical descriptions and conclusions, medication lists with dosages and routes of administration, coded and uncoded allergies, and family history scattered across different sections. Summarizing all of this coherently requires not only processing words but also understanding what is relevant to each clinical context, what can be safely omitted, how to structure the information for maximum utility, and in what tone to communicate it. Large language models trained on massive medical corpora have demonstrated surprising capabilities for these tasks, but surprisingly, they should not be confused with a guarantee.

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

12.1.1 THE DIALECTS OF SUMMARIZATION: ADAPTING TO CONTEXTS AND AUDIENCES

The use cases for automatic summarization are as diverse as the moments in clinical practice. A system can generate admission summaries for specialists receiving referred patients, highlighting active diagnoses, ongoing treatments, critical allergies, and pending tests that justify the consultation. Another can produce hospital discharge reports based on progress notes from admission, organizing the reason for admission, key findings, treatment received, follow-up recommendations, and warning signs for the patient. Another can create simplified versions for the patients themselves, translating technical jargon into understandable language without losing medical accuracy. Each of these summaries has a different audience, a different purpose, a different level of detail, and a different risk of error. A single, unadapted model serves all these needs poorly.

Customization based on the recipient is where generative artificial intelligence demonstrates its flexibility. Through appropriate prompting, the same base model can be trained to produce heart-centered summaries for the cardiologist, lung-centered summaries for the pulmonologist, or summaries focused on functional impact for the physical therapist. It can adjust the level of technical detail, the depth of past and remote history, and the emphasis on preventive versus curative aspects. This adaptability is advantageous over the rigid template systems that dominated previous clinical documentation generation. However, it is also a source of risk: if the prompt does not adequately specify the recipient's needs, the model may produce a summary that is elegantly written but clinically useless or dangerously incomplete.

Managing contradictory information is a particularly acute challenge. Real medical records contain inconsistencies: one doctor documents an allergy to penicillin, while another later prescribes amoxicillin without comment; one imaging report describes a stable lesion, while a subsequent report suggests progression; and the patient denies symptoms in one consultation that they described in detail in another. A naive summary that simply condenses all of this may convey confusion or, worse, take sides with one version without pointing out the contradiction. More sophisticated models must explicitly identify discrepancies, present them within their temporal context, and, when appropriate, indicate that they require clinical clarification before important decisions are made. This ability to handle uncertainty in the source text is a hallmark of mature systems.

12.1.2 QUALITY BEYOND FLUENCY: EVALUATING WHAT TRULY MATTERS

The evaluation of automatic medical summaries cannot be limited to traditional natural language metrics. BLEU or ROUGE scores, which compare surface-level similarity with reference summaries, are insufficient: a summary may use completely different phrvalues while better capturing essential information, or it may copy phrases from the original while critical points are omitted. Human evaluation by clinicians is the gold standard, but it is costly, subjective, and difficult to scale. Hybrid metrics that combine coverage of key information (extracted via structured language processing), verifiable factual accuracy against the original record, relevance to the requested context, and readability for the target audience provide a more comprehensive evaluation.

The omission of critical information is the most dangerous error and the most difficult to detect automatically. A summary that fails to mention an iodine allergy prior to a contrast study, that overlooks a history of long QT syndrome before prescribing certain antiarrhythmics, or that ignores polypharmacy in an elderly patient can cause direct harm. Summarization systems must incorporate context-specific checklists of critical elements, ensuring that certain information is always included when it is present in the source. This "hardcoding" of medical priorities is a necessary safeguard against model variability.

Hallucination—the generation of information not present in the source—is a characteristic risk of generative models. A summary that invents a diagnosis, attributes symptoms to the patient whom they never reported, or suggests treatments that were not administered is worse than useless; it is actively dangerous. Grounding techniques, which link each statement in the summary to its specific location in the source documentation, enable efficient human verification and automatic detection of unsupported claims. Models must be designed to prefer omission over invention when the source is ambiguous.

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

12.1.3 INTEGRATION INTO THE WORKFLOW: BETWEEN AUTOMATION AND SUPERVISION

The timing and method of summary generation determine its practical utility. Is it generated automatically upon opening a new patient's chart, presenting itself to the physician initiating the consultation? Is it produced on demand when the clinician requests a "summary for referral to cardiology"? Is it created automatically upon discharge, for review before signing? Each option has implications for workflow, the physician's perception of control, and the risk of excessive dependence. The most widely adopted systems tend to offer on-demand generation with configurable options, respecting professional autonomy.

Human review of generated summaries is an essential safeguard, but its nature evolves as the system matures. Initially, the review must be thorough, line by line, and compare the text to the sources. As evidence of reliability accumulates for certain types of summaries in certain contexts, the review can become more targeted: verification of critical elements, sampling of sections, and conditional trust in the history of previous corrections. Feedback from these reviews—what errors are detected, how often, and in what contexts—must inform the retraining or adjustment of the model. A system that does not learn from its detected errors is static in a dynamic world.

The relationship between efficiency and safety is a constant tension. A system that saves thirty minutes a day but introduces errors that require twenty minutes of correction and potentially cause harm is a bad deal. A system that saves time but reduces the physician's familiarity with the patient's complete history, creating a reliance on the summary—can degrade the quality of care in subtle ways. Impact assessment must include not only documented time but also perceived quality of care, patient satisfaction, and long-term clinical outcomes.

12.1.4 PRIVACY, CONSENT, AND THE GHOST OF MEMORIZATION

Large language models have a concerning ability to memorize information from their training and reproduce it in response to appropriate prompts. In medicine, where training data contain real—albeit anonymized—medical records, there is a risk that the model will generate summaries that include specific information about patients from the training dataset, violating privacy if these prompts are cleverly designed. Differential privacy training techniques, which add mathematically calibrated noise to ensure formal privacy, and techniques for verifying that the model does not memorize sensitive data, are areas of active research but are not yet mature enough for widespread clinical deployment.

Consent for the use of data in training generative models is evolving regulatory territory. If a patient consented to their medical record being used to "improve healthcare," does that include training AI systems that will generate summaries for other patients? What if the model—or partially trained on their medical history—produces a summary for another patient who contains phrases identical to those in their own documentation? Consent policies must be updated to specify these uses, and systems must be designed to minimize the risks of reidentification and retention.

For the developer, building medical summary systems requires hybrid skills. Not just language model engineering but also a deep understanding of clinical workflows, interface design that facilitates efficient review, feedback infrastructure, and continuous improvement. It requires close collaboration with clinical informaticians, quality professionals, and privacy specialists. It alsorequires humility in terms of limitations: generative models are powerful but imperfect tools that improve with human supervision rather than without it.

For the clinician-user, automatic summarization demands the development of new skills. The ability to critically read a generated summary, verifying it against their knowledge of the patient and the source documentation when in doubt. The discipline to systematically report errors, not just correct them individually. In addition, the judgment is to know when to trust the summary and when to dive into the full history. Technology does not eliminate professional responsibility; it transforms it.

Ultimately, the automated summary must serve a more humane medicine, not one that is more efficient but cold. If it frees up time for doctors to listen, explain, and accompany patients, it will have fulfilled its promise. If it simply accelerates an already frenetic pace, or worse, if it creates a distance between the clinician and the patient's full history, it will have betrayed the medical purpose. The direction depends on how we design, implement, and regulate these technologies. The choice is ours, and so is the responsibility.

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

12.2 AUTOMATION OF BUREAUCRATIC TASKS

There is a monster in modern medicine that does not appear in medical records or management reports but that devours hours, energy, and vocations with silent efficiency. It is called healthcare bureaucracy, and it is estimated to consume between one-third and one-half of doctors' working hours. It is not a single task but an ecosystem of small demands that accumulate: coding each diagnosis into standardized taxonomies, justifying each costly treatment to payers with shifting criteria, responding to requests from insurers who question decisions already made, filling out forms for quality indicators that no one seems to use, and drafting appointment letters and reminders that should be automated. Each of these tasks, examined in isolation, has its own administrative logic. Taken together, they constitute a burden that distracts professionals from what truly matters: caring for people who are suffering. The promise that generative artificial intelligence can take on part of this burden is, therefore, extraordinarily appealing. However, like any promise of power, it raises questions about accuracy, accountability, and the risk that technically gained efficiency will be lost in new forms of clinical detachment.

12.2.1 CODING AND ITS LABYRINTHS: WHEN DIAGNOSIS MEETS THE CODE

Diagnostic coding is perhaps the most burdensome and least rewarding bureaucratic task in medical practice. Every clinical encounter must be translated from the rich, narrative, sometimes ambiguous language of medical judgment into standardized codes: the ICD-10 for general diagnoses, the CIAP for primary care, the CPT for procedures, the SNOMED-CT for more sophisticated systems, and national variants that complicate the landscape. This translation is necessary for billing, public health records, epidemiological research, and quality management. However, it is tedious, prone to errors in selecting among similar codes, and consumes time because the physician does not spend listening to the patient.

A generative artificial intelligence system can, on the basis of clinical notes in natural language, suggest the most appropriate codes and even justify the selection. The model analyzes the case description, identifies medical entities, maps them to the target taxonomy, and presents suggestions ranked by confidence. The physician reviews, confirms, or corrects, and the system—learns from these corrections for future suggestions. In theory, this reduces coding time from minutes to seconds, improves consistency by eliminating variability among human coders, and increases specificity by suggesting codes that are more precise than those the physician would recall.

The reality is more complex. Medical coding systems are labyrinths where seemingly similar codes have very different implications. The difference between "type 2 diabetes mellitus with complications" and "type 2 diabetes mellitus without complications" is not academic; it affects billing, resource prioritization, and facility quality metrics. A model that suggests the most common code because it appears most frequently in its training, without capturing the subtleties of the particular case, introduces systematic biases. Human oversight is not optional; it is a guarantee that clinical judgment prevails over the model's statistics.

12.2.2 JUSTIFICATIONS, REQUIREMENTS, AND THE DANCE WITH PAYERS

Justifying treatments represents another frontier of bureaucratic automation. When a physician prescribes an expensive drug, a complex procedure, or a prolonged hospital stay, they must often document why it is necessary, especially when dealing with payers with budget constraints or specific eligibility criteria. This justification must be clinical, evidence-based, and tailored to the format and criteria of each payer: private insurers with their own forms, public systems with prior authorization protocols, and rare disease programs with specific requirements.

An artificial intelligence system can, by knowing the patient's history, the proposed treatment, and the relevant clinical guidelines, draft a justification that the physician reviews and submits. The model identifies elements in the clinical documentation that meet the payer's criteria, structures the argument clinically, and adapts the language to the required format. This promises to save hours of work and, potentially, improve the approval rate by ensuring that no relevant elements are omitted.

The challenge lies in the variability of requirements. Each payer has its own form, procedure codes, coverage criteria, and preferred documentation styles. A system that works perfectly for insurer A may fail completely for insurer B. Adaptability through prompting—natural language

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

instructions that specify the payer's context—is promising but not infallible. Models may "hallucinate" criteria that do not exist or omit specific requirements they do not recognize in their general training. Human verification that the draft exactly meets the specific payer's requirements remains mandatory.

The response to administrative requests follows a similar pattern. Insurers, accrediting bodies, and disease registries periodically request information that already exists in medical records but is scattered across different formats, requiring manual synthesis. An artificial intelligence system can automatically extract the relevant information, structure it according to the requested format, and generate draft responses. The potential efficiency is enormous, especially for routine and repetitive requests.

Exception handling is where automation reaches its limits. Not all cases fit standard patterns: patients with complex situations requiring unconventional reasoning, low-level treatments with emerging but not yet established evidence, and rare diagnoses where guidelines do not apply directly. In these cases, a model that always produces a response—even if it is inadequate—is more dangerous than one that recognizes its limitations and refers to the case of a human. Systems must be designed to detect when confidence is insufficient, when the case deviates from trained patterns, and when human intervention is imperative.

12.2.3 INDICATORS, ALIGNMENT, AND MEASURING WHAT MATTERS

Generating reports on quality indicators is a task that consumes substantial resources in hospitals and health centers. Dozens of indicators—wait times, readmission rates, vaccination coverage, adherence to guidelines—require data extraction from multiple sources, aggregation, and presentation in standardized formats for regulatory and accreditation bodies. An artificial intelligence system can automate much of this process, generating preliminary reports that quality managers review and finalize.

The risk here is the "dictatorship of the measurable." Quality indicators are imperfect proxies for the true quality of care. A system that optimizes report generation may inadvertently prioritize the appearance of quality over actual quality. If the model learns that certain codes or formulations improve indicators without reflecting better care, it may systematically suggest them, distorting both documentation and practice. Human oversight must ensure that indicators reflect reality, not that reality is shaped to meet indicators.

Correspondence with patients—appointment letters, test reminders, test results, and postdischarge instructions—is an area where automation promises a dual benefit: efficiency and improved communication. A system that generates personalized drafts, tailored to the patient's level of understanding, their specific clinical situation, and the tone desired by the institution, can save time on drafting while improving clarity. Reminders can be more effective if personalized; postdischarge instructions are safer if they are adapted to the patient's specific polypharmacy.

Mass personalization, however, has ethical limits. Medical correspondence is not merely the transmission of information; it is also the building of a therapeutic relationship. A patient who receives letters clearly generated by a machine may feel depersonalized, perhaps reduced to a database entry. The balance between efficiency and humanization is delicate: automation should free up time for personal interaction and not replace all interactions with efficiency. The design of these systems must include "human touches," options for physician customization, variations in tone, and spaces to add handwritten notes, all of which maintain the human connection.

12.2.4 INTEGRATION, ACCOUNTABILITY, AND CONTROL OF AUTOMATION

Integrating these capabilities into existing electronic health records systems is a technical and organizational challenge. Powerful models alone are not enough; connectivity is required with patient databases, clinical guideline repositories, up-to-date code catalogs, and constantly evolving payer systems. Interfaces are needed that enable efficient review: not only display the model's suggestion but also link it to the source in the clinical documentation, allowing for intuitive editing and capturing feedback on corrections. Workflows are required that integrate automation without disrupting practice, given that different physicians work at different paces.

The relationship between automation and legal liability is uncharted. If an incorrect diagnostic code generated by the system leads to fraudulent billing, who is liable? If a poorly drafted treatment justification causes a denial of coverage and harm to the patient, is it the

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

physician who signed off without thoroughly reviewing it? The hospital that implemented the system? The developer who trained the model? Case law is only just beginning to take shape, and in the meantime, prudence calls for defensive design: mandatory human oversight, full traceability of decisions, and clear documentation that medical liability remains with the professional.

Concern about loss of control is legitimate and profound. If the system automatically generates codes, justifications, and responses, how can we ensure that they reflect actual clinical judgment and not the statistical optimization of the model? Won't physicians, pressed for time, end up accepting suggestions without the critical review that safety demands? Aren't we silently delegating to the machine decisions that should be human? These concerns are not Luddism; they are an expression of the tension between efficiency and professional autonomy that characterizes all meaningful automation.

Careful design can mitigate these tensions. Systems must be "reviewable by design": they should display not only the final suggestion but also the reasoning behind it, allow the physician to explore alternatives, and make it explicit when the model's confidence is low. They must be "ignorable without penalty": a physician who prefers to code manually or draft their own justifications should be able to do so without friction. They must be "learners from their own corrections": When the physician corrects a suggestion, the system must incorporate that feedback to improve future suggestions, creating a virtuous cycle of adaptation.

In terms of job satisfaction, the impact is a critical factor that is often overlooked. The automation of bureaucratic tasks can free up time, but it can also strip the work of aspects that some professionals value. Some physicians prefer to write their own reports because they retain narrative control or find intellectual satisfaction in precise coding. Imposing uniform automation can lead to resistance and alienation. Implementation must be flexible, allowing professionals to choose when and how to use these tools while respecting the diversity of work styles.

12.3 MEDICAL HALLUCINATIONS: THE RISK OF INVENTION

There is a phenomenon in large language models that has evolved from academic curiosity into a critical obstacle to their adoption in medicine. When these models encounter questions with no clear answer in their training data or when they encounter ambiguous information, they do not always respond with the honesty of an experienced physician: "I don't know for sure; I need to verify." Instead, they frequently make things up. They do so with such fluency, with such an air of scientific authority, that even expert readers may be slow to detect deception. This phenomenon has been termed "hallucination," a borrowed psychiatric term that captures its quality of perception without reality. In any other context, it would be an amusing anecdote. In medicine, where decisions are made in seconds on the basis of the information presented, it is a potentially lethal risk that demands a deep understanding and rigorous safeguards.

Hallucinations are not ordinary errors of knowledge or interpretation. An ordinary error is understandable and detectable: the model confuses two drugs with similar names, misinterprets a dose, or applies a guideline to a context where it does not apply. A hallucination is qualitatively different: the model generates information that exists nowhere in its training or in reality, citing studies that were never conducted and describing symptoms that do not appear in the literature, attributing to recognized authors statements they never made. It does so with the same expressive certainty with which it describes an established fact. For a physician consulting the model in the midst of an urgent clinical decision, distinguishing between what is real and what is invented may be impossible without external verification that time does not permit.

12.3.1 ARCHITECTURE OF INVENTION: WHY MODELS HALLUCINATE

The cause of hallucinations lies in the fundamental nature of large language models. Unlike traditional knowledge-based systems, which consult structured repositories of ver d facts, these models operate through the statistical prediction of sequences. They have no internal representation of "truth" against which to test statements. What they possess is the ability to predict, given a sequence of preceding words, what the next most likely word is within the context of the trained language. When they generate text, they produce statistically plausible sequences, not necessarily true ones.

If certain combinations of concepts appeared frequently together in the training data, the model learned that they were "cooccurring" and reproduced them. If the requested combination

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

has no established true answer in the literature but the model has seen similar patterns, it can generate something that sounds convincing but is completely fictitious. Linguistic plausibility—correct grammar, appropriate terminology, and logical structure—is independent of factual accuracy. A model trained on medical texts learns to speak like a doctor, not necessarily to think like a rigorous doctor.

In medicine, hallucinations take on particularly dangerous forms because of the gravity of the consequences of each error. They can invent drug interactions that do not exist, leading doctors to avoid perfectly safe and effective therapeutic combinations. They can describe drug side effects that have never been documented, causing unnecessary anxiety in patients or leading to the unjustified discontinuation of necessary treatments. They can cite clinical studies with completely fictitious authors, dates, journals, and numerical results, giving the appearance of scientific evidence to recommendations that have no basis whatsoever.

12.3.2 DOCUMENTED CASES: WHEN FICTION RESEMBLES SCIENCE

A particularly instructive case involved a language model that was asked to generate a summary of evidence regarding an experimental treatment for a rare oncological condition. The model produced text that was flawless in form: a contextualized introduction, methods described with apparent precision, numerical results with confidence intervals, and a balanced discussion. The citations included titles from prestigious journals, names of authors recognized in the field, and recent publication years. Only when a researcher attempted to locate the cited articles did they discover that none existed. The model had invented everything: the titles, the authors, the journals, the methods, and the results. The presentation was so convincing that without access to bibliographic databases for verification, any user would have accepted the presented evidence as valid.(1)

Hallucinations are particularly difficult to detect because the models have learned to perfectly mimic scientific language. They use jargon appropriate to the subspecialty, structure their responses as an expert in systematic reviews, maintain internal consistency that inspires confidence, and cite sources in the correct format. A reader who is not a specialist in a specific subject may have no way of knowing that they are reading fiction. Even specialists may have doubts, especially if the hallucination pertains to a very specific area where their own knowledge has gaps. The model's authoritative tone, confident manner, apparent detail, and absence of signs of doubt are counterindicators of reliability, but psychologically, they function as indicators of credibility.

12.3.3 MITIGATION STRATEGIES: BETWEEN PREVENTION AND DETECTION

For developers of systems based on generative artificial intelligence, hallucinations represent the most pressing technical problem. There is no complete solution, but various approaches can reduce their frequency and severity. Retrieval-augmented generation (RAG) is a promising strategy: instead of generating responses purely from the model's internal knowledge, the system first consists of reliable and verifiable databases, indexed medical literature, official clinical guidelines, and drug registries and then generates a response based strictly on the retrieved information. This drastically reduces hallucinations but requires that the database be comprehensive and up-to-date and that the model be capable of correctly extracting relevant information without distortion.

Reinforcement learning with human feedback (RLHF) can reduce the frequency of hallucinations by rewarding verifiable answers and penalizing in s. However, it does not eliminate the following fundamental problem: the model still lacks an internal truth mechanism; it merely learns to avoid answers that humans identified as having been diagnosed with hallucinations in the past. New situations can generate new forms of invention not seen in the feedback training.

Designing systems that, when in doubt, prefer to respond "I don't know" rather than risk hallucinating is a desirable but technically challenging goal. It requires the model to have a precise calibration of its own uncertainty—a capability that current models possess only in a limited and unreliable manner. A model that underestimates its ignorance will continue to hallucinate; one that overestimates its uncertainty will be useless because of excessive caution.

Human oversight remains the most effective barrier against hallucinations in clinical contexts. Any text generated by artificial intelligence that is intended to influence medical

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

decisions must be reviewed by a professional before use. This review cannot be superficial; it must include verification of key statements, especially those affecting diagnostic or therapeutic decisions. However, this oversight limits the utility of automation: if every response requires exhaustive verification, the time savings are drastically reduced, and the risk that oversight will become routine and less rigorous over time is real.

12.3.4 EDUCATION, SKEPTICISM, AND THE FUTURE OF TRUTHFULNESS

For clinicians, the existence of hallucinations demands a radically skeptical attitude toward any text generated by artificial intelligence. They cannot trust that something is true simply because it is wellwritten and expressed with authority. They must verify important claims, especially if they are going to base clinical decisions on them. They must be alert to potential signs of hallucination: vague references ("some studies suggest..."), the absence of verifiable citations where they would be expected, details that do not align with their knowledge of the subject, or suspiciously perfect data—which in real research always contain some noise.

Education on hallucinations must be a mandatory component of any artificial intelligence training program for healthcare professionals. The goal is not to instill paralyzing fear but to provide training for safe use. A physician who understands how the models work and who knows that they can hallucinate and through what mechanisms is better positioned to use these tools in a way that leverages their benefits while mitigating their risks. They are also better positioned to demand that developers implement safeguards and to actively participate in system evaluation prior to institutional adoption.

The future of generative artificial intelligence in medicine necessarily depends on resolving the problem of hallucinations or at least rigorously containing their risks. Architectures that explicitly incorporate the notion of truth, with mechanisms to verify claims against reliable sources in real time, are an active area of research. Automatic verification methods that detect internal inconsistencies or statistical improbabilities in generated text can provide early warning signals. The development of specialized "verification models," trained specifically to detect hallucinations in the outputs of other models, is a defense-in-depth strategy.

In the meantime, prudence must guide all clinical applications. Hallucinations are neither a minor flaw nor a temporary bug; they are a manifestation of the fundamental nature of large language models. They were designed to generate plausible text, not to be accurate encyclopedias. Plausibility is not true, and in medicine, the difference between the two can have irreversible consequences. Generative artificial intelligence must be used with extreme caution, with mandatory human supervision, and with full awareness that every model output is potentially a mixture of trained wisdom and statistical invention. Only in this honesty about limitations lies the possibility of beneficial and nonharmful use.

12.4 EVALUATING LLMS IN HEALTHCARE SETTINGS

Evaluating a language model in medicine is not like evaluating a traditional predictive algorithm. When a model classifies images or predicts risk, we have a clear gold standard: a biopsy that confirms the diagnosis and an outcome that verifies the prediction. The metrics are numerical, comparable, and reproducible. However, when a model generates a medical record summary, answers a distressed patient's question, or drafts a treatment justification, against what we compare its output? Who decides if the text is correct, complete, appropriate, and safe? The evaluation of generative models is a methodological challenge of the first order, and its imperfect resolution is a necessary—although not sufficient—condition for these systems to be safely integrated into clinical practice.

12.4.1 THE ABSENCE OF A SINGLE TRUTH: WHEN "CORRECT" IS PLURAL

The first profound difficulty is the absence of an unequivocal "fundamental truth" for many generative tasks. A summary of a patient's medical history can be valid in multiple ways: focused on active problems versus comprehensive in background, detailed on medication versus centered on diagnoses, and structured chronologically versus by systems. Two expert physicians may produce summaries that are equally correct but substantially different. How, then, do we evaluate whether the model's output is acceptable? The answer inevitably involves human judgment, but

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

human judgment is costly, variable across evaluators, difficult to scale, and potentially biased by expectations regarding the technology.

This plurality of valid answers does not mean that everything is valid. There are hard limits: omission of allergies is unacceptable, invention of diagnoses is unacceptable, and recommendations contrary to established guidelines are unacceptable. However, within safety limits, there is ample room for legitimate variation. Evaluation must capture this complexity: not only detecting serious errors but also rating the quality of acceptable responses, differentiating between excellent, adequate, and improvable.

12.4.2 THE DIMENSIONS OF QUALITY: BEYOND BINARY ACCURACY

The quality of generated medical text is inherently multidimensional. Factual accuracy—are the contained data correct?—is only the first dimension. Completeness—does it include all information relevant to the context?—is distinct: a summary may be factually correct in what it includes but dangerously incomplete in what it omits. Coherence—is it easy to follow logically?—affects practical utility. Conciseness—does it avoid irrelevant, distracting information?—determines whether the physician will actually read the summary. Tailoring to the audience—does it use language appropriate for the intended reader?—varies between a specialist, a general practitioner, an educated patient, or a patient with low health literacy. In addition, safety—does it contain statements that could lead to erroneous decisions if taken at face value?—is a critical dimension that goes beyond factual accuracy.

Each dimension requires different evaluation methods. Factual accuracy can be verified against structured medical records. Completeness requires context-specific checklists of critical elements. Coherence can be assessed using readability metrics and human judgment. Suitability for the target audience requires evaluators who represent that audience. In addition, safety requires adversarial testing designed to provoke problematic responses. A model may be excellent in terms of factual accuracy but deficient in safety or brilliant in coherence but dangerous in completeness. One-dimensional evaluation is misleading.

12.4.3 DETECTING THE NONEXISTENT: THE EVALUATION OF HALLUCINATIONS

Hallucinations—convincing fabrications of false information—pose a particularly acute evaluation challenge. They are subtly plausible: a fabricated study will have a credible title, authors with real names, and reasonable numerical results. Detecting that it is false requires verification against reliable sources, a task that no human can perform exhaustively at scale. Automated methods for detecting hallucinations—verifying citations against bibliographic databases, detecting internal inconsistencies, and analyzing sequence probabilities—have their own limitations and error rates.

Assessing hallucinations in medical contexts requires the deliberate design of “traps”: questions designed to elicit fabricated responses because the true answer does not exist or is extremely rare. Ambiguous clinical cases for which there is no established consensus exist. Requests for information about drugs that do not exist. Questions about clinical guidelines for recently discovered diseases where the literature is scarce. The frequency with which the model falls into these traps and the nature of its responses are critical safety metrics.

12.4.4 PRACTICAL APPROACHES: BETWEEN THE LAB AND THE CLINIC

Faced with these complexities, multiple evaluation approaches have emerged, none perfect, all partial. Task-based evaluation defines sets of test cases with ground-truth answers crafted by experts and compares the model’s output to these ground-truths using similarity metrics or human judgment. It is the closest to traditional evaluation but requires immense effort to create cases and does not capture the legitimate variability of acceptable answers. It is useful for evaluating specific skills but is limited for general evaluation.

Clinical peer evaluation uses groups of expert clinicians who evaluate model outputs across multiple dimensions using standardized scales. It better captures the complexity of the task but is costly and slow and suffers from interrater variability that necessitates multiple evaluations per case and statistical aggregation methods. Interrater reliability is a critical metric: if experts cannot agree on what constitutes a good response, the model cannot be consistently optimized.

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

The use of language models as evaluators, known as LLM-as-a-judge, is an emerging and promising approach. A second model evaluates the quality of the first by following detailed instructions regarding dimensions and criteria. It is scalable and consistent, but it raises obvious questions: if models can hallucinate, can't they also hallucinate in their evaluations? Research suggests that with careful instructions and on well-defined tasks, this approach correlates reasonably well with human judgment, but it should not be the sole or final evaluation.

Safety evaluation through adversarial testing is a critical component. It includes cases designed to provoke failures: scenarios where incorrect information would have serious consequences, questions on topics where the model should admit ignorance, and requests that combine concepts in ways not seen during training. The frequency and nature of failures in these tests determine confidence limits for deployment.

Fairness evaluation examines whether the model generates responses of differing quality or content for patients from different demographic groups. It requires systematic evaluation across population subgroups and comparisons of multidimensional quality metrics across groups. It is complex but essential work: a model that performs well on average but systematically fails specific minorities perpetuates health inequalities.

12.4.5 REAL-WORLD VALIDATION: WHERE TESTING ENDS

Evaluation in real-world contexts is the final and indispensable step. A model may excel in laboratory tests and fail in real-world practice because of usability issues, integration challenges, changes in user behavior, or dataset shifts. Prospective studies, in which the model is used under real-world conditions with real patients and its impact on clinical processes and outcomes is evaluated, are the sole source of evidence regarding real-world utility. They are costly and complex and require specific ethical approval, but they are mandatory prior to widespread institutional adoption.

For developers, the evaluation of LLMs in medicine is a rapidly evolving field with no established standards. Prudence calls for multiple complementary methods, exhaustive documentation of limitations, transparency regarding uncertainties, and close collaboration with clinicians in the evaluation design to ensure the clinical relevance of metrics. There are no shortcuts: rigorous evaluation is a necessary investment for justified trust.

For clinicians, understanding how these models are evaluated empowers them to interpret performance claims. When a provider announces "95% accuracy," the following question should be answered: When is accuracy in what dimension? evaluated by whom? in which populations? with what types of cases? including adversarial testing? Without clear answers, the claim is marketing, not evidence. Clinicians must demand transparency and participate in validation before adoption.

The future of evaluation lies in the development of shared standards, similar to the CONSORT guidelines for clinical trials. Initiatives such as TRIPOD-LLM extend the TRIPOD guidelines to generative models. Public repositories of test cases will enable standardized comparisons between models. In addition, collaboration among novel Rs, clinicians, regulators, and patients will define what it truly means for a language model to be safe and effective. In this cocreation of standards lies the potential for generative artificial intelligence that is trustworthy in healthcare.

If we have learned to evaluate the quality and safety of generative models, how do we ensure that when these systems get it right, they also explain why, allowing clinicians not only to trust but also to understand, question, and, when necessary, correct?

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

REFERENCES

Emsley R. ChatGPT: these are not hallucinations – they're fabrications and falsifications. Schizophrenia [Internet]. December 1, 2023 [cited May 14, 2026];9(1):52. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC10439949/

https://doi.org/10.62486/978-9915-9876-8-2.Ch12

Chapter 13 / Capítulo 13

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch13
Pages: 211-224

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 13. Explainability and Trust (XAI)

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "An algorithm that gets it right but doesn't explain why is like a doctor who makes the correct diagnosis but cannot justify it. In both cases, the correct answer may be due to chance, intuition, or even an error disguised as a correct answer. Medicine is not based on faith but on reasons. In addition, reasons, to be valid, must be expressible."

Imagine that an artificial intelligence algorithm recommends against administering anticoagulant therapy to a patient at high risk of thrombosis. Its recommendation contradicts the doctor's intuition, who believes the patient would clearly benefit from the treatment. The doctor asks the algorithm why it recommends against anticoagulation. The response is a list of probabilities, weights, and statistical correlations that shed no light on the decision. Doctors face a dilemma: they follow their clinical judgment, which is based on years of experience and deep knowledge of the patient, or follow the algorithm's recommendation, which likely processes more data than he has, but whose reasoning he cannot understand. In this situation, there is no obvious correct answer, but there is one certainty: the lack of explainability undermines trust and hinders shared decision-making.

Explainability is neither a luxury nor an academic frill. It is a clinical, ethical, and legal necessity. Clinically, without understanding why an algorithm recommends something, the physician cannot integrate that recommendation with their own knowledge of the patient. Ethical, because patients have the right to understand the decisions affecting their health, even when those decisions are influenced by algorithms. In the event of an error, the courts will demand answers about how the decision was made, and "the algorithm said so" is not an admissible defense. Explainable artificial intelligence, known as XAI, is the field that seeks to endow models with the ability to explain their reasoning in a way that is understandable to humans.

In this chapter, we explore the fundamentals of explainability as applied to digital health. We begin with the fundamental premise: it is not enough for the algorithm to be correct; the physician must know why. We will continue with the most commonly used techniques for explaining models, such as SHAP and LIME, which allow predictions to be broken down into the contributions of each variable. We address the current limits of explainability, especially in complex models such as deep neural networks. Finally, we discuss how to communicate risk and uncertainty to the clinician and the patient—an aspect as important as the technical explanation itself. The goal is not to turn the reader into an expert in XAI but to equip them with the questions they should ask when faced with any system that aims to influence their clinical decisions.

13.1 IT IS NOT ENOUGH TO BE RIGHT; THE PHYSICIAN MUST KNOW WHY

> "A diagnosis without an explanation is an oracle, not medicine; and doctors should not practice divination."

The history of medicine is full of remedies that worked without anyone understanding why. Willow bark for fever, used since ancient times, only revealed its secret—acetylsalicylic acid—when organic chemistry matured. Digitalis, extracted from the purple foxglove, relieved heart failure centuries before we understood sodium and potassium channels. Quinine from the cinchona tree saved lives from malaria, while parasitology was still a dream. This functional ignorance was not a medical virtue but a historical limitation that science gradually overcame. Modern medicine is not content with knowing that something works; it demands understanding why it works because that knowledge allows us to predict side effects, extend applications to new situations, combine treatments rationally, and correct them when they fail. Explanability is not an intellectual luxury but a practical tool for improving care.

Medical artificial intelligence should be held within the same epistemological standard. An algorithm that predicts sepsis with high accuracy is useful, but its utility is greatly enhanced if it explains which factors drive that prediction. Is recent heart rate the main driver? The trend in blood pressure? A specific pattern of lab values? The nonlinear interaction of multiple subtle signals? With this information, the physician can direct diagnostic attention to the identified elements, verify whether they make physiological sense in the patient's context, explain to the patient why they are considered highrisk, and decide whether the algorithm's pattern applies to this particular situation. Without it, the prediction is a black-box verdict: it is either right or wrong, but it does not teach, does not allow for questioning, and does not respect the intelligence of the clinician who must use it.

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

13.1.1 WHEN ERROR IS KNOCKED AT THE DOOR: THE NEED TO DIAGNOSE

All predictive models make errors; in medicine, those errors can result in harm. When an algorithm fails, explainability becomes imperative, not optional. Without understanding why the model was wrong, we cannot correct it; we cannot prevent it from occurring again, and we cannot even determine whether the error was attributable to the algorithm or to an exceptional circumstance in which no reasonable model would have anticipated. A model that does not explain its errors is a model from which we cannot learn and therefore a model that cannot be fully trusted.

Imagine a sepsis risk prediction system that, in a particular case, predicts high risk, but the patient progresses without complications. If the model explains that its prediction was based primarily on mild tachycardia in a young patient with no other signs, the physician can recognize that the model overemphasizes that sign in this context, adjust their interpretation, and report the case for algorithm refinement. If the model does not explain, the physician only knows that it failed, without knowing whether it was a systematic error or an exception and without the ability to take preventive action. Explainability transforms an opaque error into an opportunity for improvement.

The ability to learn from errors is especially critical in medicine, where biological variability ensures that unusual cases will occur. A model that explains this situation allows for the creation of a repository of borderline cases, an understanding of failure conditions, and iterative improvement. Without explanation, each error is an isolated event with no discernible pattern; with them, errors become data for the system's evolution. This capacity for guided self-correction is the hallmark of mature systems versus fragile ones.

13.1.2 DETECTING THE INVISIBLE: BIASES REVEALED BY THE LIGHT OF EXPLANATION

Explainability also serves as a detector of biases operating in the dark. An algorithm may systematically discriminate against certain groups without its creators being aware of it, especially when discrimination operates through proxy variables. If the model does not explain its decisions, this bias can remain hidden for years, causing cumulative silent harm. If the model explains, even partially, the e xplanations can reveal suspicious patterns: that the ZIP code has an unusually high weight in risk predictions, suggesting confusion between geographic location and socioeconomic status; that age is used differently for men and women, indicating possible gender bias; and that certain hospital names appear as strong predictors, revealing overfitting to the training context.

These signals, visible only when the model opens up its logic, can trigger deeper investigations and corrections before widespread harm occurs. Explainability is thus a mechanism for ethical oversight, not just technical oversight. Without it, algorithmic justice is an impossible aspiration; with it, it is a difficult but attainable goal.

An example of the "zip code" as a predictor of health risk illustrates the problem. In many health systems, the zip code strongly correlates with health outcomes, not by geographical coincidence but because it captures socioeconomic variables: access to healthy food, air quality, opportunities for exercise, and environmental stress. A model that uses ZIP code as a direct predictor may seem accurate but perpetuates structural inequalities. Only when the model explains its reliance on this variable and when clinicians question whether it is appropriate can a correction toward more direct causal variables emerge.

13.1.3 TRUST AS A CONDITION OF USE: WHY CLINICIANS IGNORE WHAT THEY DO NOT UNDERSTAND

Trust is a determining factor in clinical adoption, and explainability is a necessary condition for justified trust. Repeated studies show that clinicians ignore recommendations from decision support systems when they do not understand their reasoning, even when those recommendations are correct. This resistance is not an irrational obstinacy but professional prudence: a physician who blindly follows an algorithm they do not understand abandons their responsibility for clinical judgment becomes dependent on a tool that may fail without warning and loses the ability to explain their decisions to patients and colleagues.

Explainability does not guarantee trust; a physician may understand the explanation and find it insufficient, but its absence guarantees mistrust or, worse, unfounded trust. An algorithm that does not generate trust is an algorithm that is not used, no matter how accurate it may be

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

in validation. Explainability is thus a condition for real utility, not a mere ethical embellishment.

The dynamics of trust are complex and deserve specific study. Research shows that doctors trust systems more when they explain using familiar concepts rather than technical model concepts, even when both explanations are equivalent in informational content. Familiarity with the format matters as much as the information. These findings suggest that effective explainability requires not only technical expertise but also a design centered on clinical users, which translates model outputs into language that resonates with medical reasoning.

13.1.4 LEGAL LIABILITY AND DECISION TRACEABILITY

The relationship between explainability and legal liability is a growing point of tension in algorithmic medicine. In the event of an error resulting in harm to the patient, courts will ask who is legally liable. If the physician followed an algorithmic recommendation without understanding it, does that limit their liability or increase it because of negligence in exercising their own judgment? If they ignored a correct recommendation and harm occurred, are they more liable for failing to follow the evidence or less so for not trusting a black box? Legal answers are evolving, but in any scenario, having clear explanations of what the algorithm recommended and why is essential for reasonably determining liability.

Without explanations, the legal debate devolves into speculation about the mental states and capabilities of opaque systems. With explanations, one can assess whether the model's reasoning was reasonable, whether the physician had grounds to accept or reject it, and whether the harm was preventable. Explainability is thus the infrastructure of justice, not just of clinical efficiency.

Emerging case law on medical artificial intelligence suggests that the "black box" will not be an acceptable defense for developers. If a model causes harm and cannot explain why it made the decision it did, courts will tend to attribute liability to the manufacturer for a design defect. This creates powerful legal pressure toward explainability that complements ethical and practical pressure. Opaque systems become legal risks, not just technical ones.

13.1.5 LEVELS OF EXPLAINABILITY: FROM TRANSPARENT TO APPROXIMATE

There are different levels of explainability, and not all the models achieve them equally. Some models are inherently interpretable: decision trees reveal their logic in their branching structure, logistic regression expresses its criteria through coefficients that quantify each variable's contribution,and association rules present explicit patterns. These models allow for a direct understanding of how they reach conclusions, generally at the cost of reduced predictive power for complex nonlinear relationships.

Other models—deep neural networks, gradient boosting, and support vector machines with complex kernels—are black boxes that require posthoc techniques to generate explanations. These techniques, such as SHAP and LIME, which we explore in the next section, approximate the model's behavior locally, revealing which factors influence a specific prediction without revealing the complete internal workings. They are like telescopes that show part of the model's universe, not the entire universe.

There is a trade-off between interpretability and predictive power that must be consciously decided. For high-risk decisions where understanding is a priority, interpretable models may be preferable even if their performance is slightly lower. With respect to complex patterns in which only deep architectures capture the signal, posthoc explanation techniques may be acceptable because of their limitations. The choice is a clinical decision, not merely a technical one, and must involve those who will use and rely on the system.

13.1.6 TOWARD EXPLANATIONS THAT SPEAK THE LANGUAGE OF MEDICINE

The future of explainability lies in deep integration between artificial intelligence and clinical reasoning. It is not just a matter of algorithms explaining their decisions but of those explanations aligning with how clinicians actually think. A clinician does not primarily reason in terms of variable weights or SHAP contributions; they reason in terms of pathophysiological mechanisms, coherent clinical histories, similar past experiences, and recognizable patterns. Explanations should be able to be translated into this language: instead of "variable X contributing 0.3 to the prediction," prefer

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

"the pattern of tachycardia disproportionate to fever suggests an early systemic inflammatory response."

Translating technical explanations into clinical narratives is a massive challenge that requires collaboration between engineers and clinicians, language models that rephrase explanations in understandable terms, and validation that these rephrasings remain faithful to the original reasoning. However, it is a necessary investment: explanations that do not speak the language of their recipients do not truly explain; they merely transfer opacity from one form to another.

For developers, incorporating explainability from the design phase is a responsibility, not an option. In terms of model selection, interpretability and performance should be considered. During development, explanations are consistent with clinical knowledge and useful for end users. At deployment, interfaces that present understandable explanations without overwhelming users with unnecessary details should be provided. In addition, maintain humility: an explanation is an approximation, not a complete revelation.

For clinicians, explainability is a tool to exercise judgment with more information, not to delegate it to the machine. Receiving an explanation should lead to the following questions: Does it make clinical sense? Does it align with what I know about this patient? Are there factors the algorithm did not consider? The explanation is ne, nor is it revealed truth but rather another input for clinical deliberation. The physician remains ultimately responsible for the decision, and the explanation should help them make it with greater knowledge and not abdicate their responsibility.

In that dialog between algorithmic explanation and clinical judgment, where each contributes what the other lacks, lies the potential for medical artificial intelligence that truly improves care, not just speeds it up.

13.2 EXPLAINABILITY TECHNIQUES: SHAP, LIME, AND OTHERS

The need to explain algorithmic decisions has spawned a flourishing field of research, with dozens of techniques promising to shed light on the inner workings of opaque models. Some have become de facto standards, adopted globally by developers who need to justify predictions to users and regulators. However, like any powerful tool, they have significant limitations and can be misused. Understanding what they are, how they actually work, and when they are appropriate versus when they are misleading is essential for clinicians who will rely on the generated explanations. This chapter explores the techniques most relevant to medicine, their fundamentals, their strengths, their critical weaknesses, and the dangers of interpreting them naively.

13.2.1 LIME: THE LOCAL APPROACH TO THE INCOMPREHENSIBLE

LIME, an acronym for local interpretable model-agnostic explanations ((1)), is an intuitively appealing and widely used technique. Its philosophy is elegantly simple: rather than explaining the entire model globally—a task that would be prohibitively complex for deep architectures—it focuses on explaining individual predictions one by one. To explain why the model produced a certain prediction for a specific patient, LIME generates small synthetic variations in that patient by slightly altering their characteristics and observing how the model's prediction changes. Using these perturbed data, a simple, local model—typically a linear regression—is constructed that approximates the behavior of the complex model in the immediate vicinity of that patient. The features with the greatest weight in this local model explain the original prediction.

LIME's fundamental strength lies in its model agnosticism: it can explain any algorithm, from deep neural networks to random forests or support vector machines (), without needing to know its internal architecture or have access to its parameters. This makes it universally applicable and conceptually accessible. Its implementation is relatively straightforward, and its output—a ranking of features by local importance—is easily understandable to nontechnical users.

However, LIME has critical weaknesses that must be understood. Explanations can be unstable: small variations in the random generation of synthetic data can produce substantially different explanations for the same patient, undermining confidence in any specific explanation. The quality of the explanation depends crucially on how the patient's "neighborhood" is defined and what constitutes "small variation"—a decision that has no single correct answer and whose choice can significantly impact the results. If the neighborhood is too small, the local model may

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

fail to capture the relevant behavior; if it is too large, the linear approximation may be poor. The LIME explains which features changed the prediction in the local context but not why the original model uses those features the way it does.

13.2.2 SHAP: GAME THEORY FOUNDATIONS FOR FAIR ATTRIBUTION

SHAP, which stands for SHapley Additive exPlanations ((2)), represents an approach with a more solid mathematical foundation than LIME does. It is based on Shapley values, a game theory concept that distributes the "payoff" of a cooperative game among players in a way that satisfies certain fairness axioms. In the context of explainability, the "game" is the model's prediction for a specific instance, and the "players" are the patient's features. SHAP calculates the marginal contribution of each feature to the difference between the prediction for that patient and the reference mean prediction for the entire population. The result is an additive decomposition where the sum of the contributions of all the features plus a base value exactly equals the model's prediction.

The main advantage of SHAP is theoretical consistency: unlike LIME, SHAP explanations are stable and satisfy desirable properties such as efficiency (the contributions sum to the full prediction), symmetry (identical features receive equal contributions), and additivity for linear models. There are computationally efficient implementations for specific models, such as decision trees (TreeSHAP) and neural networks (DeepSHAP), which avoid the prohibitive cost of exact calculation of Shapley values. The associated visualizations—summary plots showing the distribution of global importance, force plots illustrating how specific features push the prediction up or down, and dependence plots revealing interactions—are informative and aesthetically clear.

