Chapter 6 / Capítulo 6

SAP Superintelligence Series
ISSN: 3144-5624
ISBN (complete work): 978-9915-9851-0-7

Superintelligence Series Volume 4: One Person Company: Artificial Intelligence, Autonomous Entrepreneurship and the Future of Business
ISBN (this volume): 978-9915-9966-3-9
DOI: 10.62486/978-9915-9966-3-9.Ch06
Pages: 59-69

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.

Contributor concentration in popular open-source artificial-intelligence agent frameworks

Concentración de contribuidores en marcos de agentes de inteligencia artificial de código abierto populares

Carlos Alberto Gómez Cano[1]

1 Corporación Universitaria de Asturias. Bogotá, Colombia.

ABSTRACT

Introduction: Open-source artificial-intelligence agent frameworks are infrastructure on which downstream applications depend, and their sustainability turns on whether they are maintained by one person or by a community. Methods: The twenty most-starred public repositories matching the query "AI agent framework" were retrieved from the GitHub application programming interface, and the share of commits contributed by the top contributor and by the top three contributors was computed for each. Results: The top contributor accounted for a median of 37.7 percent of commits, with a range from 11.7 to 91.2 percent, and five of the twenty repositories had a top contributor above 50 percent. The top three contributors accounted for a median of 57.6 percent of commits. Conclusions: Contribution to the most popular open-source agent frameworks is highly concentrated, and a substantial minority of these frameworks depend on a single dominant contributor, a maintenance risk that the field's reliance on them does not currently price in.

Keywords: contributor concentration; open source; AI agents; bus factor; software sustainability

RESUMEN

Introducción: Los marcos de agentes de inteligencia artificial de código abierto son infraestructura de la que dependen aplicaciones posteriores, y su sostenibilidad depende de si los mantiene una persona o una comunidad. Métodos: Se recuperaron de la interfaz de programación de GitHub los veinte repositorios públicos con más estrellas que coinciden con la consulta "AI agent framework", y se calculó para cada uno la proporción de commits aportada por el contribuidor principal y por los tres principales. Resultados: El contribuidor principal aportó una mediana del 37,7 por ciento de los commits, con un rango de 11,7 a 91,2 por ciento, y cinco de los veinte repositorios tenían un contribuidor principal por encima del 50 por ciento. Los tres contribuidores principales aportaron una mediana del 57,6 por ciento de los commits. Conclusiones: La contribución a los marcos de agentes de código abierto más populares está muy concentrada, y una minoría sustancial de estos marcos depende de un único contribuidor dominante, un riesgo de mantenimiento que la dependencia del campo hacia ellos no valora actualmente.

Palabras clave: concentración de contribuidores; código abierto; agentes de IA; factor de bus; sostenibilidad del software

INTRODUCTION

Open-source artificial-intelligence agent frameworks have become load-bearing infrastructure, in the precise sense that a large and growing share of agent-based applications now build on them rather than on bespoke code. Frameworks such as LangChain, MetaGPT and AutoGen now mediate how a large share of agent-based applications are built, and their popularity, measured in stars, runs into the hundreds of thousands, which makes them the de facto platform layer of a fast-growing application ecosystem. The question of who maintains these frameworks is therefore not an academic one: the sustainability of open-source software depends on the distribution of its contributors, and a project that rests on a single dominant contributor is one departure away from abandonment.(1) The bus factor, the number of contributors whose loss would stall a project, is the standard shorthand for this risk, and its estimation is a mature subfield of software engineering.[2,3] For a category of infrastructure this young, the contributor distribution is not yet settled, and it is exactly the unsettled categories where the single-maintainer risk is highest, because the diversification that comes with age has not yet had time to occur.

