

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



© 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

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.

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

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

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

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

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.

SaMD Risk Classification Matrix - IMDRF				
LEVEL	DESCRIPTION	SOFTWARE FUNCTION	CLINICAL EXAMPLES	REGULATORY IMPACT
IV HIGHEST	Critical treatment	Directly controls life-sustaining therapies	- Smart infusion pumps - Radiotherapy planning - Connected defibrillators - Assisted mechanical ventilation	Most demanding process. Mandatory clinical trials. PMA / MDR Class III.
III HIGH	Diagnosis that influences critical therapy	Provides information for therapeutic decisions in serious conditions	- Arrhythmia detection (ECG-AI) - Early sepsis diagnosis - Emergency triage - Ischemic stroke detection	Notified body review. FDA 510(k)/De Novo. MDR Class IIb.
II MEDIUM	Diagnosis that influences non-critical conditions	Supports diagnosis and monitoring in controllable diseases	- Type 2 diabetes management - Hypertension monitoring - Mental health apps - COPD management	Registration with clinical performance documentation. 510(k) / MDR Class IIa.
I LOWEST	Information without diagnosis or therapy	Records or displays data without active clinical interpretation	- Schedules and reminders - Manual symptom logging - Test result portals - Medication information	Exemption or simplified registration. Manufacturer self-certification

Source: IMDRF, Software as a Medical Device (SaMD): Application of Quality Management System, IMDRF/SaMD WG/N23 FINAL 2015.

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

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

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

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.

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.

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

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?

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/>