The main disadvantage of SHAP is the computational cost of exact calculations, although modern approximations mitigate this. More fundamentally, SHAP assumes that all features are potentially interchangeable, which may not be realistic in medical data where certain variables are causally related. Interpreting SHAP values requires caution: they represent the average contribution across all possible feature permutations, not necessarily the actual causal mechanism. A high SHAP value for "heart rate" does not imply that changing the heart rate would alter the risk; only the model uses this variable to distinguish risks in this population.

13.2.3 DEPENDENCE PLOTS: FROM GLOBAL TO LOCAL

Partial dependence plots (PDPs) offer a global perspective that complements local explanations. They show how the model's average prediction changes when a specific feature is varied, while all others are held at their marginal values. They are useful for understanding the general relationship between a variable and an outcome, revealing patterns such as nonlinearities or critical thresholds. For example, a PDP may show that the risk of sepsis increases monotonically with increasing lactate concentration up to a certain point, where it stabilizes—valuable information for calibrating clinical alarms.

However, PDPs hide heterogeneity: a variable may have a positive effect in one subgroup and a negative effect in another, and the PDP would show an average effect that does not represent any real patient. This limitation is particularly problematic in medicine, where interactions between variables are common. ICE plots (individual conditional expectations) extend PDPs by displaying dependency curves for each individual patient, revealing variation in effects. If lines from different patients are parallel, the effect is homogeneous; if they cross, there are significant interactions with other variables.

Interpreting ICE plots requires careful sampling: with thousands of patients, the graph becomes unreadable. Selecting representative subgroups or using curve clustering techniques can help. Visualizing heterogeneity is powerful for identifying subpopulations where the model behaves differently, suggesting a need for stratification or specific models. However, it also increases complexity: instead of a simple story about a variable's effect, the clinician must navigate multiple possible stories.

13.2.4 SURROGATE MODELS AND THE ILLUSION OF UNDERSTANDING ()

Decision rules and surrogate trees represent a different family of techniques. They consist of training a simple and fully interpretable model—typically a decision tree with few branches—

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

to mimic the predictions of the complex model across the entire dataset or relevant subsets. The resulting tree can be examined to understand what patterns the original model has learned, presenting rules of the form "IF age > 65 AND creatinine > 1.5 THEN high risk." This form of explanation resonates with rule-based clinical reasoning, making it intuitive for physicians.

The fundamental limitation is that the replacement tree only approximates the behavior of the original model. If the true model is highly nonlinear with complex interactions, the tree needed to approximate it faithfully may become so large and intricate that it loses interpretability. A tree with one hundred nodes is technically interpretable but practically incomprehensible. Furthermore, multiple different trees may approximate the complex model equally well but suggest different explanations, creating a problem of nonuniqueness. The surrogate tree explains which rules approximate the model, not necessarily which rules the model uses internally.

Salience maps for medical imaging deserve special mention given the importance of images in diagnosis. Techniques such as Grad-CAM or layerwise relevance propagation generate heatmaps over medical images, highlighting regions that contributed most to the classification. (2) For a radiologist, seeing which areas of an X-ray triggered a diagnosis of pneumonia can be more useful than an abstract numerical explanation. These techniques are rapidly advancing toward more granular explanations, identifying not only regions but also specific patterns within them.

However, saliency maps can be misleading. They may highlight regions that are technical artifacts or spurious correlations rather than genuine pathological findings. A model that has learned that patient ID tags correlate with a certain diagnosis (due to training bias) will show saliency over the tag, not over relevant anatomy. Validating that highlighted regions have clinical significance requires medical expertise and systematic correlation studies with expert interpretation.

13.2.5 BEYOND THE TECHNIQUE: TOWARD MEDICALLY MEANINGFUL EXPLANATIONS

For developers, the choice of explainability technique depends on multiple interrelated factors: model type, data nature, the audience for the explanations, and the specific purpose. There is no universally superior technique; each has strengths and weaknesses that must be weighed. The most sensible practice is to use multiple complementary techniques, compare their results, and verify the consistency among them and with established clinical knowledge. If LIME and SHAP agree that heart rate is important for this patient, confidence increases; if they disagree, further investigation is needed.

For clinicians, understanding the limitations of these techniques is just as important as understanding their contributions. A SHAP explanation is not "the reason" why the model made its decision; it is a specific mathematical attribution that could be different under another reasonable methodology. A LIME explanation is a local approximation that may fail if the model is highly nonlinear in that region. Salience maps may highlight regions without clinical significance. Explanations should be taken as hypotheses generated by the model, not as revealed truths, and are subject to clinical verification.

The distinction between explainability and causality is a common pitfall. SHAP may indicate that "heart rate" is important for predicting sepsis, but this does not imply that heart rate causes sepsis or that modifying it reduces risk. Explainability reveals correlations that the model has learned, not causal relationships in the real world. Clinicians, accustomed to causal reasoning, must be explicitly aware of this distinction so as not to misinterpret explanations as justifications for interventions.

The future of medical explainability lies in deeper integration with clinical knowledge. Instead of generic explanations in terms of variables, methods are being developed that generate explanations in clinical concepts: "This patient is at high risk because they present a pattern of tachycardia disproportionate to their fever, with elevated lactate, consistent with an early systemic inflammatory response." These explanations, closer to medical reasoning, are more useful and generate greater justified confidence.

Interactive explanation techniques are also emerging, where clinicians can ask "what if creatinine were normal?" and see how the prediction would change. This capacity for counterfactual analysis allows us to explore the robustness of predictions and understand which variables are critical versus which are noise. However, it also requires caution: models lack true causal knowledge, and counterfactual predictions are extrapolations that may not correspond to biological reality.

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

13.3 CURRENT LIMITS OF EXPLAINABLE AI

There is a natural tendency, with any new tool, to attribute powers to it that it does not possess. With explainable artificial intelligence, this is precisely what has happened: a collective illusion has been created that techniques such as SHAP or LIME allow us to open the black box and truly understand why models make their decisions. The reality is more modest and, in a certain sense, more unsettling. The explanations we obtain are not transparent windows into the model's inner workings but rather partial, approximate, potentially misleading interpretations and are always mediated by mathematical choices that are neither unique nor neutral. Understanding the fundamental limits of explainable AI is an ethical and practical imperative: Without this understanding, we fall into a false sense of security that can be just as dangerous as the opacity we are trying to combat.

13.3.1 APPROXIMATION AS A FEATURE, NOT A FLAW

The first limitation, perhaps the most fundamental yet frequently overlooked, is that posthoc explainability techniques do not reveal the model's actual inner workings but rather generate local approximations of its observable behavior. When SHAP assigns specific importance to a variable, it does not show how the model actually uses that variable in its internal layers—calculations that may be opaque even to designers—but rather calculates a contribution on the basis of specific game theory axioms. Another reasonable technique, with different but equally valid mathematical foundations, might attribute different importance to the same variable. The explanation is not "the truth" about the model; it is one perspective among many possible ones, each with its own implicit assumptions.

This multiplicity of valid explanations is uncomfortable but inevitable. The complex model operates in a high-dimensional space with nonlinear interactions that do not admit a simple, unique decomposition. Any explainability technique imposes a simplifying structure—local linearity, additivity, and independence of features—which is an approximation, not a revelation. The danger arises when we forget this condition of approximation and treat the explanations as if they were faithful photographs of the model's interior. They are maps, and like any map, they distort the territory they represent.

13.3.2 INSTABILITY, LACK OF FIDELITY, AND THE VERIFICATION PROBLEM

The second critical limitation is the instability of explanations in the face of small variations. In LIME, the random sampling process used to generate neighborhoods can produce substantially different explanations for the same patient across different runs, even when the model's prediction barely varies. This instability is not an implementation bug but a characteristic of the stochastic approach. For clinical decision-making, an explanation that changes drastically with a different random seed is unreliable, but detecting this instability requires specific tests that are rarely performed in clinical practice.

The lack of fidelity constitutes a third fundamental limitation. An explanation is faithful if it correctly reflects the model's behavior, but measuring this fidelity is a circular problem: we have no independent access to the "truth" about how the model works against which to validate the explanation. Surrogate metrics—the ability of key features to alter predictions when modified—are imperfect. A model can generate plausible explanations, consistent with human expectations, that do not faithfully represent its internal workings. This "misleading plausibility" is particularly dangerous because it generates unwarranted confidence.

The difficulty of capturing complex interactions is a fourth major limitation. Modern models learn highly nonlinear relationships where the effect of variable A depends radically on the values of variables B, C, and D. Techniques that attribute importance to variables individually—including SHAP in its standard form—may miss these essential interactions. A variable may be critical only in a specific combination with another, and univariate explanations suggest moderate importance for both without capturing synergy. Techniques that explicitly model interactions exist but are computationally demanding and conceptually complex, limiting their adoption.

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

13.3.3 CAUSAL TRAP AND VALIDATION OF HUMAN PROBLEMS

The confusion between learned correlation and causal relationships constitutes a fifth limitation with serious practical consequences. An explanation may indicate that "heart rate" is important for predicting sepsis, but this does not imply that heart rate causes sepsis or that modifying it, if possible, would change the risk. The statistical correlation of the training data in the model revealed that a high heart rate may be a marker of an inflammatory response, or it may be a consequence of another unmeasured variable that is the true causal driver. Clinicians trained in causal reasoning may misinterpret such explanations as justifications for interventions targeting highlighted variables, with potentially harmful results if the relationship is not causal.

The difficulty of validating explanations with human experts is the sixth methodological limitation. In theory, expert clinicians should be able to judge whether explanations make medical sense. In practice, this assumes that experts can assess the plausibility of complex combinations of variables that may not correspond to familiar clinical patterns. Furthermore, experts have their own cognitive biases, potentially preferring explanations that align with their prior knowledge even if they are not faithful to the model or reject valid but counterintuitive explanations. Human validation is necessary but not sufficient and can introduce biases of its own.

The tension between explainability and privacy emerges as a seventh limitation in healthcare settings. Detailed explanations, especially when they reveal specific contributions of features to individual predictions, can leak information about training data. A sophisticated attacker could use sequences of explanations to infer patient characteristics from the training dataset or carry out membership attacks. This vulnerability is particularly relevant where privacy is a fundamental value, requiring that explanations be consciously designed to minimize the risk of information leakage, possibly at the expense of granularity.

13.3.4 LACK OF STANDARDIZATION AND COGNITIVE OVERLOAD

The lack of standards constitutes the eighth structural limitation. There is no standard equivalent to IEC 62304 or ISO 14971 for AI explainability. Each developer chooses techniques arbitrarily, presents explanations in proprietary formats, and makes inconsistent claims about their reliability. This lack of standardization hinders comparisons between systems, consistent regulatory evaluation, and the accumulation of evidence on which approaches work best in specific clinical contexts. Regulatory bodies are beginning to address this gap, but the pace of technological development outstrips that of regulation.

Cog tive overload is a frequently overlooked practical limitation. Providing detailed explanations, with multiple variables, interactions, and visualizations, can overwhelm clinicians who are short on time. A physician in the emergency department does not have the ability to study complex SHAP plots while attending to multiple critically ill patients. Explanations must be concise, contextually relevant, and presented at the appropriate moment in the workflow. However,determining what is relevant requires a deep understanding of the decision context, the clinician's specific needs at each moment, and the opportunity costs of care. Designing effective explanations is as much a problem of information engineering as it is of mathematics.

The tenth and most insidious limitation is that explainability itself can create a false sense of security. A physician who receives a plausible explanation may place greater trust in the algorithm, even if the explanation is mathematically misleading or clinically inadequate. This "misplaced trust" can lead to the acceptance of erroneous recommendations without the critical scrutiny that would be applied to a prediction without an explanation. Misunderstood, explainability becomes a veneer of legitimacy for decisions that deserve to be questioned. It is a dangerous paradox: the tool designed to increase transparency can, in the hands of users unaware of its limitations, reduce critical vigilance.

13.3.5 NAVIGATING THE LIMITS: HUMILITY AND MINDFUL DESIGN

For developers, these limits imply that explainability must be approached with epistemic humility and explicit transparency regarding limitations. One cannot promise that explanations reveal the model's inner workings; one must communicate that they are approximations subject to uncertainty. One cannot assume that a single technique is sufficient; multiple approaches must be used, their consistency compared, and the discrepancies explicitly noted. Decision-making responsibility cannot be delegated to the explanation; the clinician remains ultimately

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

responsible, and the explanation is an input for their judgment, not a substitute for it.

For clinicians, these limits require maintaining a critical attitude even in the face of more plausible explanations. An explanation is not revealed truth but a hypothesis about why the model predicted what it did. It must be contrasted with clinical knowledge, the specific patient context, and medical common sense. If the explanation does not make clinical sense, the model is likely failing or the explainability technique is inadequate for this case. Trust should not be placed in the explanation as an object but in the process of informed deliberation that the explanation facilitates.

More robust techniques are being developed through the aggregation of multiple explanations, more accurate methods validated against the model's empirical behavior, and approaches that integrate explicit causal knowledge to generate more medically meaningful explanations. Standards for evaluating and presenting explanations and guidelines for effectively communicating associated uncertainty are being explored. However, while these advances mature, prudence advises against attributing more power to explanations than they actually possess. The black box has not become transparent; we have merely learned to project interpretations onto some of its surfaces that must be treated with constructive skepticism. It is in this humility regarding what we truly understand—rather than in excessive confidence in explainability tools—that the possibility of safe and effective use of medical artificial intelligence lies.

13.4 COMMUNICATING RISK TO CLINICIANS AND PATIENTS

Let's imagine that an algorithm has calculated that a patient has a 15% risk of developing postoperative complications. That number, in isolation, is almost useless for decision-making. Is the risk high or low? Compared to what baseline? What does "15% risk" mean for this specific patient, at this point in their life, with this unique medical history and with these personal values and preferences? The answer lies not in the number itself but in how it is communicated, how it is contextualized, and how it is integrated into the conversation between clinician and patient. Risk communication is an art that combines science, psychology, and empathy, and the arrival of artificial intelligence at the patient's bedside does not simplify this complexity; it amplifies it with new technical and ethical dimensions that we must learn to navigate.

13.4.1 PSYCHOLOGY OF PROBABILITY: WHY NUMBERS ARE NOT ENOUGH

The first fundamental difficulty is that humans, including doctors, do not process probabilities in an intuitively correct way. We tend to think in binary terms: this will happen or it will not happen. A 15% risk may be interpreted as "almost certainly won't happen" by natural optimists or as "quite likely to happen" by those who have had previous negative experiences, depending on their cognitive biases, personality, and emotional context. Doctors, despite their scientific training, are not immune to these biases; they can also misinterpret probabilities if they are not presented carefully. That is why the way information is communicated is just as important as the number itself—and sometimes even more so.

Presentation formats dramatically impact understanding. Numerical probabilities can be expressed as percentages (15%), natural frequencies (15 out of every 100 patients similar to you), or relative comparisons (the risk is 1.5 times the average for your age). Research in health psychology consistently shows that natural frequencies are better understood than percentages are, especially among patients with lower levels of formal education. However, the frequencies can be less accurate when the numbers are small. Comparisons help provide context, but they can be misleading if the reference group is not chosen carefully. Visual formats—pictograms with human figures, bar graphs, and risk thermometers—complement numerical information and facilitate intuitive understanding, although they introduce their own design biases.

13.4.2 CLINICAL CONTEXTUALIZATION: THE MEANING OF NUMBERS IN REAL LIFE

The clinical context radically transforms the meaning of any probability. A 15% risk of postoperative complications may be entirely acceptable if surgery is necessary to remove a malignant tumor with a good prognosis and unacceptable if it is an elective cosmetic procedure. The same number has different ethical implications depending on the situation. Therefore, risk communication cannot be limited to presenting a figure; it must include a discussion of available

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

alternatives, the expected benefits of the intervention, the consequences of inaction, and the patient's values and preferences. The algorithm provides the calculated probability, but the interpretation of that probability is inherently clinical, personal, and situational.

The uncertainty associated with predictions must be explicitly communicated, even if this is technically challenging. A 15% risk calculated by a model has a confidence interval, variability due to model limitations, and uncertainty regarding the representativeness of training data for this specific patient. If the interval is wide—for example, between 5% and 35%—the point estimate may be deceptively precise. While communicating this uncertainty is ethically necessary for genuine informed consent, it is conceptually difficult. Confidence intervals are statistical abstractions that many patients—and not a few physicians—find confusing. Visual simulations showing the distribution of possible outcomes can help, but they do not resolve the tension between epistemic honesty and communicative clarity.

13.4.3 TIMING, CHANNEL, AND HUMAN MEDIATION

Risk communication must take into account the timing and channel of delivery. Communicating risk during a calm consultation with time for questions is not the same as displaying it on a website without clinical supervision. Health care systems are beginning to make algorithmic predictions directly available to patients through mobile apps and online portals without clinical oversight. This poses risks of misinterpretation, anxiety generated by decontextualized information, and hasty decisions based on partial understanding. Risk communication cannot be completely delegated to technology; the human clinical encounter remains irreplaceable for the emotional and comprehensive assimilation of information.

For developers, designing systems that effectively communicate risk requires multidisciplinary collaboration with experts in risk communication, cognitive psychologists, and experienced clinicians. It is not enough to simply display numbers; interfaces must be designed to automatically provide context, allow for "what-if" scenario exploration, and facilitate clinical conversation rather than replace it. We must avoid "presentism," which displays results without adequate explanation, and "alarmism," which exaggerates minor risks to grab attention. Interfaces must be tested with real users of diverse backgrounds, refined for comprehensibility, and evaluated for their impact on real-world decisions, not just in the laboratory.

13.4.4 CLINICAL TRAINING AND SHARED DECISION-MAKING

For clinicians, the advent of algorithmic predictions adds a layer of complexity to an already difficult task. They must understand the predictions technically, explain them to patients in an understandable way, and integrate them into shared decision-making processes. This requires specific training not only in interpreting metrics but also in risk communication techniques, managing emotions in the face of probabilities, and navigating divergent values. It also requires time, which is a scarce resource in contemporary clinical practice. If systems are not designed to facilitate this communication—by providing suggested language, answers to frequently asked questions, and visual support—predictions may be ignored because of information overload or used in an inappropriately simplified manner.

The relationship between risk communication and shared decision-making is intrinsic. The goal is not for the physician to convey information and the patient to passively accept it but for both, with their respective knowledge and perspectives, to reach a joint, informed decision. The algorithm can inform this decision by providing quantified estimates, but it cannot replace the deliberative process. Risk communication should facilitate this dialog, not shut it down with numerical authority. It should open up questions about values and preferences, not close them with apparent objectivity. It should acknowledge uncertainty as a human condition, not hide it behind false precision.

13.4.5 CULTURAL DIVERSITY AND PERSONALIZATION OF COMMUNICATION

Risk communication to patients from different cultures, educational levels, and prior experiences adds a crucial layer of complexity. What works for a patient with scientific training and prior experience with healthcare systems may not work for a patient with low health literacy or a history of distrust toward medical institutions. Systems must be flexible, allowing for different

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

formats and levels of detail on the basis of the patient's explicit or inferred preferences. The personalization of risk communication is an emerging field with the potential to improve equity and effectiveness but also carries the risk of algorithmic biases if inferences about preferences are incorrect.

The cultural dimension is particularly sensitive. Concepts of risk, probability, and personal responsibility vary across cultures. In some cases, direct communication of bad news is considered insensitive; in others, withholding information is considered paternalistic. Risk communication systems must be culturally adaptable and not universally identical. This requires ethnographic research on communication practices in different communities and technical flexibility to accommodate variations.

13.4.6 HORIZONS: TECHNOLOGY IN THE SERVICE OF HUMANITY

The future of risk communication in medicine lies in deeper integration between artificial intelligence and human communication, but with the aim of enhancing, not replacing, the clinical relationship. Systems will be able to generate personalized explanations tailored to the patient's level of understanding, format preferences, and situational context. They will be able to simulate scenarios showing how risk would change with different decisions, facilitating informed deliberation. They will be able to suggest questions the patient might want to ask, on the basis of profiles of similar users, enriching the dialog. However, these technical capabilities will not replace the clinician; they will free them from routine tasks so that they can devote more time to what only humans can do: listen with full attention, empathize with genuine suffering, and accompany patients through existential uncertainty.

In closing, it is worth remembering that behind every calculated risk, there is a specific person. A person with legitimate fears, fragile hopes, and uncertain plans for the future. A person who needs to understand the information but also needs to be heard in their full humanity. A person who deserves for us to speak honestly about what we know and what we do not know, clearly about implications, and with compassion about limitations. Algorithms can calculate risks with increasing precision, but they cannot communicate them with humanity. That task remains our inalienable responsibility, and we must do it well because it depends not only on the quality of individual decisions but also on the trust in the entire medical enterprise and on the very possibility of technologically assisted yet humanly centered care.

Once we have learned to predict, explain, and communicate risk, how do we design studies that demonstrate that all this effort truly improves clinical outcomes and not just technical metrics?

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

REFERENCES

Ribeiro MT, Singh S, Guestrin C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. NAACL-HLT 2016 - 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstration Session [Internet]. February 16, 2016 [cited May 14, 2026];97-101. Available at: https://arxiv.org/pdf/1602.04938

Lundberg SM, Lee SI. A Unified Approach to Interpreting Model Predictions. Adv Neural Inf Process Syst [Internet]. May 22, 2017 [cited May 14, 2026];2017-December:4766-75. Available from: https://arxiv.org/pdf/1705.07874

https://doi.org/10.62486/978-9915-9876-8-2.Ch13

Chapter 14 / Capítulo 14

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch14
Pages: 225-245

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 14. Study Design for Medical Software

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "A clinical trial for a drug and a study for medical software are like open surgery and a laparoscopic procedure: they pursue the same goals, but the methods, risks, and ways of evaluating them are radically different. Confusing them is not only a methodological error but *also* a source of delays, cost overruns, and, worst of all, insufficient evidence to make informed decisions."

Considering the following case, before implementing it, we want to demonstrate that it actually improves patient outcomes. We designed a randomized clinical trial, similar to those used for drugs, where half of the patients received care using the algorithm and the other half did not. However, problems soon arise: Doctors cannot be blinded to the treatment because they know whether they are using the algorithm or not. Patients may be at risk of bias because the same physician treats patients from both groups. The outcome measure, mortality, may be influenced by many factors in addition to the algorithm. The required sample size is enormous because the algorithm is only an aid and not a direct treatment. This scenario illustrates the difficulties of applying traditional evaluation methods to digital interventions and the need for specific approaches for medical software.

The evaluation of medical software poses unique methodological challenges that distinguish it from drug evaluation. Software is not ingested, has no pharmacological effects, and is not metabolized. Its impact is always indirect and mediated by the decisions of the clinicians who use it. The same algorithm can be highly effective in the hands of some doctors and useless in the hands of others, depending on how they integrate it into their practice. It can improve outcomes in some contexts and worsen them in others, depending on the population, resources, and local culture. Evaluating medical software involves evaluating not only a technology but also a complex sociotechnical system.

In this chapter, we explore the fundamentals of the study design for medical software. We will begin with the fundamental differences between clinical trials of drugs and studies of digital interventions. We will continue with prospective and retrospective designs and their advantages and limitations in this context. We address the practical aspect of how to speak with department heads to secure their collaboration, an oftenoverlooked but essential aspect. Finally, we discuss the preparation of applications for ethics committees, a process that can be particularly complex when the intervention is software. This is not an exhaustive guide to research methodology but rather a practical reflection for those who need to demonstrate that their software actually serves a purpose.

14.1 DIFFERENCES FROM DRUG CLINICAL TRIALS

Modern medicine has been built on a methodological pillar that seems unshakable: randomized clinical trials. For decades, this design has proven its worth distinguishing between an effective drug and an inert drug and between a genuine cure and a placebo effect. The logic is seductive in its simplicity: we randomize, compare results, and randomize the rest of the work, balancing known and unknown factors between the groups. However, when the intervention comes not in a vial but on a screen, when it is not a molecule but an algorithm, that methodological elegance collides with a much more complex reality. Medical software does not behave like a drug and does not act like a drug; therefore,it cannot be evaluated exactly like a drug. This assertion is the starting point for developing more appropriate, more honest methodologies.(1,2)

After all, regulators are familiar with them, journals publish them enthusiastically, and scientific prestige seems guaranteed. However, this approach can lead to studies that are brilliant in appearance yet deeply misleading in substance. A trial demonstrating that an algorithm "works" under controlled conditions can tell us very little about whether it will work in the operating room of a regional hospital three times in the morning, with an exhausted doctor and a computer system that crashes every ten minutes. External validity—the ability to generalize results beyond the study's context—becomes the true methodological challenge. To address this issue, we must first understand precisely how digital intervention differs from chemical intervention.

14.1.1 THE HUMAN INTERMEDIARY: WHY THE DOCTOR IS PART OF THE MEDICINE

The first methodological obstacle is so obvious that it is often overlooked: a drug acts directly on the patient's physiology, regardless of who prescribes it. The active ingredient in metoprolol reduces heart rate whether it is prescribed by an expert cardiologist or administered by a resident

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

on their first shift. Medical software, on the other hand, is completely dependent on the human intermediary. A decision-support algorithm does not diagnose the patient; it suggests to the physician that they consider a certain diagnosis. It does not prescribe treatment; it presents options that the clinician can accept, modify, or ignore entirely. This human mediation introduces enormous variability such that traditional trials are not designed to capture or analyze adequately.

Let us imagine a hypothetical study of an early sepsis alert system in the emergency department. The algorithm analyzes data in real time and notifies the physician when it detects a suspicious pattern. In the intervention group, the early diagnosis rate improved dramatically. Does this mean the algorithm is effective? It depends. If we breakdown the data, we might discover that young, digitally native physicians act quickly on the alert and improve their outcomes. However, veteran physicians, who rely more on their clinical experience, may systematically ignore notifications that they consider to be false alarms. The average shows success, but the reality is a mix of success and failure that depends on user characteristics, not the software. Traditional clinical trials, with their focus on intention-to-treat, would lose this crucial heterogeneity in terms of a deceptively positive average.

This variability is not a software design flaw; it is its inherent nature. A drug is a physical entity that can enter the body and trigger predictable biochemical reactions. Software is a tool that amplifies, modifies, or distorts the clinical judgment of the user. Therefore, evaluating software without characterizing its users is similar to evaluating a scalpel without considering the surgeon's skill. The relevant question is not just "does the algorithm work?" but "with what kind of physicians, in what contexts, and with what additional training does it work best?" Classical experimental designs, which seek to eliminate variability between subjects, can be counterproductive here because we precisely need to study that variability to understand where and how to implement the tool.

Furthermore, the relationship between the clinician and the algorithm evolves over time. During the first few weeks of use, the clinician may be cautious and verify each recommendation. With familiarity, they may develop overconfidence, accepting suggestions without initial scrutiny. Alternatively, the opposite may occur: after several false alerts, the doctor may become desensitized and start ignoring the system, a phenomenon known as "alert fatigue." Drug trials assume that the effect is relatively constant over time, except for pharmacokinetics. Software trials must incorporate longitudinal analyses that capture this learning and adaptation curve, which may be more decisive than the technical characteristics of the algorithm itself.

The practical implication is that studies must collect detailed data on users' years of experience, specialties, attitudes toward technology, and workloads during on-call shifts when the system is used. This allows for subgroup analyses that reveal for whom the intervention actually works. y trial showing a null effect in the general population but a large effect in first-year residents remains valuable; it simply tells us that the software needs to be implemented selectively or accompanied by mentoring programs. Without this granularity, we run the risk of discarding useful tools or, worse still, of massively implementing systems that benefit only a minority.

14.1.2 THE BROKEN MIRROR: THE IMPOSSIBILITY OF BLINDING AND THE KNOWLEDGE EFFECT

In pharmacology, a blind trial is the methodological shield against expectation biases. If neither the patient nor the physician knows who is receiving the active treatment, differences in outcomes must be attributed to the drug and not to the psychology of expectations. With medical software, this shield inevitably cracks. It is impossible for a physician to be unaware of whether a decision-support algorithm is available or whether they are practicing with their usual resources. Knowledge of group assignment is complete, and that knowledge inevitably alters behavior in subtle but potentially decisive ways for the interpretation of results.

Consider the phenomenon of compensatory effort. Physicians assigned to the control group, aware that they do not have the "modern" tool at their disposal, may redouble their clinical attention to demonstrate that they do not need the technology. They may consult more literature, order more diagnostic tests, and devote more time to each patient. This extra effort, motivated by the knowledge of being in the placebo group, can level the differences between groups, making the software appear ineffective when in reality it is being masked by a predictable psychological reaction. Conversely, physicians in the intervention group may experience a novelty effect that temporarily improves their performance, regardless of the algorithm's actual quality.

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

There is also the risk of excessive dependence or paralyzing mistrust. Doctors who blindly trust the algorithm may stop exercising their own critical judgment, accepting inappropriate recommendations in atypical cases where the algorithmic pattern fails. Another physician, who is distrustful of the technology, may have the algorithm available but systematically ignore it, or worse still, may feel insecure and make more conservative decisions than usual. These behavioral responses are not methodological artifacts to be eliminated; they are an integral part of the "effect" of the intervention. However, they greatly complicate causal attribution: if outcomes improve, is it because of the algorithm or the greater caution it induces? If they worsen, is it due to software failures or the complacency it generates?

Methodological designs must incorporate strategies to detect and quantify these biases. One approach is to measure intermediate processes, not just final outcomes. If the algorithm recommends a certain treatment, how often does the physician follow it? Does this adherence vary depending on the complexity of the case? Another strategy is to include washout or follow-up periods after discontinuing the intervention to determine whether the effects persist (suggesting genuine learning) or disappear (suggesting a novelty effect). Qualitative interviews can also be used to explore physicians' subjective experiences, revealing mechanisms invisible in quantitative data.

Transparency regarding these limitations is essential. An article reporting on an open-label software trial must explicitly discuss how knowledge of the assignment may have influenced the results. Readers, many of whom are clinicians who intuitively understand these dynamics, will value an honest study about its limitations more than one that pretends to have unattainable methodological purity. Science advances not by denying complexity but by acknowledging it and systematically incorporating it into the analysis.

14.1.3 INVISIBLE CONTAMINATION

One of the most insidious threats to the internal validity of software trials is cross-group contamination. In a pharmacological study, contamination is relatively rare and generally involves administration errors that can be detected and corrected. In a software trial, contamination is almost inevitable if the same professional treats patients from both groups, and it operates through cognitive mechanisms that are impossible to control through the protocol. When a physician uses an algorithm for some patients, the knowledge acquired cannot be selectively compartmentalized; it becomes integrated into their pattern of clinical reasoning and is applied, consciously or unconsciously, to all their patients.

The mechanism is simple yet powerful. Over weeks of use, the physician exposed to the algorithm learns to recognize patterns that have been previously overlooked. A pneumonia diagnosis support system can teach them to evaluate certain subtle radiological findings, to combine clinical data in novel ways, and to consider differential diagnoses they would typically rule out. This learning, which is precisely one of the desirable goals of the technology, becomes a methodological problem when the physician applies these new criteria to patients in the control group. The difference between groups is diluted not because the software is ineffective but because the control group has been "contaminated" by indirect exposure.

The magnitude of this effect depends on the nature of the software. Systems that provide factual information, such as drug databases, are less prone to contamination because the knowledge is not internalized in the same way. However, machine learning algorithms that identify complex patterns, especially in medical images or continuous monitoring data, modify clinicians' perceptions in a lasting way. A radiologist who has used a lung nodule detection system for months will develop an "algorithmic vision" that will persist even when analyzing studies without computer assistance.

Strategies to minimize bias are costly but necessary. The most robust is cluster randomization, where entire hospitals or complete departments are assigned to one group or another. This eliminates cross-exposure at the individual level, although it introduces new complexities: hospitals differ in many ways beyond the intervention, and much larger samples are needed to maintain statistical power. Another approach is a phased design, where all centers eventually receive the intervention but at different randomized times. During the period prior to implementation at each center, they function as controls, and the comparison between centers in different phases allows us to estimate the effect.

It is also possible to measure contamination directly. If we suspect that clinicians are applying knowledge of the algorithm to patients in the control group, we can analyze whether

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

clinical decisions in the control group change over the course of the study, converging toward the algorithmic recommendations. If we find this trend, we can statistically adjust the analysis or, at the very least, discuss it as a limitation. The worst thing a researcher can do is ignore the possibility of contamination, assuming that the groups remain isolated when clinical reality suggests otherwise.

Contamination has an interesting philosophical implication: it challenges the dichotomy between intervention and control. In pharmacology, placebo and treatment are distinct entities. In software, the intervention may be a process of knowledge transformation that, once initiated, cannot be selectively reversed. Perhaps we should think less about evaluating "the software" as a static object and more about evaluating "the introduction of the software" as a dynamic process that irreversibly changes clinical practice. This perspective opens the door to study designs that embrace, rather than combat, the inevitability of learning and adaptation.

14.1.4 TIME RUNS FASTER THAN THE PROTOCOL

Perhaps no difference is characteristic of software because of its ability to change during the study itself. Once manufactured and encapsulated, a drug remains chemically stable for years, allowing for long-term trials that evaluate exactly the same substance from start to finish. Software, on the other hand, is inherently mutable. Developers may detect critical errors during the trial and feel an ethical obligation to correct them. They may implement improvements on the basis of user feedback. They may adapt to changes in operating systems or hospital integrations. Every update, however small it may seem, raises an uncomfortable methodological question: are we evaluating the same intervention as at the start?

This volatility is not a minor flaw; it is a fundamental characteristic that challenges one of the basic assumptions of experimental methodology: the constancy of the intervention. When a software clinical trial lasts twelve months, the version deployed at month six is likely to differ substantially from the initial one. Machine learning algorithms can be retrained with new data, changing their recommendation patterns. User interfaces can be redesigned to improve usability. Workflows can be optimized. From the end-user's perspective, these improvements are desirable. From a methodological perspective, they introduce variability that complicates the interpretation of results.(3)

The question of when to "freeze" the software for evaluation purposes has no single answer. Freezing at the start of the trial ensures temporal comparability but may mean evaluating an obsolete or defective version. Allowing updates maintains clinical relevance but makes it difficult to know which specific version is responsible for the observed results. Some studies adopt a compromise: they freeze the software during specific evaluation periods, allowing updates between these periods and restarting data collection after each major change. Others meticulously document all modifications and perform sensitivity analyses to determine whether the results vary depending on the version used.

The speed of software evolution has implications for external validity that go beyond trial design. Even if we demonstrate that a specific version of the software is effective under controlled conditions, how long will that evidence remain valid? An algorithm trained on 2022 data may become less accurate by 2024 because of changes in epidemiology, diagnostic practices, or the population served. This suggests that software evaluation should not be a one-time event culminating in regulatory approval but rather a continuous process of postmarket surveillance, more akin to pharmacovigilance but adapted to the speed of technological change.

For developers, this means that trial design must incorporate flexibility from the outset. The protocol must clearly specify which types of changes are considered minor (correction of typos in the interface) versus major (modification of the core algorithm), and this st establishes procedures for each case. The exact version used by each participant at every point in the study must be documented. A per-protocol analysis comparing results among users of different versions must be planned, in addition to the conventional intention-to-treat analysis. This added complexity is the price of evaluating an intervention that, unlike a drug, does not remain inert in a vial.

14.1.5 GAME-CHANGING CONTEXTS. WHY IS ONE HOSPITAL NOT THE SAME AS ANOTHER?

Generalizing results has always been a challenge in clinical research, but in medical software, it takes on qualitatively different dimensions. A drug administered at equivalent doses

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

should produce similar biological effects in different populations, with similarities in metabolism and comorbidities assumed. Software, on the other hand, is deeply embedded in complex sociotechnical systems where context is not a neutral backdrop but an active determinant of performance. The same algorithm can thrive in a tertiary care hospital and fail spectacularly in a regional center, not because of technical flaws but because of differences in infrastructure, organizational culture, and workflows.

Consider an electronic prescribing system with drug interaction alerts. In a hospital with robust information systems, technical support staff available around the clock, and a safety culture that values reviewing alerts, the system can significantly reduce medication errors. In a hospital with intermittent connectivity, where alerts take a long time to load and doctors are accustomed to ignoring intrusive pop-ups, the same system may increase frustration without improving safety, or worse, it may foster complacency if clinicians assume that the system will catch all errors. The software's effect is entirely mediated by these contextual characteristics.

This contextual heterogeneity has profound methodological implications. Single-center trials, while easier to conduct, have limited external validity. A study conducted at an elite teaching hospital may overestimate the effectiveness of the software by failing to reflect the resource constraints of most facilities. Even traditional multicenter trials may fail to capture this variability if they select hospitals that are too similar to one another. Deliberately heterogeneous designs are needed that include centers of different sizes, resource levels, and organizational cultures, allowing for moderation analyses that identify in which contexts the intervention works best.

The interaction between software and context is not static; it evolves over time. A system implemented in a hospital may perform poorly initially because of resistance to change, improve as users adapt, and subsequently deteriorate if training is not maintained in the face of staff turnover. Capturing this dynamic trajectory requires longitudinal study designs with multiple measurement points, moving away from the static view of the classic clinical trial. It also requires mixed methods that combine quantitative outcome data with qualitative assessments of contextual factors, using theoretical frameworks such as implementation science to interpret why the same software produces different results in different settings.

For clinicians involved in these studies, understanding this contextual variability is essential for critically interpreting the literature. An article reporting positive results should be read with the following implicit question: "In what type of hospital was this study conducted, and does it resemble mine?" If the answer is no, the evidence remains valuable but should be applied with caution, perhaps if local adaptation or pilot reevaluation is needed before large-scale implementation.

14.1.6 COMPARATORS, METRICS, AND THE NATURE OF SUCCESS

Choosing the right comparator is a critical decision in any clinical trial, but in medical software, there are no obvious answers. In pharmacology, the natural comparator is usually a placebo or the standard of care. In software, what constitutes the equivalent of a placebo? Standard practice without the algorithm? Standard practice plus additional training to match the time spent on the intervention? Another competing software product? Each choice answers a different question and yields conclusions with varying practical implications, and none is inherently correct.

If we compare against standard practice without software, a positive effect demonstrates that the algorithm adds value relative to the status quo. However, it does not tell us whether the benefit is due to the specific technology or simply to the additional care involved in its implementation. If we compare against an equivalent training intervention, we can isolate the algorithm's specific added value. However, this design may underestimate the actual value if, in practice, intensive training without software is never implemented, making the comparison artificially unfavorable to the technology. The choice should be guided by the relevant clinical question: do we want to know if this software is better than nothing or if it is better than other ways of spending the same resources?

The outcome metrics present parallel dilemmas. In pharmaceuticals, the endpoints are typically hard clinical outcomes: mortality, myocardial infarction, and cancer progression. In software, direct effects on these endpoints are rare and difficult to attribute. An algorithm that improves diagnostic accuracy will improve mortality only if subsequent treatment is adequate and timely; if the limitation lies in the availability of treatments, a better diagnosis will not save lives. Therefore, many software trials use the following process endpoints: time to diagnosis, adherence to clinical guidelines, and interobserver variability in test interpretation. These are more

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

sensitive but less compelling to patients and payers, who naturally care about health outcomes, not intermediate processes.

Causal attribution is complicated by the multiplicity of intervening factors. A decision support system may recommend the optimal treatment, but if the patient does not adhere to the medication for social or economic reasons, the clinical outcome will not improve. If the system is implemented in a hospital where surgical wait times are long, diagnosing a surgical condition more quickly will not accelerate the definitive intervention. This does not invalidate the software, but it does require that trials measure not only the direct effect but also the mediators and moderators of the effect. Mediation analyses can decompose the total effect into the portion due to diagnostic improvement versus the portion due to changes in treatment, revealing exactly where the technology is adding value.

The sample size required to detect software effects is typically larger than that for potent drugs because the effects are smaller and variability is greater. This makes traditional trials prohibitively expensive, driving interest in alternative designs. Interrupted timeseries studies, which compare trends before and after implementation, may be more feasible. Before-and-after designs with histori e controls, although vulnerable to selection bias, can provide sufficient evidence for implementation decisions when randomized trials are not feasible. The methodology must serve the clinical question, not the other way around, and sometimes the best methodology is the one that produces reproducible evidence rather than the one that maximizes internal validity.

14.1.7 BEYOND THE CLASSIC TRIAL: TOWARD ADAPTIVE METHODOLOGIES

The fundamental differences between software and drugs should not lead us to methodological discouragement but rather to creative innovation. The scientific community is developing designs specifically tailored to the peculiarities of digital interventions—designs that maintain rigor while embracing the complexity inherent in medical software. These methodologies are not concessions to scientific weakness but are sophisticated responses to complex questions that traditional designs simply cannot adequately address.

Cluster randomization, mentioned earlier, is perhaps the most established adaptation. Assigning entire hospitals or departments rather than individual patients eliminates cross-group contamination and allows for the evaluation of effects at the organizational level. However, this requires special considerations in statistical analysis, as the unit of observation (the patient) is nested within the unit of randomization (the hospital), violating the assumption of independence of observations. Mixed-effects methods or generalized estimation equations are necessary to obtain valid standard errors, and the number of clusters must be sufficiently large to maintain statistical power, typically more than twenty per group.

Staged or phased implementation designs offer an elegant compromise between rigor and feasibility. All the participating centers eventually receive the intervention, but at randomized times. During the period when a center has not yet received the software, it serves as a control for the centers that already have it. This design is politically appealing because no site is permanently disadvantaged, and it allows for the continuous accumulation of evidence, enabling the study to be stopped early if the results are convincing or extended if more data are needed. Staged randomization also facilitates the analysis of implementation effects, examining how outcomes vary according to the time elapsed since introduction.

Hybrid effectiveness-implementation studies represent perhaps the most promising evolution. They recognize that technically excellent software can fail if not implemented properly and that mediocre software can have a disproportionate impact in the right context. These designs simultaneously evaluate clinical outcomes and the contextual factors that influence adoption and effectiveness. They do not settle to ask “does it work?”, but rather investigate “does it work, for whom, under what circumstances, and why?”. This approach requires multidisciplinary teams that bring together clinical experts, implementation scientists, health economists, and software developers, reflecting the inherently interdisciplinary nature of digital medicine.

For software developers, understanding these methodologies is just as important as knowing the technical requirements. An algorithm designed without considering how it will be evaluated may prove impossible to study rigorously. For example, if the system requires extensive customization for each hospital, a multicenter trial becomes methodologically problematic because the intervention differs substantially across sites. If the system generates usage data that cannot be exported for privacy reasons, analyses of adherence and mechanisms are limited. Early collaboration between developers and research methodologists can lead to the design of

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

systems that are not only clinically useful but also scientifically evaluable.

For clinicians, familiarity with these alternative methodologies expands their ability to contribute to the evidence base. Not all physicians have access to the resources of a traditional randomized clinical trial, but many can participate in timeseries studies or implementation evaluations. Their knowledge of the local context is invaluable for interpreting why a software program performs differently in their hospital than in others do and for identifying local modifications that could improve its effectiveness. Research on medical software should not be the exclusive prerogative of large academic centers; adaptive methodology allows meaningful contributions from everyday clinical practice.

The methodological horizon continues to expand. Adaptive designs, where the protocol can be modified during the study on the basis of interim results, are particularly relevant for rapidly evolving software. Platform trials, where multiple interventions are tested on a common infrastructure, allow for more efficient comparisons. The use of electronic health data for real-world effectiveness observational studies, although with well-known limitations, can complement controlled trials by providing evidence on long-term effects in diverse populations. The methodology is alive, evolving alongside the technology it seeks to evaluate.

14.2 PROSPECTIVE VS. RETROSPECTIVE STUDIES

Clinical research has built an entire cosmology around a seemingly simple distinction: do we look forward or backward? Do we plan data collection before events occur, or do we analyze existing information that was accumulated for other purposes? This dichotomy between prospective and retrospective studies is not merely technical; it is philosophical, economic, and temporal. It defines which questions we can answer, with what confidence we can answer them, and at what cost. When the subject of study is medical software, this distinction takes on additional layers of complexity because the object of study changes faster than the study itself and because its effects are so intertwined with the human context that the temporal direction of our gaze determines not only what we see but also how we interpret it.

The choice between the two approaches is not neutral. A prospective study allows us to design data collection with surgical precision, defining exactly what to measure, how to measure it, and when. A retrospective study forces us to accept what the medical record chose to document—with its omissions, inconsistencies, and eloquent silences. However, this prospective precision comes at a price: time. While the prospective study progresses slowly and methodically, the software being evaluated may have been updated three times, changed versions, or withdrawn from the market because of obsolescence or competition. Rigorous evidence runs the risk of reaching the world as a postmortem diagnosis of a product that no longer exists. This tension between methodological rigor and temporal relevance is the central dilemma that every digital health researcher must navigate.

14.2.1 PROSPECTIVE STUDIES AS THE GOLD STANDARD

Prospective studies occupy the highest rung in the hierarchy of medical evidence, and this privileged position is not arbitrary. By planning data collection before the events of interest occur, the researcher exercises almost artisanal control over the process of knowledge generation. They can precisely define the variables they consider relevant, standardize measurement instruments to ensure interobserver reliability, and establish collection protocols that minimize unwanted variability. In the context of medical software, this meticulous planning allows us to go beyond the final results to explore intermediate mechanisms: how clinicians interact with the interface, how often they follow the algorithm's recommendations, and whether there are unanticipated effects that only direct observation can reveal.

Considering the following case,a prospective design allows us to install specific sensors to capture high-frequency physiological data, train nursing staff in the standardized use of the system, and define composite endpoints that combine mortality with intermediate events such as transfers to intensive care units. We can collect process measures, such as the time between the alert and the medical intervention, which help us understand why the system works or fails. This wealth of information is impossible to obtain retrospectively, where we depend on what the clinical record deemed worth noting, typically at discrete moments and often insufficient to capture the dynamics of acute deterioration.