The concern is sharper for agent frameworks than for ordinary libraries, for two reasons. The category is also, by the nature of what it builds, a dependency multiplier: an agent framework is not a terminal application but a substrate that other applications embed, so the maintenance risk of the framework propagates to everything that embeds it, and a single-maintainer framework that fails takes all of its downstream users with it in a way that a standalone tool does not. This propagation is what distinguishes the concentration risk in a framework from the same risk in a standalone tool, and it is why the head of the category, the frameworks with the most downstream use, is the part that matters most. First, these frameworks are new and fast-moving, which means their contributor bases have had less time to diversify, and a single founder who built the framework may still dominate it. Second, the frameworks sit at the frontier of a field whose research and engineering norms are themselves still forming, so the maintenance burden may be concentrated in the hands of whoever understands the design deeply, which is often one person.(4) The literature on core-contributor dynamics documents exactly this pattern of concentration and its risks across open-source software more broadly,[5,6] and the abandonment of once-popular projects is a documented outcome of that concentration.(7) The ownership structure of highly starred repositories is itself a measured property of the ecosystem, and its distribution across software and artificial-intelligence domains has been shown to follow the same concentration logic.(8) The question is made more acute by the fact that many such frameworks are company-hosted, and company-hosted projects carry their own governance and contributor dynamics,[9] including the risk that a vendor changes the terms of openness and sheds its volunteer contributors.(10)

The field's own self-understanding is still settling, and the maintenance question is usually asked only after a failure has already occurred, which is the point at which the answer is no longer useful to anyone. When a popular framework loses its maintainer, the community discovers the concentration retroactively, and the discovery is expensive because the framework is already embedded in downstream code. The measurement reported here is intended to make the question askable before the failure, by quantifying the concentration that a maintainer's departure would expose. This is the specific gap the study addresses, and it is a gap of measurement rather than of theory, because the theory of single-maintainer risk is already well established in the software-engineering literature.

What the literature has not supplied is a measurement of contributor concentration specific to the most popular agent frameworks, the ones that downstream work actually depends on. Studies of the bus factor and of core contributors exist,[11,12] and the popularity-maintenance trade-off in highly starred repositories has been examined across software and artificial-intelligence domains,[8] but the head of the agent-framework category has not been isolated and measured. The gap matters because a category-level average would hide the heterogeneity between the mature frameworks, which have diversified, and the newer entrants, which have not, and it is precisely that heterogeneity that a funder or an adopter needs to see. This study supplies that measurement. The objective is to measure the concentration of code contribution in the most popular open-source artificial-intelligence agent frameworks, and to characterize how many of them depend on a single dominant contributor.

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

METHODS

DATA SOURCE AND SAMPLE

The GitHub public application programming interface was queried without authentication. The twenty most-starred public repositories returned by the search query "AI agent framework" were selected as the sample, on the grounds that stars are the field's own proxy for a framework that downstream work depends on, and the search-based selection is a reproducible way to define the head of the category without relying on the authors' prior knowledge of which frameworks are important. The query term was chosen to be broad enough to capture the frameworks that describe themselves as agent frameworks, and the top-twenty cut was chosen to keep the sample small enough for a complete contributor retrieval while remaining large enough to admit several distinct concentration regimes. For each selected repository, the contributors endpoint was queried to retrieve the list of contributors ordered by contribution count, with anonymous contributors excluded. The extraction is a cross-sectional snapshot of the platform's contribution data at the time of retrieval. GitHub has long been established as a valid empirical source for large-scale studies of code and development activity,[13] and the contribution metadata it exposes through the contributors endpoint is the same signal that the bus-factor and core-contributor literatures rely on.[1,5]

CONCENTRATION MEASURE

The primary measure was the share of total commits contributed by the single most prolific contributor, computed as that contributor's commit count divided by the sum of all listed contributors' commit counts, expressed as a percentage. The secondary measure was the share of the three most prolific contributors. The contributors endpoint returns at most one hundred contributors per repository, so the denominator is the sum over the top one hundred listed contributors; this cap is a measurement limitation that the Discussion returns to. The measure is a concentration ratio rather than a formal bus-factor estimate, and it was chosen over the formal estimate because it is transparent, reproducible from a single endpoint call, and directly interpretable by the practitioners and funders who are the intended audience of an applied measurement. The 50 percent threshold used to flag single-contributor dominance follows the natural meaning of the share: a single contributor who writes more than half the code is, by that measure alone, the effective maintainer of the project.