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

The ability to establish causal relationships with greater confidence is perhaps the greatest strength of prospective studies. By controlling the assignment of the intervention—whether through randomization or careful quasiexperimental designs—we can argue more strongly that the observed differences are due to the software and not to confounding factors. If we randomize hospitals to implement the algorithm in a phased manner, we can compare departments that are similar in both observable and unobservable characteristics, thereby strengthening causal inference. This methodological strength is especially crucial when the intervention is complex and its effect is mediated by multiple contextual factors that would be impossible to statistically adjust for in a retrospective analysis.

However, prospective studies have their drawbacks. The financial cost of a well-designed prospective study can be prohibitive for startups or academic researchers without industry funding. The resources required to recruit participants, monitor data quality, and maintain protocol integrity over months or years scale rapidly. Furthermore, the time required from study conception to publication of results may be incompatible with the pace of the technology market. A machine learning algorithm trained on 2022 data may require retraining in 2023 because of changes in epidemiology or clinical practices, making the results of the prospective study—however rigorous—applicable only to a historical version of the software.

External validity represents another persistent concern. Prospective studies, in their quest for methodological control, often implement software under conditions that are more ideal than those in real-world practice are. Technical support is more intensive, staff training is more comprehensive, and integration with existing systems is more carefully planned. When the same software is subsequently deployed in hospitals without these research resources, the results may be less favorable. This gap between efficacy (what the software can achieve under optimal conditions) and effectiveness (what it achieves in real-world practice) is particularly pronounced in digital interventions, where implementation is as critical as the technology itself.

14.2.2 THE RETROSPECTIVE VIEW

Retrospective studies offer an irresistible temptation: the promise of quick and inexpensive answers by tapping into the hidden treasures in existing health records. Instead of waiting years for events to occur, the researcher can analyze thousands of patients in a matter of months, explore associations and generate hypotheses at a speed that prospective designs cannot match. For medical software, this approach is especially valuable in the early stages of development, when it is necessary to quickly evaluate an algorithm's technical performance, compare it with historical practice, or identify subgroups where it might be particularly beneficial or harmful.

The main strength of retrospective studies is their ability to reflect real-world clinical practice in terms of all its variability and imperfection. Electronic health record data capture the authentic noise of everyday medicine: constant interruptions, decisions made with incomplete information, and deviations from protocols for clinical or logistical reasons. An algorithm that performs well on these "dirty" data is more likely to work in the real world than one validated solely on clean, curated databases. Furthermore, compared with prospective studies, retrospective studies can include much larger and more diverse populations, allowing for an assessment of the generalizability of the software and the detection of rare effects that emerge only in massive samples.

Let us consider a concrete example. A team developed an algorithm to predict the risk of readmission in patients with heart failure. A retrospective study analyzed five years of records from ten hospitals, including tens of thousands of patients with various demographic characteristics and comorbidities. It could compare the algorithm's predictions with actual discharge decisions, identifying cases where the algorithm would have suggested a prolonged hospitalization that turned out to be unnecessary, or vice versa. This large-scale evaluation provides invaluable insights into the software's performance under real-world conditions—insights that would be prohibitively expensive to obtain prospectively.

However, retrospectivity involves systemic vulnerabilities that must be honestly acknowledged. Data quality is the Achilles' heel of these studies. Clinical records were designed for patient care, not for research, and reflect the priorities, habits, and limitations of those who completed them. Diagnoses may be coded inconsist, using different classification systems or levels of specificity. Important variables may be systematically missing, such as the patient's socioeconomic status, their actual adherence to medication, and their treatment preferences. Timestamps may be documented approximately, with events grouped by date without the hourly granularity required for certain analyses.

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

Selection and information biases are particularly insidious in retrospective studies of medical software. Selection bias arises when the included patients do not adequately represent the target population. For example, if we evaluate a sepsis detection algorithm using only patients who were ultimately diagnosed with sepsis, we are ignoring the false positives it would have generated in the general emergency department population. Information bias occurs when the way data are recorded is related to the exposure or outcome. A physician using a diagnostic support algorithm may record certain symptoms in greater detail precisely because the algorithm suggested them, creating an artificial appearance of agreement between prediction and reality.

Establishing causal relationships from retrospective data requires considerable statistical sophistication. Techniques such as propensity score matching, discontinuity regression, or instrumental variable models attempt to simulate the randomization we lacked, but all rely on assumptions that are rarely perfectly met. The researcher must assume that they have correctly identified and measured all important confounding factors—a bold assumption in the complex clinical context. Therefore, retrospective studies are excellent for generating hypotheses and evaluating technical performance but are limited in demonstrating that the software actually improves health outcomes in a causal manner.

14.2.3 HYBRID AND ALTERNATIVE DESIGNS

The rigid dichotomy between prospective and retrospective has given way to more flexible designs that combine elements of both approaches, seeking to maximize strengths while minimizing weaknesses. These hybrid designs are particularly relevant for medical software, where the need for rigorous evidence competes with time-to-market pressures and where the intervention itself may be modified during the study. Understanding these methodological alternatives expands the researcher's repertoire and allows us to address questions that no single design could adequately answer.

Before-and-after studies with historical controls represent a widely used intermediate category. In this design, we implement the software at a specific point in time and compare the results with those of the same service prior to implementation or with those of a similar service that has not received the intervention. This approach is more feasible than a randomized trial because it does not require randomization or simultaneous parallel groups, but it introduces specific vulnerabilities. Temporal changes in the quality of care, the epidemiology of diseases, or the composition of the served population can be confounded by the effect of the software. If we implement a COVID-19 diagnostic algorithm in 2021 and compare it with that in 2020, we compare periods that are radically different in terms of medical knowledge and healthcare pressure.

Discontinuous timeseries designs offer an additional level of sophistication that makes them particularly appealing for evaluations at the service or hospital level. Rather than comparing simple pre- and postimplementation averages, these designs analyze the trend of the indicator of interest over time, looking for changes in the level or slope of the trend coinciding with the implementation of the software. If the medication error rate gradually decreased before the algorithm and continued to decline at the same rate afterward, the software likely had no additional impact. However, if there is a sharp decrease at the time of implementation or if the rate of improvement accelerates, we have more robust evidence of a causal effect. This design controls for preexisting trends and can detect effects that simple before-and-after comparisons would overlook.

Staged randomization, mentioned in the previous section, represents another elegant synthesis. All participating centers eventually receive the intervention, eliminating ethical objections regarding the deprivation of potential benefits, but they do so at randomized times. During the period when some centers already have software and others do not, there is a rigorous prospective comparison. As time progresses, more centers enter the intervention group, leading to the accumulation of ongoing evidence that allows the study to be stopped if the results are convincing or expanded if more precision is needed. This design is particularly useful when the intervention is politically popular and it is difficult to retain controls or when contamination between groups is inevitable if they coexist in the same time and space.

Hybrid effectiveness-implementation designs go a step further, recognizing that technically excellent software can fail if not implemented properly and that mediocre software can have a disproportionate impact in the right context. These studies simultaneously evaluate clinical outcomes and the contextual factors that influence adoption: organizational culture, local leadership, available resources, and barriers perceived by users or im. They do not settle to ask

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

whether the software works but rather investigate for whom it works, under what circumstances, and why. This approach requires mixed methods that combine the quantification of outcomes with the qualitative assessment of processes, but it produces evidence that is more accessible to those who must decide on the adoption and scaling of the technology.

Understanding these alternative designs is crucial for the strategic planning of the evidence program. In the early phases, retrospective studies can validate technical performance and justify investment in more costly studies. Timeseries or staggeredrandomization designs can provide sufficient evidence of effectiveness for initial regulatory approval, while broader prospective studies are planned for the postmarketing phase. The key is to align the methodological design with the specific question that needs to be answered at each stage of the product lifecycle, recognizing that different audiences (regulators, payers, clinicians) may require different types of evidence.

14.2.4 PRACTICAL IMPLICATIONS: SELECTION, COMBINATION, AND COMMUNICATION

The choice between prospective and retrospective approaches, or between their hybrid variants, should not be an abstract decision based purely on methodology but rather a pragmatic strategy aligned with the study's objectives, available resources, and the software's stage of development. For teams developing medical technology, this choice has direct consequences for commercial viability, regulatory acceptance, and, ultimately, patient health. A poorly designed evidence strategy can doom technically excellent software to failure, whereas a smart strategy can accelerate patient access to beneficial tools.

In the early stages of development, when the algorithm is still being refined, retrospective studies are appropriate and efficient. They allow for rapid iteration, testing different versions of the software against historical data, identifying edge cases where the algorithm fails, and adjusting parameters without putting patients at risk. This internal validation phase is comparable to preclinical drug trials: it is necessary but insufficient. A common pitfall is to remain in this phase, publishing articles on the algorithm's excellent performance in curated databases, without moving toward evaluations under real-world clinical conditions.

The transition to prospective studies must be carefully planned, typically when the software has reached a level of technical maturity that justifies the investment and when clinical added value needs to be demonstrated for regulatory approval or reimbursement. This transition does not need to be abrupt; hybrid designs can serve as a bridge. A prospective study with a systematic collection of primary data can be complemented by retrospective registry analyses to expand the sample size or evaluate long-term effects. The integration of research platforms into electronic health records systems facilitates this combination, allowing prospectively collected data to be fed into registries that can be analyzed retrospectively in the future.

For clinicians who participate in these studies—whether as principal investigators, collaborators, or simply as users whose work generates the data—understanding these methodological distinctions empowers their critical engagement. They can assess whether the study design is appropriate for the research question, identify potential sources of bias that researchers may have overlooked, and contribute their tacit knowledge of real-world clinical practice. A physician who knows that emergency department records systematically underestimate alcohol consumption can point out this limitation in a retrospective study. A clinician familiar with team dynamics in their department can suggest that the implementation of the software coincided with a change in the department head, confounding the effect of the intervention.

The communication of results must be equally sophisticated. Researchers have a responsibility to be transparent about the limitations of their design, avoiding the temptation to present a retrospective study as if it were prospective or to extrapolate causal conclusions from observational associations. Readers, many of whom are busy clinicians who rely on summaries and abstracts, must be educated to always ask, "Was this a prospective or retrospective study? What type of data was used? How were confounding factors handled?" The quality of evidence in digital medicine depends as much on the sophistication of those who produce it as on the critical literacy of those who consume it.

Relationships with regulators add another layer of complexity. Drug and medical device agencies are developing specific guidelines for software evaluation, and these guidelines often distinguish between evidence requirements on the basis of the device's risk. With respect to low-risk software, well-designed retrospective studies may be sufficient. For high-risk devices

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

that support critical decisions, prospective studies are typically needed. Understanding these regulatory thresholds from the earliest stages of development allows for the design of an evidence program that efficiently moves toward approval, avoiding the costly mistake of having to repeat studies because the initial design did not meet regulatory standards.

The future points toward increasingly close integration between these two worlds. Electronic health records are becoming more standardized and comprehensive, narrowing the quality gap between retrospective and prospective data. Federated machine learning platforms allow algorithms to be trained and validated across multiple institutions without centralizing sensitive data, facilitating large-scale retrospective studies that maintain privacy. Postmarket continuous surveillance systems, where the software itself collects data on its performance in real-world use, blur the line between research and clinical practice. In this emerging landscape, the distinction between prospective and retrospective data may become less relevant than the distinction between high-quality, contextualized data and poor or decontextualized data.

One persistent question must be echoing: once we have generated this evidence, whether looking forward or backward, how do we translate it into action? How do we convince a skeptical department head that it is worth altering their workflows to incorporate an algorithm? How do we navigate the complex ethical and regulatory procedures that protect patients but often seem to hinder innovation? Methodology tells us what works, but implementation confronts us with the question of how to make it work in the real world, with its organizational resistance, limited budgets, and imperfect humans.

14.3 HOW TO TALK TO A DEPARTMENT HEAD TO COLLABORATE

In the world of clinical research with medical software, there is an invisible threshold that separates projects that thrive from those that sink into administrative oblivion. It is not the threshold of ethical approval, however, that is crucial. It is not that of funding, although without it nothing moves forward. It is the moment when someone—usually a young, enthusiastic researcher—sits down with a department head to ask for something seemingly simple: permission to conduct a study within their department. That conversation, which may last twenty minutes or an hour, determines whether a project secures the most powerful ally it can have or whether it becomes just another folder piled on a desk already overflowing with unfulfilled good intentions. The difference between these two outcomes rarely depends on the technical quality of the algorithm but almost always on the human ability of the person seeking access to connect with the real concerns of the person holding the key.

Hospital department heads face a reality of converging pressures that those coming from the software industry often underestimate. They are not merely dealing with bureaucratic hurdles or passively receiving academic proposals; they are managers of extremely scarce resources, ultimately responsible for the quality of care, coordinators of complex teams, trainers of the next generation of specialists, and often clinicians who maintain their own practice while working grueling shifts. In this context, a research project is not a brilliant opportunity that lights up its desk; it is an additional burden that competes with life-threatening emergencies, endless waiting lists, residents who need supervision, and quality targets that must be met quarter after quarter. Understanding this ecosystem of pressure is the first step toward designing a conversation that has any chance of success.

14.3.1 FROM THE OTHER SIDE OF THE TABLE: UNDERSTANDING THE CLINICAL MANAGER

The most common mistake when a department head is approaching is to treat them as if they were fellow researchers who are interested in the methodological nuances of the study design. Nothing could be further from the truth. The department head is, above all, a manager whose success is measured by indicators of productivity, quality, and team stability. Their primary concern is not whether the algorithm uses transformer architectures or convolutional networks but whether the study's implementation will make their department function better or worse during the coming months. This fundamental understanding must shape every aspect of the preparation and execution of the initial conversation.

To connect with these priorities, it is necessary to translate the project into the language of its tangible benefits for the department. It is not enough to say that the algorithm has a sensitivity of ninety-five percent; one must explain that this means detecting three more cases of sepsis per

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

week, which could translate into two fewer ICU admissions and a reduction in the average length of stay. It is not enough to mention that the study will include 200 patients; you must highlight that this represents coauthorship on a high-impact publication for the department, visibility at international conferences, and priority access to technology that other centers do not yet have. Department heads value academic prestige, but they value it even more when it is clearly linked to improvements in daily clinical practice and public recognition of their team.

Concerns about staff are particularly sensitive. Department heads live with the constant fear of their physicians burning out, the turnover of experienced nurses, and the difficulty in retaining talent. Any project they perceive as an additional source of stress for a team already at its limit will be rejected, regardless of its scientific merit. Therefore, the proposal must explicitly address how staff will be protected: by hiring dedicated support staff for the study, by integrating data collection into existing workflows without duplicating efforts, and by scheduling data collection and collection times that avoid peak hours of patient care. This study demonstrates that the team's well-being builds more trust than any promise of an impending technological revolution does.

It is also crucial to understand the internal power dynamics of departments. The department head must answer hospital management, quality committees, and patient associations. A project that could be interpreted as experimental, risky, or likely to generate complaints represents a political burden that many prefer to avoid. Therefore, the proposal must include elements of security and risk management: clear protocols for adverse situations, data security committees, and patient communication plans that minimize misunderstandings. The department head needs to feel that they can defend the project to their superiors with solid arguments regarding security and potential benefits.

The timing of the approach is also strategic. Approaching the department head immediately after a tense meeting with management or during a period of peak clinical pressure is a surefire way to reject the proposal because of burnout. Conversely, identifying moments of relative calm, or of particular interest in innovation (for example, following an audit visit that highlighted areas for improvement), can significantly increase the chances of success. This contextual intelligence requires prior networking and informal conversations with department members who can provide insight into the head's current mood and priorities.

14.3.2 ART OF TRANSLATION

The language gap between software developers and clinicians is deep and mutually bewildering. For those coming from the tech world, it is natural to discuss microservice architectures, RESTful APIs, and performance metrics such as the AUC or F1score. For the clinician, this vocabulary is white noise that obscures the essential message: does this help my patients or not? The inability to translate technical concepts into tangible clinical impact is among the main causes of failure in these conversations, and overcoming it requires a deliberate effort at empathy and clarification.

Effective translation is not trivial simplification; it is conceptual reconstruction. It is not about omitting technical complexity but about reconnecting it to outcomes that the clinician values. Instead of saying "our algorithm has a ninety-two percent accuracy rate in detecting intracranial hemorrhages," one could say "this means that out of every hundred scans the algorithm flags as urgent, ninety-two actually require immediate intervention, reducing the false alarms that currently overwhelm your radiology department." Instead of mentioning "training with fifty thousand MRI images," one can explain, "the system has learned from more cases than any radiologist would see in their entire career, identifying subtle patterns that human experience might overlook during night shifts." Technical metrics become clinical benefits; data volume transforms into superior accumulated expertise.

This translation must extend to the entire project narrative. The methodological design, which for the researcher is the heart of the study, must be presented as a plan to minimize disruptions. Data collection, which for the methodologist is crucial for validity, must be explained as something that will occur in the background, without interfering with patient care. Quality controls, which ensure scientific rigor, must be framed as safeguards for the safety of the service's patients. Every technical element must pass through the filter of immediate clinical utility.

Preparing for this translation requires preliminary work. The researcher must study the target department, read its recent publications, understand its prioritized lines of research, and be familiar with the clinical problems most frequently mentioned in meetings or internal

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

communications. This field research allows for personalizing the message: if the department has published on emergency department workload, emphasize how the algorithm reduces decision time; if they have highlighted the importance of training, underscore how the project will include training sessions for residents. A generic message, however technically brilliant, will compete with dozens of similar proposals; a personalized message demonstrates respect and genuine commitment.

It is also important to anticipate the questions the department head will ask, which are rarely technical. “How much time will this take away from my doctors?” “What happens if the algorithm makes a mistake and we don’t notice?” “How will this affect our relationship with patients?” “Who takes responsibility if there is a problem?” These questions reveal concerns about workflow, safety, clinical ethics, and legal liability. Having prepared, honest, and detailed answers makes the difference between a professional interlocutor and a seller of empty promises. It is particularly important not to downplay potential risks; acknowledging them and explaining how they are mitigated builds more trust than implicitly denying them.

14.3.3 THE ECONOMICS OF THE EXCHANGE

A successful research project cannot be an extractive relationship where the clinical service provides labor, patients, and data, while external researchers take the publications, patents, and commercial benefits. This perception of inequality, which is too common in the history of biomedical research, is poisons for sustainable collaboration. Building a genuinely reciprocal relationship requires explicitly designing what the department receives in exchange for its participation and ensuring that these benefits are valued and tangible.

Reciprocity can take many forms, and the key is to identify which forms align with the department’s specific priorities. For some, coauthorship on high-impact publications is a crucial academic asset for departmental accreditation or the promotion of team members. For others, priority access to developed software—potentially before its general commercial release—represents a tangible competitive advantage. Some departments value the specialized training the project can offer their residents, exposing them to research methodologies or emerging technologies. Others may need funding to hire additional staff to alleviate the clinical workload or IT infrastructure to enhance their capabilities beyond the specific project.

It is important that these considerations are not vague future promises but rather concrete commitments documented from the outset. A memorandum of understanding specifies who will be a coauthor on which publications, what access the department will have to the software poststudy, what training will be provided and for how long, serves to align expectations and avoid future conflicts. This document does not need to be a complex legal contract, but it must be clear and signed by both parties. Formalizing the agreement demonstrates seriousness and protects both sides from misunderstandings that may arise months later, when the original participants in the conversation are no longer available.

Reciprocity also operates on an emotional and relational level. The department head and their team must feel respected as intelligent and valuable partners, not as mere executors of a protocol designed elsewhere. This involves genuinely involving them in the study design, soliciting their opinion on the feasibility of procedures, adapting the protocol to their suggestions, and publicly acknowledging their contribution in presentations and publications. A researcher who treats the clinician as an equal collaborator fosters loyalty that extends beyond the individual project, opening doors for future collaborations and creating advocates for the work within the clinical community.

It is also wise to consider institutional reciprocity, not just individual reciprocity. Agreement with the department head should include provisions that benefit the hospital: recognition in press releases, participation in outreach events, and networking opportunities with other centers participating in the study. These institutional benefits strengthen the department head’s support with hospital management and contribute to an organizational culture favorable to research that extends beyond the specific project. A hospital that perceives research as providing prestige and resources is more willing to facilitate future studies.

14.3.4 THE DANCE OF FLEXIBILITY: ADAPTING TO MOVE FORWARD

Approaching the conversation with a rigid, unchangeable protocol is a strategy guaranteed

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

to fail. Department heads know the nuances of their own reality better than any external researcher does, and their suggestions for adaptation are not capricious obstacles but are invaluable contributions of local insight. Methodological flexibility is not a scientific weakness; it is intelligent pragmatism that increases the study's feasibility and often improves its quality. A researcher who actively listens and adapts builds trust; one who imposes resistance.

This flexibility can manifest in multiple dimensions. Temporally, the department head may suggest that certain study procedures be conducted at specific times of the day or week to minimize conflicts with peak clinical activity. Methodologically, they may propose excluding certain patient subgroups for ethical or logistical reasons that the researcher had not considered. Technically, they may point out limitations in the IT infrastructure that require modifications to the software integration. In all these cases, the appropriate response is not to defend the original protocol at all costs but to assess whether the proposed adaptation seriously compromises the scientific objectives or, conversely, improves external validity by aligning with actual practice.

Anticipating objections is an advanced skill that distinguishes experienced researchers from novices. Instead of passively waiting for the department head to raise issues, the prepared researcher brings them up first, demonstrating that they have considered the project from the department's perspective. "I know that one of your concerns will be the additional burden on residents; that is why we have budgeted for the hiring of a dedicated research coordinator to handle all administrative tasks." "I understand that the emergency department is extremely busy on weekends; the protocol allows for excluding those days from intensive data collection." This foresight demonstrates empathy and professionalism and shifts the conversation from a defensive negotiation to a constructive collaboration.

However, flexibility has ethical and scientific limits that must not be crossed. Adaptations that compromise patient safety, introduce unacceptable systematic biases, or violate fundamental regulatory principles must be politely but firmly rejected. The key is to explain why certain elements are nonnegotiable while offering creative alternatives that address the underlying concern without sacrificing the study's integrity. For example, if the head of a department wishes to exclude the most severely ill patients out of fear of complications, one can explain that this would skew the results but offer a specific intensive monitoring protocol for that subgroup that ensures safety without excluding them from the analysis.

Documenting agreed-upon adjustments is essential. Following the conversation, a written summary outlining the agreed-upon points, modifications to the original protocol, mutual commitments, and specific next steps helps prevent misunderstandings and provides a reference as the project progresses and individual memories fade. This document should be sent promptly, ideally the day after the meeting, along with a personalized thank-you email that reinforces the rapport established. Professionalism in these administrative details is interpreted by the department head as an indicator of how the project as a whole will be managed.

14.3.5 BUILDING TRUST: CREDIBILITY, PATIENCE, AND PERSISTENCE

Department heads have witnessed too many projects that promised much and delivered little, researchers who disappeared after obtaining the data and technologies that never materialized. This history of broken promises has fostered a healthy skepticism that the researcher must overcome through credible signs of seriousness and the ability to execute. Credibility is not declared; it is built through tangible evidence that this project is different and that the person proposing it has the resources, experience, and commitment to see it through to a successful conclusion.

Signs of credibility include previous results from similar projects, preferably documented in publications or presentations that the department head can consult. Support from other prestigious departments or institutions that have already approved the project serves as social validation: if the general hospital has accepted it, perhaps it is worth considering. secured funding, demonstrable through documentation from the funding agency or the sponsoring company's contract, eliminates concerns that the project will run out of resources halfway through. A visible multidisciplinary team, with clear roles and documented experience, demonstrates the capacity to address the project's various technical, clinical, and regulatory dimensions.

Internal endorsement is particularly powerful. If someone from within the department itself—someone respected by the head—can attest to having participated in the project's design and can vouch for its viability, barriers to mistrust are significantly broken down. For this reason, prior networking with clinicians in the department, building informal relationships, and

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

participating in scientific meetings where preliminary ideas are presented are investments that greatly facilitate formal conversation with the head. An internal clinical "champion" who opens the door and accompanies the presentation is an invaluable asset that justifies the time spent cultivating these relationships.

Patience is an essential virtue in this process. Department heads have schedules filled with unforeseen emergencies, and a research proposal, no matter how interesting, is rarely the top priority. It may take weeks to receive a response, and during that time, the researcher must maintain polite and regular contact that keeps interest alive without coming across as a nuisance. A reminder every two or three weeks, with some relevant progress (new funding secured, ethics committee approval, recent related publication), demonstrates that the project is moving forward and keeps the conversation on the recipient's mind. Patient persistence trumps aggressive pressure.

When the response finally arrives, whether positive or negative, the researcher's reaction makes all the difference for true interactions. An acceptance should be celebrated with genuine gratitude and immediately followed by the specific steps agreed upon, demonstrating efficiency that validates the trust placed in you. A rejection should be accepted professionally, requesting feedback on what could have been done differently and leaving the door open for future collaborations when circumstances change. Burning bridges with a frustrated department head close not only that project but also potentially an entire network of contacts within that institution.

For software developers, mastering these communication and relationship management skills is just as crucial as any technical competency. A perfect algorithm that never makes it out of the lab because the necessary clinical partnerships were not established is, from the patient's perspective, equivalent to not existing. Investing time in understanding the hospital environment, in cultivating authentic human relationships, and in learning the language and priorities of those who lead the departments is among the highest-return investments a healthcare innovation team can make. Technology does not spread on its own merit; it spreads because specific people decide to trust other specific people to implement it.

14.4 INSTITUTIONAL REVIEW BOARDS (IRBS): HOW TO PREPARE AN APPLICATION

For those in the world of technology development, accustomed to the breakneck speed of iterative cycles and the "fail fast, learn fast" culture, the first encounter with a Drug Research Ethics Committee (CEIm) can feel like a journey through alien territory. Deadlines measured in weeks rather than hours, seemingly endless forms, questions that challenge assumptions the team takes for granted—all of these conspires to generate frustration and, at times, disdain. The temptation to treat ethical approval as a bureaucratic formality to be overcome with the least effort is understandable, but it represents a catastrophic mistake that can paralyze projects for months or, worse still, expose patients to avoidable risks. Understanding that behind every committee requirement lies a history of human protection—of past abuses that society decided not to repeat—transforms the experience from an obstacle into an opportunity for professional growth. (4)

Ethics committees are not abstract entities designed to stifle innovation; they are the institutional embodiment of principles forged after decades of unethical research. The Nuremberg Code emerged from the trials of doctors who experimented on prisoners without consent; the Declaration of Helsinki consolidated international standards following decades of abuses in developing countries; and the Belmont Report established the principles of respect for persons, beneficence, and justice that continue to guide contemporary research. When a committee requires detailed informed consent, it is not being nitpicky; it is ensuring that no person is reduced to a mere means for the ends of others. When it requests a rigorous analysis of risks and benefits, it does not create unnecessary obstacles; it is compelling researchers to confront the real consequences of their work on human lives. This historical genealogy must inform every step of the preparation of an ethics application. (5,6,7)

14.4.1 THE ART OF PREPARATION: TEN RULES FOR A SUCCESSFUL APPLICATION

Start the process early, when the study is still in the conceptual design phase, rather than when the software has already been developed and all that is left is the "go-ahead" to test it. Ethics committees, with their accumulated expertise in methodology and participant protection,

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

can fer suggestions that modify fundamental aspects of the project: the definition of inclusion and exclusion criteria, the selection of primary outcome measures, and consent procedures. If these recommendations arrive when the protocol is already set in stone in code and procedures, the changes may be technically impossible or financially ruinous. If they arrive during the design phase, they can be organically integrated, improving the study without disruption. Many committees offer nonbinding preliminary consultations that allow for validation of the general direction before investing resources in full development; ignoring this opportunity constitutes methodological negligence.

It seems obvious but is systematically overlooked: read and follow the specific instructions of the committee to which the application is addressed. Each IRB has its own forms, document checklists, submission deadlines, and scheduled meetings. Submitting a generic application downloaded from the internet—or worse, adapted from another committee without verifying specific local requirements—guarantees unnecessary delays and gives the impression of a lack of professionalism. Taking the time to familiarize yourself with the committee's website, download the updated documentation, and understand the process flow from receipt to resolution is an investment that pays off in subsequent efficiency. If the committee offers informational sessions or has support staff available for preliminary consultations, the use of these resources demonstrates research maturity and can prevent costly errors.(4)

It requires a conceptual translation of the project into the language of medical ethics. For the development team, the focus may be on the model architecture, hyperparameter optimization, ROC curves, and confusion matrices. For the committee, these are technical abstractions that only acquire ethical relevance when connected to concrete human impacts. The application must answer fundamental questions with absolute clarity: What specific risks——does each individual participant face? How are these risks minimized through study design or safety procedures? What benefits are anticipated, who will receive them, and with what level of evidence? How is the confidentiality of sensitive data guaranteed? How will informed consent be obtained and documented? An application that does not explicitly address these ethical dimensions forces the committee to request clarifications, initiating back-and-forth cycles that consume valuable weeks.

It is essential to anticipate the difficult questions the committee will inevitably raise regarding sensitive issues. IRBs are particularly vigilant with respect to the inclusion of vulnerable people who cannot protect themselves, such as minors; individuals with significant cognitive disabilities; patients in medical emergencies where informed consent seems impossible;and pregnant women, prisoners, and individuals in situations of extreme economic dependence. If the study involves any of these groups, the application must contain a solid ethical justification for why their inclusion is scientifically necessary and cannot be replaced by less vulnerable populations, along with a detailed description of special protective measures. Equally sensitive is the use of retrospective data without specific consent, where it must be argued why obtaining consent is impractical and how the irreversibility of anonymization is guaranteed.(4)

Advocate for radical transparency regarding the study's limitations. No research design is perfect, and members of ethics committees—generally experienced researchers—are well aware of this. Attempting to conceal methodological limitations from the ethics committee, minimize potential risks, or exaggerate anticipated benefits is often counterproductive. If the committee discovers inconsistencies or omissions during its review, trust in the research team erodes, and the process drags on indefinitely. It is preferable to be proactively honest about the limitations, explain why they were accepted in the design, and argue why, despite them, the study maintains a favorable benefit–risk ratio and provides sufficient scientific or social value to justify exposing participants to residual risks.

Focus on the exceptional quality of the patient information and informed consent documents. These texts are the primary ethical interface between the study and potential participants, and committees scrutinize them with almost forensic thoroughness. They must be written in language accessible to people without medical or technical training, avoiding both incomprehensible specialized jargon and paternalism that underestimates patients' comprehension ability. They must clearly explain what participation entails in practice; what specific procedures it involves; how much time it requires; what physical, psychological, or social risks it entails; what direct or indirect benefits the participant can expect; the absolute right to withdraw at any time without penalty or impact on their regular medical care; and how their personal data will be handled. Dedicating the iterative time to refine these documents, testing them with focus groups of potential patients, is among the highest-return investments in the entire ethical approval process.

It addresses data management and privacy in an era of strict regulation and massive media

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

leaks. The European Union's General Data Protection Regulation and equivalent regulations in other jurisdictions have raised the bar for requirements regarding the processing of sensitive health information. The application must specifically detail which categories of data will be collected, for what legitimate purposes, who will have access at each level of identification, where and how they will be stored physically or digitally, for how long they will be retained, and what technical and organizational security measures will be implemented. If the application envisions international data transfer, legal safeguards must be specified; if cloud computing services are used, the provider's security must be justified; if data sharing with other researchers is planned, the access control mechanism must be described. A well-articulated and transparent data policy serves as a letter of credibility to the committee.

This concerns the demonstrable suitability of the research team. IRBs evaluate not only the protocol on paper but also the actual capacity of the individuals who will carry it out. The application must include updated CVs of the principal investigators, specific training in clinical research, prior experience in similar studies, and, if relevant, certifications in data protection or specialized knowledge in the clinical domain of the study. A multidisciplinary team combining medical, statistical, ethical, and technical expertise demonstrates that the project comprehensively addresses its complex dimensions. If researchers in training are involved, the supervision and mentoring system that will ensure their adequate performance must be clearly specified.(4)

Absolute transparency regarding funding and potential conflicts of interest is needed. Committees need to know who is funding the study to assess potential biases in the design, interpretation, or publication of results. The application must explicitly state all sources of funding, both public and private, and any commercial or academic relationships that could create a conflict of interest. If the software is developed by a company that plans to commercialize it later, a detailed explanation must be provided of how the independence of the statistical analysis and the right to publish negative or adverse results are guaranteed. The concealment of commercial interests, if discovered, can invalidate the entire application and irreparably damage the team's reputation.

Exercise patience and professional courtesy in all interactions with the committee. CEIm committees are generally composed of r volunteer professionals—doctors, nurses, lawyers, and patient representatives—who devote their personal time to reviewing others' projects while maintaining their own demanding professional obligations. Their responses may not be immediate, and their questions or requests for modifications may sometimes seem disproportionate or misinformed regarding technical aspects. Responding politely, providing substantive arguments without a defensive attitude, is the only viable strategy. If the committee requests specific changes, implement them unless there are compelling ethical or scientific reasons to disagree, in which case the disagreement should be explained respectfully and constructive alternatives offered. Aggressive confrontation or disdain for the ethics process does not lead to any favorable outcomes. (4)

14.4.2 BEYOND PERMISSION: ETHICS AS A COMPASS FOR DEVELOPMENT

For software developers, submitting to the scrutiny of an ethics committee represents a unique opportunity for professional reflection that rarely arises from purely technical processes. The questions the committee raises force us to consider dimensions of the project that the algorithmic approach tends to obscure: patients as people with autonomy and dignity, not as data points; psychological and social risks, not just technical risks of system failure; and the social value of the research, not just the model's metric performance. This confrontation with the ethical dimension, although initially uncomfortable, often results in software that is more mature, more responsible, and better prepared for the complexities of the real world.

Software that has survived rigorous ethical scrutiny has demonstrated its ability to justify itself against independent standards of human protection. It has had to articulate its benefits convincingly, acknowledge its limitations honestly, and demonstrate that the risks to participants are minimized and proportionate to the expected benefits. This process of public justification constitutes a form of social responsibility that distinguishes ethical research from irresponsible experimentation. The developer who goes through this process emerges with a deeper understanding that medical technology is not valued for its intrinsic sophistication but for its demonstrable contribution to human well-being.

For clinicians involved in medical software projects, their prior experience with ethics committees is an invaluable asset they can bring to the multidisciplinary team. They understand

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

regulatory language, anticipate the typical concerns of reviewers, and have navigated the process multiple times in their own research. They can guide developers in preparing documentation, review forms from the perspective of someone who knows what the committee truly values, and sometimes act as principal investigators submitting the formal application, lending clinical credibility to the project. Their role goes beyond merely facilitating access to patients; it includes expertly navigating the governance structures that regulate all research involving human subjects.

The future of the relationship between medical software developers and ethics committees points toward greater specialization and mutual understanding. Some institutions are creating specific IRBs or subcommittees dedicated to digital projects and technological innovation, incorporating members with training in health informatics, cybersecurity, and data science. These specialized committees can more accurately assess the specific risks of medical software—from algorithmic biases to security vulnerabilities—without falling into the trap of mechanically applying criteria designed for drug trials. Moreover, developers are developing greater ethical literacy, understanding that participant protection is not an obstacle but rather a prerequisite for the legitimacy of their work.(4)

As these specialized structures mature, the responsibility for preparing robust ethics applications falls on research teams. Doing this well not only speeds up the process of obtaining the necessary approval but also builds a foundation of trust with the institution that will be valuable throughout the project's lifespan. A committee that has had positive prior experience with a research team is more likely to approve future applications expeditiously, creating a virtuous cycle of collaboration. Conversely, a team that gains a reputation for superficiality or ethical opacity will face increasing scrutiny in future proposals.

If we have successfully navigated the ethical approval process, if we have demonstrated that our software respects participants' autonomy, minimizes risks, and promises proportionate benefits, we are ultimately in a position to begin data collection and clinical evaluation. However, a troubling question then arises: how do we know that the results we obtain in this study, however welldesigned it may be, will reflect what will actually happen when the software is used outside the controlled research environment? How do we distinguish between efficacy demonstrated under ideal conditions and real-world effectiveness in everyday clinical practice, with all its noise, imperfections, and contextual variations?

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

REFERENCES

Schulz KF, Altman DG, Moher D; CONSORT Group. CONSORT 2010 statement: updated guidelines for reporting parallel group randomized trials. Ann Intern Med. 2010;152(11):726-32. https://doi.org/10.7326/0003-4819-152-11-201006010-00232

International Medical Device Regulators Forum. Software as a medical device (SaMD): key definitions. IMDRF/SaMD WG/N10FINAL:2013. IMDRF; Dec 18, 2013. Available at: https://www.imdrf.org/documents/software-medical-device-samd-key-definitions

U.S. Food and Drug Administration. Artificial intelligence/machine learning (AI/ML)-based software as a medical device (SaMD) action plan. Silver Spring (MD): FDA; Jan. 2021 [cited May 14, 2025]. Available at: https://www.fda.gov/media/145022/download

Spain. Royal Decree 1090/2015, of December 4, regulating clinical trials with medicinal products, Ethics Committees for Research with Medicinal Products, and the Spanish Registry of Clinical Trials. BOE. Dec 24, 2015;(307):121923-64. Available at: https://www.boe.es/eli/es/rd/2015/12/04/1090

World Medical Association. WMA Declaration of Helsinki – ethical principles for medical research involving human participants [Internet]. Ferney-Voltaire: WMA; Oct. 19, 2024 [cited May 14, 2025]. Available at: https://www.wma.net/policies-post/wma-declaration-of-helsinki/

The Nuremberg Code. In: Trials of War Criminals before the Nuremberg Military Tribunals under Control Council Law No. 10. Vol. 2. Washington, DC: US Government Printing Office; 1949. p. 181–2. Available at: https://history.nih.gov/display/history/Nuremberg+Code

National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research. Washington, DC: U.S. Department of Health, Education, and Welfare; 1979. Available at: https://www.hhs.gov/ohrp/regulations-and-policy/belmont-report/index.html

https://doi.org/10.62486/978-9915-9876-8-2.Ch14

Chapter 15 / Capítulo 15

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch15
Pages: 246-261

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 15. Real-World Evidence and Economic Evaluation

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "An algorithm that works perfectly in the laboratory and fails in practice is not a working algorithm; it is a mirage. The truth— r evidence of artificial intelligence in medicine—lies not in the clean, carefully selected data of research, but in the noise, variability, and imperfection of real-world clinical practice."

There is a very large gap between an algorithm's performance under controlled conditions and its effectiveness in real-world clinical practice. In the laboratory, the data are clean, the variables are welldefined, and the protocols are clear. In the real world, data are missing, measurements are imperfect, patients do not follow the rules, and doctors skip the protocols. An algorithm that showed an area under the curve of 0.95 in the validation study may decrease to 0.70 when implemented in a hospital with limited resources. This gap between efficacy (what the algorithm can achieve under ideal conditions) and effectiveness (what it actually achieves in practice) is among the greatest challenges in digital health, and it can be addressed only through studies that capture the complexity of the real world.

Real-world evidence (RWE) has become a central concept in the evaluation of health technologies. It refers to information derived from data collected in routine clinical practice, not in controlled trials. It includes data from electronic health records, patient registries, administrative databases, wearable devices, and other sources that reflect what actually happens when the technology is used under normal conditions. This evidence complements that from clinical trials, providing information on how interventions work in diverse populations, in varied contexts, and over time.

In this chapter, we explore the evaluation of medical software in the real world and its economic dimension. We begin with real-world validation, analyzing the differences from laboratory validation and the methods to address them. We will continue with economic evaluation and health technology assessment (HTA), which determine whether the software is worth its cost and whether it deserves to be funded by healthcare systems. We address how to demonstrate that the software is worth its cost, which is crucial for its adoption. Finally, we discuss economic evaluations specific to public and insurance systems, which have different criteria and processes. This is not an exhaustive guide to health economics but rather an introduction to the concepts that any developer must understand so that their software has a chance of being adopted.

15.1 VALIDATION IN THE REAL WORLD, NOT JUST IN THE LABORATORY

The history of artificial intelligence applied to medicine is littered with failed technologies that should never have been adopted clinically. Algorithms that published dazzling metrics in high-impact journals, with areas under the curve that bordered perfection and sensitivities that surpassed the expert human eye, only to collapse spectacularly when someone tried to use them in a real hospital, with real patients, amid the chaos of a night shift. This isn't about fraud or dishonest researchers; it's about an uncomfortable truth that the tech community is slow to accept: the lab lies—or at least tells only part of the story. Controlled conditions, curated data, and standardized protocols create an artificial world where algorithms may seem infallible, but that world does not exist outside the pages of scientific articles. Real-world validation is not an optional phase of development; it is the only filter that separates genuinely useful tools from digital mirages.

The distance between the laboratory and everyday clinical practice is not a gap; it is an abyss. In the laboratory, the data are clean, complete, and labeled by consensus experts and sourced from calibrated equipment and carefully selected patients. In the real world, data are the messy byproduct of urgent medical care: missing values because the patient was unconscious when blood pressure was attempted, uncertain diagnoses because the confirmatory test could not be performed, multiple conditions interacting in unpredictable ways, and incomplete records because staff were overwhelmed. An algorithm trained in the purity of the laboratory faces this chaos with the naivety of a tourist who expects everything to work just like at home and fails with the inevitability of a guarded prognosis.

15.1.1 THE SIX FRONTIERS OF FAILURE: WHY THE LAB DOES NOT PREDICT THE REAL WORLD

Data quality: a variable that is obsessively controlled in the laboratory but left to chance in

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

practice. In controlled studies, every measurement follows strict protocols: blood pressure is taken with the patient seated after five minutes of rest using a recently calibrated sphygmomanometer by specifically trained personnel. In the rush of an overwhelmed emergency department, the same blood pressure may be measured while the patient is lying in a hallway, immediately after being transported on a stretcher, using an automatic device of questionable calibration, by a nurse who is simultaneously attending to three other patients. The algorithm trained on laboratory data is expected to be consistent; it can be interpreted as a signal when it is noise or as noise when it is a signal. Robustness to these variations cannot be assumed; it must be demonstrated through explicit validation in environments where inconsistency is the norm.

Data integrity, or more precisely, its systematic incompleteness. Research studies design forms that capture all relevant variables for all participants; empty fields are exceptions that require justification. In real clinical practice, missing data is the rule: the lab test was not ordered because the physician considered it added no value in that specific context; the record was not completed because the computer system crashed; and the variable was not measured because the staff was occupied with another critical patient. An algorithm that requires a complete set of inputs to produce an output becomes a useless ornament at the most critical moment, precisely when data tend to be missing. To handle absence, feature engineering must incorporate strategies from the design phase,such as context-based intelligent imputation, probabilistic models that marginalize unobserved variables, or feature selection that prioritizes those available in 100% of real clinical cases.

The heterogeneity of the population—a diversity that controlled studies systematically eliminate through exclusion criteria that leave out precisely the most challenging patients. Frail elderly individuals with multiple comorbidities, patients with language or cultural barriers that hinder communication, people with mental illnesses that affect adherence, and immigrants with inaccessible medical histories—all are excluded from studies because of methodological complexity, yet they are precisely those who need care the most in real-world practice. An algorithm trained on a homogeneous population of young adults with a single, well-defined disease approaches real patients such as a child who has studied only arithmetic and suddenly encounters algebra. External validation in diverse populations is not a methodological luxury; it is the only way to know whether the algorithm has generalizable utility or is merely a tool for selected cases.

Variability in clinical practice—a reality that research protocols seek to eliminate and that real-world medicine celebrates as an expression of individualized clinical judgment. In controlled studies, all patients received the same standardized treatment, the same follow-up tests, and the same discharge criteria. In practice, each physician develops their own style, each department adapts protocols to its resources, and each hospital prioritizes them on the basis of its specific pressures. An algorithm that assumes uniform practice will fail when it encounters a physician who makes different decisions for valid reasons that the algorithm does not know. Adaptability to context—or at least the explicit characterization of how performance varies across different contexts—is essential for real-world utility.

The algorithm's own effect on practice—a form of feedback that does not exist in the laboratory, where the algorithm is a passive observer. When introduced into a real-world setting, the algorithm becomes an active agent that modifies the behavior of those who use it. Clinicians may place excessive trust in it, delegating critical judgment to the machine and losing the ability to detect the algorithm's errors when they occur. Alternatively, they may systematically distrust it, ignoring the recommendations and maintaining their previous practices. They may learn from the algorithm and improve their performance even when they are not using it, or they may become dependent and degrade their previous skills. These effects are an integral part of the real-world impact of the technology, but they are invisible in laboratory studies where the algorithm does not interact with humans making decisions. Evaluating them requires study designs that measure not only the model's technical performance but also how clinical behavior changes in its presence.

Time drift—the recognition that medicine is a constantly flowing river, not a static lake. Clinical guidelines are updated, new treatments emerge that alter outcomes, populations fall ill in different ways, and epidemics shift the prevalence of conditions. An algorithm trained on 2020 data may face a transformed clinical reality in 2025 where its predictions no longer apply. Validation cannot be a one-time event culminating in regulatory approval; it must be a continuous process of monitoring and adaptation. Unlike drugs, algorithms can and must be updated, retrained, and adjusted to changes in the environment. However, this requires a postimplementation surveillance infrastructure that is rarely planned with the same attention as the initial development.