STATISTICAL ANALYSIS

For the set of repositories, the median, mean, minimum and maximum of the top-contributor share and of the top-three share were computed, along with the number of repositories whose top contributor exceeded 50 percent, the threshold at which a single person accounts for more than half of the code. No hypothesis test was performed, because the analysis is a descriptive census of a defined set rather than a sample drawn for inference. The measure used here is a simplified truck factor: rather than estimating the minimum number of contributors whose loss would stall the project, which is the formal object of the truck-factor literature,[2,3,14] the analysis reports the more transparent top-contributor share, which is a direct, reproducible proxy for the same underlying risk and is the quantity that a practitioner can compute from a single endpoint call.(1) The relationship between this share and the formal truck factor is well understood, and the top-contributor share is a defensible simplification for a descriptive study whose aim is to characterise the head of the category rather than to estimate the exact bus factor of each project. Analyses were performed in Python with the standard library and numpy.

RESULTS

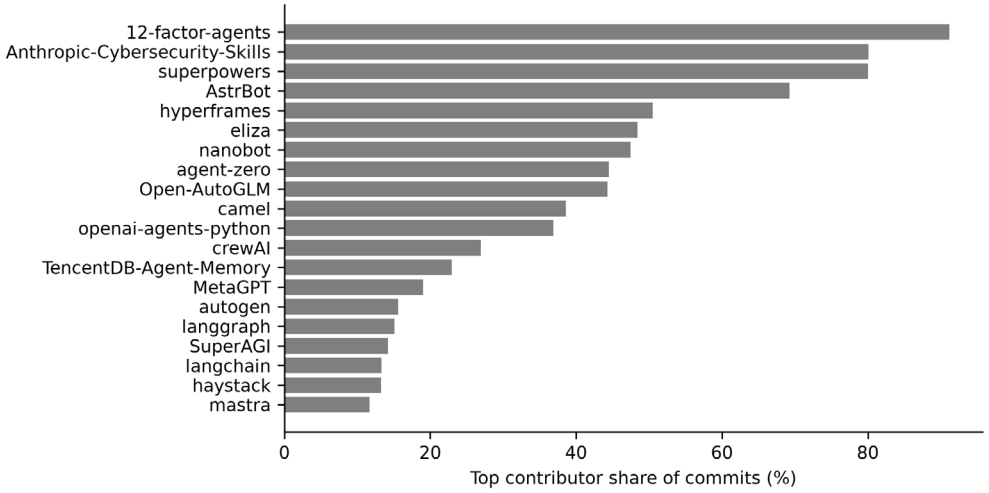
The top contributor accounted for a median of 37.7 percent of commits across the twenty frameworks, and for a mean of 39.2 percent, with the mean exceeding the median because a few very concentrated frameworks pulled the average value upward (Figure 1).

The range ran from 11.7 percent to 91.2 percent. Five of the twenty frameworks had a top contributor above 50 percent. The most concentrated framework, with a top contributor at 91.2

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

percent, stood at the opposite end from the least concentrated, at 11.7 percent. The single most concentrated repository was a twelve-factor-agent framework with only fifteen listed contributors, and the least concentrated was an agent platform with one hundred listed contributors, which is the endpoint cap.

Figure 1. Top-contributor share of commits across the twenty most-starred open-source AI agent frameworks.



Note: Each bar is a repository. The dashed vertical line marks 50 percent, above which a single contributor accounts for a majority of commits.