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

15.1.2 DESIGNS FOR REALITY: HOW TO CAPTURE THE TRUTH WITHOUT LOSING CONTROL

To cross these six frontiers and obtain valid evidence on how algorithms actually perform, research methodology must evolve beyond classic laboratory designs. Observational studies using real-world data are the fundamental tool of this new approach. These studies analyze the algorithm's performance under completely natural conditions, without researcher intervention regarding who receives the test or how the result is recorded. They capture authentic variability, real-world noise, and the real-world limitations of clinical practice. They allow us to evaluate what happens when the algorithm encounters unselected patients with incomplete data in diverse contexts. However, this richness of reality entails known methodological vulnerabilities: selection biases regarding who accesses the algorithm, information biases regarding how results are documented, and confounding biases due to unmeasured factors that simultaneously affect the algorithm's indication and the outcome.

To mitigate these limitations without abandoning reality, researchers turn to hybrid designs that maintain some control while preserving pragmatism. Before-and-after studies, which compare the period prior to implementation with the period after implementation at the same center, control for hospital characteristics that remain constant over time, although they remain vulnerable to general temporal trends. Interrupted timeseries designs analyze the evolution of indicators before and after the intervention, looking for changes in level or slope that suggest a causal effect beyond the preexisting trend. Pragmatic trials represent perhaps the most sophisticated synthesis, maintaining randomization as a guarantee of causal inference but relaxing inclusion criteria to reflect the real-world population, allowing for flexibility in implementation according to the local context, and measuring outcomes that matter in daily practice beyond laboratory surrogate end points.

Pragmatic trials deserve particular attention because they reconcile scientific rigor with external validity. Unlike explanatory trials, which seek to demonstrate biological efficacy under ideal conditions, pragmatic trials evaluate effectiveness under real-world conditions of use. Their inclusion criteria are broad, reflecting the algorithm's actual indications; their protocols are flexible, allowing adaptation to local workflows; and their outcome measures are relevant to clinical and health policy decisions. They are more costly and complex than observational studies are, but they provide higher-quality evidence for implementation decision-making. A well-designed pragmatic trial can answer not only whether the algorithm works technically but also whether it improves important outcomes when used by real clinicians on unselected patients.

Continuous postimplementation monitoring completes the validation cycle. Once the algorithm is in use, automated systems must compare its predictions with observed outcomes, detecting signs of performance degradation before they cause harm. This surveillance requires a data infrastructure that enables longitudinal tracking, a statistical methodology to detect significant changes in the model's calibration or discrimination, and clear procedures for deciding when an algorithm needs recalibration, updating, or withdrawal. Artificial intelligence can even be used to monitor itself, applying anomaly detection techniques to the algorithm's own predictions to identify when the environment has changed enough to call its validity into question.

15.1.3 SHARED RESPONSIBILITIES: DEVELOPERS, CLINICIANS, AND THE VALIDATION PACT

For software developers, real-world validation must transcend the category of a postdevelopment activity to become an integral component of the product lifecycle. From the algorithm's conception, plans must be made regarding how external validation will be conducted, what real-world data sources will be needed, what methodological designs will be used, and what financial and human resources will be needed. This early planning avoids the temptation to launch a product "validated" solely on internal data or in academic competitions. Collaborating with clinicians from the design phase ensures that validation is conducted in relevant contexts, using outcome measures that are both clinically meaningful and employing procedures that do not unduly disrupt clinical practice. Methodological humility is essential: recognizing that the laboratory does not predict the real world, that development metrics do not guarantee clinical utility and that negative validation results are learning opportunities, not personal failures.

For clinicians, real-world validation is the only guarantee that the tools they use are trustworthy. An algorithm validated within their own hospital, with their specific patients, within their particular context of resources and practices, generates operational confidence that no publication in a foreign journal can replace. That is why it is crucial for clinicians to actively

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

participate in these processes, not as passive subjects of research but as coproducers of valid knowledge. They must see the results of local validation before adopting new technologies; they must report discrepancies between the algorithm's predictions and their clinical experience; they must contribute to identifying cases where the algorithm fails in unexpected ways. Validation is not the sole responsibility of those who write code; it is a shared responsibility of those who care for patients.

The future of real-world validation points toward increasingly close integration between research and daily clinical practice. Electronic health record systems, which are becoming increasingly universal and standardized, can be configured as platforms for continuous research where each use of the algorithm contributes to its cumulative evaluation. Adaptive " " designs allow the algorithm to be adjusted in real time on the basis of observed results, merging research with quality improvement. Federated learning methodology allows models to be trained and validated across multiple institutions without centralizing sensitive data, facilitating large-scale external validation studies while preserving privacy. All of this is technically achievable, but it requires sustained investment, genuine institutional collaboration, and the political will to prioritize evidence over speed to market.

We arrive at an inescapable conclusion: real-world validation is neither a regulatory formality nor a bureaucratic obstacle to commercial launch. It is the fundamental mechanism that separates technological promise from real health benefits and genuine innovation from a profitable mirage. An algorithm that works in the laboratory but not in practice is not a frustrated breakthrough; it is a design flaw that should never have left the controlled environment. An algorithm implemented without adequate real-world validation is an undeclared experiment on patients who did not consent to be test subjects. Building the bridge between the laboratory and real life is the responsibility of all the actors in the digital health ecosystem, and doing so well is the only way for artificial intelligence to finally fulfill its promise of improving human health in a demonstrable and sustainable manner.

15.2 WHEN CLINICAL EXCELLENCE IS NOT ENOUGH

After years of effort, we have developed an algorithm that provides almost-supernatural precision in which patients with heart failure will be readmitted within the next thirty days. Rigorously designed clinical studies have shown that physicians using the system can intervene proactively and reduce readmissions by twenty-five percent. Clinicians are enthusiastic, publications in high-impact journals are racking up citations, and international conferences are requesting our presence. Everything indicates that we have a resounding success on our hands. However, when we present this technological marvel to the procurement department of the public healthcare system, the response we receive baffles us: "Interesting. How much does it cost? How much does it actually save? What resources will another patient lose out on if we invest in this?" These questions reveal an uncomfortable truth: in the world of public health, clinical excellence is necessary but insufficient. For an innovation to reach patients, it must pass an additional filter that is as rigorous as the scientific one but based on a different logic: health technology assessment (HTA), a process that translates medical success into sustainable value for the system.

Health technology assessment is neither a bureaucratic whim nor a tr aba to innovation; it is the institutional response to a dilemma that healthcare systems have faced ever since medical technology began advancing faster than the resources available for it. In essence, HTA asks not only whether something works but also whether its effectiveness justifies its cost, whether it is sustainable on a population scale, whether it distributes its benefits equitably, and whether it aligns with the values and priorities that society, through its democratic representatives, has decided to prioritize. In most developed countries with public health systems, regulatory approval of a technology by the FDA or the EMA is only the first step; the second, which is decisive for patients' actual access, is a positive evaluation by the national HTA agency, which determines whether the health system should, can, and wants to pay for it.(1,2,3)

15.2.1 THE GENEALOGY OF VALUE

The history of HTA dates back to the 1970s, when the healthcare systems of developed countries began to face an unsustainable tension: the technological explosion in medicine, with

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

its promises of unprecedented cures and improvements, clashed with the reality of finite budgets that were growing more slowly than expectations. The question “Does it work?” was no longer sufficient; the question “Is it worth it?” arose, and with it, the need for disciplines and processes that could answer it in a systematic, transparent, and reproducible manner was met. Since then, HTA has evolved from a marginal academic exercise into a central institution of health policy, with specialized agencies exerting decisive influence over which technologies reach citizens and which remain in the limbo of technical but not financial availability.

In Europe, this institutionalization takes on particularly powerful forms. The UK’s NICE (National Institute for Health and Care Excellence) has become a global benchmark, with the authority to recommend or reject technologies for the entire national health system on the basis of cost-effectiveness analyses that establish explicit thresholds of acceptability. The HAS (Haute Autorité de Santé) in France combines scientific evaluation with judgment regarding the “service médicalrendu,” the actual added therapeutic value. IQWiG (Institut für Qualität und WirtschaftlichkeitimGesundheitswesen) in Germany applies German methodological rigor to technology comparisons. These agencies, with their public procedures, detailed reports, and binding decisions, shape the environment in which any medical innovation seeking public funding must navigate. To ignore them is to set oneself up for commercial failure from the outset. (2,4)

For medical software developers, understanding this ecosystem is just as strategic as mastering machine learning architectures. An algorithm that is technically brilliant and clinically validated but fails to meet the cost-effectiveness threshold set by the NICE or its local equivalent may face an insurmountable barrier: regulatory approval exists, but public funding does not—which, in predominantly public systems, amounts to practical nonexistence for most patients. Therefore, incorporating the HTA perspective from the earliest stages of development is not a matter of business maturity; it is a condition for commercial survival. This means designing clinical studies with the aim not only of demonstrating biological efficacy but also of generating the specific data required by HTA economic models: information on implementation costs, impact on resource utilization, effects on quality of life measured with validated instruments, and appropriate comparators from the perspective of the healthcare decisionmaker.(3,4)

15.2.2 CORE: HOW VALUE IS MEASURED IN HTA

The beating heart of HTA is economic evaluation, a discipline that seeks to translate all dimensions of a technology—clinical and nonclinical—into a common metric that allows alternatives to be compared and resource allocation decisions to be made. The most widespread form of this translation is cost-effectiveness analysis, which compares the incremental costs of a new technology with the incremental benefits it generates, typically expressed in quality-adjusted life years (QALYs). A QALY represents a year of life lived in perfect health; years spent with disability or pain are weighted with lower values depending on the severity of the condition. The cost-per-QALY ratio tells us, in understandable terms, how much it costs to purchase health through the evaluated technology. (4)

Acceptability thresholds vary across countries and contexts, but they exist and are operationally decisive. In the United Kingdom, NICE traditionally uses a range of twenty to thirty thousand pounds per QALY, although it may adjust this for technologies that represent significant advances or benefit particularly vulnerable populations. In Spain, the Ministry of Health’s evaluation reports typically consider similar ranges in euros. If a technology falls below the threshold, the funding recommendation is favorable; if it falls above, it may be rejected or require price negotiation, restricted to subgroups where it is more cost-effective, or made conditional on the collection of additional evidence. These seemingly cold numbers determine the fate of innovations that have taken years of work and millions in investment.(3,4)

Calculating the cost-effectiveness of medical software presents specific challenges that models designed for drugs do not adequately capture. While a drug acts directly on the patient’s physiology with relatively predictable effects, medical software operates through complex mediations: it modifies clinicians’ behavior, alters organizational workflows, and generates ripple effects that spread throughout the system. Its impact on health outcomes is indirect, dependent on the implementation context, and mediated by factors such as user adherence, the quality of technical integration, and the system’s responsiveness to generated alerts. This makes economic models inherently more complex, requiring more structural assumptions and extensive sensitivity analyses. However, this complexity also presents opportunities: software that improves procedural

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

efficiency can generate economic benefits that go beyond direct clinical outcomes—arguments that HTA is increasingly able to appreciate.

15.2.3 FULL PICTURE: COSTS, SAVINGS, AND WHAT CAN'T BE QUANTIFIED

A rigorous economic evaluation of medical software must account for a range of costs that go far beyond the initial license price. Acquisition costs include not only user licenses but also technical installation, integration with existing hospital information systems, the potential need for additional hardware, and any infrastructure modifications that may be needed. Implementation costs encompass training for clinical and technical staff, managing organizational change, reduced productivity during the learning period, and resources dedicated to resolving initial issues. Maintenance costs include software updates, ongoing technical support, performance monitoring, and periodic recalibration of the algorithm. Finally, opportunity costs represent the value of alternative resources we forgo by investing in this software: what other interventions would we stop funding? What staff would we not hire? All these components must be inventoried and quantified honestly, not just the list price appearing in the contract.

On the benefits side, potential savings must be identified and quantified with equal rigor but also with methodological caution. Software that reduces hospital readmissions generates direct savings in length of stay and avoids procedures and untreated complications. A system that speeds up diagnosis saves costs from unnecessary diagnostic tests, days of work disability, and deferred treatments that become complicated. A tool that optimizes prescription reduces pharmaceutical costs and adverse events due to inappropriate medication. However, these savings do not materialize automatically; they depend on the healthcare organization's ability to reallocate the freed-up resources, on healthcare professionals adapting their practice to the new possibilities, and on the financing system not discouraging a reduction in activity (as occurs in fee-for-service systems). The analysis must distinguish between theoretical savings, potentially achievable savings, and savings actually realized in specific contexts.

Beyond quantifiable monetary costs and savings, HTA explicitly considers benefits that resist monetization but are central to the decision. Improvements in patients' quality of life, measured using validated instruments such as the EQ-5D or the SF-36, are incorporated into the calculation of QALYs and thus into the denominator of the cost-effectiveness ratio. The reduction in anxiety and uncertainty for patients and families, although difficult to capture in standardized surveys, can be assessed in qualitative analyses that complement quantitative models. Greater satisfaction and reduced burnout among healthcare professionals—organizational benefits that the software can generate, although traditionally underrepresented in HTA—are gaining recognition as determinants of the system's sustainability. Equity in access, the reduction in disparities among population groups, and the alleviation of the burden on informal caregivers—all these aspects are weighed in modern HTA, although their specific weighting varies across agencies and contexts.(5)

The perspective from which the economic analysis is conducted is a methodological choice with substantial consequences. The public health system perspective, which is common in evaluations for funding decisions, considers only the costs that fall directly on the public purse: payments to providers, personnel costs, pharmaceuticals, and funded devices. The broader social perspective also includes costs to patients and families: work time lost due to caregiving, transportation expenses to access treatments, and opportunity costs of informal care. The individual patient's perspective focuses on their subjective experience and the outcomes they value, even if the system does not pay for them directly. The choice between these perspectives can drastically alter the conclusions: software that reduces hospital costs but increases the burden on family caregivers may appear cost-effective from the system's perspective but not from the social perspective; an algorithm that improves quality of life but requires frequent travel may be valued differently by urban and rural patients.

15. 2.4 PREPARING FOR SCRUTINY: EVIDENCE STRATEGY FROM DESIGN

For medical software developers, adequately preparing for HTA means a strategic investment in evidence generation that begins before the first line of code is written and continues throughout the entire product lifecycle. In the early phases, this involves designing clinical trials that not only demonstrate biological efficacy but also systematically collect data on real-world implementation

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

costs, healthcare resource utilization, and effects on quality of life measured using instruments accepted by HTA agencies. It means selecting appropriate comparators from the healthcare decision-maker's perspective: not just placebo or no intervention but also the therapeutic or organizational alternatives that the system currently uses or would consider using. This means planning subanalyses by risk groups, implementation contexts, and resource levels, which allow for identifying where the software is particularly cost-effective and where it is not.

The collection of information on the actual costs of technical integration and operational maintenance is particularly critical and frequently underestimated. HTA economic models are only as good as the data that feeds them, and developers who can provide estimates on the basis of real-world implementation experience, rather than theoretical projections, gain decisive credibility. This requires close collaboration with early implementation sites, rigorous documentation of all the resources consumed, and honesty about the setbacks and unforeseen costs that inevitably arise. An economic model that incorporates real implementation data—even if it is less favorable than initial projections—is more convincing than one based on unvalidated optimistic assumptions.

For clinicians involved in the development and evaluation of medical software, HTA may initially seem like an inappropriate intrusion of economic logic into the realm of healing. However, understanding that healthcare resources are finite and that every euro spent on one technology is a euro unavailable for another transforms this perception. HTA is not a cold accounting exercise but an institutionalized attempt to allocate resources as fairlyly and efficiently as possible, maximizing the population health generated with the available budget. Clinicians play an irreplaceable role in this process: they can contribute tacit knowledge about what actually works in practice, about what patients genuinely value beyond measurable endpoints, and about the hidden costs of implementation that abstract economic models may overlook. Their participation in HTA committees, in expert groups that inform evaluations, and in the generation of realistic evidence is essential for coverage decisions to reflect the complexity of medical practice.

The future of HTA applied to medical software lies in the development of specific methodologies that capture the unique characteristics of digital interventions. Traditional economic models, designed for drugs with predictable life cycles and direct biological effects, do not adapt well to technologies that evolve weekly, generate complex indirect effects, and whose value depends critically on the implementation context. New approaches are being developed that incorporate accelerated technological obsolescence, network effects and organizational externalities, as well as the importance of local adaptation. The involvement of patients and caregivers in evaluation processes is also advancing through public deliberation techniques and preference analysis so that HTA decisions reflect not only technical efficiency but also social values regarding justice, equity, and human dignity.

15.3 HOW TO DEMONSTRATE THAT THE SOFTWARE IS WORTH THE COST

The history of medical software development is littered with technical promises that never managed to become widespread healthcare realities. Algorithms that seemed magical in controlled demonstrations, that outperformed the best human specialists in academic competitions, that accumulated citations and recognition in high-impact journal publications but that at the end of the day remained on digital shelves, used only by those who created them and by a few early enthusiasts. They did not fail because of technical flaws; they failed because their creators never learned to answer the question that every payer, public or private, eventually asks: Why should I allocate my scarce resources to this rather than to the thousand other needs competing for them? In a world of limited budgets, where every euro spent on a technology is literally a euro unavailable for another potentially life-saving intervention, demonstrating clinical efficacy is merely the starting point. The real challenge is to build a comprehensive value e that convinces decisionmakers that this specific investment deserves priority over legitimate and competing alternatives.

Building this case is not an exercise in marketing or persuasive rhetoric; it is a rigorous process that combines solid scientific evidence, sophisticated economic analysis, and a deep understanding of the institutional and political context where decisions will be made. It requires the language of algorithmic precision and software architecture to be translated into the language of budgetary impact, systemic efficiency, and return on investment in population health. It requires segmenting heterogeneous audiences—from clinicians concerned with practical utility to managers obsessed with cash flow—and building specific arguments that resonate with each

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

group's priorities. In addition, methodological honesty about limitations and risks is needed because credibility, once lost with public procurement officials or hospital executive committees, is virtually irrecoverable.

15.3.1 KNOWING YOUR AUDIENCE

The essential starting point for any value-demonstration strategy is a comprehensive mapping of who the stakeholders are who will ultimately decide on the adoption and funding of the software and what decision-making factors influence each of them. This segmentation is not a minor tactical detail; it is the difference between a message that resonates and drives action and one that elicits indifference or rejection. A hospital's medical director, for example, operates under a logic of clinical efficiency and quality of care: they are concerned with reducing variability in practice, improving patient safety indicators, and positioning their institution as an innovative leader. The procurement manager of a public health system operates under imperatives of population-level cost-effectiveness and budgetary sustainability: they must demonstrate that every euro spent generates maximum health benefits for the covered population and that decisions can be defended in the face of audits and before courts of auditors. The frontline clinician, who will ultimately use or ignore the software, values immediate practical utility, seamless integration into their already overloaded workflow, and respect for their professional autonomy.

A message designed to convince the medical director of the reduction in adverse events may be completely irrelevant to the procurement manager, who needs to see the five-year budget impact analysis. An argument that resonates with clinicians regarding the reduction in administrative time may seem futile for medical directors, who seek to affect institutional quality indicators. Therefore, the value communication strategy must be multifaceted and adaptive, maintaining a core of common evidence but presenting it with differentiated emphasis depending on the audience. The supporting materials should include different versions: a financial executive summary for managers, a clinical impact analysis for medical directors, user testimonials for clinicians, and a systems-level overview for health policy makers. Consistency among these messages is essential; there can be no contradictions between what is promised to one group versus another, but the presentation must be tailored to maximize the perceived relevance for each stakeholder.

15.3.2 THE FIVE PILLARS OF THE VALUE PROPOSITION

The first fundamental pillar is the rigorous quantification of clinical benefits, expressed in terms that go beyond vague generalities and are grounded in specific, credible numbers relevant to decision-making. It is not enough to state that "the algorithm improves the diagnosis of sepsis"; one must be able to say, backed by published evidence or well-designed proprietary studies, that "the system detects sepsis in its early stages an hour and a half before conventional diagnosis, which is associated with a 15% reduction in 30-day mortality and a two-day decrease in the average ICU stay." These concrete numbers, especially when expressed in units that decisionmakers recognize as meaningful (lives saved, complications avoided, days of hospitalization reduced), have a persuasive power that qualitative statements cannot match. The evidence must come from sources that the audience respects: publications in indexed journals, independent external validations, or data from real-world implementations in similar institutions.

The second pillar requires the translation of these clinical benefits into economic terms that are understandable to those who manage budgets. Although a life saved is priceless in the abstract, healthcare systems need to know the incremental cost of saving it using this specific technology to compare it with alternatives. This translation requires economic models that estimate, on the basis of demonstrated clinical effects, the impact on resource utilization: avoided hospital stays, canceled unnecessary diagnostic procedures, complications that do not require treatment, and staff freed from administrative tasks. Each of these effects has an estimable unit cost within the local healthcare system, and aggregating them allows for the construction of a case for savings or, at the very least, a reasonable cost per unit of benefit. This is not an exact science—economic models incorporate uncertainties that must be made explicit—but it is an unavoidable necessity for dialog with administrators.

The third pillar is comparative analysis with legitimate alternatives because software is never evaluated in a vacuum. The relevant question is not whether the algorithm adds value in

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

the abstract but whether it adds more value than the other ways of addressing the same problem that the system could fund. These alternatives may include standard practices without additional technology, other competing software already available, nontechnological organizational interventions (such as manual screening protocols or rapid response teams), or even the reallocation of resources to other higher-priority areas. An incremental cost-effectiveness analysis comparing the software to the most relevant alternative from the decision-maker's perspective is the standard tool for this evaluation. If the software costs fifty thousand euros per QA-LY gained and the current alternative costs thirty thousand, the decision to adopt it will require additional arguments regarding benefits not captured in the formal economic analysis.(4)

The fourth pillar addresses the short-term budgetary impact, a practical consideration that often determines the feasibility of adoption regardless of long-term cost-effectiveness. Even if a software solution generates net savings over five years, its implementation may require significant upfront investments in licenses, technical integration, large-scale staff training, and workflow reorganization—costs that a rigid annual budget simply cannot absorb. A detailed budget impact analysis, projecting cash flows year by year over the implementation horizon, makes it possible to identify whether there are critical years of intensive investment and whether deferred financing mechanisms or staggered payment models can be designed to make adoption viable. This analysis demonstrates to decision makers that their actual budget constraints are understood and that practical solutions have been considered, not just theoretical arguments.

The fifth pillar incorporates benefits that resist direct monetary quantification but can tip the balance when formal economic analyses are very close. Improved healthcare staff satisfaction and retention are difficult to monetize but critical in the context of professional shortages. The reduction in administrative burden allows for more time for direct patient care. Greater equity in access to quality diagnostics across centers with different levels of specialization. The generation of data and knowledge that facilitates future innovations in the system. These benefits must be described qualitatively in a compelling manner, supported where possible by satisfaction surveys, staff turnover analyses, or equity studies. Although they do not directly enter into the cost-per-QALY calculations, they form part of the comprehensive value argument that decisionmakers consider, especially in systems where the sustainability of the professional model and equity are explicitly prioritized values.(4)

15.3.3 ANTICIPATION, TRANSPARENCY, AND LONG-TERM COMMITMENT

Healthcare procurement decisionmakers are skeptical of the profession and of accumulated experience. They have witnessed too many product presentations promising radical transformations that failed to deliver results, too many optimistic economic analyses that fell apart during actual implementation, and too many technologies abandoned after significant initial investments. Therefore, an effective value demonstration strategy actively anticipates the objections that will inevitably arise and has prepared solid, honest responses backed by evidence or planning.

Typical objections include doubts about generalizability: "Do your results from teaching hospitals apply to our context with more limited resources?" Response: present subanalyses at the resource level or propose a local pilot study before widespread adoption. Concerns about implementation costs: "What if the costs of integrating with our legacy systems exceed your estimates?" Response: offer integration budgets with pricecap guarantees or shared-risk models where the provider assumes part of the cost overrun. Skepticism about adoption by clinicians: "What if our doctors simply ignore the system?" Response: Present adoption data from similar implementations and offer training programs and organizational change initiatives with contractual adoption metrics. Every objection that is anticipated, addressed, and answered convincingly builds credibility; every objection that captures the presenter off guard erodes trust.

Transparency about limitations is a counterintuitive but powerful strategy. No software is perfect, no study is definitive, and no economic analysis is exact. Attempting to hide these universal truths breeds suspicion that more serious problems are being concealed, and when they are inevitably discovered, the presenter's credibility is destroyed. It is far more effective to proactively acknowledge limitations: "Our algorithm is less accurate in patients with advanced kidney disease, a subgroup representing eight percent of the target population; we are working on a version specifically for this group, but in the meantime, we recommend enhanced clinical monitoring in these cases," and to explain why, despite these limitations, the software deserves consideration. This honesty builds trust and allows for a constructive discussion about risk

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

management rather than a defensive confrontation.

The value proposition must include an explicit long-term vision that extends beyond the initial commercial transaction. The adoption of medical software is not a one-time event but the beginning of a relationship that will last for years, with implications for maintenance, updates, performance monitoring, and adaptation to changes in the clinical and technological environment. Demonstrating that one has a robust plan for this evolution, guaranteed technical support commitments, shared development roadmaps, user feedback mechanisms that drive continuous improvements, and assurances that the software will remain compatible with emerging standards is especially important for public systems that cannot afford to be trapped with obsolete technologies or defunct vendors. This long-term vision transforms the vendor's perception from a transactional supplier to a strategic partner.

15.3.4 PARTNERS, REAL-TIME INSIGHTS, AND THE FUTURE OF VALUE

Clinicians who have used the software are the most powerful allies in building the value proposition. A hospital director may be skeptical of a sales representative's claims, even if they are accompanied by impressive PowerPoint slides, but will listen with genuine interest to their own doctors if they endorse the technology on the basis of their first-hand experience. That is why investing in ensuring that early adopters are not only satisfied but also equipped to articulate their experience is an essential strategy. Their testimonials about specific cases where the software made a difference, their real-world usage data demonstrating sustained adoption, and their stories of patients who benefited—which can be shared (while respecting confidentiality)—are invaluable assets that no economic analysis can replace. The case for technical and economic value must be combined with the case for human value, as told by those on the front lines of care.

The future of demonstrating value in medical software points toward greater methodological standardization and closer integration of real-time evidence. As healthcare systems become fully digitized, it will be technically possible to continuously monitor the impact of implemented software, automatically compare outcomes between exposed and unexposed patients, detect signs of performance degradation, and adjust funding decisions on the basis of current data rather than historical studies. Value-based payment models, where the manufacturer receives revenue contingent on actual clinical outcomes achieved, are gaining ground as a mechanism for aligning incentives. In these models, demonstrating value is not—a presale exercise but an ongoing process: the software is funded as long as it demonstrates that it works, and funding ceases if it stops doing so. This logic, although technically demanding in terms of measurement infrastructure and complex contractual agreements, represents perhaps the purest way to ensure that payment is made only for value actually generated.

15.4 ECONOMIC EVALUATION FOR PUBLIC SYSTEMS AND INSURANCE

The difference between a tax-funded public health system and a private insurer competing for customers is not merely an accounting matter of who signs the checks. It is a profound divergence in the ultimate objectives they pursue, in the incentives driving decisionmakers, and consequently, in the criteria they use to assess whether a technology merits adoption. A public system, accountable to an entire population and funded with resources derived from legitimate fiscal coercion, seeks to maximize population health with available resources, prioritizing equity in access and distributive justice. A private insurer, which must attract customers in a competitive market and generate returns for its shareholders, seeks to offer a product perceived as attractive while maintaining the company's financial sustainability. These different objective worlds generate different questions, different decision-making processes, and different evaluation cultures that medical software developers ignore at their own commercial and, ultimately, ethical risk.

Understanding these differences is not an academic exercise in comparative health policy; it is a strategic necessity for anyone who want their technological innovation to move beyond the prototype phase and actually reach patients. Software designed and evaluated solely from the perspective of a public system can fail spectacularly in the private market, and vice versa. The population-level cost-effectiveness metric that convinces a European HTA agency may be irrelevant to a U.S. insurer concerned with retaining high-spending customers. Demonstrating a reduction in health inequalities that excites a public system director may not resonate with a private insurance sales director. Consciously navigating between these logics, adapting the

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

message and evidence to each context without compromising the product's integrity, is among the most sophisticated skills a digital health innovator must develop.

15.4.1 AT THE HEART OF THE STATE: THE LOGIC OF PUBLIC SYSTEMS

Public health systems, which are predominant in Europe and are expanding in many Latin American countries, operate under the logic of maximizing population health subject to budgetary constraints and principles of equity. The decision to fund a new technology, including medical software, typically goes through a formal evaluation process by an HTA agency, such as the NICE in the United Kingdom, the HAS in France, or the IQWiG in Geria, which publicly analyzes the evidence and issues binding or highly influential recommendations. This process is deliberately transparent, with explicit criteria, published methods, and opportunities for manufacturers to provide information and answer questions. The central question guiding the evaluation is whether the technology provides sufficient health value relative to its cost, compared to available alternatives, considering the system's budgetary constraints and equity obligations.(2, 3)

The dominant metric in these contexts is the cost per quality-adjusted life year (QALY), which allows for the comparison of very different technologies on a common scale: how much it costs, in terms of system resources, to gain one year of life in perfect health through this specific intervention. There are implicit or explicit thresholds, typically between twenty thousand and thirty thousand pounds or euros per QALY in most European systems, which function as rough filters: technologies below the threshold have a high probability of a favorable recommendation, while those above require additional justification or price negotiation. For medical software, calculating this ratio involves economic models that project the algorithm's clinical effects on mortality and morbidity, convert them into QALY gains, and relate them to all incremental system costs attributable to the software.(4)

However, the final decision in public systems is rarely based solely on the calculated cost-effectiveness. HTA agencies explicitly consider additional factors that capture dimensions of value not reflected in the economic ratio. Equity is crucial: software that disproportionately benefits disadvantaged groups, reduces health disparities, or improves access in underserved areas may receive a favorable recommendation even if its cost per QALY exceeds the standard threshold. The severity of the condition being treated matters: technologies for conditions with high mortality or disability, especially if no effective alternatives exist, are given greater flexibility. The total budgetary impact—how much it would cost to implement the technology at a population-wide scale and with full—can block adoptions that are cost-effective individually but unsustainable when aggregated. The quality of available evidence determines the confidence in effectiveness estimates. In addition, social value, defined through public consultations and stakeholder engagement, can tip the balance when uncertainty exists.(4)

The decision-making process in public systems is inherently long and complex, involving multiple sequential stages that can span months or years. The technical evaluation by the HTA agency generates a preliminary report on which the manufacturer can comment. A formal recommendation is submitted to public procurement officials, who initiate price negotiations on the basis of demonstrated value and international price comparisons. Final inclusion in the list of covered benefits requires contractual agreements specifying conditions for implementation, monitoring, and potential revisions. Throughout this process, the manufacturer has structured opportunities to provide additional information, address methodological questions, and propose risk-sharing agreements or outcome-based payments. This protracted process requires dedicated teams, strategic patience, and comprehensive documentation that cannot be improvised at the last minute.

15.4.2 IN THE HEALTHCARE MARKET: THE LOGIC OF PRIVATE INSURERS

Private insurers, which predominate in the U.S. system but are also present as a complement in Europe and expanding in Latin America, operate under a radically different logic that combines cost concerns with commercial imperatives of competitiveness. Here, there is no centralized agency that evaluates technologies for the entire system; each insurer makes independent decisions on the basis of its own criteria, which are often opaque and vary from one entity to another. The central question is not only about population health value but also about the product's appeal to consumers and companies purchasing insurance and its impact on the

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

insurer's financial sustainability.

Cost-effectiveness remains relevant, but it is evaluated with a shorter time horizon and a different perspective. Insurers are particularly interested in the impact on their operating costs over one or two years, not just in long-term health value. Software that reduces hospital admissions is valuable because it lowers payments to providers; a tool that improves the management of chronic diseases is attractive because it reduces the use of costly services. However, the satisfaction perceived by policyholders also matters greatly: software that improves the patient experience, facilitates access to care, and provides a sense of control and personalization can be a powerful marketing tool that differentiates the insurer's product in a competitive market, even if it does not generate measurable direct savings. In this context, "effectiveness" includes customer retention and the acquisition of new policyholders—business metrics foreign to the vocabulary of public systems.

The decision-making process at insurers is faster and less formal than that at public HTA agencies but also more opaque and unpredictable. Decisions are made by internal medical committees, often without public guidelines or explicit thresholds, on the basis of available evidence, direct negotiations with manufacturers, and strategic considerations regarding market positioning. For the software developer, this means that simply submitting a standardized evidence dossier is not enough; it is necessary to build business relationships with each relevant insurer, understand their specific priorities—are they focused on aggressively reducing costs or on differentiating themselves through perceived quality? Do they prioritize technologies for chronic or acute care ()—and adapt the value proposition to each context? With respect to what convinces United Healthcare may not work with Aetna or Cigna, segmentation and personalization are essential.

Payment models in the private sector also differ substantially. While public systems typically negotiate prices on the basis of demonstrated value using international benchmarks, private insurers operate according to a more commercial logic: prices based on expected usage volume, exclusivity discounts, bundling agreements with other products, and, increasingly, outcome-based payment models in which the manufacturer assumes part of the risk that the software will not deliver the promised benefits. This commercial flexibility can be an opportunity for developers who can offer risk-sharing structures, but it also entails contractual complexity and the need for outcome monitoring systems that many manufacturers have not yet developed.

15.4.3 BETWEEN TWO WORLDS: MIXED SYSTEMS AND LATIN AMERICAN COMPLEXITY

Latin American health systems present particularly complex configurations where public provision, private insurance, and social security systems with hybrid characteristics coexist, often with permeable boundaries. In Mexico, INSSTE and IMSS operate under a social security framework for formal workers, while the public health system serves the uninsured population; in Brazil, the public SUS coexists with private health plans that cover a significant portion of the population; in Chile, FONASA and ISAPRE represent the public–private duality with complex interactions. Each of these components has its own funding rules, technology assessment processes, and adoption criteria.(5)

In some Latin American countries, there are HTA agencies that evaluate technologies for the public component of the system—CONITEC in Brazil and INVIMA in Colombia, with expanding evaluative functions—but private insurers and employer-sponsored plans may make independent decisions on the basis of commercial criteria. In others, funding decisions are decentralized at the provincial or state level, creating a patchwork of regulations that manufacturers must navigate. Local evidence is of paramount importance here: a study conducted in the United States or Europe, however rigorous, has limited validity in contexts where epidemiology, personnel costs, and clinical practices differ substantially. Generating evidence with local patients, conducting cost analyses using country-specific data, and making comparisons with standard practices within that specific system are essential investments for credibility. (2,3,6)

Transparency emerges as a common value across all these systemic complexities. Both public system evaluators and medical committees of private insurers in Latin America as well as in Europe or North America, value manufacturers being open to their methods, their primary data, and the limitations of their analyses. Hiding inconvenient information, presenting economic analyses biased by unwarranted optimism, or denying access to underlying data is the quickest way to destroy credibility and close doors. Conversely, being explicit about how results were obtained, acknowledging uncertainties and wide confidence intervals, and offering independent

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

audits of the data builds the trust that facilitates evaluation and, eventually, adoption.

Price negotiation is another cross-cutting element, although it takes different forms in each context. The list price is rarely the final price; in public systems, negotiation is formal and value-based, while in private insurers, it is more commercial and volume-based, but in both cases, it is inevitable. The developer must have a clear pricing strategy, with a maximum price that reflects the product's demonstrated value and a minimum price below which the business' sustainability is called into question. Within this range, the flexibility to offer discounts for volume commitments, staggered payments aligned with implementation, or shared-risk models where the final price depends on the results obtained can be decisive in closing deals.

15.4.4 STRATEGY, LOCAL EVIDENCE, AND THE FUTURE OF EVALUATION

For medical software developers, navigating this complexity requires a strategic approach that begins long before the product is ready for market. It is essential to identify, from the design phase, which markets are priorities and what types of payers predominate in them to guide the generation of evidence and the construction of the value proposition. Clinical studies must be designed with the specific questions that evaluators from each system will consider: equity analyses for European public systems, patient satisfaction studies for U.S. insurers, and implementation data in resource-limited settings for Latin American markets. Proactive relationships must be built with key stakeholders—HTA agencies, insurance procurement directors, scientific societies that influence practice guidelines, and patient associations advocating for coverage—before their approval is urgently needed.

Patience and persistence are essential virtues. Evaluation processes in public systems are deliberately designed to be lengthy to ensure democratic deliberation; processes at private insurers can be unpredictable because of staff turnover and changes in corporate strategy. The developer must be prepared to adapt to changes in evaluation criteria, to new evidence emerging from competitors, and to budget crises that shift priorities without losing sight of the long-term vision or the product's integrity. They must be prepared to generate ongoing evidence because economic evaluation is no longer a one-time event prior to commercialization but a process that continues postimplementation, especially in outcome-based payment models.

The role of clinicians in these ecosystems can be decisive in multiple ways. Their testimony regarding the software's real-world utility in practice, their participation in local studies that generate contextualized evidence, and their endorsement before scientific societies that issue usage recommendations—all contribute to the product's credibility. However, they can also help developers understand the particularities of the local system: the informal actors who influence decisions, the unwritten processes that are sometimes more important than official protocols, and the political sensitivities that determine which technologies are prioritized. Their on-the-ground knowledge is an asset that cannot be replaced by international consultancies.

The future of economic evaluation for medical software points toward greater harmonization between countries and between the public and private sectors. Initiatives such as EUnetHTA in Europe seek to coordinate evaluations and reduce duplication of effort, although implementation is slow and complex. Value-based payment agreements, where the final price depends on the health outcomes actually achieved in real-world practice, are gaining ground in both sectors because they align the incentives of manufacturers and payers. Dynamic evaluations, which are automatically updated as digital health systems generate continuous real-world evidence, are technically feasible and increasingly in demand by payers who do not want to be trapped in decisions on the basis of outdated evidence. All of these findings suggest a future where health technology assessment will be more efficient, more transparent, and more genuinely aligned with the value that technologies generate in practice.

We arrive at a conclusion that transcends business strategy: economic evaluation, in all its institutional forms, is not an end in itself but a means to ensure that healthcare resources—whether derived from taxes, insurance premiums, or direct payments—are allocated in the most beneficial way.

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

REFERENCES

US Food and Drug Administration. Artificial intelligence in software as a medical device [Internet]. Silver Spring (MD): FDA; updated Jan 6, 2025 [cited May 14, 2025]. Available from: https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-software-medical-device

Oortwijn W, Sampietro-Colom L, Habens F, Cranovsky R, Dehne M, Granados A, et al. The new definition of health technology assessment: a milestone in international collaboration. Int J Technol Assess Health Care. 2021;37(1):e54. https://doi.org/10.1017/S0266462321000416

National Institute for Health and Care Excellence (NICE). NICE health technology evaluations: the manual. PMG36 [Internet]. London: NICE; 2022 Jan 31 [updated 2025 Jul; cited 2025 Mar May 14]. Available from: https://www.nice.org.uk/process/pmg36

Claxton K, Martin S, Soares M, Rice N, Spackman E, Hinde S, et al. Methods for the estimation of the National Institute for Health and Care Excellence cost-effectiveness threshold. Health Technol Assess. 2015;19(14):1-503. https://doi.org/10.3310/hta19140

EuroQol Group. EuroQol—a new facility for the measurement of health-related quality of life. Health Policy. 1990;16(3):199-208. https://doi.org/10.1016/0168-8510(90)90421-9

Colombia. Decree 4725 of December 21, 2005, regulating the system of health registrations, marketing authorization, and health surveillance of medical devices for human use. Official Gazette. Dec. 26, 2005;(46.134). Available at: https://www.invima.gov.co/productos-vigilados/dispositivos-medicos/dispositivos-medicos-equipos-biomedicos

https://doi.org/10.62486/978-9915-9876-8-2.Ch15

Chapter 16 / Capítulo 16

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch16
Pages: 262-274

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 16. Ethics and Algorithmic Bias

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "An algorithm is not ethical or unethical in and of itself; it is so because of what it does, how it is used, and the consequences it has for people. An algorithm that discriminates is not necessarily a poorly programmed algorithm; it may be an algorithm that has perfectly learned the biases of the society that trained it. The ethics of artificial intelligence is not an optional add-on; it is a requirement of justice."

There is a naive belief that algorithms are objective because they are based on mathematics, and mathematics has no ideology. This belief is dangerously false. Algorithms learn from data, and data reflect society with all its biases, inequalities, and injustices. An algorithm trained on historical data will learn historical practices, including those we now consider discriminatory. It will learn that women were treated one way and men another, that ethnic minorities were offered fewer options, and that the poor were allocated lower-quality resources. It then reproduces and amplifies those biases, presenting them as if they were objective mathematical truths.

The history of artificial intelligence in medicine is full of examples of algorithms that have been discriminated. Algorithms that underestimated pain in Black patients because the training data reflected the erroneous belief that they feel less pain. Algorithms that recommend fewer cardiac tests for women because the symptoms of heart attacks in women are underrepresented in the literature. The algorithms allocated fewer resources to poor neighborhoods because the historical data revealed that less resources were spent there. These are not technical failures; they are ethical failures with real consequences for people's health.

In this chapter, we explore the fundamental ethical aspects of medical software development. We will begin with the most basic question: Does your software work the same for men as it does for women? We will continue with socioeconomic bias, which is often invisible but deeply ingrained in the data. We address informed consent for the use of secondary data, an increasingly relevant topic in the era of big data. Finally, we discuss legal liability and liability insurance—uncomfortable but necessary questions about who is responsible when an algorithm causes harm. This is not a treatise on philosophical ethics but a practical guide to developing software that does not perpetuate injustices.

16.1 DOES YOUR SOFTWARE WORK THE SAME FOR MEN AS IT DOES FOR WOMEN?

> "The most dangerous question is not whether your algorithm is biased but whether you have the courage to ask it out loud."

At first glance, the question seems almost naive in its simplicity: does your software work the same for men as it does for women? It could be interpreted as a minor technical curiosity, a methodological detail that developers resolve with a quick statistical check before moving on to more pressing matters. However, behind this apparent simplicity lies one of the deepest and least explored chasms in medical software development. It is not enough to answer yes or no; one must systematically demonstrate that the answer is true; one must understand with biological and social precision why differences emerge when they exist; and one must be prepared, morally and technically, to act decisively when the answer, as is often the case, is that the algorithm works better for one group than for another. The uncomfortable reality is that for most current medical algorithms, especially those developed without conscious attention to equity, the answer is no—even if their creators, trapped in the blindness of their own data, ignore this uncomfortable truth.

The training data that feed machine learning algorithms are often deeply unbalanced in terms of gender representation, reflecting historical biases in medical research that for decades considered the male body the universal norm and the female body a deviant variant. Cardiovascular diseases, for ex, were studied predominantly in men, ignoring the fact that they manifest qualitatively differently in women. The symptoms that algorithms learn to recognize as indicators of acute myocardial infarction—severe chest pain, radiation to the left arm, and profuse sweating—are typically male, whereas the most common manifestations in women—insidious fatigue, vague nausea, and jaw or back pain—remain underrepresented in the training data. Ignoring these differences does not neutralize them; it renders them invisible, making algorithmic bias more difficult to detect and correct because it is not even actively sought out.

16.1.1 THE FOUR BARRIERS TO GENDER BIAS

The first frontier that hinders an honest assessment of performance by gender is the

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

availability and quality of disaggregated data. To determine whether an algorithm performs equitably, we need sufficiently large samples of men and women, with adequate clinical variability, and reliably labeled for diagnosis. This combination is surprisingly rare. In many historical datasets, women are systematically underrepresented, especially in specialties such as interventional cardiology, vascular neurology, or critical care pulmonology. The large clinical trials that generated reference data for training modern algorithms historically excluded women of childbearing age and underrepresented older women. When these biased data feed algorithms, underrepresentation becomes algorithmic invisibility: the system simply has not seen enough female cases to learn reliable patterns, yet it operates with undifferentiated confidence across all patients.

The second frontier goes beyond mere data quantity to address qualitative differences in disease manifestation. Pathologies are not abstract biological entities that affect all human organisms equally; they are expressed through hormonal, metabolic, and sociocultural filters that differentiate men and women. Acute myocardial infarction, again the paradigmatic example, presents different symptoms in approximately one-third of female cases compared with male cases. An algorithm trained predominantly on male data learns a presentation pat ron that is literally dangerous when applied to women, generating false negatives that can cost lives. It is not that the algorithm is consciously sexist; it has learned an incomplete and historically androcentric version of clinical reality.

The third frontier is more insidious because it operates on the “fundamental truth” we use for training: the diagnostic labels issued by human clinicians who themselves operated under historical biases. If, for decades, women were underdiagnosed for certain diseases—because their symptoms did not fit standard male patterns and because their complaints were attributed to psychosomatic causes because they underwent fewer confirmatory tests—then the training data reflect this historical underdiagnosis as if it were biological variation. The algorithm learns that women are less likely to suffer from certain conditions, not because it is true but because that is how it was recorded in medical history. This vicious cycle is particularly difficult to break because the reference “truth” is contaminated at its source, and the algorithm, far from correcting the bias, systematizes and amplifies it by applying it with the apparent neutrality of mathematics.

The fourth frontier recognizes that gender identities do not operate in isolation but intersect with other demographic and structural variables. Being a woman is not the same as being a woman over eighty, a woman of African descent, or a woman with limited financial resources that hinder access to preventive care. Inequalities intersect and reinforce one another in complex configurations that a simple binary men–women analysis obscures. An algorithm that appears to work reasonably well “on average” for women may fail catastrophically for Black women, rural women, or women with disabilities. Intersectional analyses, although they require even larger samples and more sophisticated methodologies, are indispensable for a comprehensive ethical evaluation of algorithmic performance.

16.1.2 CONCRETE ACTIONS: FROM DIAGNOSIS TO CORRECTION

For medical software developers, addressing gender equity is not an exercise in political correctness but a technical and ethical obligation that requires specific and verifiable actions. The first step is the active curation of training data to ensure demographic representativeness that reflects the population to which the algorithm will be applied, with oversampling of historically underrepresented groups when necessary and with transparent documentation of limitations when full representativeness is not achievable. This means rejecting the convenience of readily available public datasets to actively seek out more diverse sources, even if this increases development costs and complexity.