The top three contributors accounted for a median of 57.6 percent of commits, and for a mean of 61.9 percent, so a majority of the code in a typical framework came from just three people (Table 1).

Table 1. Contributor concentration in the twenty most-starred open-source AI agent frameworks.

Repository	Stars	Contributors	Top-1 share (%)	Top-3 share (%)
humanlayer/12-factor-agents	26903	15	91.2	94.5
mukul975/Anthropic-Cybersecurity-Skills	35188	15	80.1	93.2
obra/superpowers	307316	37	80.0	93.2
AstrBotDevs/AstrBot	44486	100	69.2	75.5
heygen-com/hyperframes	74622	100	50.5	84.2
elizaOS/eliza	19698	100	48.4	81.9
HKUDS/nanobot	49585	100	47.5	79.0
agent0ai/agent-zero	19593	58	44.5	85.9
zai-org/Open-AutoGLM	26497	10	44.3	84.1
camel-ai/camel	17989	100	38.6	50.4
openai/openai-agents-python	30355	100	36.9	59.5
crewAIInc/crewAI	60221	100	26.9	55.7
TencentCloud/TencentDB-Agent-Memory	30058	33	23.0	34.4
FoundationAgents/MetaGPT	71058	100	19.0	47.0
microsoft/autogen	61607	100	15.6	43.3
langchain-ai/langgraph	44060	100	15.1	39.8
TransformerOptimus/SuperAGI	17721	63	14.2	40.2
langchain-ai/langchain	148625	100	13.4	36.4
deepset-ai/haystack	26871	100	13.2	31.6
mastra-ai/mastra	29156	100	11.7	27.6

Note: Top-1 and Top-3 share are the percentage of total listed commits contributed by the most prolific contributor and the three most prolific contributors. Contributors is capped at 100 by the endpoint.

Below that value, the distribution was skewed: the largest and oldest frameworks, LangChain at 13.4 percent and MetaGPT at 19.0 percent for the top contributor, sat at the low-

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

concentration end, while smaller and newer frameworks dominated the high-concentration end. The inverse relationship between popularity and concentration was consistent across the set, with the highest-starred repositories showing the lowest top-contributor shares and the youngest repositories showing the highest.

To a lesser extent, the three-contributor share told the same story, with the mature frameworks sitting near one-third and the concentrated frameworks above ninety percent, which means the gap between the two ends of the category is not a matter of a single outlier but of a systematic gradient.

Second, the concentration was not a function of total contributor count alone. The median listed contributor count reached 100, the endpoint cap, which means the true contributor count is higher and the top-contributor share is computed over the most active one hundred contributors, a denominator that slightly overstates concentration.

At the same time, five frameworks fell below fifty listed contributors, and these were also the most concentrated, with top-contributor shares from 44.3 to 91.2 percent. The five frameworks above 50 percent were spread across both the commercial and the community sectors of the category, which indicates that the concentration is not specific to a single origin.

DISCUSSION

The concentration of code contribution in the most popular agent frameworks is high, and a substantial minority of these frameworks depend on a single dominant contributor, a finding with direct consequences for who bears the maintenance risk of the ecosystem and for how that same risk is currently being systematically under-priced. The practical implication that leads is that downstream adopters are, without necessarily knowing it, betting on the continued availability of a handful of individuals, and the field's reliance on agent frameworks does not currently price in that maintenance risk. For a developer choosing a framework to build on, the top-contributor share is a cheap, direct signal of that risk, and it should sit alongside stars and activity as a first-order selection criterion, because a framework with a 90 percent single contributor and a framework with a 13 percent single contributor carry very different abandonment odds even when their star counts are comparable.