The second step is the explicit and systematic evaluation of performance broken down by sex/gender, reported publicly regardless of the results. Developers must calculate performance metrics—sensitivity, specificity, predictive value, and calibration—separately for men and women and for intersectional subgroups when sample size permits. These analyses must be included in scientific publications and regulatory documentation, not as optional appendices but as central components of the product’s characterization. Transparency regarding performance disparities, while commercially inconvenient, is ethically mandatory and increasingly expected by regulators.

The third action is causal investigation of differences when they are detected. Not all performance disparities are equally concerning or require the same interventions. Some may be due to remediable biases in the training data, others to real biological differences that the

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

algorithm should learn to handle,and others to technical design issues such as inadequate feature selection. Diagnosing the etiology of the bias is a prerequisite for designing an appropriate remedy. If the problem is a lack of female data, the solution is additional data collection; if it is a qualitative difference in manifestation, the solution may be specific feature engineering; if it is biased in historical labels, the solution requires learning strategies that are more robust to systematic noise.

The fourth course of action, when differences are substantial and persistent, is to consider developing algorithms specific to population subgroups or at least explicitly incorporating sex/gender as an input variable rather than expecting the model to implicitly learn these differences. Sex-specific algorithms, although controversial because of their apparent essentialism, can save lives when disease manifestations are qualitatively different. Alternatively, modeling architectures that treat sex/gender as an explicit covariable may allow the algorithm to adjust its predictions in a differentiated manner without requiring completely separate systems.

16.1.3 CLINICAL RESPONSIBILITY AND THE FUTURE OF ALGORITHMIC EQUITY

For clinicians who use or are considering adopting medical software, a critical assessment of gender equity must be a standard part of their due diligence. Before they rely on a decision support system, they should demand documented answers to specific questions: In which population was the algorithm trained and validated? Was performance explicitly analyzed by sex/gender? Are there known differences, and how have they been addressed? If the manufacturer cannot provide this information or if the evidence suggests unresolved disparities, this should be a cause for concern and potentially for rejection. In daily clinical practice, clinicians must maintain active vigilance regarding potential discrepancies: if the algorithm appears to work well in certain patients but generates inappropriate alerts or dangerous false negatives in others, these patterns should be systematically reported as warning signs of algorithmic bias.

The issue of gender equity in medical algorithms is, of course, the tip of a much larger iceberg that includes age, ethnicity, socioeconomic status, sexual orientation, gender identity, disability, and residential geography. Each of these dimensions can introduce bias if not explicitly considered in the design and evaluation. However, sex is a particularly valuable starting point because it is universally applicable, because biological and clinical differences are relatively welldocumented in the medical literature, and because historical biases are particularly evident and wellstudied. A development team that learns to seriously address gender equity will be better prepared to identify and manage other dimensions of algorithmic distributive justice.

The future of ethically sustainable medical artificial intelligence lies in a design that is conscious of equity from the earliest stages of development, not as a quality control measure added at the end. This means that equity must be an explicit design goal, with quantifiable metrics and acceptable thresholds for disparity, just as accuracy or sensitivity are technical goals. This refers to the proactive selection of representative data, the continuous evaluation of performance across subgroups during deployment, and the organizational commitment to redesign when unacceptable inequalities are detected. This means that the diverse participation of affected communities in development, such as the implicit biases of developers—typically young men from urban areas with a technical education—does not crystallize into products that perpetuate their blind spots. In addition, this means, above all, recognizing that equity is not a state achieved through a single verification but a continuous process of monitoring, auditing, and iterative improvement.

We have reached a conclusion that transcends algorithmic technique: behind every detected or unnoticed bias are real people who suffer concrete consequences, sometimes fatal ones. A woman whose heart attack goes unnoticed because the algorithm fails to recognize her atypical symptoms may die at home because of a delayed diagnosis. A man who receives inadequate doses of medication because dosing studies were conducted predominantly on women may suffer preventable adverse effects. A person of limited means for whom the system assigns lower priority because historical data show that “less is spent there” may see their health compromised by algorithmic decisions that reflect structural inequalities. These are not ethical abstractions; they are concrete clinical injustices that developers have a moral responsibility to prevent. The question “Does your software work the same for men as it does for women?” is not a political or ideological demand; it is a fundamental ethical requirement and an indispensable clinical necessity. Answering it properly, with the methodological rigor it deserves, is one of the most important contributions we can make to a genuinely just 21st-century medicine.

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

16.2 SOCIOECONOMIC BIAS: EVIDENCE AND MITIGATION

Socioeconomic bias in medical algorithms is perhaps the most insidious form of algorithmic discrimination because it operates in the shadows of apparent technical neutrality. Unlike gender or race, which are relatively visible categories that conscientious developers can actively seek out to assess equity, socioeconomic status is an elusive, multidimensional variable that is often completely absent from the clinical databases that feed machine learning models. However, its influence on health is overwhelming: people from lower socioeconomic backgrounds experience poorer chronic health, less access to preventive services, less favorable clinical outcomes, and a life expectancy that can differ by more than a decade from that of their more privileged fellow citizens. When algorithms learn from data saturated with these structural inequalities, they neither correct nor ignore them; they systematize, amplify, and render them invisible under the guise of mathematical objectivity, creating vicious cycles of discrimination that are particularly difficult to detect and dismantle.

The problem of socioeconomic bias dramatically increased public consciousness, with a study published in 2019 that analyzed a commercial algorithm used by millions of patients in the United States for the allocation of healthcare resources. The system, designed to identify patients with greater medical needs and offer them additional support programs, was designed to perpetuate massive racial discrimination without its creators or users being aware of it. The cause was not an intentionally racist design but the use of a seemingly innocuous proxy variable: historical healthcare costs.(1) The algorithm logically assumed that those who had generated higher medical costs in the past were those with greater current needs. However, because black patients have historically been less often covered because of barriers to access, discrimination in clinical practice, and a lack of insurance, the algorithm systematically learned that black patients were less sick and needed less care when they were just as sick or sicker than their white counterparts with higher documented costs were. This paradigmatic case illustrates how socioeconomic bias can infiltrate algorithms without ever being explicitly named, carried by variables that appear technically neutral but are deeply imbued with structural inequality.

16.2.1 THE TUNNELS OF BIAS: HOW INEQUALITY BECOMES A PROXY

The pathways through which socioeconomic status influences clinical data are multiple, complex, and mutually reinforcing. People of lower socioeconomic status face systematic barriers to accessing healthcare: inadequate transportation, inflexible work schedules that conflict with clinic hours, a lack of insurance coverage, or simply the inability to pay copayments and medications. As a result, their illnesses are diagnosed later, at more advanced stages, and with poorer prognoses. These diagnostic delays do not reflect lower disease severity but rather reduced access to the screening system; however, the algorithm that learns from these data interprets the delayed presentation as lower risk or less need for intervention.

In addition to access, there are differences in the quality of care received. Low-income patients are less likely to receive state-of-the-art treatments, be referred to specialists, or participate in chronic disease management programs. Their doctors may be less experienced, work in settings with higher patient volumes, or have less time for individualized care. Clinical outcomes are worse, but these poorer outcomes are not attributable to patient biology but rather to the unequal organization of care. The algorithm, however, learns that certain patient profiles, correlated with poverty, have worse outcomes and may erroneously infer that they are less likely to benefit from expensive interventions.

Health literacy and communication styles also vary systematically with socioeconomic status. People with less formal education may describe their symptoms differently, using vocabulary that medical coding systems do not adequately capture. They may have less ability to navigate the healthcare system, less persistence in seeking diagnoses, and less trust in medical authority. All these factors are reflected in the data as “lower adherence,” “atypical presentation,” or “poor compliance”—labels that the algorithm learns to recognize as predictors of poor prognosis, without understanding that they are manifestations of structural inequality rather than individual characteristics.

The physical and social environment adds another layer of complexity. Low-income individuals are more likely to live in neighborhoods with environmental pollution, food insecurity, inadequate housing, and exposure to violence. Their jobs are more physically demanding, more precarious, and more stressful. All of these affect their health and are recorded in medical data,

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

but the algorithm lacks the ability to distinguish between what is due to biological disease and what is due to social inequality. The ZIP code becomes a powerful proxy for health, not because geography determines biology but because geography determines exposure to social determinants of health.

16.2.2 CASE STUDIES: WHEN PREDICTING RISK IS CONFUSED WITH PREDICTING PRIVILEGE

The example of hospital readmission prediction algorithms clearly illustrates how socioeconomic bias can completely reverse the logic of intervention. These systems, widely used in U.S. healthcare systems and expanding in other countries, are designed to identify patients at high risk of readmission within thirty days of discharge, allowing hospitals to allocate preventive transition resources to those who need them most. However, independent studies have shown that these algorithms systematically overestimate the risk of readmission in patients from high socioeconomic backgrounds and underestimate it in those from low socioeconomic backgrounds.

The reason for this paradox is that the algorithm is, in fact, predicting access to the healthcare system rather than actual medical needs. Privileged patients are more likely to be readmitted because they have better access to hospitals, because their primary care physicians are more available to refer them, because they have insurance that covers readmissions, and because the medical culture is more inclined to hospitalize them at the slightest sign of doubt. Low-income patients, on the other hand, may not be readmitted simply because they cannot afford transportation to the hospital, because they fear the costs even if they have formal insurance, because they cannot afford the time off work for recovery, or because they have learned that the system does not meet their needs. The algorithm interprets the absence of readmissions as an absence of risk; in reality, it may indicate abandonment of the system because of insurmountable economic barriers.

The accumulated evidence on socioeconomic bias in medical algorithms, although still limited compared with the literature on sex or racial bias, is consistently concerning. A 2021 systematic review revealed that the vast majority of predictive algorithms in medicine simply do not evaluate performance by socioeconomic status, as if this dimension were irrelevant to equity. When studies do disaggregate data by available socioeconomic indicators, they frequently find significant and substantial differences. Some algorithms are completely useless and perform no better than chance when applied to low-income populations, even though they work reasonably well in privileged populations. Implementing these systems without specific adjustments is not a neutral technological improvement; it is an active way of amplifying existing inequalities, allocating preventive resources to those who already have them and denying them to those who need them most.

16.2.3 DEVELOPMENT CHALLENGES: DETECTING THE INVISIBLE

For medical software developers, addressing socioeconomic bias presents specific challenges that have no purely technical solutions. The first challenge is the lack of direct data. Many historical clinical databases simply do not include information on income, education, occupation, or any direct measure of socioeconomic status. Hospitals do not routinely ask patients how much they earn, and when the information exists in linked administrative records, privacy and regulatory barriers hinder its use for research. Without this basic information, it is technically impossible to assess whether the algorithm performs equitably across the socioeconomic spectrum.

A partial solution is the use of proxy variables such as the home ZIP code, which allows for an approximation of socioeconomic status on the basis of neighborhood characteristics using census data. However, these approximations are crude: a ZIP code can include both mansions and public housing, and assigning average characteristics to all its residents introduces substantial classification errors. Furthermore, using the ZIP code as a proxy can introduce new biases, particularly racial ones, in contexts where residential segregation is pronounced. Another approach is to enrich clinical databases with information from external sources, tax records, household surveys, and data from social programs, but this raises complex issues of privacy, informed consent, and integration logistics that many projects cannot resolve.

The second challenge is the complexity of the causal relationships between socioeconomic status and health. Socioeconomic status is not simply a confounding variable that can be

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

statistically adjusted for using standard techniques; it is a fundamental social determinant of health that acts through multiple simultaneous mediating pathways. Statistically adjusting for socioeconomic status can obscure important effects or create interpretive paradoxes. For example, if we adjust a model for "access to care," we may eliminate precisely the variable that explains why certain patients have worse outcomes, erroneously concluding that there is no disparity when in fact we have made it invisible through the adjustment. There are no simple statistical solutions; careful, e-nalysis and explicit theorization of causal mechanisms are needed, as well as frequent interdisciplinary collaboration with social epidemiologists and health economists.

The third challenge is intersectionality—the recognition that socioeconomic status interacts with other dimensions of inequality in ways that are not additive but multiplicative. Low-income women face different and potentially greater barriers than men do in the same economic situation, including unpaid caregiving responsibilities, employment discrimination, and gender-based violence. Low-income immigrants face additional challenges related to language, documentation, and legal status. Poor older adults face social isolation and increasing dependence. Analyses that consider only one inequality variable at a time miss these crucial interactions, and evaluation designs that capture intersections require massive samples and sophisticated methodologies that are rarely available.

16.2.4 MITIGATION STRATEGIES: FROM AWARENESS TO ACTION

Despite these difficulties, there are concrete measures that development teams can implement to mitigate socioeconomic bias, although none offer a complete guarantee. The first step is, as always, awareness of the problem as an active dimension of design. Many medical software development teams, dominated by technical and biological health science profiles, simply do not consider social determinants relevant to their work. They implicitly assume that their data are neutral records of biological reality and that any pattern the algorithm detects is true pathology rather than a social artifact. Basic training in social epidemiology, health determinants, and the history of health inequalities should be a standard component of any medical software developer's curriculum—not as a humanities elective but as an essential technical skill.

The second measure is the explicit evaluation of performance across different socioeconomic groups, using available variables even if they are imperfect. If ZIP codes are available, performance should be analyzed by neighborhood income quintiles. If the insurance type is known, performance should be compared between public and private systems or between different coverage levels. If educational information is available, performance should be evaluated by educational attainment. When significant differences are detected, investigating their causes must be a priority: Are they due to biases in the training data that underrepresent certain groups? To real differences in disease manifestation that require specific models? To technical design issues such as inadequate feature selection? Without this diagnostic analysis, any attempt at correction is blind.

The third step is to consider the use of bias-correction techniques that have been developed in the fair machine learning literature. Rebalancing training samples to overrepresent historically underserved groups can improve performance for these groups, albeit at the expense of overall accuracy. The explicit inclusion of socioeconomic variables in the model—far from being eliminated for the sake of "neutrality"—can allow the algorithm to adjust its predictions in a differentiated manner. Adjusting decision thresholds by group, by setting different cutoff points for risk classification on the basis of socioeconomic context, can equalize false positive and false negative rates across groups. These techniques have known limitations; they may introduce new biases, require access to protected variables that are not always available, and raise ethical dilemmas regarding when it is legitimate to treat different groups differently, but they are preferable to inaction, which perpetuates the discriminatory status quo.

The fourth measure is the genuine participation of diverse communities in the development and evaluation of the software. People with direct experience with the inequalities the algorithm aims to address can identify problems, assumptions, and blind spots that developers in positions of privilege do not foresee. Usability and acceptability studies must deliberately include patients from different socioeconomic backgrounds, not just those who are the easiest to recruit in university or private settings. Patient advisory boards must include voices from historically marginalized communities. This participation is not a symbolic gesture of inclusion; it is a mechanism for detecting biases that purely technical methods cannot capture.

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

16.2.5 THE CLINICIAN'S RESPONSIBILITY AND THE ETHICAL BOUNDARY

For clinicians who use algorithms in their daily practice, socioeconomic bias is an additional reason to maintain a critical and active stance. When an algorithm recommends fewer resources for a patient from a poor neighborhood or classifies their risk as low when clinical intuition suggests otherwise, the physician must ask the following: Does this recommendation reflect actual medical needs, or does it reflect historical patterns of neglect toward this type of patient? Is the algorithm considering the specific barriers this individual faces in accessing care? If the answer is no, the clinician should feel justified—indeed, obligated—to disregard the algorithmic recommendation and rely on their professional judgment, explicitly documenting the reasons for the discrepancy to inform system improvements.

The future of the fight against socioeconomic bias in medical algorithms lies in greater integration of social determinants into routine clinical data. Initiatives such as the inclusion of Z codes in the International Classification of Diseases (ICD-11) for social factors influencing health—including issues related to housing, employment, education, and the social environment—are advancing slowly. Public health information systems are developing the ability to link with social registries, enabling more comprehensive analyses.(2) Statistical methods for detecting and correcting biases in machine learning continue to evolve, with adversarial learning and causal modeling techniques promising greater sophistication.

However, the most fundamental change is cultural: the widespread recognition that health is not a purely biological phenomenon but a biopsychosocial one and that algorithms that ignore this multidimensional reality are inherently incomplete and potentially harmful. Developers have an inescapable ethical responsibility not to exacerbate existing inequalities but not to create systems that amplify structural discrimination. However, the responsibility to build more just societies and to eliminate the roots of poverty and exclusion that biased data generate in the first place is collective and political. Algorithms can make inequalities visible, they can monitor their evolution, and they can even mitigate their effects to some extent. However, they cannot, on their own, transform the power structures that produce them. That requires bold public policies, sustained investment in social services, and an organized fight against poverty—tasks that transcend software engineering but must constantly inform it.

If we have internalized the gravity of socioeconomic bias and the responsibility to mitigate it, the question inevitably arises of how these efforts toward algorithmic justice align with the rights of the people whose data we are using. How do we ensure that the use of data to detect and correct inequalities does not violate the autonomy of those who never explicitly consent to their medical records being used to train algorithms that will eventually make decisions about the care of others?

16.3 INFORMED CONSENT FOR THE USE OF SECONDARY DATA

The relationship between patients and the information they generate about their own bodies constitutes one of the most intimate and simultaneously most opaque aspects of contemporary medicine. Every consultation, every diagnostic test, every prescription, and every MRI image leaves a digital trail that is recorded in hospital information systems, stored on servers, potentially shared among institutions, and eventually used for purposes that go far beyond the individual care that prompted its initial collection.

This information is undoubtedly essential for the continuity of care for the patient who generated it, but it is also extraordinarily valuable for other purposes: epidemiological research, the development of predictive algorithms, the improvement of quality protocols, the planning of population-based healthcare resources, and even, in commercial contexts, the generation of economic profits for private companies. The conflict between the patient's fundamental right to control their personal information and the social or commercial interest in using it to advance knowledge or profit represents one of the most complex and leastresolved ethical dilemmas of the digital era in healthcare.

Informed consent has traditionally been the preferred ethical and legal tool for managing this conflict. The logic is seemingly simple: before a person's data can be used for any purpose other than direct and immediate medical care, the individual must be informed of what is proposed to be done with their information and what the risks and benefits of that use are, and explicit, voluntary, and revocable authorization must be obtained. However, this conceptual simplicity quickly fades when we confront the realities of big data and medical artificial intelligence. How

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

do we inform a patient about future uses of their data when even the researchers themselves cannot foresee what scientific questions will be asked five or ten years from now? How do we obtain consent for algorithms that do not yet exist, trained with technologies that have not yet been invented, applied to clinical contexts we cannot yet imagine? How do we manage consent when data flow across multiple institutions, national borders, and cloud ecosystems where the physical location of the information becomes irrelevant?

16.3.1 BEYOND SPECIFIC CONSENT: ALTERNATIVE MODELS FOR THE BIG DATA ERA

Traditional consent, designed for clinical research studies with defined purposes, static protocols, and limited timeframes, proves profoundly inadequate for big data research where uses are emerging, analyses are iterative, and time horizons are indefinite. In light of this inadequacy, alternative approaches have emerged that attempt to balance patient autonomy with the feasibility of large-scale research, although none fully resolve the inherent tension.

Broad consent, also known as "broadband consent," represents the most widespread approach currently used in biobanks and large population databases. In this model, the patient authorizes the use of their data for a broad, predefined category of future research, provided that such research complies with certain general ethical and institutional governance conditions. The patient does not know the specific uses to which their data will be put but trusts in an oversight framework that will ensure that any subsequent research is ethical and scientifically valid. This approach has the virtue of practical feasibility—it allows for secondary research without requiring repeated contact with millions of participants—but it raises fundamental questions about whether it genuinely meets the "informed" standard of ethical consent. Can consent that authorizes unknown uses be considered informed? Is it legitimate to waive the right to know the specific purposes of the research in advance?

Dynamic consent has emerged as a response to these concerns, using digital platforms to maintain an ongoing, two-way relationship between researchers and participants. Rather than a single, static act of consent, patients access digital portals where they can manage their preferences over time, receive notifications about specific new projects that use their data, and actively decide whether they wish to participate in each particular initiative. This approach more faithfully respects patient autonomy and the right to information but faces challenges in large-scale implementation: it requires sophisticated technological infrastructure, digital literacy among participants, and institutional organization capable of managing individualized preferences at a population scale. Furthermore, the risk of "nsent fatigue," where participants disengage because of the impossibility of continuously processing new requests, can erode the protection it aims to provide.

A third approach, more controversial but operationally necessary in certain contexts, is the waiver of consent authorized by research ethics committees. When a retrospective study uses large databases where contacting all patients would be practically unfeasible and the risks to participants are minimal because the data are adequately anonymized, ethics committees may authorize the use of data without explicit individual consent. This exemption is contingent on the study having significant social value, as there being no known prior objections from patients, and robust privacy safeguards being implemented. However, this practice is ethically contested, with legitimate arguments that consent should always be required even if it is impractical and that the convenience of researchers should not take precedence over the autonomy of participants.

16.3.2 THE COMMERCIAL FRONTIER: WHEN RESEARCH BECOMES A PRODUCT

In the specific context of medical software development, the issue of consent takes on additional layers of complexity because data are frequently used not only for academic research aimed at advancing public knowledge but also to build commercial products that will subsequently be sold to healthcare institutions. A patient may be genuinely willing to share their cancer data for university research aimed at improving diagnosis but not for a private company to develop an algorithm that it will later market, with profits that will not benefit the patient or their community. The distinction between research and commercial development is ethically relevant; it implies different distributions of benefits, different levels of public oversight, and different levels of exploitation, but this distinction is often blurred in current practice.

Many patients who generously authorized the use of their data for "medical research"

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

discover with dismay, years later, that their information fed algorithms owned by corporations that are now reaping millions in profits. This sense of betrayal—even though broad consent forms might legally cover these uses—erodes public trust in medical research and may cause future patients to refuse to participate. Medical software developers face the ethical responsibility to explicitly clarify, in the consent processes that feed their databases, when data may be used for commercial purposes and what the implications of such use are.

The issue of anonymization repeatedly emerges as a justification for consent waivers or less formal consent procedures. The promise that "the data are anonymous" is used to reassure patients and regulators about the risks of secondary use. However, perfect anonymization is technically increasingly difficult to achieve in the world of big data. Demonstrative studies have shown that combining relatively few seemingly innocuous demographic data—zip code, date of birth, and sex—allows for the unique reidentification of a surprisingly high proportion of individuals in "anonymous" databases. When these databases are combined with other commercially or publicly available sources of information, reidentification becomes nearly inevitable. Patients have a right to know that the promise of anonymity may be technically untenable and that "anonymous data" does not necessarily equate to "data without a risk of identification."

16.3.3 PRACTICAL RESPONSIBILITIES: FROM CODE TO ETH

For medical software developers, these abstract ethical issues have immediate and nonnegotiable practical implications. The first is the obligation of due diligence regarding the origin of the data used to train algorithms. It cannot be assumed that data acquired from third parties, or even generated internally, are free of ethical and legal restrictions. It is necessary to systematically verify the consent obtained at the source, authorizations from ethics committees, specified usage limitations, and the expiration dates of authorizations. Ignoring these aspects is not only an ethical failure; it is a potentially catastrophic legal exposure in a data protection regulatory environment that imposes multimillion-dollar penalties for noncompliance, as demonstrated by recent fines under the European GDPR.

The second implication is the imperative for transparency regarding the current use of data. When a patient interacts with a healthcare system that uses their data to improve algorithms—whether through implicit feedback on its use or through explicit analysis of its results—they should be clearly informed of this secondary processing. When a healthcare institution shares data with an external developer, there should be a clear contractual agreement specifying permitted purposes, explicit prohibitions, audit mechanisms, and the consequences of noncompliance. This transparency is not merely a matter of legal compliance; it is a necessary condition for maintaining the public trust that underpins the entire digital health enterprise.

The third technical implication is the design of systems that materially respect patient preferences. If an individual has expressed opposition to their data being used for certain purposes—for example, commercial research or international sharing—the system's architecture must be capable of verifying the exclusion of their data from the relevant training datasets. This requires sophisticated technical mechanisms for managing consent at scale: database flags, filters in processing pipelines, and audits to verify that exclusions are effectively honored. Implementing these capabilities is no small feat from a software engineering perspective, but it is increasingly expected by regulators and patients alike.

The fourth implication is active participation in the broader social debate on health data governance. Decisions regarding acceptable uses of medical data cannot be delegated exclusively to technologists, regulators, or markets. Developers, as actors with expert knowledge of technological capabilities and limitations, have a responsibility to contribute to the collective reflection on which applications are socially desirable and which should be restricted. They also have a responsibility for public education: to help patients and citizens understand, in accessible yet honest terms, the potential benefits and real risks of using big data in medicine so that decisions regarding consent are genuinely informed and not based on myths or unfounded fears.

16.3.4 THE ROLE OF THE CLINICIAN AND THE DIMENSION OF JUSTICE

For clinicians operating at the interface between patients and information systems, consent for secondary use of data is a matter of direct and growing responsibility. It is often they who collect the initial data, who explains procedures and their implications to patients, and who obtain

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

consent signatures on forms that patients often do not read in full. They are also the ones who may see their patients' trust betrayed if patients discover that their data have been used in ways they did not anticipate or would not approve of. That is why it is essential that clinicians deeply understand these issues, be a ble to explain them in a way that patients can understand during consultations and actively participate in ethics committees and institutional decisions regarding data policies.

The relationship between consent and social equity is particularly relevant and concerning. The most vulnerable patients—those with less formal education, language barriers, a reduced ability to process complex information, or less power to negotiate with medical institutions—are precisely those most likely to have their data used without truly informed consent. They are also those who stand to benefit most from research advances but who may suffer the most from its risks if the research perpetuates inequalities. Ensuring that consent is truly informed for everyone—not just for privileged patients who can afford lawyers and advisors—is a requirement of distributive justice that health systems must explicitly address through the design of accessible informational materials, the use of interpreters, and adequate time for deliberation.

Consent management technologies, natively integrated into electronic health record systems, will allow individuals to set detailed preferences regarding which specific uses they authorize: academic research, commercial research, no domestic use, no international use, no cancer research, and no mental health research. Citizen research platforms will transform patients from passive providers of data into active participants in the design and direction of research projects. Shared governance models, where representatives of patients, researchers, developers, and regulators collaborate on decision-making regarding data use, are gaining ground as alternatives to technical dictatorship or pure bureaucracy.

However, if we have established that medical artificial intelligence must be ethical, responsible, and fair in the distribution of risks, the pragmatic question arises of how these imperatives translate into viable business models. How do we monetize an algorithm that saves lives without resorting to the exploitation of sensitive data or creating inequalities in access? How do we build economic ecosystems where innovation is rewarded but social benefit comes first?

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

REFERENCES

Obermeyer Z, Powers B, Vogeli C, Mullainathan S. Dissecting racial bias in an algorithm used to manage the health of populations. Science (80-) [Internet]. October 25, 2019 [cited May 15, 2026];366(6464):447-53. Available at: /doi/pdf/10.1126/science.aax2342

Hau C, Grubber JM, Ferguson RE, Cushman WC, Ishani A, Glassman PA, et al. Social Determinants of Health ICD-10 Code Use by a Large Integrated Healthcare System. Healthc [Internet]. November 1, 2025 [cited May 15, 2026];13(21):2710. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC12608873/

https://doi.org/10.62486/978-9915-9876-8-2.Ch16

Chapter 17 / Capítulo 17

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch17
Pages: 275-289

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 17. Business Models in Digital Health

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "No matter how brilliant an innovation may be, no matter how rigorous its validation *is, no matter how ethical its design is*, if there is no sustainable business model to fund it, it will never reach the patients who need it. Digital health is not just science and technology; it is also economics, and those who ignore this reality are doomed to see their ideas die in the laboratory."

Medical software developers tend to believe that technical quality and clinical evidence are sufficient to guarantee success. They develop a brilliant algorithm, rigorously validate it, publish excellent results, and expect the world to recognize it. However, the world does not work that way. Between an excellent product and its widespread adoption lies a chasm that can only be bridged with a solid business model. Who is going to pay for this? How much are they willing to pay? How is payment structured? What incentives do buyers have? These questions, which many developers view as mundane, are actually what determines whether an innovation makes it into practice or remains on paper.

The business model is not an afterthought to development; it should be part of the design from the very beginning. It determines which features are viable, what level of evidence is needed, which marketing channels are used, and which partnerships are established. A well-developed business model can compensate for technical weaknesses; a poorly developed business model can sink even the best product. That is why it is essential for development teams to include, from the very beginning, people with knowledge of the healthcare market, financing, regulation, and competition.

In this chapter, we explore the different business models in digital health, their advantages, and their limitations. We begin with the fundamental question: who pays. We analyze the three major payers: public systems, private insurers, and the patients themselves. We will continue with digital therapeutics (DTx) and reimbursement pathways, emerging fields with great promise and complexity. We address digital prescribing, which turns software into a product that doctors can prescribe like a drug. Finally, we discuss sustainability and technology transfer, the long-term challenges that determine whether a project survives beyond initial funding.

17.1 WHO PAYS: PUBLIC SYSTEMS, INSURANCE, PATIENTS

> "No algorithm is so brilliant that it can survive a poorly chosen payment model; technology heals, but money distributes it."

The question "who pays?" seems, in its formulation, almost naively simple. However, this apparent simplicity conceals a strategic complexity that determines, with the force of commercial destiny, everything else: the price that can be set, the type and volume of evidence that must be generated, the distribution channels that will be viable, the communication strategy that will resonate, and the institutional partnerships that will be necessary. A product designed to be purchased directly by patient-consumers has absolutely nothing in common in its business architecture, with one designed to be financed by a public health system. Confusing these markets, ignoring their distinct logics, or attempting to construct a hybrid model without understanding the tensions between its components is one of the surest paths to commercial failure, no matter how sophisticated the underlying technology may be.

17.1.1 THE HEALTHCARE LEVIATHAN: SELLING TO THE PUBLIC SYSTEM

Public health systems, funded predominantly through taxation and responsible for universal coverage of entire populations, are the dominant payers in Europe, Canada, Australia, and a growing number of countries in Latin America and Asia. Their decision-making logic is not that of the consumer market but rather the maximization of population health subject to rigid budgetary constraints. They are not attracted primarily by technological innovation for its own sake or by abstract algorithmic sophistication; they are interested exclusively in the added health value that the product can generate relative to its cost compared with available alternatives. Software that costs the same as standard practice but does not significantly improve outcomes will not be adopted; software that costs substantially more will be funded only if the additional clinical benefit, rigorously demonstrated, justifies the incremental expenditure from a public health perspective.

The commercial process for public systems is a marathon rather than a sprint. It typically involves a formal evaluation by a health technology assessment agency that will analyze the

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

evidence of efficacy, safety, and cost-effectiveness; a price negotiation that can last for months and in which the manufacturer must demonstrate flexibility; and eventuallynt to inclusion in the catalog of covered services, which enables access for the population. From the initial submission to actual availability for patients, two to five years may elapse, during which the manufacturer must keep the product up to date, generate additional evidence if requested by evaluators, and manage complex relationships with multiple stakeholders. It is a process that is costly in terms of financial and human resources, but the potential reward is enormous: once included in public coverage, the product gains access to populations of millions of patients with predictable and stable revenue streams for years, often with protection against competitors through contractual exclusivity or regulatory barriers to entry.

Public systems have developed increasingly sophisticated purchasing mechanisms that developers must understand and navigate. Risk-sharing agreements, where the final price paid depends on the actual health outcomes achieved in the treated population, align the manufacturer's incentives with those of the system and reduce the adoption risk for the payer, but they require an outcome monitoring infrastructure that many hospitals do not yet have. Early access agreements allow for product promotion while additional evidence is gathered, with price reviews once more robust data become available. Staggered payments, where the cost is spread over time in alignment with implementation, facilitate short-term budget viability. Each of these models requires careful contractual design and the ability to demonstrate value on an ongoing basis.

17.1.2 MARKET LOGIC: NEGOTIATING WITH PRIVATE INSURERS

Private health insurers, which dominate the U.S. system but are also present as complements or alternatives in almost every country, operate under a radically different logic that combines concerns about clinical value with commercial imperatives of competitiveness. Their goal is not only to maximize the health of the covered population but also to offer an attractive insurance product that retains existing customers and captures new ones while maintaining corporate profit margins. This calculation introduces variables that public systems ignore: perceived satisfaction among policyholders, differentiation from competitors, brand value associated with technological innovation, and the impact on costs over shorter time horizons than those used in public HTA evaluations.

An insurer may be genuinely interested in funding software that substantially improves the patient experience—by streamlining appointments, reducing wait times, and facilitating smooth communication—even if evidence of direct cost savings is limited because retaining satisfied customers has a quantifiable commercial value. It may prioritize interventions that reduce hospital admissions in the medium term, even if the initial investment is high, because its business model makes it particularly sensitive to the use of costly services. It may seek technologies that attract corporate employers who purchase insurance for their employees, a particularly lucrative market segment. The calculation is more complex, less publicly transparent, and highly variable among insurers with different positioning strategies.

The process of selling to insurers is fragmented and relationshipdriven. There is no central agency that evaluates and determines the entire market; each entity makes independent decisions on the basis of its internal medical committees, its actuarial analyses, and its commercial negotiations. This means that the manufacturer must build commercial relationships with each relevant insurer, tailor the value proposition to their specific priorities—one may prioritize cost reduction, another product differentiation, and another customer satisfaction for those with chronic conditions—and maintain regional sales teams covering specific geographies. The market is more fragmented than the public sector but also more dynamic: an insurer can become a strategic partner that accelerates mass adoption if it includes the product in its standard catalog, generating significant revenue before public processes have even begun.

17.1.3 THE PATIENT AS A CUSTOMER: THE DIRECT-TO-CONSUMER MODEL

The third revenue stream involves the patient themselves, who purchases the product directly out-of-pocket. This model is dominant in wellness, preventive health, digital fitness, and lifestyle monitoring applications, but it also operates in specific clinical areas, such as mental health (cognitive–behavioral therapy apps), diabetes management (over-the-counter continuous

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

glucose monitoring), and home-based cardiac rehabilitation programs. The fundamental advantage of this model is its apparent simplicity: there are no institutional intermediaries to convince, no formal evaluation processes to go through, only the direct market where the product must compete for the consumer's attention and money.

However, this simplicity is deceptive. The price must be affordable for individuals paying with limited personal resources, which drastically compresses margins and limits the ability to recoup large investments in development and clinical validation. Adoption depends entirely on marketing, visibility in app stores, user reviews, and word of mouth rather than on recommendations from healthcare professionals that confer medical legi ty. The model raises acute ethical issues of equity: only patients with available financial resources can access it, potentially widening the health inequalities that technology is supposed to reduce.

The direct-to-consumer payment model is sustainable for products with low development costs, high digital scalability, and massive potential user volume. It is difficult to sustain complex products that require rigorous clinical evidence, costly regulatory validation, and sophisticated maintenance. For this reason, many products that start with the direct-to-consumer model eventually seek to transition to public or insurance funding to expand their reach and stabilize their revenue or adopt hybrid models where basic functionality is free and access to advanced features requires a subscription.

17.1.4 HYBRID MODELS AND THE ARCHITECTURE OF STRATEGIC CHOICE

There are hybrid configurations where payments are distributed among multiple actors, creating funding ecosystems that are more resilient but also more complex to manage. A public system may fund the base cost of medical software while the patient pays a copay for premium features. An insurer may offer the product at no direct cost to its policyholders as part of their benefits package, having negotiated a corporate price with the manufacturer. An employer may pay for a digital wellness program for its workforce as part of its strategy to reduce insurance costs and absenteeism. These hybrid models may be more viable than relying on a single payer, but they require sophisticated contractual architecture and a clear allocation of responsibilities among the parties.

For medical software developers, choosing the right payment model is not a tactical decision that can be postponed until the product is finished; it is a fundamental strategic determination that must inform the design from the outset. A product for a rare disease, with small target populations and a need for rigorous evidence, should likely target exclusively public systems that can finance the high cost per patient through the logic of healthcare value. A general wellness product, with the potential for massive viral reach and low marginal costs, can thrive in the direct-to-consumer model. A product for a common chronic disease—such as diabetes or hypertension—may have viable options in multiple markets and require careful analysis of competition, the ability to generate evidence, and available commercial resources to determine the optimal strategy. The critical point is that this decision should not be left until the end but rather should guide the product architecture, the design of evidence studies, and the building of organizational capabilities from day one.

For clinicians operating in these ecosystems, understanding payment models allows them to be realistic about what they can expect from the technology and how to advise their patients. A product available only through out-of-pocket payment raises ethical dilemmas of unequal access that must be acknowledged. A product dependent on public funding may take years to reach everyday practice. A product subject to changing insurer policies may disappear from coverage without warning. This understanding allows physicians to participate informedly in debates on coverage and prioritization policies and to advise individual patients on their actual options.

Value-based payment agreements, where the price is dynamically adjusted on the basis of achieved health outcomes, become more widespread as monitoring infrastructure improves. Institutional subscription models, where hospitals pay annual fees for access to product catalogs, will gain ground as a mechanism for technology portfolio management. Freemium models, with free basic features funded by advertising or data and paid advanced features, will be used for patient-facing products. Innovation in business models will be as critical to success as technological innovation itself.

Behind every payment model are real patients who hope to benefit from the technology. An excellent product that cannot find a viable business model is, from a public health perspective, a product that does not exist. However, a business model that works commercially yet is ethically

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

unacceptable—because it exploits sensitive data and because it widens inequalities in access because it shifts costs onto the most vulnerable—is equally unacceptable. True success is not just about finding who pays, but doing so in a way that ensures that the technology reaches everyone who can benefit from it—fairly, sustainably, and in a manner worthy of the trust patients place in us.

17.2 DIGITAL THERAPIES (DTX) AND REIMBURSEMENT PATHWAYS

Over the past decade, a category of health products has emerged that challenges traditional classifications and redefines the boundaries between software and medical therapy. Digital therapeutics, known internationally as DTx (digital therapeutics), represent a qualitative evolution from conventional health apps: they are interventions based on rigorous scientific evidence, explicitly regulated as medical devices by agencies such as the FDA or the EMA, that prevent, manage, or treat specific medical conditions exclusively through software.(1) Unlike general wellness apps that promise vague lifestyle improvements, DTx must demonstrate clinical efficacy comparable to that of drugs, obtain regulatory approval under the same safety and efficacy standards, and be prescribed by physicians as integral components of formal therapeutic plans. Their arrival in the healthcare ecosystem presents transformative opportunities for access to care but also presents complex challenges in terms of methodological development, regulatory validation, and, perhaps most critically, establishing sustainable reimbursement pathways that enable access for patients who need them.

The line between a conventional health app and legitimate digital therapy is not always visible to the casual consumer, but there are clear distinguishing criteria that professionals must understand. A genuine DTx must have a specific and explicit therapeutic purpose on the basis of a reasonably well-characterized biological or psychological mechanism of action. Efficacy must be demonstrated in clinical studies that meet the methodological standards required for drugs: randomized controlled trials where ethically feasible, clinically relevant and validated outcome measures, subgroup analyses, and adequate safety monitoring. It must be formally regulated as a medical device, with the corresponding quality certification and traceability. It must be explicitly designed for use under medical supervision and integrated into broader therapeutic plans rather than functioning as a standalone self-care intervention. These criteria radically distinguish DTx from mass-market wellness apps, which require neither clinical evidence nor professional supervision for commercialization.

17.2.1 DEVELOPMENT AS A SCIENCE: CLOSER TO A DRUG THAN TO AN APP

The development of digital therapy is, in its fundamental architecture, more similar to the development of a drug than to that of a conventional commercial app. It requires explicit conceptualization of the therapeutic mechanism of action: how does the software produce its clinical effect? By educating the patient about their condition? By systematically modifying risk behaviors through digitally implemented cognitive–behavioral psychology techniques? By continuously monitoring physiological parameters with real-time feedback that allows for therapeutic adjustments? By activating neuroplastic processes through digital neurorehabilitation interventions? This understanding of the mechanism is not merely an academic option; it is necessary to design appropriate clinical studies, to convince skeptical regulatory agencies, and to di rentiate commercially from the competition of unvalidated applications.

The cost of bringing a DTx to the regulated market is typically estimated at between two and ten million euros, depending on the complexity of the intervention, the number of indications sought, and the specific requirements of regulators in the target jurisdictions. This investment includes the technical development of software to meet the required medical quality standards, the conduct of clinical trials that can last for years, the preparation and submission of comprehensive regulatory dossiers, and initial market access activities. It is a considerable investment, but one that is manageable for many health technology companies, especially compared to the more than one billion dollars typically required to develop a new molecular entity drug. This more favorable cost ratio, combined with shorter development cycles and the ability to update products postmarket, makes DTx a particularly attractive therapeutic innovation opportunity for investors and entrepreneurs.

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

17.2.2 THE BIG CHALLENGE: WHO PAYS AND WHY

The central challenge in determining the commercial success or failure of digital therapy is not its technical or clinical development but rather the establishment of sustainable reimbursement mechanisms. Who pays for these interventions? How is their price determined in the absence of clear reference markets? What additional evidence, beyond regulatory approval, do payers need to include them in their covered benefits catalogs? These questions lack universal answers and vary dramatically across healthcare systems, creating a fragmented reimbursement landscape that developers must navigate strategically.

In Germany, the digital health applications (DiGA) procedure began in place in 2019, a pioneering regulatory and reimbursement framework that has enabled dozens of DTx to be funded by the public health system.(2) The German model allows apps that meet basic safety, functionality, and quality requirements to be provisionally included in the benefits catalog, with a price initially set by the manufacturer for the first year. During this temporary inclusion period, the manufacturer must complete a clinical study demonstrating the specific benefits of the app for the German population. If the study confirms the therapeutic value, the inclusion becomes permanent, and the price is negotiated with the health insurance associations. If the study is negative or not completed, the app is removed from the catalog. This hybrid model lowers the initial barrier to entry, allowing early access for patients while ensuring that only apps with solid evidence of real benefit remain funded in the long term.

The German model has proven to be a significant operational success, with dozens of approved applications covering indications ranging from type 2 diabetes to major depression, benefiting millions of patients with guaranteed access at no direct cost. It has been empirically demonstrated that it is technically and politically possible to systematically integrate therapeutic software into the benefits portfolio of a public health system, using transparent criteria and agile processes that do not sacrifice scientific rigor. Other European countries, including France, Belgium, and the Netherlands, are developing similar processes inspired by the German experience, suggesting that this model will likely expand in the coming years. For DTx developers, a deep understanding of the German framework is strategically essential, as it sets precedents that influence the evolution of reimbursement across multiple markets.

In the United States, the reimbursement landscape is more fragmented and evolving. Some large-scale private insurers have begun to selectively cover certain DTx, particularly in areas such as mental health and diabetes, where cost-effectiveness evidence is more robust. Medicare, the federal program for older adults, is actively exploring pathways for including digital therapies, but the process is complex and slow.(3) The FDA has established accelerated approval pathways for DTx, but regulatory approval does not guarantee reimbursement in the U.S. system, where each payer makes independent decisions. This fragmentation creates opportunities for creative negotiations with innovative insurers but also significant operational complexity.

17.2.3 VALUE, PRICE, AND THE ECONOMICS OF DIGITAL THERAPEUTICS

For a payer—whether a public system or a private insurer—to agree to fund digital therapy, the manufacturer must build a compelling case for value that justifies the requested price. This value can manifest in clinical, economic, or human dimensions, and typically, the most robust combination integrates all three dimensions. Clinical value is demonstrated through measurable improvements in health outcomes, such as a reduction in symptoms, improvement in biological markers, prevention of complications, or increased remission of chronic diseases. Economic value is quantified through savings in healthcare costs: reduced hospital admissions, fewer emergency room visits, optimized use of medications, or prevention of costly procedures. Human value, which is more difficult to monetize but equally important, includes improvements in quality of life, reduced treatment burden, greater patient autonomy, and satisfaction with the care received.

Pricing DTx is inherently complex because it lacks established reference markets comparable to those for generic pharmaceuticals. The price must reflect demonstrated value but also recoup development costs, consider the prices of existing or developing competitor products, and respect the willingness to pay of institutional buyers. In the German model, the manufacturer freely sets the price during the provisional inclusion year but then faces negotiations with powerful collective buyers. In other markets, the price may be determined through direct negotiation from the outset or through competitive bidding processes. In all the cases, a pricing strategy grounded in rigorous value analysis, with clear sensitivity scenarios and reserve points, is indispensable for

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

commercial negotiations.

17.2.4 MARKET STRATEGY AND THE ROLE OF THE CLINICIAN

For developers, entering the digital therapeutics market requires a specific strategy that goes beyond the mere creation of good software. Clinical studies must be designed from the product's conception, in collaboration with methodological experts in clinical research, ensuring that endpoints will be recognized as relevant by regulators and payers. Regulatory dossiers must be prepared with the same rigor as would be applied to a new drug, documenting not only efficacy but also safety, software quality, and traceability of the development process. It is necessary to understand the reimbursement processes specific to each target country and adapt the evidence and commercialization strategy to them. It is necessary to develop strategic relationships with payers, relevant scientific societies, and patient associations that can influence coverage decisions.

For clinicians, DTx represents a new therapeutic class that they must become familiar with, critically evaluate, and learn to integrate into their practice. Just as previous generations of physicians had to learn to prescribe antibiotics, antihypertensives, or antidepressants when these emerged, the current generation must develop competence in "digital prescribing." This requires specific training on which DTx are available for which indications, what evidence supports them, how to monitor the patient's response, and when to adjust or discontinue the intervention. However, it also requires a shift in mindset regarding the therapeutic relationship: a DTx is not a passive drug that the patient consumes; it is an active intervention that requires the patient's involvement in their own care. The role of the physician is evolving toward that of motivator, monitor, and supporter rather than a mere prescriber.

We are likely to see rapid expansion in areas where the evidence is more mature: mental health (depression, anxiety, insomnia), diabetes mellitus, cardiovascular diseases, and neurological rehabilitation. Hybrid models are likely to emerge that combine DTx with medications—for example, apps that optimize medication adherence—or with wearable devices that provide input data for therapeutic algorithms. Artificial intelligence is likely to enable increasingly personalized digital interventions, tailoring content and frequency to each patient's individual characteristics and responses, thereby increasing efficacy. However, there are also significant risks: the proliferation of DTx of varying quality that confuses the market; the lack of long-term evidence for many applications; the slowness of reimbursement processes in many markets; and potential resistance from clinicians who perceive DTx as threats to their autonomy or as an outsourcing of responsibility.

DTx solutions are not an end in themselves but rather a means to tangibly improve the lives of people living with chronic conditions. Behind every approved app are real patients who can benefit from better diabetes management, reduced symptoms of depression, and a higher quality of life during rehabilitation. The ultimate goal is not to create a profitable business—although financial sustainability is a necessary condition—but to help these patients live better lives. If developers keep this human purpose at the center of their decisions regarding design, evidence, pricing, and reimbursement, their strategies will be more coherent, more ethical, and more resilient. In the end, what justifies the existence of digital therapies is not the sophistication of the code but the genuine therapeutic value they provide to those who need it most.

17.3 THE PRESCRIPTION OF THE FUTURE: HOW DOCTORS PRESCRIBE BITS AND BYTES

> "Prescribing an app is not *technophilia*; it is recognizing that healing can come in the f r code when that code is validated, regulated, and prescribed with the same responsibility as any molecule."

The idea that a doctor could prescribe an app as if it were a drug contains something of a technological revolution and, simultaneously, something deeply natural in therapeutic logic. After all, if software can demonstrate measurable physiological or psychological effects, if it can reduce symptoms, modify risky behaviors, or prevent medical complications with the same robustness as a pharmacological intervention, why should it not occupy a legitimate place in the clinician's therapeutic arsenal? Digital prescribing is precisely this: the formal act by which a healthcare professional recommends, indicates, or explicitly prescribes a digital therapy as an integral component of a specific patient's treatment plan. However, behind this apparent

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

conceptual simplicity lies a profound transformation of clinical practice, with implications ranging from operational logistics to the very nature of the therapeutic relationship between doctor and patient.

Digital prescribing cannot be reduced to the casual conversation in the office where the doctor suggests, "Download this app; it might help you." It is a structured, documented, and technically integrated process that must fit seamlessly into the existing clinical workflow, or doctors simply will not adopt it. This process typically involves five sequential steps that transform a casual recommendation into a formal medical intervention.

17.3.1 THE FIVE STEPS OF THE DIGITAL THERAPEUTIC CHAIN

The first step is clinical diagnosis and determination of indications. The physician evaluates the patient and determines, on the basis of their specific medical condition, individual characteristics, comorbidities, preferences, and capabilities, whether they might benefit from a particular digital therapy. This selection does not differ in principle from the prescription of medication: not all antidepressants are suitable for all patients with depression, and not all digital therapies are suitable for all potential candidates. Physicians must be familiar with the available options, their efficacy profiles in subgroups, their relative contraindications, and their patient engagement requirements. Inappropriate selection—prescribing a digital therapy that is in, that the patient is unable to use, or that is not indicated for their specific profile—is just as problematic as inappropriate drug prescribing.

The second step is generating the formal prescription. In the most advanced systems, this involves creating a structured "digital prescription" that the patient can redeem on an authorized platform or specialized app store. This digital prescription includes crucial metadata: the specific therapy prescribed, the indicated duration of treatment, the agreed-upon therapeutic goals, personalized instructions for use, and follow-up criteria. In many cases, the prescription is linked to a unique activation code that unlocks the full therapeutic version of the app, while the publicly downloadable version may be limited in functionality or require direct payment. This activation code mechanism is what allows the healthcare system or insurer to cover the cost, while the patient gains access without immediate financial burden.

The third step is actual dispensing. The patient receives the prescription, typically via text message, email, or notification on a personal health portal, and accesses the digital therapy through the designated platform. The system verifies in real time that the prescription is valid, that it has not been redeemed previously, that the patient is eligible for funded access and that the therapeutic version is appropriate for their condition. In some models, payment is processed at this stage via or through direct billing to the healthcare system; in others, the patient pays and then requests reimbursement; in more advanced models, the cost is completely invisible to the patient and absorbed by institutional funding.

The fourth step is therapeutic use by the patient. Over weeks or months, the patient interacts with the app according to the prescribed protocol: completing intervention modules, recording symptoms or behaviors, receiving feedback, and potentially participating in interactions with human therapists coordinated through the platform. The app collects data on adherence (is the patient actually using the tool?), symptom progression, side effects or technical difficulties, and other parameters relevant to clinical follow-up. These data, always with the patient's explicit consent, can be transmitted back to the prescribing physician to inform follow-up.

The fifth step is clinical follow-up and therapeutic adjustment. The physician receives information on the patient's progress, typically through dashboards integrated into the electronic health record or automated periodic reports, and assesses whether the digital therapy is producing the expected results, whether the patient maintains acceptable adherence, whether there are adverse effects, or whether it is necessary to adjust the dose (frequency of use), switch to alternative therapy, or supplement with other interventions. Digital prescribing is not an isolated event but the start of a dynamic therapeutic process that requires continuous monitoring, just as would occur with any chronic pharmacological treatment.

17.3.2 THE INVISIBLE INFRASTRUCTURE: INTEGRATION AND STANDARDS

The technical integration of digital prescribing into existing health information systems represents a considerable challenge that is frequently underestimated by application developers.

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

Digital prescriptions must be able to be generated directly from the electronic health record interface that the physician already uses, without the need for parallel systems that increase the workload. They must be interoperable with dispensing platforms—, public app stores, and institutional health portals—using recognized data exchange standards. They must also allow for the secure bidirectional flow of follow-up data from the patient's app back to the physician's clinical system.

This integration requirement calls for common technical standards that are still under development in many jurisdictions. Well-designed APIs that enable communication between heterogeneous systems; standardized digital prescription formats; authentication and authorization protocols that ensure that only legitimately prescribed patients access therapeutic versions; and integrated electronic billing mechanisms are infrastructure components that require coordinated investment among developers, health systems, and regulators. In countries such as Germany, where the DiGA model has been in operation for several years, specific infrastructure has been developed to facilitate this integration, including centralized directories of approved applications, prescription verification and platforms, and standardized data reporting systems.(2) Other markets are still building these capabilities.

17.3.3 NEW SKILLS AND NEW RESPONSIBILITIES

For physicians, digital prescribing adds powerful tools to their therapeutic arsenal, but it also imposes new competencies and responsibilities that traditional medical training does not always prepare them for. They must develop up-to-date knowledge about which digital therapies are available in their jurisdiction, which have robust evidence for specific indications, which are covered by reimbursement, and what the specific contraindications or precautions are. They must learn to explain these interventions to patients who may have unrealistic expectations about the technology or, conversely, resistance due to digital mistrust. They must develop skills to encourage continued use, recognizing that adherence to digital therapies follows dropout curves similar to those of other chronic treatments. They must be vigilant about unanticipated adverse effects: patient frustration if they do not see quick results, excessive dependence on the app, or "digital fatigue" that reduces long-term effectiveness.

Training in digital prescribing should be a standard component of continuing medical education, not a peripheral option. Medical professional societies are responsible for developing prescribing guidelines, training programs, and certifications that ensure that physicians prescribing digital therapies have the necessary competencies to do so safely and effectively. Hospitals and health systems must integrate digital prescribing into their care protocols, with clinical support decisions that suggest appropriate digital therapies when indicated and with monitoring systems that alert patients to inadequate adherence or suboptimal outcomes.

17.3.4 PATIENT-CENTERED CARE: EMPOWER AND FATIGUE

For patients, digital prescriptions represent both an opportunity for empowerment and a risk of added burden. Digital therapies typically require active participation, time commitment, and cognitive effort that some patients find motivating but others experience as overwhelming. They are not passive interventions such as taking a pill; they are programs that the patient must follow, with tasks to complete, records to maintain, and lessons to integrate. This can be deeply empowering for patients who desire greater control over their condition, but it can be exclusionary for those with limitations in digital literacy, technological access, or the capacity for sustained engagement.

The careful selection of appropriate candidates is an essential part of the art of digital prescribing. Not all patients are good candidates for digital therapies, regardless of the intervention's demonstrated efficacy in general populations. Physicians must assess patients' comfort with technology, their access to devices and the internet, their organizational ability to maintain usage routines, and their personal preference for treatment modalities. Prescribing digital therapy to a patient who does not own a smartphone or who cannot commit to regular use is as futile as prescribing an inhaler to a patient who cannot coordinate inhalation.

The relationship between digital prescribing and treatment adherence is complex and bidirectional. On the one hand, compared with traditional interventions, apps can incorporate gamification elements, smart reminders, immediate feedback, and peer support, which

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

substantially improve adherence. On the other hand, "digital fatigue" is a real and well-documented phenomenon: after weeks or months of use, many patients lose interest, ignore notifications, and abandon the therapy. The design of effective digital therapies must explicitly incorporate long-term retention strategies: variation in content, adjustment of intensity on the basis of progress, elements of the social community, and a planned transition toward autonomous maintenance.

17.3.5 PRIVACY, SECURITY, AND THE PERSONALIZED FUTURE

Data privacy and security take on critical dimensions in digital prescribing. The data that patients generate while using the therapy—symptoms, behaviors, usage patterns, locations, and interactions—are sensitive health data protected by regulations such as the GDPR or HIPAA, with severe ethical and legal implications if mishandled. The exchange of these data with the prescribing physician must be secured through encryption, explicitly consented to by the patient with an understanding of what is being shared, and strictly limited to what is necessary for clinical follow-up. Patients must retain control over their information, with the ability to review what data have been shared, with whom, and for what purposes and to revoke their consent if they change their mind.

The future of digital prescribing points toward ever-greater personalization through artificial intelligence. Digital therapies will be able to dynamically adapt to each patient's individual characteristics, learning pace, content preferences, and response patterns, modifying protocols in real time to maximize efficacy. Recommendation algorithms can suggest to physicians the most appropriate digital therapy for each specific case on the basis of accumulated evidence from thousands of previous prescriptions. In addition, digital prescribing evolves from a one-time event into a continuous and adaptive process, where the physician adjusts the digital "dose," frequency of interaction, content intensity, and level of complementary human support on the basis of real-time response data.

For developers of digital therapies, digital prescribing provides the most prestigious and sustainable marketing channel—formal medical endorsement—but imposes design requirements that many consumer applications ignore. Therapies must be designed with integration into clinical workflows in mind, as well as the generation of reports useful to physicians, ease of use for patients with varying levels of digital literacy, and navigation of complex reimbursement processes. They must generate clinical evidence that convinces physicians that incorporating them into their practice is worthwhile and that the results justify the effort of prescription and follow-up. It is a major challenge but also an opportunity to construct products that transcend mass marketing to become genuinely transformative medical interventions.

If we understand the architecture of digital prescribing, the question arises of how these business and intervention models are sustainable in the long term. How do we ensure that digital therapies are not just passing fads but are permanent components of the healthcare system? How do we build technology transfer mechanisms that allow innovations developed in one context to reach populations that need them in others?

17.4 SUSTAINABILITY AND TECHNOLOGY TRANSFER

> "The best algorithm in the world, abandoned *owing* to a lack of a maintenance plan, is a monument to failed planning, not a contribution to public health."

There is a painfully familiar scene in university hospitals and research departments around the world. A multidisciplinary team, combining clinicians, software engineers, and statisticians, spends years developing a genuinely promising prototype: an app that improves glycemic control in patients with type 2 diabetes; a deep learning algorithm for the early detection of diabetic retinopathy;and a decision support system that reduces medication errors in the emergency room. They secure public funding for initial development, publish impressive results in high-impact journals, generate excitement at international conferences, and raise expectations among patients and clinicians. However, when the research project ends, funding runs out, the original researchers disperse to new projects or institutions, and the prototype—which could have benefited thousands or millions of patients—is abandoned in a forgotten code repository, a victim of the lack of a sustainability plan. This unfortunately common story illustrates one of the most serious and least discussed problems in digital health innovation: the persistent gap between successful research and sustainable implementation that truly improves lives.

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

The sustainability of a medical software project is ne; it is neither a fortuitous accident nor an afterthought that can be planned once the product is finished; it is a property that must be explicitly designed from the earliest stages of conception. Sustainability means that the product can remain operational, relevant, and secure over time: updating when new security threats arise or underlying operating systems change; correcting errors when they are detected; adapting to changes in clinical practices and technological standards; and evolving to incorporate improvements on the basis of user experience. This means that there is a business or funding model that generates sufficient revenue or resources to cover these ongoing costs or an institution that explicitly and permanently assumes responsibility for maintenance. This means that there is a dedicated team, with institutional continuity, that has a deep understanding of the product and the capacity to evolve it. Without this built-in sustainability, even the most technically brilliant product is doomed to accelerated obsolescence and eventual abandonment, becoming a dead digital asset that takes up space on servers but provides no value to anyone.

17.4.1 FOUR PILLARS OF SUSTAINABILITY

The first pillar of sustainability is technical. Software, unlike pharmaceuticals, is not a stable chemical entity that remains unchanged over time; it is a complex system that interacts with constantly evolving technological ecosystems. It requires continuous maintenance: fixing bugs that emerge under unforeseen usage conditions, critical security updates in response to newly discovered vulnerabilities, adaptation to new versions of mobile or desktop operating systems that render previous ones obsolete, and integration with hospital information systems that evolve their interfaces and protocols. Many research projects dramatically underestimate these recurring maintenance costs, budgeting only for initial development and running out of resources when the first compatibility or security issues arise. Realistic technical planning must include, from the outset, conservative estimates of annual maintenance costs—typically between fifteen and twenty-five percent of the initial development cost—and an explicit strategy to secure these resources: maintenance contracts, subscriptions, committed institutional funding, or hybrid models.

The second pillar is organizational. The product needs a clear institutional “home”: a commercial company, a nonprofit foundation, or a dedicated hospital department that formally assumes responsibility for its continuity. This involves fundamental decisions regarding intellectual property, governance structures, and institutional leadership. Who makes strategic decisions about the product’s future? Who is legally and financially liable if serious problems arise? Who decides on the incorporation of new features, expansion into new indications, or withdrawal from the market when it is no longer viable? Without a defined and stable organizational structure, the product becomes orphaned, vulnerable to the shifting priorities of institutions that never formally committed to its maintenance, and exposed to discontinuation when the key individuals who supported it leave the project.

The third pillar is financial. The product requires predictable and sufficient revenue streams to cover the operational costs of maintenance and development. This revenue can come from multiple sources: direct product sales, institutional subscriptions, pay-per-use licenses, maintenance service contracts, public grants for sustained innovation and research, transactional research funds, or institutional support from hospitals or universities that consider the product strategic to their mission. The critical factor is that these revenues are diversified whenever possible and predictable over time. Relying exclusively on public research funding is inherently unsustainable in the long term because these sources are temporary by design, geared toward generating new knowledge rather than maintaining existing systems, and subject to political and budgetary cycles that do not respect the continuity needs of medical software.

The fourth pillar is humans. The product needs people who know it in depth, who are capable of diagnosing complex problems, and who can evolve it without breaking existing functionalities. Staff turnover is among the greatest risks to the sustainability of medical software. When the original developers leave—for other projects, other institutions, or other industries—tacit knowledge regarding design decisions, technical trade-offs, and edge-case behaviors is irretrievably lost, and maintaining the product becomes exponentially more difficult and risky. That is why it is crucial not only to thoroughly document the code and architecture but also to build teams with planned continuity, with onboarding processes that systematically transfer knowledge, with succession plans for critical roles, and with an organizational culture that values evolutionary maintenance as much as innovative development.

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

17.4.2 TECHNOLOGY TRANSFER

Technology transfer is the process by which a product developed in an academic or clinical research environment—driven by publication requirements, temporary funding, and proof-of-concept objectives—transitions to a sustainable production environment where it can be maintained, scaled, and benefit broad populations. It is a complex process involving legal, financial, organizational, and human dimensions and is, unfortunately, the point at which many promising projects meet their premature end—the "valley of death" between successful research and a sustainable product.

The first obstacle in the transfer is intellectual property. Who owns the source code? The university that employs the researchers? The hospital that provided the data and clinical context? The individual researchers who wrote the code? The company that partially funded the project through a research contract? Without legal clarity on the ownership of exploitation rights, it is impossible to negotiate commercial licenses, sell the product, seek venture capital investment, or even continue development in an organized manner. Research transfer offices (RTOs) at universities and hospitals exist precisely to help resolve these issues, but too many projects reach them too late, when ambiguities have already generated conflicts among stakeholders or when commercial windows of opportunity have closed.

The second obstacle is product maturity. Research prototypes, while functionally impressive in controlled demonstrations, are often far from being robust marketable products. They may have user interfaces that only their creators understand, lack technical or user documentation, be unprepared to scale to thousands of concurrent users, fail to meet regulatory security or privacy requirements, or depend on technological infrastructures that are not sustainable outside the research environment. Transition typically requires substantial "product engineering" work to transform the prototype into a robust, documented, scalable, and compliant system. This work entails significant costs that are rarely funded by original research grants and requires industrial software engineering skills that are not always available on academic research teams.

The third obstacle is real-world validation. A product that demonstrated efficacy in a controlled study with selected patients and optimized conditions has not necessarily been proven to work in actual clinical practice, with diverse users, varied contexts, and less-than-ideal conditions. Translation requires implementation studies that evaluate effectiveness under real-world conditions, adaptations to environments different from the original, and potentially new regulatory validations if the indication or target population is expanded. This additional validation process takes time—months or years—and money and resources that are not always available once the original research funding has ended.

The fourth obstacle is the shift in mindset from a research-driven approach to a business-driven approach. Researchers are accustomed to thinking in terms of hypotheses, publications, and contributions to public knowledge. Successful translation requires people with expertise in marketing, sales, financial management, customer service operations, and competitive strategy. It requires understanding the potential market, analyzing existing and emerging competition, identifying viable distribution channels, and building relationships with payers and regulators. Above all, it requires a mindset focused on long-term sustainability, return on investment, and the generation of measurable commercial value—which can conflict with the academic values of openness and public contribution to knowledge.

The fifth obstacle is managing the team. The researchers who developed the product may not be the right people to manage its commercialization: they may lack entrepreneurial skills, prefer to continue researching new ideas, or be unwilling to take on the risks and full-time commitment required to launch a company. Technology transfer often requires bringing in new managerial and commercial talent or licensing the product to an existing company with established commercialization capabilities. This can lead to emotional tensions and conflicts of interest if not managed with transparency and clear agreements from the outset regarding roles, compensation, and ownership.

17.4.3 TRANSFER MODELS

There are different models for technology transfer, each with its own advantages and trade-offs. The spin-off model—creating a new company specifically to commercialize the product—allows for maintaining ties with the original researchers, who typically retain equity stakes and advisory or executive roles and offers maximum flexibility to seek external investment,

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

scale operations, and pivot strategies on the basis of market insights. It does, however, require entrepreneurs willing to assume the financial and personal risk of dedicating themselves full-time to the project and the willingness of the originating institution to license or transfer intellectual property rights to the new entity.

The licensing model to an existing company, where the university or hospital transfers exploitation rights to an established company in exchange for royalties or upfront payments, is less risky for individual researchers but offers less control over the product's future. The licensee may decide not to commercialize the product if it does not fit its strategy, may modify it in ways that the original creators do not approve of, or may abandon it if commercial results do not meet expectations. L icense agreements must be carefully negotiated to include due diligence clauses that require effective exploitation and reversion mechanisms if the company fails to comply.

The institutional adoption model, where the hospital or university decides to maintain the product internally as its own service, can work for products that are highly specific to the institution and strategic to its clinical or academic mission. However, it is rarely sustainable in the long term because healthcare and academic institutions are not organized to operate software businesses: maintenance costs compete with other budget priorities, the lack of external revenue limits the capacity to invest in development, and when budget cuts occur, these nonessential products are typically the first to be sacrificed.

17.4.4 RESPONSIBILITIES AND THE FUTURE OF SUSTAINABILITY

For medical software developers, thinking about sustainability and handover from day one of the project is not pessimism but professional realism. The uncomfortable questions must be asked early on: Who will maintain this when I move on to another project or institution? Where will the resources for maintenance come from when research funding ends? What happens if the hospital decides it is no longer a strategic priority? If these questions have clear answers and contingency plans, the project is likely to move beyond the research phase; if they are ignored, the project is doomed to obsolescence regardless of its technical brilliance.

For clinicians considering the adoption of medical software tools, sustainability is an evaluation criterion that is just as important as proven efficacy. They should explicitly ask: What is the five-year maintenance plan? What guarantees are there that the product will continue to be updated and supported? What happens if the developer company disappears or is acquired? A product without a clear sustainability plan is a clinical risk: it may have uncorrected security flaws, it may stop working when operating systems change, or it may become obsolete relative to evolving standards of care. Caution regarding "orphaned" products is a professional responsibility.

The future of digital health sustainability depends on greater professionalization of technology transfer within academic and clinical institutions. Technology transfer offices are gaining experience, resources, and clear mandates to facilitate the transition from research to impact. Investment funds specializing in digital health are emerging that understand the sector's long cycles and complex regulations. Acceleration programs are being developed to provide researchers with the entrepreneurial skills and networks needed to launch successful spin-offs. Collaboration between academia and industry is becoming increasingly structured, with cocreation models in which research and commercialization are planned simultaneously from the outset.

The entire digital health innovation strategy: sustainability is not an end in itself but rather the necessary means to ensure that innovations actually reach the patients who need them and remain available for as long as they are needed. An unsustainable product, no matter how effective it may have been in research studies, is ultimately useless for public health. Sustainability is, ultimately, an ethical responsibility toward patients who rely on these tools, toward the professionals who integrate them into their practice, and toward society, which invests public resources in research expecting a return in the form of better health. To ignore it is to doom even the most technically brilliant projects to failure.

However, once we have ensured the technical and commercial sustainability of our product, we face a global regulatory maze where the rules of the game vary dramatically across continents. How do we navigate the landscape of regulatory agencies that determine whether we can market our medical software, under which risk classification, and with what evidence is required?

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

REFERENCES

Wang C, Lee C, Shin H. Digital therapeutics from bench to bedside. NPJ Digit Med [Internet]. December 1, 2023 [cited May 18, 2026];6(1):38. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC10006069/
Schliess F, Affini Dicenzo T, Gaus N, Bourez JM, Stegbauer C, Szecsenyi J, et al. The German Fast Track Toward Reimbursement of Digital Health Applications: Opportunities and Challenges for Manufacturers, Healthcare Providers, and People With Diabetes.J Diabetes Sci Technol [Internet]. March 1, 2022 [cited May 18, 2026];18(2):470. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC10973846/
Reilly L, Molnar A. Policy and reimbursement strategies to improve patient access to prescription digital therapeutics. NPJ Digit Med [Internet]. December 1, 2025 [cited May 18, 2026];8(1):740. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC12672704/

https://doi.org/10.62486/978-9915-9876-8-2.Ch17

Chapter 18 / Capítulo 18

Primum non nocere: Software engineering for medicine
ISBN: 978-9915-9876-8-2
DOI: 10.62486/978-9915-9876-8-2.Ch18
Pages: 290-308

Check for updates

© 2026 The authors. This is an open access book chapter distributed under the terms of the Creative Commons Attribution (CC BY) 4.0 License.

Chapter 18. Comparative Regulatory Landscape

Annier Jesús Fajardo Quesada
Eduardo Antonio Hernández González
René Herrero Pacheco
Rolando Javier Álvarez Pérez

1 Granma University of Medical Sciences. Granma, Cuba.
2 University of Medical Sciences of Pinar del Río. Pinar del Río, Cuba.

> "Medical software is not a global product simply because it is written in binary code. What is safe and effective in one country may not be in another, not because the software changes but because laws, criteria, and expectations change. Navigating the international regulatory maze is not an option for those who aspire to bring their products to patients around the world; it is a condition for survival."

Let us imagine that we developed an algorithm to detect diabetic retinopathy with exceptional performance. We've validated it in thousands of patients, obtained certification in our country, and are ready to bring it to market. Then, someone asks us: what about the FDA? What about the CE mark? What about Brazil's ANVISA? What about Mexico's COFEPRIS? We discover that every country has its own regulatory body, its own requirements, and its own timelines. What is a simple procedure in one place can be a long and costly process in another. The low-risk product in one country may require additional clinical trials in another. Regulatory harmonization is still a distant dream, and those who want to operate internationally must learn to navigate this complexity.

The regulatory landscape for medical software is a mosaic of standards, institutions, and procedures that reflect the diversity of healthcare systems, legal traditions, and political priorities. In the United States, the FDA has a well-established system with decades of experience. In Europe, the new Medical Device Regulation (MDR) has tightened requirements and centralized evaluation. In Latin America, each country has its own agency, with varying levels of maturity and stringency. Added to this is the recent regulation of artificial intelligence, such as the European AI Act, which adds an additional layer of complexity. Understanding this landscape is not a luxury; it is a strategic necessity.

In this chapter, we explore the main regulatory frameworks for medical software around the world. We will begin with the U.S. FDA, its product classification, and its approval pathways. We will continue with the European Medical Device Regulation (MDR) and its counterpart for in vitro diagnostics (IVDR). We address the AI Act, the first regulation specifically targeting artificial intelligence, and how it affects medical softwa. Finally, we provide an overview of regulations in Latin America and Spain, including their specific characteristics and challenges. This is not an exhaustive legal guide but rather an introduction to the questions every developer should ask and the resources they should consult.

18.1. FDA: CLASSIFICATION AND APPROVAL PATHWAYS

> "In the world of digital health, the FDA is not just another agency: it is the beacon that many other regulators follow, and its approval can mean the difference between a product that changes lives and one that never leaves the drawer."

The U.S. Food and Drug Administration, known simply as the FDA, exerts a disproportionate influence on the global regulatory landscape. This is no exaggeration: when the FDA makes a decision regarding a new medical device—from a diagnostic algorithm to a chronic monitoring app—its criteria resonate with those of agencies across the globe.(1) Many countries, especially in Latin America and Asia, view FDA approval as a seal of quality that facilitates their own authorization processes. For any development team with international ambitions, understanding how this agency works is not an academic luxury but a business survival skill. However, the FDA is not a monolithic, immutable entity; it is a complex organization with multiple centers, specialized divisions, and, most importantly, different regulatory pathways tailored to the risk level and nature of each product.

The relationship between the FDA and medical software has been a story of constant adaptation. It began in the 1980s, when the first programmable devices began appearing in U.S. hospitals. For years, the agency applied the same criteria to these new products as it did to traditional physical devices, which often led to confusing and inappropriate outcomes. A turning point came in 2017 with the publication of specific guidance on Software as a Medical Device (SaMD), which finally recognized that code is not a screw or a circuit but rather something with its own characteristics that requires a different framework. Since then, the FDA has continued to evolve, attempting to keep pace with rapid advances in artificial intelligence and machine learning, although technology sometimes advances faster than the regulatory capacity to adapt. (2)

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

18.1.1. THREE RISK CLASSES: THE FIRST STEP IN NAVIGATING THE SYSTEM

The absolute starting point for any developer looking toward the U.S. market is understanding how the FDA classifies medical devices. The agency organizes all products into three categories on the basis of their level of potential risk to the patient, and this classification determines everything that follows: the necessary documentation, timelines, costs, and business strategy. Class I encompasses the lowest-risk products—those that, if they fail, would cause minimal or no harm. In the world of software, this category includes administrative management systems, basic electronic health records, and appointment scheduling applications. These products are subject to general quality controls and good manufacturing practices but rarely require a comprehensive review by the FDA before being marketed, provided that they meet basic safety and privacy standards.(3)

Class II represents the category where most of the medical software currently being developed falls. This includes moderate-risk products—those that can influence important clinical decisions but do not sustain vital functions on their own. Diagnostic imaging support algorithms, applications that calculate drug doses on the basis of patient parameters, and clinical alert systems fall into this category. For these products, the FDA requires a premarket notification known as a 510(k), a process that requires demonstrating that the new device is substantially equivalent to another device already authorized and legally marketed in the United States. This equivalence refers not only to appearance or branding but also to safety and clinical efficacy under the same intended conditions of use.

Class III is the highest-risk category and, therefore, is subject to the most scrutiny. This class includes devices that sustain or maintain human life, implants, diagnostic products for life-threatening diseases with no alternative treatments, and those whose failure could cause serious injury or death. This class of medical software is relatively rare but not nonexistent: an algorithm that directly controls an insulin infusion pump or a fully autonomous diagnostic system for cancer could fall into this category. These products require a premarket approval (PMA), a rigorous process that demands robust clinical evidence, typically through randomized controlled trials, and can take years and millions of dollars to prepare.

Class determination is not a theoretical exercise that the developer undertakes alone; it is a decision that must be documented and that the FDA may challenge or modify. The manufacturer must carefully analyze the intended use of its product, the specific indications for which it is designed, and the risks associated with its normal operation as well as its potential failures. To facilitate this task, the FDA maintains an extensive public database of previously classified products, known as “predicates” or reference devices. These predicate devices serve as the compass guiding the regulatory strategy: if your product is substantially equivalent to one already on the market, you can follow the 510(k) pathway. If no suitable predicate exists or if your product is highrisk, you will face more complex processes.

18.1.2. APPROVAL PATHWAYS: FROM EQUIVALENCE TO RADICAL INNOVATION

Once the product class has been determined, the developer must choose and prepare the appropriate regulatory pathway, and this is where experience and specialized advice make the difference between a smooth process and a tortuous one. The 510(k) pathway is by far the most commonly used for Class II medical software, and its logic is relatively simple to understand, although complex to execute: You must demonstrate that your new product is substantially equivalent to a predicate device already on the market in terms of safety and effectiveness. This equivalence encompasses multiple dimensions: the indications for use must match or be more restrictive, the underlying technology must be comparable, and the clinical performance must demonstrate that it does not introduce significant new risks. New clinical trials are not always required for this pathway, but sufficient evidence is required—generally through concordance studies, analytical validation, or, in some cases, real-world performance data—to demonstrate that the equivalence is genuine and not merely declarative.

The 510(k) process has a variable duration that typically ranges from three to twelve months, depending on the product’s technical complexity, the quality of the submitted documentation, and the reviewers’ workload at the time. A common mistake among first-time developers is the underestimation of the importance of advance preparation: incomplete documentation, a poorly structured comparison with the predicate device, or an ambiguous definition of the intended use can lead to additional questions from the FDA that extend timelines by months or even years.

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

The agency sets time targets, but these are not guarantees, and a well-advised developer knows that investing in document quality from the first draft pays dividends in terms of approval speed.

For products that do not fit comfortably into any existing predicate device but also do not reach the level of risk that would justify a PMA, the FDA offers the de novo pathway, an alternative that has gained popularity with the proliferation of innovative software. This route allows the manufacturer to request a specific classification for its product, effectively establishing a new predicate for future competitors. It is a pathway that requires a more detailed evaluation than the standard 510(k), generally including a more thorough characterization of risks and benefits, but avoids the overwhelming burden of a PMA.(4) Success in a de novo application not only authorizes your product but also creates a new regulatory category that can pave the way for an entire emerging industry, as occurred with the first AI-based diagnostic imaging algorithms.

The PMA pathway represents the other end of the spectrum: it is the most demanding, the longest, the most expensive, and, fortunately, for most software developers, the least frequently needed. Robust clinical evidence, which is typically obtained through prospective controlled trials that convincingly demonstrate the product's safety and efficacy, is needed. The process includes multiple reviews by independent advisory committees, inspections of manufacturing facilities, and a significantly higher rejection rate than in the other pathways. With respect to medical software, except in very specific cases involving devices that support vital functions or autonomous diagnoses of serious diseases, this pathway remains a theoretical possibility rather than a common practical reality.

18.1.3. SPECIAL PROGRAMS AND THE CHALLENGE OF CONTINUOUS INNOVATION

Given that traditional processes could stifle innovation in a field evolving as rapidly as medical software does, the FDA has developed special programs in recent years designed to accelerate the market entry of promising products without sacrificing safety standards. The Breakthrough Devices Program offers fast track for products that provide more effective treatment or diagnosis of life-threatening or debilitating diseases or that offer significant advantages over currently approved alternatives.(5) Participation in this program does not guarantee approval, but it does provide priority access to engagement with the FDA, guidance on developing clinical evidence, and flexibility in study design—elements that can drastically reduce development timelines for truly transformative products.

Another forward-looking regulatory experiment is the Software Precertification Program, or Pre-Cert Program, a pilot initiative that represents a fundamental paradigm shift in how the FDA views its relationship with developers.(6) Instead of evaluating each product individually, this program proposes evaluating the developing organizations themselves on the basis of their quality processes, compliance history, postmarket monitoring capabilities, and culture of continuous improvement. The logic is that a company with robust and transparent processes can iterate and improve its products with lighter regulatory oversight, indicating that its own quality control systems will detect and mitigate risks. Although this program is still in the development and evaluation phases and its final scope remains uncertain, it points in a clear direction: the FDA is exploring models that recognize that software is not a static product manufactured once but a living system that evolves.

This evolutionary nature of software poses one of the most complex and as yet unresolved regulatory challenges: the management of updates. An artificial intelligence algorithm, by definition, can continuously improve its performance as it is exposed to new data. A clinical decision support system can add new features on the basis of user feedback. The FDA has published specific guidelines on when an update constitutes a major change requiring new regulatory authorization and when it may be considered minor and managed internally by the manufacturer. In general, changes that directly affect clinical performance, introduce new indications for use, or modify the target population require a new review by the agency. Changes that improve usability without altering clinical logic, correct minor errors, or optimize technical performance without impacting safety may be implemented with appropriate documentation but without additional prior authorization.

Artificial intelligence and machine learning have been a particular focus of the FDA in recent years. In 2021, the agency published an action plan for the regulation of AI/ML that establishes five strategic priorities: fostering a culture of quality in developer organizations, gathering real-world evidence on the performance of these algorithms, developing evaluation methodologies adapted to the adaptive nature of AI, ensuring transparency toward users and patients, and establishing

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

mechanisms for continuous postmarket monitoring.(7) The FDA has already approved hundreds of algorithms based on these technologies, but its approach goes beyond initial approval: it expects manufacturers not only to demonstrate that their algorithm works at the time of launch but also to establish systems to keep it functioning correctly over time, detecting and correcting performance deviations that may arise owing to changes in input data, clinical practice, or the patient population.

18.1.4. LONG-TERM RELATIONSHIPS: BEYOND INITIAL APPROVAL

Preparing an application for the FDA is a meticulous process that begins long before the first form is drafted. It requires thoroughly documenting the entire development process, from initial requirements to final validation testing, including systematic risk analyses per ISO 14971 and usability testing per IEC 62366. It requires a crystal-clear and well-defined intended use because any ambiguity at this stage can result in a classification different from what was expected or in unexpected market restrictions.(8) It requires identifying and analyzing the relevant predicate devices and preparing detailed comparisons that anticipate reviewers' questions. It requires being prepared for a dialog that can extend for months, answering technical questions, clarifying methodologies, or providing additional data that the agency deems necessary for its evaluation.

However, the relationship with the FDA does not end with the issuance of the letter of authorization. Manufacturers must maintain robust postmarket surveillance systems, reporting to the agency on significant changes to the device, on adverse events that meet certain criteria for severity or frequency, and on updates that may affect safety or efficacy. This ongoing reporting obligation is particularly relevant for software, where the frequency of updates is inherently higher than that for physical devices. A developer who ignores this postmarket dimension exposes themselves to regulatory actions that may include product recalls, significant fines, or bans on future marketing.

For healthcare professionals who use these products, understanding the meaning of different FDA approvals is an increasingly necessary digital literacy skill. A 510(k) approval does not imply that the product has been tested in prospective clinical trials in the same population that the clinician treats; it guarantees only that it is substantially equivalent to another product already on the market, whose own evidence may date back decades. De Novo approval offers greater assurances of specific evaluation, while a PMA provides the highest level of premarket clinical evidence. Products designated as breakthrough products have been explicitly recognized by the FDA as representing significant advances. All this information is publicly available in the FDA's databases, and clinicians who know how to interpret it can make more informed decisions about which technologies to adopt in their practice.(4)

The future of medical software regulation in the United States is taking shape in two directions that appear contradictory but are actually complementary: toward greater international harmonization and toward an increasingly sophisticated adaptation to the unique characteristics of software as a distinct product. The FDA actively participates in the International Medical Device Regulators Forum (IMDRF), collaborating with its counterparts in Europe, Canada, Australia, Japan, and other countries to align criteria and facilitate global access to healthcare innovation. Moreover, it explores specific regulatory frameworks for artificial intelligence that will likely serve as a model for other regulators in the coming years, recognizing that software cannot be treated as a physical device with a defined lifespan but rather as a complex system that evolves, learns, and requires continuous monitoring.

Ultimately, it is worth remembering that the FDA, with all its complex processes and seemingly bureaucratic requirements, is not an arbitrary obstacle but an institutionalized safeguard. Its requirements exist to protect patients from unsafe, ineffective, or poorly designed products, and approval from this agency remains among the most recognized and respected seals of quality in the healthcare world. For the strategic developer, investing in doing things right from day one, rather than seeking regulatory shortcuts, is not just a matter of ethical compliance but of business acumen. At the end of the day, what truly matters is not simply obtaining authorization to place a product on the market but reaching that market with a solution that doctors and patients can fully trust, knowing it has been evaluated against one of the most demanding standards on the planet.

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

18.2. EUROPEAN MDR AND IVDR REGULATIONS

In recent years, the European regulatory landscape has undergone a transformation of historic magnitude, comparable in its impact only to the creation of the single European market itself. In May 2017, two legal texts published almost simultaneously radically altered the terrain on which all medical device manufacturers operate, explicitly and for the first time, systematically including medical software. Regulation (EU) 2017/745, known as the MDR (Medical Device Regulation), and its counterpart Regulation (EU) 2017/746, known as the IVDR (In Vitro Diagnostic Regulation),(9) are not mere technical updates to previous regulations; they represent a complete overhaul of the European regulatory framework, replacing directives that had governed the sector for more than two decades with significantly stricter requirements, more intrusive control mechanisms, and unprecedented levels of transparency through the European database EUDAMED.

The distinction between a directive and a regulation, seemingly subtle to those unfamiliar with EU law, has enormous practical consequences for medical software developers. A European directive sets out objectives that each member state must achieve but leaves room for countries to choose how to implement them in their national legislation, which has historically led to differences in interpretation, uneven timelines, and, occasionally, gray areas that manufacturers could exploit. A regulation, on the other hand, is directly applicable and binding in all Member States from the moment it enters into force, without the need for national legislative transposition. This means that the rules are exactly the same in Stockholm, Seville, or Sofia, which facilitates market harmonization but eliminates any possibility of seeking out jurisdictions with more permissive interpretations or less stringent requirements.

The new regulations did not arise from a legislative vacuum or a mere bureaucratic desire to complicate the market; they were a response to profound crises of confidence that shook the pre ystem to its foundations. The PIP breast implant scandal, in which it was discovered that a French manufacturer had used unauthorized industrial silicone gel instead of approved medical-grade material, revealed catastrophic failures in the oversight of notified bodies and in postmarket surveillance. Similar cases involving defective hip implants and other problematic medical devices demonstrated that the previous CE certification system, while functional in many respects, had structural cracks that allowed unsafe products to reach patients for years before being detected. That is why the MDR and IVDR place a renewed and almost obsessive emphasis on total transparency, the absolute traceability of every device from manufacture to end use, and continuous surveillance that does not end with initial certification.

18.2.1. THE FIVE PILLARS OF CHANGE: WHAT THE NEW FRAMEWORK TRULY TRANSFORMS

For medical software development teams, the transition from the previous framework to the new MDR and IVDR regime involves the assimilation of five structural changes that profoundly alter how a digital medical device is conceived, developed, and marketed. The first pillar—and perhaps the one that has generated the most surprises among developers accustomed to the previous regulation—is the reclassification of software risk. Under the previous directives, much of the medical software was typically classified into low-risk classes, often Class I, which meant that the manufacturer could self-assess its conformity without third-party intervention. The new regulations establish specific and much more restrictive rules for software that, in practice, substantially increase the classification of most digital products with real clinical impact.

Rule 11 of the MDR is particularly decisive in this regard: it establishes that any software intended to provide information used to make decisions for diagnostic or therapeutic purposes is automatically classified as at least Class IIa. If these decisions can have serious consequences for the patient's health or even survival, the classification increases to Class IIb or, in the most extreme cases, to Class III. This means that diagnostic imaging support algorithms, clinical alert systems, medication dosing applications, or physiological signal interpretation software—which could previously be marketed on the basis of a manufacturer's self-assessment—now inevitably require the intervention of a notified body, with the associated costs, timelines, and complexities. The transition has forced many companies to reconsider their business models or even withdraw products from the European market.

The second transformative pillar is the radical strengthening of the role of notified bodies—those private entities designated by the national authorities of Member States that act as independent conformity assessors for medium- and high-risk products. Under the MDR, the requirements for being designated as a notified body have been drastically tightened, demanding

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

greater technical capacity, more medical specialists on staff, more rigorous audit procedures, and stricter financial independence from the manufacturers they assess. Many bodies that operated under the old directive have failed to renew their accreditation, creating a severe bottleneck in the system: fewer entities are available to assess more products that, moreover, now require mandatory certification because they have been reclassified. Waiting times for an evaluation appointment have multiplied, and manufacturers have learned the hard way that planning for certification years in advance—not months—is now a matter of survival.

The third pillar introduces a new, mandatory professional role: the person responsible for regulatory compliance (PRRC). Every manufacturer must designate at least one person with specific qualifications to assume legal responsibility for ensuring that each product complies with all applicable regulatory requirements. This person must have a university degree in law, medicine, engineering, or related sciences and demonstrate prior practical experience in medical device regulation. This is not merely an external consultant who reviews documentation occasionally; it is a key figure who must be permanently available to notified bodies and competent authorities, who must actively participate in decisions regarding significant product changes and whose absence or incompetence may result in the suspension of the marketing authorization.

The fourth pillar exponentially increases the requirements for clinical evaluation, understood as the systematic collection, evaluation, and analysis of clinical data related to a product to verify its safety and clinical performance in accordance with its intended use. Whereas previously technical validation tests in the laboratory or a claim of substantial equivalence with products already on the market might have sufficed, the MDR requires manufacturers to actively demonstrate that their product fulfills its specific clinical purpose on the basis of sufficient and appropriate evidence. For medical software, this often translates into the need to conduct prospective clinical studies specifically designed for the product, not just technical performance tests or algorithm validation against historical datasets. The possibility of claiming equivalence with existing products has been significantly restricted, and the required standards for the quality and relevance of clinical data are notably more ambitious.

The fifth pillar is transparency operationalized through EUDAMED, the European database for medical devices, which, once fully operational, will radically transform the sector's visibility. Manufacturers must register each product with its unique identifier, essential technical characteristics, intended use, and classification. Notified bodies must publish the certificates of conformity issued and their exact scopes. Serious incidents and safety field actions must be reported in real time through the system.(10) Crucially, EUDAMED will be accessible to the general public, not just professionals, allowing any patient or clinician to check which products are legally authorized, what safety incidents have been reported, and what clinical studies support their marketing. Although the full implementation of EUDAMED has faced significant delays because of technical complexities, its eventual operation will eliminate the shadows where deficiencies could previously be hidden.

18.2.2. IVDR: THE SISTER REGULATION FOR IN VITRO DIAGNOSTICS

While the MDR has garnered the most media and academic attention, its counterpart, the IVDR, poses an equal or greater regulatory challenge for a specific but growing segment of medical software: that intended for the analysis of biological samples, the interpretation of clinical laboratory results, diagnosis on the basis of molecular or genetic biomarkers and the monitoring of biochemical parameters. The In Vitro Diagnostic Medical Devices Regulation is, in many respects, even more demanding than the MDR because it has massively reclassified products previously considered low- or moderate-risk into categories requiring mandatory evaluation by a notified body. Laboratory tests that could previously be marketed with minimal oversight, such as certain genetic tests or tumor markers, now require rigorous certification through specific clinical studies demonstrating their sensitivity, specificity, and predictive value in real-world populations.

For software that processes, interprets, or reports in vitro diagnostic results, the implications are profound. An algorithm that analyzes digital histopathology images, a system that interprets genomic sequencing results, or an application that calculates genetic risks on the basis of SNP profiles now falls under the scope of the IVDR with very strict clinical validation requirements. The risk classification for this type of software tends to be high by default, especially when the results influence clinical management, therapeutic, or public health decisions. The manufacturer must not only demonstrate that its algorithm works technically but also that its clinical interpretation is correct, reproducible, and clinically useful in the care setting where it will be used.

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

The transition from the old In Vitro Diagnostic Directive to the IVDR has been particularly traumatic for the sector because of the magnitude of the reclassification. Thousands of products that were legally circulating under manufacturer self-assessment have suddenly required third-party certification, further overwhelming the available notified bodies. The shortage of evaluation capacity has been especially acute in this area, forcing transitional extensions that have prolonged regulatory uncertainty. For diagnostic software developers, this has meant that planning commercial launches has become extremely difficult, with market availability dates depending more on obtaining certificates than on the completion of technical development.

The interaction between the MDR and the IVDR creates areas of additional complexity when software combines functionalities from both regulations. A system that collects data from an in vitro diagnostic device, processes it using its own algorithms, and generates therapeutic recommendations may be subject simultaneously to requirements from both legal texts. Determining which one takes precedence, or whether both apply in a complementary manner, requires sophisticated regulatory analysis and often formal consultations with the competent authorities. This duality reinforces the need for specialized advice and regulatory planning that begins in parallel with technical design, not as a final phase prior to launch.

18.2.3. ADAPTING TO THE NEW PARADIGM: OPERATIONAL IMPLICATIONS FOR DEVELOPERS

The transition to MDR and IVDR compliance demands a profound shift in mindset for medical software developers that goes beyond merely updating documentation procedures. It is no longer feasible to develop a complete product, validate it technically, and only then seek regulatory certification as a final step. The new regulations require the integration of regulatory requirements from the initial design phase, the so-called “quality by design” approach. Clinical evaluation must be planned years in advance because prospective studies demonstrating clinical performance under real-world conditions require time for design, recruitment, execution, and analysis. Documentation is not an afterthought but a living, audited record of every decision made during development.

Software change management takes on a complex regulatory dimension under the new framework. In the previous system, many manufacturers, especially in the agile world of software, were accustomed to continuous update cycles, deploying new versions weekly or even daily, often without explicitly notifying the authorities of every minor modification. The MDR clearly states that significant changes to the device’s design or intended use require a new assessment by the notified body, which can drastically slow the pace of innovation. The distinction between major changes requiring recertification and minor changes that the manufacturer can manage autonomously is not always clear and requires organizations to develop formal internal procedures to make these decisions, document them thoroughly, and defend them during audits.

The practical implementation of the MDR and IVDR has, it must be acknowledged, been problematic and fraught with unforeseen difficulties. The original deadlines have been repeatedly delayed because of the lack of preparedness of many notified bodies to meet the new accreditation requirements and because of the technical complexity of implementing EUDAMED as a functional pan-European system. Many products, especially from small and medium-sized manufacturers, have had to be withdrawn from the European market because their certificates under the previous directive expired before they could obtain the new MDR certification. Small businesses, innovative startups, and individual developers have been disproportionately affected by increased certification costs, bureaucratic complexity, and the need to hire specialized staff or costly external consultants.

Despite these implementation hurdles, the strategic direction is unequivocal, and there is no turning back. Europe has decided that it wants medical devices, including medical software, that are safer, better monitored, and more transparent, and manufacturers wishing to access the market of nearly 500 million consumers will have to adapt to these requirements. Those who succeed will find, paradoxically, a more predictable and uniform market than under the old directive, where certification in one member state guarantees access to all and where EUDAMED’s transparency can become a competitive advantage for truly robust products against lower-quality competitors.

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

18.2.4. CLINICIAN'S PERSPECTIVE AND FUTURE REGULATORY CONVERGENCE

For healthcare professionals who use medical software in their daily clinical practice, the new MDR and IVDR framework offers additional safeguards that they must know how to interpret correctly. When a clinician now sees a product with the CE mark, they must pay attention to which regulation underpins that mark: if it is under the new MDR or IVDR, they will know that the product has undergone a substantially more rigorous evaluation than under the old directive, with more demanding clinical studies, more intrusive supervision by a notified body, and stricter postmarket surveillance obligations. Once fully operational, the EUDAMED database will allow you to consult the product's complete history: when it was certified, by which notified body, which clinical studies support its marketing, what safety incidents have been recorded, and what corrective actions the manufacturer has implemented.

However, clinicians will also need to develop regulatory expertise to navigate the transitional periods during which products certified under the old directive and under the new regulation coexist. During these years of transition, the meaning of the CE marking is not uniform; a product with an old certificate may have been evaluated under less stringent standards than one recently certified under the MDR. This temporary duality requires healthcare professionals, especially those involved in hospital procurement or the selection of clinical tools, to actively inquire under which regulatory framework each product was certified and to understand the implications of that difference.

This growing complexity, although intimidating for developers and clinicians alike, serves a noble and shared ultimate purpose: to build a digital health system in which patients can fully trust, where professionals have sufficient information to decide which tools to adopt, and where responsible innovators who invest in real quality can compete on equal footing against those seeking regulatory shortcuts. The radical transparency imposed by EUDAMED, the clinical rigor demanded by the new regulations, and the mandatory role of the compliance officer are mechanisms designed to ensure that patient safety does not depend on the good faith of manufacturers but rather on robust systems of independent verification. In this context, medical software will cease to be a special, less-regulated category and will become a medical device in its own right, with the same guarantees and standards as a cardiac implant or a biologic drug.

18.3. AI ACT: WHEN THE ALGORITHM BECOMES A REGULATED PRODUCT

The European Union has taken a regulatory step unprecedented in the global history of technology: it has created the world's first comprehensive law dedicated exclusively to regulating artificial intelligence as a specific category, with its own risks, obligations, and oversight mechanisms. The Artificial Intelligence Regulation, universally known as the AI Act, does not treat AI as just another technology among many others but rather recognizes that algorithmic systems possess unique characteristics—such as autonomous learning capabilities, inherent opacity, potential for systematic bias, and critical dependence on data quality—that warrant their own regulatory framework. Approved in 2024 after years of intense debate, this regulation establishes a system of obligations proportional to the level of risk each system presents, creating a new regulatory layer that overlaps and complements existing sector-specific regulations such as the MDR and the IVDR.(11) For any team developing medical software with artificial intelligence components, understanding this regulation is no longer a strategic option but a legal necessity for survival.

It is essential to understand from the outset that the AI Act does not replace or supersede sector-specific medical device regulations; rather, it complements them and, in many respects, reinforces them. An artificial intelligence system used in medical diagnosis, clinical decision support, or patient monitoring must simultaneously comply with all MDR requirements as a medical device and with the specific obligations of the AI Act as a high-risk system. This dual regulation may seem, at first glance, like an overwhelming bureaucratic burden that unnecessarily complicates the path to the market. However, for strategically minded developers, this overlap also offers opportunities for differentiation: those who master both frameworks will be able to navigate with an advantage in an ecosystem where many competitors will be left behind by the complexity. The AI Act is genuinely new because it provides a perspective focused on the specific risks of artificial intelligence that traditional health regulations address in a generic or indirect manner.

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

18.3.1. THE RISK PYRAMID: CLASSIFICATION AND SCOPE IN MEDICINE

The cornerstone upon which the entire AI Act is built is a system for classifying artificial intelligence systems into four levels of increasing risk, each with specific and proportionate obligations. At the base of the pyramid are systems with minimal risk, such as video games that use AI to generate the behavior of nonplayable characters, email spam filters, or optical character recognition applications for digitizing administrative documents. These systems have no specific obligations beyond the general ones applicable to any digital product and represent the vast majority of everyday applications of artificial intelligence. At an intermediate level are limited-risk systems, such as customer service chatbots, which must primarily comply with transparency obligations, clearly informing the user that they are interacting with a machine and not a human.

The top tier—and where the true regulatory transformation lies—is occupied by high-risk systems. These are defined as those that can significantly affect people's health, safety, or fundamental rights, and the regulation includes specific lists in Annexes II and III detailing the categories automatically considered highrisk. For the healthcare sector, the classification is clear-cut and virtually all-encompassing: Annex III explicitly includes artificial intelligence systems used in medical diagnosis, clinical decision support, real-time patient monitoring, management of critical hospital resources, clinical research, and health insurance management systems that determine eligibility or premiums. This means that the vast majority of medical software incorporating any artificial intelligence component—from an algorithm for detecting diabetic retinopathy to a sepsis prediction system in intensive care units—will automatically be considered a high-risk system and must comply with the full set of the regulation's most stringent obligations.

At the apex of the pyramid, completely separate and prohibited, are systems of unacceptable risk. These include practices considered contrary to the fundamental values of the European Union, such as social scoring systems that classify citizens on the basis of their behavior, subliminal manipulation systems that operate beyond people's awareness, or emotional recognition systems in workplaces and institutional education settings for classification purposes. Although these prohibitions seem far removed from the medical field, developers must remain vigilant: a system that evaluates the "cooperation" of chronic patients to adjust their priority on waiting lists, or that scores the behavior of medical residents, could skirt these prohibited boundaries if not designed with extreme ethical care.

The practical implication of this classification for medical software is that there is virtually no room for medium- or low-risk artificial intelligence systems in the clinical setting. With the exception of purely administrative applications with no impact on healthcare decisions, every medical algorithm will be highrisk by legal definition. This eliminates the temptation to seek favorable classifications and establishes a uniformly high regulatory floor for the entire industry. Developers must assume from the product's conception that they will be subject to the strictest regime, which dictates technical architectures, development budgets, clinical trial plans, and documentation strategies.

18.3.2. SIX FUNDAMENTAL OBLIGATIONS OF A HIGH-RISK SYSTEM

Being classified as a high-risk artificial intelligence system entails assuming a set of six fundamental obligations covering the entire product lifecycle, from conception to market withdrawal. The first of these obligations concerns transparency and comprehensive technical documentation. Manufacturers must create and maintain a documentation dossier that verifies how the system was developed, which design and validation methodologies were used, how AI-specific risks were managed, and how each applicable requirement of the regulation is met. This documentation is not mere archival formality; it must be permanently available to national supervisory authorities, who may request it at any time and must be updated with every significant change to the system. The level of detail required far exceeds standard practices in agile software development, forcing organizations to adopt rigorous document management processes.

The second fundamental obligation is data governance, a critical pillar that recognizes that the quality of an artificial intelligence system cannot exceed the quality of the data with which it is trained and validated. High-risk systems must be developed using training, validation, and test datasets that are relevant to the intended use, representative of the actual target population, and free from discriminatory biases to the greatest extent possible. Manufacturers must meticulously document the origin of all the data; the demographic and clinical characteristics of the population, including exclusions and their justifications; and the technical and methodological measures

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

adopted to detect, quantify, and correct biases. In medicine, where algorithmic biases can result in misdiagnoses for minority groups or systematic exclusions of certain patient profiles, this obligation has far-reaching ethical and legal implications.

The third obligation focuses on transparency for end users, which is distinct from documentation for authorities. Systems must be designed so that physicians, nurses, patients, or caregivers who are interacting with them are fully aware that they are dealing with artificial intelligence, understand its actual capabilities and specific limitations, and have sufficient information to exercise critical judgment. This involves providing clear, nontechnical instructions for use that explain the system's expected performance, validation metrics in relevant populations, known circumstances in which the system may not function properly, and warning signs indicating that the algorithm's output should be questioned. This operational transparency is essential for human oversight to be effective rather than merely nominal.

The fourth obligation establishes strict requirements for effective human oversight. High-risk systems must be technically designed to allow competent humans to monitor their operation, intervene at any point in the decision-making process, have the actual ability to override automated decisions when they deem it appropriate, and completely halt the system's operation if they detect anomalous or risky behavior. Human oversight is not a cosmetic addition or a legal fig leaf; it is a functional safety requirement that must be integrated into the system's architecture from the initial design phase. In the medical context, this means that the algorithm may recommend, alert, or suggest, but the final clinical decision must always rest with a human healthcare professional who has the information, training, and technical authority to disregard or modify the algorithmic recommendation.

The fifth requirement demands adequate levels of technical robustness, accuracy, and cybersecurity. Systems must achieve clinically validated performance metrics and be resilient to input errors, adversarial attacks, variations in the quality of input data, and gradual performance degradation over time. They must incorporate automatic mechanisms for detecting deviations in their operation, alerts when performance falls below predefined thresholds, and the ability to adapt or undergo controlled retraining when necessary. This requirement for continuous performance monitoring is particularly relevant for medical artificial intelligence, where a model can degrade silently as patient populations, clinical reference practices, or the characteristics of data acquisition equipment change.

The sixth requirement mandates a formalized risk management system specific to artificial intelligence. Manufacturers must establish processes that identify, assess, mitigate, and continuously monitor the risks associated with the system's use, including not only technical risks of malfunction but also risks of misuse by inadequately trained operators, risks of discriminatory bias, risks to privacy and data protection, and risks of overreliance or deskilling among professionals using the system. This risk management system must be proactively updated throughout the product's commercial lifecycle, incorporating lessons learned from incidents, new scientific evidence, and changes in the clinical environment of use.

18.3.3. GOVERNANCE, SANDBOXES, AND THE SURVEILLANCE ECOSYSTEM

The AI Act established a multilayered governance system of considerable institutional complexity. The national market surveillance authorities of each member state—generally the same bodies that already regulate medical devices—will be primarily responsible for monitoring compliance with the regulation within their territory. At the European level, a European Artificial Intelligence Committee coordinated by the Commission is established, which will be responsible for harmonizing interpretations, resolving conflicts between national authorities, and proposing updates to the regulation as technology evolves.(12) Notified bodies—those accredited private entities that already assess MDR and IVDR compliance—will also assume responsibility for evaluating compliance with the AI Act's requirements where applicable, creating institutional continuity that facilitates regulatory integration for manufacturers.

A particularly valuable development for the innovation ecosystem is the institutionalization of regulatory testing environments, known as regulatory sandboxes. These spaces allow developers to test innovative artificial intelligence systems under real-world but controlled conditions under the active supervision of competent authorities before the full burden of regulatory compliance for general marketing is assumed. With respect to medical software, where traditional clinical validation can be extremely costly and time-consuming, sandboxes offer a unique opportunity to generate real-world evidence of safety and efficacy with mitigated risks. Member states are

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

required to establish these sandboxes and actively encourage their use by developers, particularly small and medium-sized enterprises, startups, and public sector entities. Participation in a sandbox does not exempt participants from future compliance, but it allows them to accumulate operational experience and performance data that greatly facilitate the preparation of final documentation.

Another significant innovation is the formalized obligation to report serious incidents. When a high-risk artificial intelligence system causes or contributes to serious harm to human health, a substantial violation of fundamental rights, or significant damage to property or the environment, the manufacturer must notify the competent authorities without undue delay. This obligation complements the existing one in the MDR for incidents involving medical devices, creating a more comprehensive and multidimensional surveillance system. Reported incidents must be thoroughly investigated; findings must be shared with other operators and authorities to prevent recurrence; and, when necessary, safety field actions must be implemented, which may include corrective updates, usage restrictions, or market withdrawals.

For medical software developers, adapting to the AI Act undoubtedly represents a considerable challenge. The additional requirements translate into more technical documentation to maintain, more data governance processes to implement, higher certification costs to bear, and greater organizational complexity to manage. Small businesses and startups, particularly those with small teams and limited resources, may feel disproportionately affected by this regulatory burden, with the real risk that the barrier to entry into the European market will rise so high that it favors large corporations with specialized regulatory departments. However, the compensatory opportunity is substantial: the AI Act creates, for the first time, a common, clear, and predictable framework across the European Union, where products that demonstrate compliance will have a credible quality differentiator compared with competitors from less-regulated markets.

18.3.4. INTERACTION WITH THE HEALTHCARE FRAMEWORK AND FUTURE OUTLOOK

The relationship between the AI Act and sector-specific medical device regulations, particularly the MDR, is not without gray areas and practical complexities that the first years of implementation will clarify. In some respects, the requirements of both regulations overlap redundantly: both require risk management, both require postmarket surveillance, and both mandate comprehensive technical documentation. In other respects, they complement each other usefully: the MDR provides expertise in the clinical evaluation of medical devices, while the AI Act provides specificity regarding the unique risks of artificial intelligence, such as algorithmic transparency, bias mitigation, and human oversight. The European Commission has published guidance documents to help manufacturers navigate this interaction, but some uncertainty inevitably persists during the first few years as interpretive practices and administrative case law take shape.

What is indisputable is that no manufacturer of medical software with artificial intelligence components can afford to ignore either of these frameworks. Products must cumulatively comply with all applicable requirements of both regulations, which demands integrated regulatory planning from the earliest stages of development. Development teams must include hybrid regulatory expertise encompassing software engineering, traditional healthcare regulations, and the specific governance requirements of artificial intelligence. Here, the role of the P r MDR Compliance Officer takes on additional dimensions of technical complexity.

The future of the AI Act's application in the medical field will depend critically on how it is implemented in practice and how artificial intelligence technology itself evolves. The regulation is deliberately designed as an adaptive framework, with built-in periodic review mechanisms and the flexibility to update the annexes defining high-risk systems without needing to reopen the entire legislative text. As new algorithmic architectures, foundational multimodal language models, generative AI systems with clinical applications, and federated learning algorithms operating on distributed data emerge, the regulation will require evolving interpretations or specific amendments. International cooperation will also be crucial: the European Union has been a pioneer, but other economic blocs are developing their own regulatory frameworks, creating a global mosaic that international manufacturers will need to navigate with agility.

In conclusion, it is important to bear in mind that the AI Act is not a regulatory end in itself but rather an instrumental means to achieve higher social goals: ensuring that artificial intelligence is developed and deployed in a manner that scrupulously respects fundamental rights, human dignity, and the physical and psychological safety of individuals. In the field of medicine, where

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

algorithmic decisions can determine diagnoses of serious illnesses, painful treatments, or even the continuation of life, this ethical imperative takes on particular urgency. Stringent regulation is not an irrational obstacle to technological innovation; it is the necessary condition for that innovation to be socially sustainable, clinically reliable, and ethically legitimate. Developers who understand this underlying logic and work constructively with regulation, rather than seeking shortcuts or jurisdictions with lower standards, will inevitably lead the future of digital health in Europe and, likely, the world.

18.4. OVERVIEW IN LATIN AMERICA AND SPAIN

The regulatory journey for medical software does not end at the FDA offices in Maryland or with the European Union's notified bodies. For companies hoping to bring their diagnostic algorithms, chronic condition monitoring apps, or clinical decision support systems to Spanish-speaking markets, the landscape becomes considerably more complex, fragmented, and often unpredictable. Latin America, in particular, is by no means a homogeneous bloc; it is a mosaic of twenty countries with levels of regulatory development ranging from sophisticated to nascent, with technical requirements that vary dramatically and with approval processes that can range from relatively streamlined to extremely bureaucratic and opaque. Spain, for its part, although geographically European and culturally Latin, presents its own unique challenges in applying EU regulations but with local particularities in implementation, healthcare financing, and technology adoption that the foreign developer ignores at their own risk. Understanding this landscape in depth is not a strategic luxury for ambitious companies; it is an inescapable operational necessity for any organization seeking to operate with sustainable success in these markets.

Let us begin by examining Spain, which, owing to its population of forty-seven million, has a geographical position as a natural bridge between Europe and the Americas and its historical role as a hub for the dissemination of Spanish-language culture, often serves as the preferred gateway to the Spanish-speaking market for many healthcare technology companies. As a full member of the European Union, Spain fully implements the European Medical Device Regulation (MDR) and the In Vitro Diagnostic Regulation (IVDR) without national transposition, exactly as Germany, France, or Italy do. This means that any medical software intended for sale in Spain must bear the CE marking in accordance with these regulations, obtained in advance through a European notified body accredited for the evaluation of products in its risk class. There is no additional national premarket health authorization process; once the manufacturer holds a valid CE mark, they may legally market their product throughout the European Economic Area, including Spain, without any additional health registration procedures.(13)

14.4.1. SPAIN: BEYOND THE CE MARKING, THE MAZE OF PUBLIC FUNDING

However—and this nuance is crucial for actual commercial viability—CE marking represents only half the battle in the Spanish healthcare system. For a digital health product to be funded by the public National Health System and thus accessible to the vast majority of patients at no direct cost, it must undergo additional processes of economic evaluation, budget impact analysis, and a formal decision on inclusion in the National Health System's common portfolio of services or in the service portfolios of the autonomous communities. The Spanish Agency for Medicines and Health Products (AEMPS) acts as the competent authority for the regulation, surveillance, and safety of medical devices, but the decision on public funding rests with the General Directorate of the Common Portfolio of Services of the National Health System within the Ministry of Health and ultimately with the health departments of the seventeen autonomous communities, which directly manage virtually all public healthcare spending. This process of evaluating health technologies and deciding on funding can take years, involves political competition between regions, and is not guaranteed even when the product bears the CE mark and demonstrates clinical superiority over existing alternatives.

AEMPS, aware that traditional regulatory rigidity can stifle digital innovation, has developed specific initiatives in recent years to facilitate the entry of software-based medical devices into the Spanish market. Of particular note is the regulatory sandbox program for digital health, which allows developers to test their products in controlled and real-world clinical settings under the agency's active supervision before undertaking the full burden of formal certification. This regulatory testing environment offers a valuable pathway to generate Spanish clinical evidence,

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

validate cultural or linguistic adaptations of the product, and establish a constructive relationship with national regulators. The AEMPS has also published specific guidelines on software as a medical device, clarifying risk classification criteria, interpretations of MDR Rule 11 for algorithms, and specific documentation requirements. For developers, especially those whose products are genuinely innovative and do not fit comfortably into existing regulatory categories, establishing an early and ongoing dialog with the AEMPS can prove strategically invaluable.(13)

Another aspect of growing importance in the Spanish ecosystem is the role of regional health technology assessment agencies. Regions such as Catalonia, through the Agència de QualitatiAvaluacióSanitàries de Catalunya (AQuAS), the Basque Country with its agency Osteba, or Andalusia with the Agencia de Evaluación de TecnologíasSanitarias (AETSA), maintain their own medical device evaluation processes that can decisively influence the adoption and funding of the product within their respective territories. These regional agencies evaluate not only clinical efficacy and safety but also cost-effectiveness, budgetary impact, and alignment with the local needs of the healthcare system. For a company aspiring to an effective national presence in Spain, it is highly recommended to familiarize itself with these agencies and their economic evaluation methodologies and to consider requesting specific evaluations targeted at particular regional health services where the product may have a greater impact or where there are more streamlined channels for technology adoption.

18.4.2. BRAZIL AND MEXICO: THE GIANTS OF THE LATIN AMERICAN MARKET

Turning our gaze across the Atlantic, we find a Latin American landscape of extraordinary regulatory diversity where each country demands its own approval pathway without mutual recognition mechanisms equivalent to the European single market. Brazil, owing to its immense population of 215 million, the size of its economy, and the relative sophistication of its healthcare infrastructure, is unquestionably the most attractive and complex market in Latin America. Its regulatory agency, the Agência Nacional de VigilânciaSanitária (ANVISA), is also among the most demanding, technically wellequipped, and has the strongest tradition of professional independence in the entire region. ANVISA maintains a medical device registration process that can be complex, documentintensive, and timevarying but has made significant progress in recent years toward harmonization with recognized international standards.(14)

For medical software specifically, ANVISA has published specific resolutions and technical guidelines clarifying its regulatory treatment and offers the possibility of using international reference certifications—particularly FDA approval or the European CE mark—as supporting evidence that can substantially streamline the Brazilian registration process.(14) However, it is important not to fall into the misconception that foreign certification completely replaces local registration: ANVISA continues to require a specific health registration for Brazil, technical documentation translated into Portuguese by certified translators, adaptations to local labeling regulations and instructions for use, and in some cases, additional validation studies demonstrating the product's performance in the Brazilian population or under local conditions of use. The process, although improved, remains slow and costly, often taking twelve to eighteen months from initial submission to obtaining final registration.

Mexico is the second-largest Latin American market, with a population of 130 million and a complex healthcare system that combines public, private, and social security segments. The Federal Commission for Protection against Health Risks (COFEPRIS) is the competent regulatory authority, and its medical device registration process has traditionally been criticized for its complexity, its highly variable timelines—which ranged from six months to three years with no predictability—and its documentation requirements, which often changed without prior notice.(15) However, in recent years, COFEPRIS has implemented significant reforms aimed at streamlining the process, making it more predictable and better align it with international standards. The agency has established priority review pathways for innovative products and has improved its electronic submission platforms.

COFEPRIS, like ANVISA, accepts international reference certifications as proof of compliance, although with specific requirements that must be verified on a case-by-case basis. A particularly important regulatory aspect in Mexico is the requirement to have a local legal representative established in the Mexican territory, who acts as the holder of the health registration with COFEPRIS and assumes legal responsibility before the authority for the marketed product. For foreign companies without a Mexican subsidiary, this necessarily involves finding a local distributor, a strategic partner, or a reliable legal representative to fulfill this role, which adds a dimension of

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

business relationship management to the purely regulatory complexity. The choice of this local partner is critical, as their actions or interactions can affect the validity of the registration and the product's reputation in the market.

18.4.3. EMERGING MARKETS: ARGENTINA, CHILE, COLOMBIA, AND THE REST OF THE REGION

Argentina, through its National Administration of Medicines, Food, and Medical Technology (ANMAT), has one of the regulatory agencies with the longest history and strongest institutional tradition in all of Latin America, building on decades of pharmaceutical regulatory development. ANMAT has shown a growing and explicit interest in digital health, publishing specific guidelines for mobile health applications, telemedicine services, and, more recently, guidance on artificial intelligence in medical devices. ANMAT's registration process is wellestablished, with documented procedures and relatively predictable timelines, although Argentina's macroeconomic situation—with its recurring currency crises, high inflation, and trade restrictions—can severely impact the actual commercial viability of approved products. For foreign developers, it is imperative to assess not only regulatory feasibility but also the actual market: the payment capacity of healthcare systems, reimbursement mechanisms, and the stability of technology imports.(16)

Chile, through its Institute of Public Health (ISP), often offers a relatively streamlined entry route for companies wishing to test the South American region before committing to larger and more complex markets. The ISP has made consistent progress in harmonizing with international standards, actively participates in the Pan American Network for the Harmonization of Pharmaceutical Regulations, and accepts reference certifications with simplified procedures. In addition, Chile maintains trade agreements with a broad network of countries, including the United States, the European Union, and numerous Asian partners, which facilitate the importation of technological products and reduce tariff barriers. The Chilean market, although smaller than the Brazilian or Mexican markets, offers a predictable regulatory environment, a healthcare system capable of adopting technology, and a strategic geographic position as a platform for expansion into other Andean countries.(17)

Colombia, through the National Institute for Food and Drug Surveillance (INVIMA), maintains a registration process that can be slow but offers a degree of predictability once the applicant understands its specific requirements. The country has made a firm commitment to telemedicine and digital health as a health policy strategy, accelerated particularly by experiences during the COVID-19 pandemic, which has created specific opportunities for medical software focused on remote consultations, home monitoring, and chronic disease management. INVIMA accepts international reference certifications, although with additional local documentation requirements that must be carefully prepared. The Colombian market, with its mixed healthcare system and population of fifty million, represents an opportunity that falls between regional giants and smaller markets.(18)

Peru, Uruguay, Paraguay, Ecuador, and other countries with smaller populations have smaller regulatory agencies, fewer specialized technical resources, and less developed and often more informal registration processes. These markets, although individually limited in size, can offer entry opportunities with lower initial regulatory investment and can serve as learning platforms for the region. However, the decision to enter these markets must be based on a rigorous cost–benefit analysis that considers not only formal regulatory requirements but also the actual size of the addressable market, the intensity of local and international competition, nonregulatorycommercial barriers to entry, and the ability to generate returns that justify investment in local adaptations, translations, and registration management.

18.4.4. NAVIGATION STRATEGIES AND THE VALUE OF DIVERSITY

A cross-cutting challenge throughout Latin America, regardless of the specific country, is the extreme variability in actual regulatory approval times. While in some countries with optimized processes, a marketing authorization can be obtained in three to six months by submitting complete documentation and reference certifications, in other less efficient or more congested systems, the process can drag on for two, three, or even five years with no predictability. This time-related uncertainty severely hinders business planning, the management of development resources, negotiations with investors, and the coordination of multinational commercial

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

launches. For this reason, it is highly recommended that companies without prior experience in the region work with local regulatory consultants with a proven track record, who are familiar not only with formal processes but also with actual timelines, the unwritten nuances of each agency, and the professional contacts that can facilitate the resolution of administrative hurdles.

Another common challenge is the management of technical and legal documentation. Although many Latin American countries accept reference certifications as a starting point, almost all req require specific local documentation, translations certified by official translators recognized in each country, and in some cases, additional clinical validation studies demonstrating safety and efficacy in the local population or under country-specific conditions of use. This can entail significant costs, especially when translation into multiple languages is needed: Portuguese for Brazil, Spanish with local variants for the other countries, and occasionally specific document format requirements. Planning for these regulatory localization costs from the product design phase and including them in international marketing budgets is essential to avoid surprises that could compromise the economic viability of regional expansion.

Regulatory harmonization in Latin America remains, however, a distant goal rather than an imminent reality. Mercosur has established some mechanisms for technical coordination among its full members—Argentina, Brazil, Paraguay, and Uruguay—and associates, but these agreements remain limited in scope and do not result in automatic mutual recognition of health registrations. The Pan American Network for the Harmonization of Pharmaceutical Regulations (RED PARF), under the umbrella of the Pan American Health Organization, actively promotes regulatory convergence and has made progress in specific areas such as good manufacturing practice inspections, but its progress toward the harmonization of software medical device registrations is slow and piecemeal. As these regional mechanisms mature, medical software manufacturers must accept diversity as the operating norm and develop internal capabilities for multinational regulatory management.

For developers facing this patchwork landscape, the recommended strategy must be based on rigorous and realistic prioritization. It is neither operationally feasible nor strategically necessary to attempt simultaneous registration in all Latin American countries. The sensible approach involves identifying the two or three most attractive markets on the basis of criteria such as size, purchasing power, alignment with the product profile, and manageable regulatory complexity; studying them thoroughly through market research and local consulting; and establishing a realistic sequential expansion roadmap. It often makes strategic sense to start with countries that have more streamlined and predictable processes—Chile or Uruguay, for example—to gain operational experience, accumulate local clinical references, and validate product adaptations before addressing more complex and competitive markets such as Brazil or Mexico. It may also be valuable to explore strategic alliances with established local partners who understand the regulatory landscape, have health registration infrastructure in place, and can facilitate access to distribution channels and relationships with payers.

For healthcare professionals in Latin America and Spain, understanding this fragmented regulatory landscape provides valuable tools for assessing the quality and legitimacy of the products they encounter. Medical software that holds a valid ANVISA health registration in Brazil has undergone a rigorous technical evaluation process and can be considered of proven quality; conversely, a product that only bears the European CE mark but lacks specific registration in the Latin American country where it is marketed may not be legally authorized and warrants further scrutiny. This regulatory knowledge also empowers clinicians to actively participate in national health policy debates, suggesting that regulatory processes that effectively protect patients from unsafe or ineffective products without creating unnecessary bureaucratic barriers that stifle legitimate and beneficial innovation.

The future of medical software regulation in Latin America will inevitably move toward greater regional harmonization and the institutional strengthening of national agencies. The experience during the COVID-19 pandemic has forcefully demonstrated the importance of having agile yet robust regulatory systems capable of rapidly evaluating technologies critical to public health without sacrificing safety standards. It has also highlighted the potential of regional cooperation to accelerate equitable access to essential health technologies. In the coming years, we are likely to see significant progress in these areas, although the pace will necessarily be gradual given the differences in institutional development across countries. In the meantime, regulat y diversity will remain the operational reality, and those who develop the ability to navigate it with intelligence, patience, and respect for local particularities will have a substantial competitive advantage in one of the world's digital health markets with the greatest growth potential.

In closing, it is worth remembering that behind every documentation requirement, every

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

registration procedure, and every evaluation process, there are real patients who trust—often without knowing it—that the technological products reaching the market where they live are safe and effective and have been evaluated by competent authorities. The regulatory diversity that characterizes Spanish-speaking markets is not a bureaucratic whim but a reflection of the diversity of healthcare systems, public policy priorities, cultures of the relationship between the state and citizens, and levels of economic development. Respecting this diversity, striving to genuinely understand it and working constructively with it rather than against it is the only way to build sustainable long-term commercial relationships and, more importantly, to contribute ethically and effectively to the health and well-being of people in these countries.

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

REFERENCES

CDRH International Affairs | FDA [Internet]. [Cited May 18, 2026]. Available from: https://www.fda.gov/medical-devices/cdrh-international-affairs?trk=article-ssr-frontend-pulse_little-text-block
Brannigan V. The regulation of medical computer software as a “device” under the Food, Drug, and Cosmetic Act. Comput Methods Programs Biomed [Internet]. 1987 [cited 2026 May 18];25(2):219-28. Available at: https://pubmed.ncbi.nlm.nih.gov/3677634/
Global Approach to Software as a Medical Device | FDA [Internet]. [cited May 18, 2026]. Available at: https://www.fda.gov/medical-devices/software-medical-device-samd/global-approach-software-medical-device?source=post_page-----58d9ee0a1ae2----------
Aboy M, Crespo C, Stern A. Beyond the 510(k): The regulation of novel moderate-risk medical devices, intellectual property considerations, and innovation incentives in the FDA’s De Novo pathway. NPJ Digit Med [Internet]. December 1, 2024 [cited May 18, 2026];7(1):29. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC10853500/
Breakthrough Devices Program | FDA [Internet]. [cited 2026-05-18]. Available from: https://www.fda.gov/medical-devices/how-study-and-market-your-device/breakthrough-devices-program?app_v2=true//statistiche
Informing the FDA’s digital health precert program | Presidential Innovation Fellows [Internet]. [cited May 18, 2026]. Avai
A Closer Look at FDA’s Newly Released AI/ML Action Plan - Petrie-Flom Center [Internet]. [cited May 18, 2026]. Available at: https://petrieflom.law.harvard.edu/2021/02/18/a-closer-look-at-fdas-newly released-ai-ml-action-plan/
Lee YK, Lih E, Kim MY, Kim SY, Nam KC, Choi J, et al. Healthcare Experts’ Advisory Unit and Support (HAUS) Program for Medical Device Development in Korea: Introduction of Clinical Unmet Needs-Based Intended Use Establishment (CLUE) Templates. J Korean Med Sci [Internet]. 2024 [cited 2026 May 18];39(40):e311. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC11496563/
Regulation - 2017/745 - EN - Medical Device Regulation - EUR-Lex [Internet]. [cited 2026-05-18]. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2017.117.01.0001.01.ENG
Overview - Public Health - European Commission [Internet]. [cited May 18, 2026]. Available at: https://health.ec.europa.eu/medical-devices-eudamed/overview_en
Regulation - EU - 2024/1689 - EN - EUR-Lex [Internet]. [cited 2026-05-18]. Available from: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689
Novelli C, Hacker P, McDougall S, Morley J, Rotolo A, Floridi L. Getting Regulatory Sandboxes Right: Design and Governance Under the AI Act. Eur J Risk Regul [Internet]. April 1, 2026 [cited May 18, 2026];1-27. Available at: https://www.cambridge.org/core/journals/european-journal-of-risk-regulation/article/getting-regulatory-sandboxes-right-design-and-governance-underthe-ai-act/E85D318E227C721BF4823F7930DDA937
OJEU-L-2017-80916 Regulation (EU) 2017/745 of the European Parliament and of the Council of April 5, 2017, on medical devices, amending Directive 2001/83/EC, Regulation (EC) No.º 178/2002, and Regulation (EC) No. 1223/... [Internet]. [cited May 18, 2026]. Available from: https://www.boe.es/buscar/doc.php?id=DOUE-L-2017-80916
Insights into regional medical devices regulatory affairs to reach LATAM [Internet]. [Cited May 18, 2026]. Available at: https://www.worldhealthexpo.com/insights/iomt/insights-into-regional-medical-devices-regulatory-affairs-to-reach-latam
Mexico: A Strategic Global Hub for Medical Device Manufacturing [Internet]. [cited May 18, 2026]. Available at: https://mexicobusiness.news/health/news/mexico-strategic-global-hub-medical-device-manufacturing?tag=health?tag=emotional-regulation
Submission of Affidavit for Imported Medical Products - Provision No. 236/2026 | Argentina.gob.ar [Internet]. [cited May 18, 2026]. Available at: https://www.argentina.gob.ar/servicio/presentacion-de-declaracion-jurada-de-productos-medicos-importados-disposicion-ndeg-2362026
León CV. Regulatory Update-ISP-Chile.
Levitt G. Pharma & Medical Device Regulation Consulting editors Lexology GTDT-Pharma & Medical Device Regulation. 2006 [cited May 18, 2026]; Available at: www.lexology.com/gtdt

https://doi.org/10.62486/978-9915-9876-8-2.Ch18

Epilogue, appendices and bibliography

EPILOGUE

As we close this book, we have traversed a vast landscape, from the fundamentals of medical software to complex business models and algorithmic ethics. However, at the end of this journey, the most important question is not technical but human: Who is the professional capable of leading this future?

The answer is not a doctor who programs or an engineer who reads medical literature. This field urgently needs translators between worlds. People can sit in a hospital consultation room to interpret a patient's pain and then translate it onto an architectural whiteboard in the form of technical requirements.

Becoming that translator does not require a superpower but rather cultivating a set of deeply human competencies:

- Intellectual curiosity to take nothing for granted.
- Communication to find the metaphor that illuminates both the engineer and the clinician.
- Systems thinking to see how a line of code impacts patient safety.
- Humility to acknowledge what you do not know and have the confidence to ask.
- Resilience to persevere when a regulator requests change or when a study does not turn out as expected.

This book ends here, but your role as a translator is just beginning. There is no single career path for this; your expertise is built through practice by combining a solid disciplinary foundation with real-world experience on the front lines. The future of digital health depends on enough people to prevent investments from going to waste and, above all, to ensure that patients receive the benefits that technology promises them.

We are writing the future of medicine with every clinical decision and every line of code. Thank you for accepting the challenge of building a digital health system that is brighter, safer, and, above all, more humane.

APPENDIX A. COMPLETE GLOSSARY OF TERMS

AEMPS: Spanish Agency for Medicines and Health Products. It is the competent authority in Spain for the regulation of medical devices, including medical software. It oversees compliance with European and national regulations and participates in postmarket evaluation and surveillance processes.

Agile: A software development methodology based on iteration, collaboration, and responsiveness to change rather than adherence to rigid plans. In regulated environments, adaptations are needed to maintain the traceability and documentation required by standards such as IEC 62304.

AI Act: European Regulation on Artificial Intelligence, adopted in 2024, which establishes a regulatory framework for AI systems on the basis of their risk level. Medical software, which is classified as high risk in most cases, imposes requirements for transparency, data governance, human oversight, and continuous monitoring.

Algorithm: A sequence of instructions or rules designed to perform a specific task or solve a problem. In the context of digital health, algorithms can range from simple calculation formulas to complex artificial intelligence models that learn from data.

Hallucination: A phenomenon whereby a language model generates false but seemingly plausible information, inventing data, quotes, or statements with complete conviction. It is among the main risks of generative AI in medicine, where it can have serious consequences.

FMEA: Failure Mode and Effects Analysis. A systematic technique for identifying potential failures in a product or process, analyzing their causes and consequences, and prioritizing preventive actions. It is a fundamental tool in risk management according to ISO 14971.

Survival analysis: A set of statistical techniques for analyzing data where the variable of interest is the time until an event occurs (death, relapse, hospital discharge). It includes methods such as Kaplan–Meier curves and the Cox proportional hazards model.

Anonymization: The process of removing or modifying information that could identify a person so that it is no longer possible to link the data to a specific individual. It is a key measure for protecting privacy in the secondary use of clinical data.

ANMAT: National Administration of Medicines, Food, and Medical Technology. It is Argentina's regulatory agency and is responsible for the authorization and control of health products, including medical software.

ANVISA: National Health Surveillance Agency. It is Brazil's regulatory agency, one of the most stringent in Latin America. To market medical software in Brazil, it is necessary to obtain ANVISA registration, which may be based on international certifications but requires local documentation in Portuguese.

API: application programming interface. A set of rules and protocols that allow different software applications to communicate with one another. In digital health, APIs are essential for interoperability, enabling electronic health record systems, laboratories, pharmacies, and patient applications to exchange information in a standardized manner.

Software architecture: The fundamental structure of a software system, comprising its components, the relationships between them, and the principles guiding its design and evolution. In the healthcare environment, architectural decisions have direct implications for security, scalability, and interoperability.

ATC: Anatomical therapeutic chemical classification system. A drug coding system maintained by the World Health Organization that classifies drugs according to the organ or system upon which they act and their therapeutic properties. It is particularly useful for epidemiological studies and analyses of prescribing patterns.

AUC: Area under the curve (ROC). A metric that summarizes the overall performance of a binary classifier for all possible decision thresholds. An AUC of 1 indicates a perfect classifier; an AUC of 0.5 indicates a random classifier. It is among the most widely used metrics for evaluating diagnostic and predictive algorithms.

Audit: A systematic process of reviewing processes, documentation, and practices to verify compliance with standards and requirements. In regulated environments, audits can be internal (conducted by the organization itself) or external (conducted by notified bodies or health authorities).

Authentication: The process of verifying the identity of a user or system attempting to access a resource. In healthcare settings, authentication must be particularly robust, often using multiple factors (something the user knows, something they have, something they are) to protect sensitive data.

Authorization: The process that determines what actions an authenticated user can perform on system resources. In digital health, authorization must be granular, role-based, and context-sensitive (for example, a doctor can view their own patients' records but not those of others).

QALY: Quality-Adjusted Life Year. A measure that combines the quantity and quality of life and is used in economic evaluation to compare the benefits of different healthcare interventions. One QALY is equivalent to one year of life in perfect health. It is the most widely used metric in cost-effectiveness analyses.

Backend: The part of an application that runs on the server and is responsible for business logic, data processing, and storage. It is not visible to the user but is essential for the application's operation.

Big Data: A dataset whose volume, velocity, or variety exceeds the processing capacity of traditional tools. In healthcare, big data includes data from electronic health records, wearable devices, genomics, and other sources and promises more predictive and personalized medicine.

Calibration: In predictive models, calibration measures whether predicted probabilities correspond to observed frequencies. A well-calibrated model predicts, for example, that 70% of patients with a 70% probability will actually experience the event. Calibration is just as important as discrimination (the ability to distinguish between cases and controls).

Cognitive load: The amount of mental effort required to perform a task. In the design of clinical interfaces, minimizing the cognitive load is essential for reducing errors and improving usability, especially in high-pressure environments such as emergency rooms or intensive care units.

Cybersecurity: A set of technical, organizational, and human measures to protect information systems against unauthorized access, attacks, data theft, or service disruptions. In the healthcare setting, cybersecurity is an extension of clinical safety.

Safety class (IEC 62304): Classification of medical software based on the potential risk of failure: Class A (no possibility of injury), Class B (possibility of nonserious injury), and Class C (possibility of death or serious injury). It determines the required development, documentation, and verification activities.

Cloud computing: A computing model that provides on-demand access to configurable shared resources (servers, storage, applications) via the internet. In healthcare, it offers scalability and flexibility but poses challenges regarding privacy and reliance on connectivity.

COFEPRIS: Federal Commission for Protection against Health Risks. It is Mexico's regulatory

agency and is responsible for the authorization and control of health products, including medical software.

Ethics Committee (CEIm): An independent, multidisciplinary committee responsible for evaluating the ethical aspects of research projects involving human subjects. Its approval is required to conduct clinical studies using medical software.

Informed consent: The process by which a patient or research participant voluntarily authorizes a procedure or the use of their data after receiving understandable information about its risks, benefits, and alternatives. It is a cornerstone of ethics in medicine and research.

Cost-effectiveness: The relationship between the costs of an intervention and its health benefits, generally expressed as cost per QALY gained. It is the fundamental criterion in the economic evaluation of health technologies.

ROC curve: A graphical representation of the performance of a binary classifier that shows the true positive rate (sensitivity) versus the false positive rate (1 minus specificity) for all possible decision thresholds. It allows one to visualize the trade-off between sensitivity and specificity and choose the most appropriate threshold for each clinical context.

Dataset shift: A change in the distribution of data between a model's training phase and its implementation in real-world practice. It is among the main causes of performance degradation in algorithms during production. This may be due to changes in the population, clinical practice, or measurement conditions.

Defense in depth: A security principle that involves establishing multiple layers of protection so that if one layer fails, the subsequent layers continue to provide protection. In healthcare cybersecurity, this involves combining technical, organizational, and human measures.

Design history file (DHF): A set of records documenting the design history of a medical device. It includes requirements, drawings, risk analyses, test results, and any other relevant information to demonstrate that the product has been developed following the appropriate processes.

DICOM: Digital Imaging and Communications in Medicine. An international standard for the management, storage, printing, and transmission of medical images. It is not just a file format but a complete protocol that defines how imaging devices should communicate and how the associated metadata should be structured.

DTx (digital therapy): A software-based therapeutic intervention, regulated as a medical device, that prevents, manages, or treats diseases on the basis of proven clinical evidence. Unlike general wellness apps, DTx requires clinical trials and regulatory approval.

Edge computing: A computing model that processes data close to where it is generated rather than sending it to the cloud. It is essential for applications requiring low latency, such as real-time monitoring of critically ill patients, or for environments with limited connectivity.

Clinical trial: This prospective study evaluates the effects of an intervention (drug, device, or procedure) in humans, following a rigorous protocol and with random allocation of participants to intervention and control groups. It is the gold standard for demonstrating the efficacy of healthcare interventions.

Specificity: The proportion of negative cases that a test or algorithm correctly identifies. High specificity means that the algorithm is good at avoiding false alarms. It is especially important when the cost of a false positive is high.

EUDAMED: European database of medical devices created by the MDR and the IVDR, which contains information on all products marketed in Europe, the economic operators involved, the certificates issued, and the incidents reported. It will be accessible to the public, increasing transparency in the sector.

Clinical evaluation: A systematic process of analyzing and evaluating clinical data related to a medical device to verify its safety and effectiveness under the intended conditions of use. It is a fundamental requirement of the MDR and the IVDR.

Explainability (XAI): The ability of an artificial intelligence system to provide understandable explanations for its decisions or predictions. It is essential for building trust, detecting biases, and complying with regulatory requirements such as those of the AI Act.

FDA: U.S. Food and Drug Administration. It is the world's most influential regulatory agency for medical devices. It classifies devices into Classes I, II, and III on the basis of their risk and offers different approval pathways (510(k), De Novo, PMA).

FHIR: Fast Healthcare Interoperability Resources. A modern standard for the exchange of health information is based on REST APIs and formats such as JSON. It is simpler and more flexible than previous standards and is being adopted by leading electronic health record providers and government agencies worldwide.

Frontend: The part of an application with the user that directly interacts with the user: interface, screens, buttons, menus. A good frontend design is essential for usability and to minimize user errors.

GDPR: The European Union's General Data Protection Regulation. It establishes a rigorous framework for the protection of personal data, with special protections for health data. It includes rights such as explicit consent, the right to be forgotten, and the requirement to report data breaches within 72 hours.

Digital twin: A virtual representation of a physical system that is updated with real-time data and can be used for simulation, prediction, and optimization. In medicine, partial digital twins of organs such as the heart are used to plan procedures, and systems such as glucose metabolism are used in artificial pancreases.

Generalization: A model can maintain its performance when faced with new data from different populations, distinct contexts, or time periods after its development. It is the difference between a model that memorizes and one that truly learns.

Hazard ratio: A measure used in survival analysis that indicates how many times greater (or lower) the risk of experiencing an event is in one group than in another at any point during follow-up. A hazard ratio of 2 means that the risk doubles; a hazard ratio of 0.5 means that it is halved.

HIPAA: The Health Insurance Portability and Accountability Act of the United States. It establishes requirements to protect identifiable medical information, including rules on privacy, security, and breach notification. For decades, it has been the global benchmark for health data protection.

HL7 v2: A standard for the electronic exchange of health information developed in the 1980s and 1990s. Despite its age and its cryptic, delimiter-based syntax, it remains the most widely used standard for communication between hospital systems. It uses plain text messages with segments such as MSH, PID, PV1, OBR, and OBX.

HTA: Health Technology Assessment. A multidisciplinary process that analyzes the medical, social, ethical, and economic implications of adopting a health technology. It determines whether a product should be funded by public health systems.

IEC 62304: International standard establishing requirements for the software lifecycle of medical devices. It defines development, verification, validation, and maintenance activities on the basis of the software's safety class (A, B, C).

IEC 62366: International standard that establishes requirements for the application of usability engineering to medical devices. Its objective is to ensure that products are safe and effective when used by their intended users in the environments where they are actually used.

Generative Artificial Intelligence: A branch of artificial intelligence focused on creating new content (text, images, and audio) on the basis of patterns learned from existing data. Large language models such as the GPT are its best-known examples. In medicine, it has promising applications but also significant risks.

Interoperability: The ability of different information systems to communicate, exchange data, and utilize the exchanged information. In the healthcare sector, ensuring continuity of care and patient safety is a clinical necessity.

INVIMA: National Institute for the Surveillance of Medicines and Foods. It is Colombia's regulatory agency and is responsible for the authorization and control of medical devices, including medical software.

IoMT: internet of Medical Things. A network of medical devices connected to the internet, such as glucose monitors, blood pressure monitors, pacemakers, or wearables, that generate continuous data on patients' health.

ISO 13485: An international standard specifying the requirements for a quality management system applicable to organizations that design, manufacture, install, and service medical devices. It is not a product standard but rather a standard for organizational processes.

ISO 14971: International standard that establishes the principles and processes for risk management in medical devices. It requires identifying hazards, estimating and evaluating risks, implementing control measures, and monitoring the effectiveness of these measures throughout the entire product lifecycle.

Kappa (coefficient): A statistical measure that quantifies the degree of agreement between two or more evaluators, taking into account the agreement that would be expected by chance. It is essential for assessing the reliability of annotations in clinical datasets.

Latency: The time that elapses between an action being performed and the system responding. In real-time clinical applications, such as monitoring critically ill patients, low latency is essential for safety.

LIME: Local Interpretable Model-agnostic Explanations. An explainability technique that generates local explanations for individual predictions by constructing a simple model that approximates the behavior of the complex model in the vicinity of the analyzed case.

LOINC: Logical observation identifier names and codes. Reference terminology for identifying laboratory tests and other clinical observations. It provides a unique code for each type of test, with sufficient granularity to distinguish clinically relevant variants.

CE Marking: A marking indicating that a product complies with the essential requirements of applicable European directives or regulations. For medical devices, CE marking under the MDR or IVDR is mandatory for marketing within the European Union.

MDR: Regulation (EU) 2017/745 on medical devices. It establishes the requirements for the marketing of medical devices in the European Union, including medical software. It is more stringent than previous directives, with greater oversight of notified bodies, rigorous clinical evaluation, and transparency through EUDAMED.

Microservices: An architectural style that structures an application as a set of small, independent services, each with its own database and lifecycle, which communicate via APIs. It offers flexibility and scalability but increases complexity.

Least privilege: A security principle according to which each user, process, or component should have only the permissions strictly necessary to perform its function. Applying it reduces the attack surface and limits the damage that a compromised component can cause.

MLOps: A set of practices aimed at professionalizing the lifecycle of artificial intelligence models, from initial training through to their maintenance in production. It includes the continuous integration, monitoring, updating, and governance of the models.

Cox Model: A proportional hazards model was used in survival analysis to estimate how multiple variables affect the risk of experiencing an event. It is among the most widely used tools in clinical research.

Monolith: A software application in which all functionalities are integrated into a single block, with a central database and a unified development process. It offers operational simplicity and consistency but can be rigid and difficult to scale.

Notified Body: An independent entity designated by the national authorities of an EU member state to assess the conformity of medical devices with the requirements of the MDR or the IVDR. Its involvement is required for medium- and high-risk devices.

PACS: Picture Archiving and Communication System. A system that receives, stores, distributes, and displays medical images in DICOM format. It is the heart of the medical imaging infrastructure in any modern hospital.

Profile (FHIR): A set of constraints on FHIR resources to adapt them to a specific use, defining which elements are mandatory, which vocabularies must be used, and which extensions are needed. Profiles allow the FHIR to be both a global standard and a tool tailored to local needs.

PMA: Pre-Market Approval. This is the FDA's most stringent approval pathway, reserved for Class III (high-risk) devices. Robust clinical evidence of safety and efficacy is needed, typically through controlled trials.

Statistical power: The probability of detecting an effect if it actually exists. In the development of predictive models, a study with low power has a high probability of producing misleading results, including both false positives and false negatives.

Prevalence: The proportion of individuals in a population who have a specific disease or condition at a given time. It is a key factor in interpreting the predictive value of diagnostic tests.

Logistic regression: A statistical technique for modeling the probability of a binary event occurring (presence/absence of disease, success/failure of a treatment) on the basis of one or more predictor variables. It is widely used in the construction of clinical predictive models because of its simplicity and interpretability.

Residual risk: The risk that remains after all control measures have been implemented. It must be assessed to determine whether it is acceptable by comparing it to previously defined acceptability criteria.

RxNorm: A terminology system that standardizes drug names, providing unique identifiers for different levels of abstraction: active ingredients, combinations, and commercial formulations. It maintains relationships between these levels, allowing navigation from the generic to the specific.

SaMD: Software as a medical device. Software intended for medical purposes that is not part of a physical device. It includes applications that interpret heart rates, algorithms that analyze X-rays, or systems that calculate medication doses.

Security by Design: A development approach where security is considered from the outset

as a core requirement, not as an afterthought. It involves early threat identification, defense in depth, least privilege, and other practices that integrate security into the system architecture.

Sensitivity: The proportion of positive cases that a test or algorithm correctly identifies. High sensitivity means that the algorithm is effective when actual cases are not missing. This is especially important when the cost of a false negative is high.

Bias: A systematic distortion in the data or algorithms that causes them to learn patterns that do not reflect reality or that reflect only part of it. This may be due to the training sample, measurements, labeling, or socioeconomic factors, among others.

SHAP: SHapley Additive exPlanations. An explainability technique based on game theory that calculates the contribution of each feature to the difference between the prediction for a specific case and the average prediction. It provides consistent, theoretically grounded explanations.

SMART on FHIR: An open standard that allows third-party applications to integrate securely and seamlessly into electronic health record systems. It combines the FHIR for data access with OAuth2 for authorization.

SNOMED-CT: Systematized Nomenclature of Medicine Clinical Terms. A comprehensive, multilingual clinical terminology covering most of the knowledge required for healthcare practice. It contains more than 350,000 concepts organized into hierarchies and linked through multiple types of relationships.

Sample size: The number of cases required for a study to have sufficient statistical power to detect effects of interest. This depends on the complexity of the problem, the prevalence of the event, the heterogeneity of the population, and the complexity of the model.

Traceability: The ability to reconstruct the path from a clinical need to the code that implements it and from a code snippet to the need that justifies it. It is essential for security, change management, and the demonstration of regulatory compliance.

External validation: Evaluation of a model's performance on data different from that used for training, ideally from different centers, distinct populations, or later time periods. It is the only way to demonstrate that a model generalizes to new contexts.

Internal validation: Evaluation of a model's performance using the same data available for training through techniques such as training/test split or cross-validation. It is necessary for development but insufficient to demonstrate generalizability.

Positive predictive value: The probability that a patient with a positive test result actually has the disease. It depends on sensitivity, specificity, and, crucially, the prevalence of the disease in the population.

Negative predictive value: The probability that a patient with a negative test result is actually healthy. It also depends on sensitivity, specificity, and prevalence.

Wearable: Wearable electronic devices, such as smartwatches, fitness trackers, or skin patches, that continuously monitor physiological parameters such as heart rate, activity, sleep, or glucose.

XAI: eXplainable artificial intelligence. A field that seeks to equip AI models with the ability to explain their reasoning in a way that is understandable to humans.

510(k): FDA approval pathway for Class II devices. It requires demonstrating that the new product is substantially equivalent to a product already on the market (predicate device) in terms of safety and effectiveness. It does not always require clinical trials.

APPENDIX B. CHECKLIST: IS MY SOFTWARE READY FOR PRODUCTION?

This checklist is designed to help development teams, quality managers, and clinicians assess whether medical software is ready for deployment in real-world clinical settings. It is not an exhaustive list of regulatory requirements but a practical tool for identifying potential gaps before moving to production. It is organized into sections covering the fundamental aspects discussed throughout the book: from problem definition to postlaunch monitoring.

1. DEFINITION AND SCOPE

- **Clearly defined clinical problem**: Have we identified and documented the specific clinical problem that the software solves? Is there evidence that it is a relevant problem for patients or healthcare professionals?
- **Identified users**: Have we defined who the system's users will be (doctors, nurses, patients, administrative staff)? Have we considered their characteristics, limitations, and context of use?
- e **usage environment**: Have we analyzed the actual environment where the software will be used (hospital, clinic, home)? Have we considered factors such as noise, lighting, interruptions, and connectivity?
- **Indications and contraindications**: Have we clearly defined for which patients and in which situations the use of the software is indicated? In which situations should it not be used?
- **Software purpose**: Have we clearly distinguished whether the software is a medical device (diagnosis, treatment) or a support tool (management, information)? This distinction affects all regulatory aspects.

2. REGULATION AND CLASSIFICATION

- **Device classification**: Have we determined the risk class of the software according to applicable regulations (FDA, MDR)? Have we documented the justification for this classification?
- **Applicable regulations**: Have we identified all regulations applicable to our product (MDR, IVDR, FDA, AI Act, HIPAA, GDPR) on the basis of the target markets?
- **Notified Body**: If applicable, have we selected and contacted a notified body? Have we agreed on the scope of the assessment?
- **Regulatory pathway**: Have we defined the specific approval pathway (510(k), De Novo, PMA, CE marking, DiGA) and planned the necessary resources and timelines?
- **Compliance plan**: Have we developed a plan detailing all necessary regulatory activities, including responsible parties and deadlines?

3. QUALITY SYSTEM AND PROCESSES

- **Quality management system**: Have we implemented a quality management system in accordance with ISO 13485 or another applicable standard? Is it documented and operational?
- **Defined processes**: Have we defined and documented the key processes: development, change control, risk management, verification, validation, and maintenance?
- **Roles and responsibilities**: Have we clearly assigned responsibilities for each process? Is there a person responsible for regulatory compliance?
- **Staff training**: Have all involved staff received training on quality processes and the specific requirements for medical software?
- **Audits**: Have we conducted internal audits to verify that processes are being followed correctly? Are the results and corrective actions documented?

4. RISK MANAGEMENT (ISO 14971)

- **Risk management plan**: Have we developed a plan that defines risk management activities, acceptability criteria, and responsibilities?
- **Risk Analysis**: Have we systematically identified all potential hazards associated with the use

of the software? Have we considered scenarios involving normal use, misuse, and failures?
- **Estimation and evaluation**: Have we estimated the probability and severity of each risk and evaluated its acceptability according to the defined criteria?
- **Risk control**: Have we implemented control measures for unacceptable risks, following the hierarchy (inherent safety, protection, information)?
- **Verification of controls**: Have we verified that the control measures are effective and do not introduce new risks? Is this verification documented?
- **Residual risk**: Have we assessed the residual risk after implementing control measures and documented its acceptability?
- **Risk management report**: Have we documented the entire process in a report that can be audited?

5. DEVELOPMENT AND LIFECYCLE (IEC 62304)

- **Software classification**: Have we correctly classified the software according to IEC 62304 (Class A, B, or C) on the basis of the risk analysis?
- **Development plan**: Have we developed a software development plan that includes the phases, activities, deliverables, and responsible parties, tailored to the software class?
- **Software requirements**: Have we documented all functional, interface, performance, and safety requirements? Are they traceable to their source?
- **Architecture and design**: Have we documented the software architecture, including components, interfaces, and requirement mapping? For Class C, is the design detailed?
- **Implementation**: Have we followed coding standards appropriate to the software class? For Class C, have documented code reviews been performed?
- **Integration**: Have we planned and executed the integration of components according to a defined strategy? Are the results documented?
- **Verification**: Have we performed the necessary tests according to the software class (unit, integration, system)? Are they traceable to the requirements? Are the results documented?
- **Validation**: Have we demonstrated that the software meets user needs in a real or simulated environment? For Class C, has this been performed in clinical settings?
- **Traceability**: Have we established bidirectional traceability between requirements, design, code, tests, and risks? Is it kept up to date?
- **Configuration management**: Have we identified and controlled all versions of the components? Can we reconstruct any released version?

6. USABILITY AND HUMAN FACTORS (IEC 62366)

- **Usage specification**: Have we documented user characteristics, the tasks they will perform, and the usage environment?
- **Usage risk analysis**: Have we identified potential usage errors and their consequences? Have we integrated them into the overall risk analysis?
- **User-centered design**: Have we involved real users in the iterative design of the interface? Have we tested prototypes with them?
- **Usability evaluation**: Have we conducted usability tests with representative users under conditions that simulate a real-world environment (wearing gloves, dealing with interruptions, and experiencing fatigue)?
- **Cognitive load**: Have we assessed the cognitive load imposed by the software and minimized it in the design?
- **User manual**: Have we developed clear, understandable, and accessible instructions for use? Do they include warnings about residual risks?
- **Training**: Have we developed training materials for users? Is training planned prior to implementation?
- **Usability report**: Have we documented the entire usability engineering process, including the results of the evaluations?

7. DATA AND TERMINOLOGIES

- **Data model**: Have we designed a data model that adequately reflects the complexity of the

clinical domain? Does it use standards (FHIR, SNOMED-CT, LOINC) where possible?
- **Data quality**: Have we assessed the quality of the data the software will use (integrity, accuracy, consistency)? Are there mechanisms in place to detect and correct errors?
- **Clinical terminologies**: Have we used standardized terminologies to code clinical concepts, medications, and laboratory tests? Are there plans to keep them up to date?
- **Terminology mappings**: If mapping between different terminologies is necessary, have we documented and validated these mappings? Are updates managed?
- **Interoperability**: Have we designed the software with interoperability in mind from the start? Does it expose standard APIs (FHIRs) that allow integration with other systems?
- **Master data management**: Are there mechanisms for the unique identification of patients, professionals, and organizations? Are duplicates handled appropriately?

8. CYBERSECURITY

- **Threat Analysis**: Have we identified potential threats to the security of the software and the data it handles? Have we assessed their likelihood and impact?
- **Authentication**: Have we implemented robust authentication mechanisms for all users? For critical access, is multifactor authentication used?
- **Authorization**: Have we implemented role-based access control, following the principle of least privilege? Are permissions reviewed periodically?
- **Encryption**: Is data encrypted in transit and at rest? Are robust algorithms used, and are keys properly managed?
- **Audit logs**: Are all significant accesses to sensitive data logged? Are the logs protected against tampering and analyzable?
- **Protection against common attacks**: Have we implemented measures against SQL injection, XSS, CSRF, and other common web attacks?
- **Supply chain security**: Have we assessed the security of the third-party components (libraries, frameworks) we use? Is there a process in place to keep them up to date?
- **Incident response plan**: Do we have a documented plan for responding to security incidents, including notifications to authorities and affected parties?

9. ARTIFICIAL INTELLIGENCE (IF APPLICABLE)

- **Training data**: Is the data used to train the model representative, diverse, and free of bias to the extent possible? Are its origin and characteristics documented?
- **Data Annotation**: Have the annotations been performed by qualified personnel, following clear protocols and verifying agreement among annotators (Kappa coefficient)?
- **Model validation**: Have we evaluated the model's performance in internal and external validation? Has it been analyzed by subgroups (age, sex, ethnicity) to detect biases?
- **Explainability**: Does the model provide understandable explanations for its predictions? For high-risk models or, are these explanations sufficient for the clinician to exercise their judgment?
- **Robustness**: Have we evaluated the model's robustness against variations in input data, noise, or small perturbations?
- **Monitoring**: Are there mechanisms to monitor the model's performance in production and detect degradation due to dataset shift?
- **Updating**: Is there a defined process for updating the model when necessary, including validation and regulatory recertification if applicable?
- **Hallucination management** (for generative AI): Are there mechanisms to detect and mitigate hallucinations? Are users warned of this risk?

10. CLINICAL VALIDATION AND EVIDENCE

- **Existing evidence**: Have we systematically reviewed the literature to identify prior evidence on similar interventions?
- **Clinical studies**: If necessary, have we designed and conducted clinical studies using appropriate methodology? Have they been approved by an ethics committee?
- **External validation**: Have we validated the software in populations other than the training

set? At multiple centers?
- **Real-world validation**: Have we evaluated the software under real-world conditions of use, with all the noise and imperfections of clinical practice?
- **Clinically relevant metrics**: Have the evaluation metrics (sensitivity, specificity, predictive values) been chosen with clinical relevance in mind, not just statistical significance?
- **Subgroup analysis**: Have we evaluated performance across different population subgroups to detect potential disparities?
- **Reporting of results**: Have we documented all validation results, including limitations and bias analyses?

11. ETHICS AND EQUITY

- **Bias assessment**: Have we explicitly assessed whether the software performs equitably across different groups (gender, age, ethnicity, socioeconomic status)?
- **Bias mitigation**: If biases have been detected, have we implemented measures to mitigate them prior to deployment?
- **Transparency**: Are users (clinicians, patients) aware that they are using an AI system? Are they informed of its capabilities and limitations?
- **Consent**: If the software uses patient data for training or continuous improvement, has appropriate consent been obtained or has an exemption been granted by the ethics committee?
- **Human oversight**: Is the software designed so that a human can monitor, intervene, and override its decisions when necessary?
- **Impact on the clinical relationship**: Have we considered how the software affects the relationship between clinicians and patients? Could it dehumanize care?
- **Equitable access**: Have we considered whether the software would be accessible to all patients who could benefit or whether it could exacerbate existing inequalities?

12. DOCUMENTATION

- **Design History File (DHF)**: Have we created and kept the DHF up to date with all development documentation?
- **Technical documentation**: Does it include a device description, specifications, drawings, risk analysis, and verification and validation reports?
- **Instructions for use**: Are they clear, complete, and accessible to the intended users? Do they include warnings about residual risks?
- **Patient Information**: If applicable, have we developed understandable information for patients regarding the use of the software?
- **Quality documentation**: Are all quality system processes and their compliance documented?
- **Version control**: Are all the documents versioned, and is a change history maintained?

13. BUSINESS MODEL AND SUSTAINABILITY

- **Value proposition**: Have we clearly defined what value the software provides and for whom (patients, clinicians, administrators, payers)?
- **Cost-effectiveness analysis**: Have we conducted an analysis that demonstrated that compared with existing alternatives, the cost of the software is worth its cost?
- **Budgetary impact**: Have we estimated the budgetary impact of adopting the software in the short, medium, and long term?
- **Payment model**: Have we defined who will pay for the software (public systems, insurers, patients) and how will payment be structured (subscription, license, pay-per-use, pay-for-results)?
- **Reimbursement strategy**: For digital therapies, have we identified reimbursement pathways in the target markets (DiGA in Germany, specific codes in other countries)?
- **Sustainability plan**: Is there a plan to maintain the software in the long term, including updates, support, and evolution? Have resources been allocated?
- **Technology transfer**: If the software was developed in an academic setting, is there a plan to transfer it to an entity that can commercialize and maintain it?

14. IMPLEMENTATION AND DEPLOYMENT

- **Implementation Plan**: Have we developed a detailed plan for implementation at target sites, including a timeline, resources, and responsible parties?
- **Integration with existing systems**: Have we tested integration with the centers' information systems (electronic health records, laboratory, and pharmacy)?
- **User training**: Have we prepared training materials and sessions for the professionals who will use the software? Is training planned prior to deployment?
- **Technical support**: Is a support channel available for users? Are response times and escalation procedures defined?
- **Contingency plan**: What happens if the software fails? Are there procedures in place to ensure patient safety while service is restored?
- **Communication**: Have we planned internal and external communication regarding the new software? Will patients be adequately informed?

15. POSTMARKET MONITORING

- **Surveillance plan**: Have we developed a plan to monitor the software's performance after launch, including key performance indicators?
- **Deviation detection**: Are there mechanisms to automatically detect performance degradation or changes in data distribution?
- **Incident management**: Is there a process for investigating and responding to incidents that occur with the software in production?
- **Notification to authorities**: Do we know the deadlines and procedures for reporting serious incidents to regulatory authorities?
- **Updates**: Is there a process for managing software updates, including impact assessment, verification, and recertification, if necessary?
- **Product recall**: Is there a plan to recall the software from the market if necessary, ensuring patient safety?
- **User feedback**: Are there mechanisms in place to systematically collect user feedback and use it to improve the product?

16. LEGAL ASPECTS

- **Intellectual property**: Are the intellectual property rights regarding the software and its components clearly defined and documented?
- **Contracts**: Have we established clear contracts with suppliers, distributors, and customers, including liability and indemnification clauses?
- **Insurance**: Do we have liability insurance appropriate to the product's risks?
- **Terms of Use**: Have we drafted clear terms of use for software users, including limitations of liability?
- **Privacy Policy**: Have we developed a privacy policy that complies with applicable regulations and is understandable to users?

17. PREPARING FOR AUDITS

- **Complete Documentation**: Is all required documentation complete, up-to-date, and organized in an accessible manner?
- **Audit simulations**: Have we conducted audit simulations with internal or external personnel to identify potential gaps?
- **Response plan**: Do we have a plan to respond to auditors' questions, including who will respond in each area?
- **Evidence of compliance**: Can we demonstrate, with documentary evidence, that we follow the stated processes at every stage of development?

Instructions for use:

1. The complete checklist at the start of the project was reviewed to plan the necessary activities.
2. Update the status of each item periodically throughout development.

3. Before launch, a thorough review was conducted with the entire team.

4. Document the decisions made for items that do not apply or that have been addressed specifically.

5. The results are used to prepare documentation for audits and regulatory evaluations.

Interpretation:

- **Green (completed)**: The item is fully addressed and documented.
- **Yellow (in progress)**: Work is underway on the item, but it is not yet finalized.
- **Red (pending)**: The item has not been addressed and requires urgent attention prior to launch.
- **Not applicable**: This item is not relevant to this project (documented justification required).

APPENDIX C. TEMPLATES: RISK MATRIX, VERIFICATION PLAN

This appendix provides practical templates for two essential documents in medical software development: the risk management matrix (based on ISO 14971) and the verification plan (based on IEC 62304). These templates are designed to be adapted to the specific needs of each project and should not be considered substitutes for a rigorous, context-specific analysis. Illustrative examples are included to aid understanding.

TEMPLATE C.1: RISK MANAGEMENT MATRIX

The risk matrix is the central document for risk management according to ISO 14971. It identifies hazards, estimates and evaluates risks, documents control measures, and verifies their effectiveness. This template is organized to facilitate traceability between hazards, control measures, and verification.

DOCUMENT HEADER

Field	Value
Project Title	[Software Name]
Document ID	RG-[Project Acronym]-001
Version	1.0
Date	[DD/MM/YYYY]
Prepared by	[Name and title]
Reviewed by	[Name and title]
Approved by	[Name and title]

1. OBJECTIVE AND SCOPE

The purpose of this document is to identify, analyze, evaluate, and control the risks associated with the [project name] software throughout its entire lifecycle. The analysis covers risks to patients, users, and third parties arising from normal use, reasonably foreseeable misuse, and software failure.

2. DEFINITIONS AND ACCEPTABILITY CRITERIA

Concept	Definition
Hazard	Potential source of harm
Severity (S)	Severity of the consequences of the harm
Probability (P)	Frequency with which the harm may occur
Risk	Combination of the probability of the damage occurring and its severity
Acceptable risk	Risk that is accepted in a given context, based on society's current values

SEVERITY SCALES:

Level	Category	Description
5	Catastrophic	Death or irreversible loss of vital function
4	Severe	Severe reversible injury or loss of nonvital function
3	Moderate	Moderate injury requiring medical intervention
2	Mild	Mild injury with no lasting effects, temporary discomfort
1	Insignificant	No injury, only inconvenience

PROBABILITY SCALES:

Level	Category	Description	Estimated frequency
5	Frequent	May occur repeatedly	> 1 in 100 uses
4	Likely	May occur several times	1 in 1,000 to 1 in 100 uses
3	Occasional	May occur occasionally	1 in 10,000 to 1 in 1,000 uses
2	Rare	Unlikely to occur	1 in 100,000 to 1 in 10,000 uses
1	Unlikely	Very unlikely to occur	< 1 in 100,000 uses

RISK ASSESSMENT MATRIX:

	Severity 1	Severity 2	Severity 3	Severity 4	Severity 5
Probability 5	Acceptable	Acceptable	Unacceptable	Unacceptable	Unacceptable
Probability 4	Acceptable	Acceptable	Acceptable	Unacceptable	Unacceptable
Probability 3	Acceptable	Acceptable	Acceptable	Acceptable	Unacceptable
Probability 2	Acceptable	Acceptable	Acceptable	Acceptable	Acceptable
Probability 1	Acceptable	Acceptable	Acceptable	Acceptable	Acceptable

Note: This matrix is an example; each organization must define its own acceptability criteria on the basis of the clinical and regulatory context.

3. RISK IDENTIFICATION AND ANALYSIS

ID	Hazard	Potential Cause	Usage scenario	Potential Consequence	S	P	Initial risk	Control measures	Residual S	Residual P	Residual risk	Verification

EXAMPLE:

ID	Hazard	Potential cause	Usage situation	Potential consequence	S	P	Initial risk	Control measures	Residual S	Residual P	Residual risk	Verification
R-001	Incorrect calculation of insulin dose	Error in the calculation formula	Patient enters incorrect data	Severe hypoglycemia	5	3	Not acceptable	1. Data entry validation (acceptable ranges). 2. Double confirmation before administration. 3. Alarm if the dose exceeds the safety threshold. 4. Traceability of all recommendations	5	2	Acceptable	Verified through integration tests (report V-001)

4. RECORD OF CONTROL MEASURES

Measure ID	Description	Type of control	Responsible	Implementation Date	Evidence
		(Design/Protection/Information)			

EXAMPLE:

Measure ID	Description	Type of control	Responsible	Implementation Date	Evidence
C-001	Data entry validation: only numeric values within clinically safe ranges are accepted	Design	J. Pérez	03/15/2024	Requirements Specification REQ-023; Unit Tests TU-045

5. CONTROL MEASURE VERIFICATION LOG

Measure ID	Verification Method	Acceptance Criteria	Result	Date	Responsible	Related document
			(OK/Not OK)			

EXAMPLE:

Measure ID	Verification method	Acceptance criteria	Result	Date	Responsible	Related document
C-001	Automated unit tests	100% of out-of-range values are rejected; 100% of in-range values are accepted	OK	03/20/2024	M. García	Test Report V-001

6. OVERALL RESIDUAL RISK ANALYSIS

Once all control measures have been implemented and verified, the overall residual risk of the system has been assessed. The residual risks are considered acceptable because:

- All individual risks have been reduced to acceptable levels according to the defined criteria.
- The expected benefits of the product outweigh the residual risks when the target population and the severity of the condition being treated are considered.
- Measures have been implemented to inform users about residual risks.

7. CONCLUSIONS

The risk management team concluded that the risks associated with the [project name] software have been adequately identified, assessed, and controlled. The residual risks are acceptable in the context of the intended use. It is recommended that postmarket surveillance be maintained to detect potential unidentified risks or changes in risk levels.

8. CHANGE CONTROL

Version	Date	Description of Change	Author	Approved by
1.0	[DD/MM/YYYY]	Initial version	[Name]	[Name]

TEMPLATE C.2: VERIFICATION PLAN

The verification plan is the document that defines how it will be demonstrated that the software meets its specified requirements. On the basis of IEC 62304, it must be adapted to the software safety class (A, B, or C). This template covers the essential aspects of a verification plan for medical software.

DOCUMENT HEADER

Field	Value
Project Title	[Software Name]
Document ID	PV-[Project Acronym]-001
Version	1.0
Date	[DD/MM/YYYY]
Prepared by	[Name and title]
Reviewed by	[Name and title]
Approved by	[Name and title]

1. OBJECTIVE AND SCOPE

This document defines the verification plan for the [project name] software and establishes strategies, methods, resources, and criteria to demonstrate that the software meets its specified requirements. The scope includes the verification of functional, interface, performance, security, and usability requirements.

2. REFERENCE DOCUMENTS

ID	Document	Version	Date
REF-01	Software Requirements Specification	1.0	[DD/MM/YYYY]
REF-02	Software Architecture Document	1.0	[DD/MM/YYYY]
REF-03	Risk Analysis	1.0	[DD/MM/YYYY]
REF-04	Coding Standard	1.0	[DD/MM/YYYY]

3. SOFTWARE CLASSIFICATION

Classification	(A/B/C)
Justification	[Explain why this class was assigned based on the risk analysis]

EXAMPLE:

Classification	C
Justification	The software provides insulin dosing recommendations that, if incorrect, can cause severe hypoglycemia or death. According to MDR Rule 11, it is classified as Class IIb, and according to IEC 62304, as Class C.

4. VERIFICATION STRATEGY

Verification Level	Description	Responsible	Deliverables
Code Reviews	Systematic peer review of source code to detect defects and verify compliance with standards	Development team	Review reports
Unit testing	Verification of individual components in isolation	Development team	Test cases, results, coverage
Integration testing	Verification of interaction between components	Development team	Test cases, results
System testing	Verification of the complete system against requirements	Testing team	Test cases, results
Regression testing	Verification that changes do not introduce new defects	Testing team	Automated results

5. INPUT AND OUTPUT CRITERIA

ENTRY CRITERIA TO START VERIFICATION:

- The requirements have been approved and are under version control.
- The code has been developed in accordance with established standards.
- The test environment is configured and validated.
- The test cases have been designed and reviewed.

EXIT CRITERIA FOR DECLARING VERIFICATION TO BE COMPLETE:

- All the planned test cases have been executed.
- All critical and major defects have been corrected and verified.
- Code coverage meets the established minimum (specified by class).
- The verification reports have been reviewed and approved.

6. COVERAGE REQUIREMENTS

IEC 62304 Class	Statement coverage	Decision coverage	Coverage of conditions
A	Not required	Not required	Not required
B	Recommended	Not required	Not required
C	≥ 80%	≥ 80%	Required for critical conditions

FOR THIS PROJECT (CLASS [A/B/C]):

Type of coverage	Minimum target	Measurement tool
Statements	[specify]	[tool]
Decisions	[specify]	[tool]
Conditions	[specify]	[tool]

7. IDENTIFICATION OF REQUIREMENTS TO BE VERIFIED

Requirement ID	Description	Verification Type	Priority	Responsible
		(Unit/Integration/System)	(High/Medium/Low)	

EXAMPLE:

Requirement ID	Description	Verification Type	Priority	Responsible
REQ-001	The system must calculate the insulin dose based on current glucose levels and carbohydrates consumed	System	High	Testing Team
REQ-002	The interface must clearly display the recommended dose before confirmation	System	High	Testing Team
REQ-003	The system must record all recommendations in a nonmodifiable log	Integration	Medium	Development Team

8. TEST CASE DESIGN

For each identified requirement, test cases are designed that include the following:

Element	Description
Case ID	Unique test case code
Associated Requirement	ID of the requirement being verified
Description	What is being tested and how
Preconditions	Required state before the test
Input data	Specific values to enter
Steps	Sequence of actions to be performed
Expected result	Expected system behavior
Result obtained	(Completed after execution)
Status	(OK/Not OK/Pending)

EXAMPLE:

Element	Description
Case ID	CP-001-01
Associated Requirement	REQ-001
Description	Verify dose calculation for a type 1 patient with a blood glucose level of 180 mg/dL and 60 g of carbohydrates
Prerequisites	Patient configured with an insulin-to-carbohydrate ratio of 1:10 and a sensitivity factor of 1:50
Input Data	Glucose: 180 mg/dL, Carbohydrates: 60 g
Steps	1. Access the calculation screen. 2. Enter glucose. 3. Enter carbohydrates. 4. Press calculate
Expected result	Calculated dose: (180-100)/50 + 60/10 = 1.6 + 6 = 7.6 units
Result obtained	7.6 units
Status	OK

9. VERIFICATION ENVIRONMENT

Component	Description
Hardware	Specifications of the equipment used for testing
Software	Operating systems, databases, dependencies
Testing tools	Unit testing frameworks, automation tools, data generators
Test data	Description of the datasets used (real, anonymized, synthetic)
Network configuration	If applicable, network configuration for integration testing

10. VERIFICATION SCHEDULE

Phase	Activity	Start Date	End Date	Responsible
Phase 1	Test case design	[DD/MM/YYYY]	[DD/MM/YYYY]	[Name]
Phase 2	Unit Testing	[DD/MM/YYYY]	[DD/MM/YYYY]	[Name]
Phase 3	Code reviews	[DD/MM/YYYY]	[DD/MM/YYYY]	[Name]
Phase 4	Integration testing	[DD/MM/YYYY]	[DD/MM/YYYY]	[Name]
Phase 5	System testing	[DD/MM/YYYY]	[DD/MM/YYYY]	[Name]
Phase 6	Regression Testing	[DD/MM/YYYY]	[DD/MM/YYYY]	[Name]
Phase 7	Final Report	[DD/MM/YYYY]	[DD/MM/YYYY]	[Name]

11. DEFECT MANAGEMENT

Defect Type	Definition	Required Action	Maximum Resolution Time
Critical	Prevents the use of the system or may cause harm to patients	Immediate correction, full reverification	24 hours
Major	Affects important but noncritical functionality	Priority fix, verification of affected functionality	1 week
Minor	Affects usability or minor nonfunctional aspects	Scheduled fix, verification in next cycle	Next version

DEFECT MANAGEMENT PROCESS:

1. Detection and recording of defects in the tracking system.
2. Classification by severity.
3. Assignment to the responsible party.
4. Correction and verification.
5. Close the defect with documentation of the solution.

12. VERIFICATION REPORTS

At the end of each verification phase, a report is generated that includes the following:

Section	Description
Executive Summary	Overview of results
Results by Requirement	Verification status of each requirement
Defects found	List of defects with their status
Coverage achieved	Code and requirement coverage metrics
Conclusions	Assessment of whether exit criteria are met
Recommendations	Actions to take before the next phase

13. RESPONSIBILITIES

Role	Responsibilities
Verification Lead	Plan, supervise, and coordinate all verification activities
Developers	Perform unit tests and code reviews; fix defects
Testing Team	Design and execute integration and system tests
Quality Manager	Verify that processes are followed correctly and that documentation is adequate

14. APPROVAL

Role	Name	Signature	Date
Verification Lead			
Quality Manager			
Project Manager			

15. CHANGE CONTROL

Version	Date	Description of Change	Author	Approved by
1.0	[DD/MM/YYYY]	Initial version	[Name]	[Name]

INSTRUCTIONS FOR USE

FOR THE RISK MATRIX:

1. The severity and probability scales are adapted to the specific context of the project.
2. The acceptability criteria are clearly defined before beginning the analysis.
3. Involve a multidisciplinary team (engineers, clinicians, quality) in hazard identification.
4. Document not only the identified risks but also why certain hazards were excluded.
5. Keep the matrix up to date throughout the entire product lifecycle.
6. Each risk is linked to control measures and their verification.

FOR THE VERIFICATION PLAN:

1. The coverage requirements are adapted to the software's safety class.
2. Design test cases that cover normal, boundary, and error scenarios.
3. Traceability between requirements, test cases, and results should be ensured.
4. Automate tests whenever possible to facilitate repeatability.
5. Document not only successful results but also failures and their resolution.
6. Keep the plan up to date as requirements evolve.

These templates are starting points; each project will require specific adaptations on the basis of its complexity, risks, and regulatory requirements. Rigorous documentation is not an end in itself but a means to ensure the safety and effectiveness of medical software.

REFERENCES

INTERNATIONAL REGULATORY STANDARDS

U.S. Food and Drug Administration (FDA). *Software as a Medical Device (SaMD): Clinical Evaluation*. Guidance for Industry and Food and Drug Administration Staff. FDA, 2017. Available at: www.fda.gov. This foundational guidance establishes the specific regulatory framework for medical software in the United States, defining classification and clinical evaluation criteria that have influenced global regulators.
U.S. Food and Drug Administration (FDA). *Artificial Intelligence/Machine Learning-Based Software as a Medical Device Action Plan*. FDA, 2021. Document outlining the agency's strategy for regulating artificial intelligence systems in healthcare, including considerations regarding continuous adaptation and algorithmic transparency.
European Commission. *Regulation (EU) 2017/745 of the European Parliament and of the Council of April 5, 2017 on medical devices*. Official Journal of the European Union, 2017. Fundamental legal text modernizing the regulation of medical devices in Europe, with specific implications for software as a medical device.
European Commission. *Regulation (EU) 2017/746 of the European Parliament and of the Council of 5 April 2017 on* in vitro *diagnostic medical devices*. Official Journal of the European Union, 2017. Regulatory framework for in vitro diagnostics with significant implications for biological data analysis software.
European Commission. *Proposal for a Regulation of the European Parliament and of the Council establishing harmonized rules on artificial intelligence (AI Act)*. 2021–2024. The world's first comprehensive regulatory framework specifically dedicated to artificial intelligence systems, with risk categorization and proportionate obligations.
International Organization for Standardization. *ISO 13485:2016 Medical devices — Quality management systems — Requirements for regulatory purposes*. ISO, 2016. International standard for quality management systems for medical devices, fully applicable to medical software.
International Electrotechnical Commission. *IEC 62304:2006 Medical device software — Software life cycle processes*. IEC, 2006 (2015 amendment). Technical standard establishing the life cycle processes for medical software according to safety levels.
International Electrotechnical Commission. *IEC 62366-1:2015 Medical devices — Application of usability engineering to medical devices*. IEC, 2015. Framework for usability engineering applied to medical devices, including software.
International Organization for Standardization. *ISO 14971:2019 Medical devices — Application of risk management to medical devices*. ISO, 2019. Risk management standard applicable to the entire life cycle of medical devices.

HEALTHCARE INTEROPERABILITY AND DATA STANDARDS

Health Level Seven International. *HL7 FHIR Release 4*. HL7, 2020. Technical specification of the Fast Healthcare Interoperability Resources standard, fundamental to modern interoperability in digital health.
Digital Imaging and Communications in Medicine. *DICOM Standard*. National Electrical Manufacturers Association, continuously updated. Universal standards for the communication of medical images and related information.
SNOMED International. *SNOMED CT Technical Implementation Guide*. SNOMED International, annual edition. Implementation guide for the most comprehensive clinical terminology available.
Regenstrief Institute. *Logical Observation Identifiers Names and Codes (LOINC) Database*. Regenstrief Institute, continuously updated. Coding system for clinical laboratory observations.
World Health Organization. *ATC/DDD index*. WHO Collaborating Centre for Drug Statistics Methodology, updated annually. Anatomical, therapeutic, and chemical classification systems for drugs.

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING IN HEALTHCARE

Char, D. S., Shah, N. H., & Magnus, D. (2018). Implementing machine learning in health care — addressing ethical challenges. *Annals of Internal Medicine*, 168(9), 619-625. This seminal article identifies the specific ethical challenges of implementing machine learning in clinical contexts.
Ghassemi, M., Oakden-Rayner, L., & Beam, A. L. (2021). The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, 3(11), e745-e750. A critical analysis of the limitations of current explainability techniques in medical applications.
Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464), 447-453. This is a landmark study documenting algorithmic biases in health care systems and their consequences.
Rajpurkar, P., Chen, E., Banerjee, O., & Topol, E. J. (2022). AI in health and medicine. *Nature Medicine*, 28(1), 31-38. This is a comprehensive review of the state of the art in artificial intelligence in medicine.
Topol, E. J. (2019). *Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again*. Basic Books. An accessible yet well-researched book that argues for medicine that is enhanced by, but not replaced

by, artificial intelligence.

SOFTWARE ENGINEERING AND METHODOLOGIES IN REGULATED CONTEXTS

McHugh, F., O'Connor, P., & McCaffery, F. (2018). Agile practices in the medical device software domain *2018 IEEE International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. Analysis of the application of agile methodologies in medical software development.
Vogel, D. A. (2011). *Medical Device Software Verification, Validation, and Compliance*. Artech House. Technical reference text on the verification and validation of medical software.
García-Valls, M., Calva-Urrego, C., & de la Fuente, M. A. (2018). The dependability of component-based real-time systems can be increased through formal verification. *Journal of Systems Architecture*, 88, 1-16. An approach to formal verification in critical real-time systems.

RISK MANAGEMENT AND PATIENT SAFETY

Institute of Medicine. (2000). *To Err Is Human: Building a Safer Health System*. National Academy Press. Foundational report on patient safety in modern healthcare systems.
Reason, J. (1990). *Human Error*. Cambridge University Press. Classic work on the nature of human error in complex systems is applicable to the design of medical software.
Vincent, C., & Amalberti, R. (2016). *Safer Healthcare: Strategies for the Real World*. Springer Open. Practical strategies for improving safety in real clinical settings.

CASE STUDIES AND SPECIFIC APPLICATIONS

Beede, E., Baylor, E., Hersch, F., Iurchenko, A., Wilcox, L., Ruamviboonsuk, P., & Vardoulakis, L. M. (2020). A human-centered evaluation of a deep learning system deployed in clinics for the detection of diabetic retinopathy. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–12. User-centered evaluation of artificial intelligence systems for diabetic retinopathy.
Muehlematter, U. J., Daniore, P., & Vokinger, K. N. (2021). Approval of artificial intelligence and machine learning-based medical devices in the USA and Europe (2015–20): a comparative analysis. *The Lancet Digital Health*, 3(3), e195-e203. Comparative analysis of regulatory approvals for artificial intelligence devices.
Sendak, M., Gao, M., Nichols, C., Lin, A., & Balu, S. (2020). Real-world implementation of machine learning in healthcare: a systematic review of challenges and opportunities. *NEJM Catalyst*, 1(4). Systematic review of real-world implementations of machine learning in healthcare.

ETHICS, BIAS, AND EQUITY IN DIGITAL HEALTH

Chen, J. H., & Asch, S. M. (2017). Natural language processing for clinical documentation: challenges and opportunities. *Journal of General Internal Medicine*, 32(12), 1362-1364. Reflection on the implications of natural language processing in clinical documentation.
Price, W. N., & Cohen, I. G. (2019). Privacy- s in the age of medical big data. *Nature Medicine*, 25(1), 37-43. Analysis of privacy challenges in the era of big data in healthcare.

ORGANIZATIONS AND REFERENCE RESOURCES

European Medicines Agency (EMA). *Guidelines on the qualification and reporting of digital health technologies*. EMA, 2020-2023. Specific guidelines for digital health technologies in the European context.
International Medical Device Regulators Forum (IMDRF). *Software as a Medical Device (SaMD): Possible Framework for Risk Categorization and Corresponding Considerations*. IMDRF, 2014. International framework for the risk categorization of medical software.
National Institute of Standards and Technology (NIST). *Cybersecurity for IoT Programs*. NIST, continuously updated. Resources for cybersecurity of connected devices in healthcare.
World Health Organization. *Digital Health: A Call to Action*. WHO, 2023. Global strategic framework for digital health from the World Health Organization.

RECOMMENDED FURTHER READING

For readers wishing to delve deeper into specific topics, the following additional reference works are suggested:
Coiera, E. (2015). *Guide to Health Informatics* (3rd ed.). CRC Press. Comprehensive text on health informatics.

Friedman, C. P., & Wyatt, J. C. (2006). *Evaluation Methods in Medical Informatics* (2nd ed.). Springer. Evaluation methods in medical informatics.
Hripcsak, G., & Albers, D. J. (2013). Next-generation phenotyping of electronic health records. *Journal of the American Medical Informatics Association*, 20(1), 117-121. Phenotyping was performed on the basis of electronic health records.
Kahn, M. G., Callahan, T. J., Barnard, J., Bauck, A. E., Brown, J., Davidson, B. N., ... & Bernstam, E. V. (2021). A harmonized data quality assessment terminology and framework for the secondary use of electronic health record data. *eGEMs*, 4(1). Data quality framework for the secondary use of electronic health records.