The phrase "one-person infrastructure" deserves to be taken quite literally here, because the wider open-source world has already shown what it means in practice. The maintainer of a critical package is a known figure of the software commons, and the failure of a single-maintainer project is a recurring incident rather than a hypothetical, which is why the bus factor is estimated at all.(1) The agent frameworks measured here are, in several cases, the same shape of object: a single author who built the system and still writes the majority of its code. The difference is one of scale and stakes, because an agent framework is a dependency of many other projects rather than a leaf package, so its single-maintainer risk propagates downstream more widely. The present measurement makes that propagation concrete by identifying the frameworks where the risk is highest, and by showing that the risk is not evenly spread across the category.

The contribution of the study is therefore genuinely twofold, and the two parts should be kept distinct in the reader's mind. The first is the measurement itself, which establishes that the head of the agent-framework category carries a median top-contributor share near forty percent and a one-person majority in a full quarter of its members. The second is the gradient, which establishes that the concentration is not uniform but declines with maturity, so the risk is highest exactly where the field's newest and least-settled dependencies are forming. The two parts together turn a vague worry about single-maintainer infrastructure into a measurable, located risk, and that transformation is what makes the result actionable rather than merely cautionary. The measurement does not by itself tell anyone which framework to choose, but it tells them what to look at before choosing, and in an ecosystem where the choice is often made on stars alone, that is a genuinely useful addition.

The artificial-intelligence dimension adds one further consideration that the classic bus-factor literature did not have to face. Agent frameworks are themselves now being written with the assistance of artificial-intelligence coding tools, which could either relieve the single-maintainer burden, by lowering the cost of maintenance, or deepen it, by lowering the cost of a single person producing and maintaining a system that others do not understand. The direction of this effect is genuinely open, and it is exactly the kind of question that a follow-up measurement, re-running the present analysis a year later, would begin to answer empirically. The ai4se taxonomy and the research-software-engineering frameworks that describe this new mode of work[15,16] are the

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

context in which this question will be answered, and the concentration reported here is a baseline against which the effect of that tooling on contributor distribution can later be measured. For the present, the data show that whatever assistance exists has not yet diversified the contributor base of the most concentrated frameworks, which is itself a finding worth recording.

The comparison with the software-engineering literature confirms that this concentration is the normal pathology of open-source projects rather than an anomaly of the agent category, and that reading it as normal should never be confused with reading it as benign. A normal pathology is still a pathology, and the literature's consistent finding that concentration predicts fragility is precisely why the bus factor is worth measuring. The agent frameworks are not special in being concentrated; they are special only in that their concentration has not yet been measured, and in that their downstream reach is large. The present result closes the first gap and makes the second visible, which is the contribution that a measurement can make when the theory is already settled. Core-contributor disengagement is known to depress project activity,[17] core-developer turnover is a measured predictor of abandonment,[6] and the survival of open-source projects is itself contingent on the maintenance of a contributor base.(7) The truck-factor literature has operationalised exactly this risk,[2,3,14] and the bus factor is estimated in practice precisely because a single person can carry a project.(1) The finding that five of twenty frameworks exceed a 50 percent top-contributor share places the agent category at the concentrated end of the distribution that this literature has mapped.

The adopter's checklist that follows from the result is deliberately short, because a decision signal that is not short will not be used. First, read the top-contributor share alongside the star count, not after it. Second, treat a share above 50 percent as a red flag and a share below 20 percent as a sign of a diversified base. Third, weigh the share against the framework's age, because a young framework with a high share is the normal starting condition of every successful project, whereas an old framework with a high share is a maintenance risk that has not yet resolved. These three rules compress the measurement into a form that a practitioner can apply while scanning a list of candidate frameworks, and they are the practical form of the study's finding. The checklist is not a substitute for the deeper truck-factor tools, but it is a screening step that costs one endpoint call and no specialised software.

The funder's decision can now be stated with the same directness. A foundation that wants to harden the agent-ecosystem infrastructure should target the concentrated, high-dependency frameworks before their single maintainer leaves, because the abandonment literature predicts that a project resting on one person fails when that person departs, and the failure is always cheaper to prevent than to recover from.(7) The present measurement identifies those frameworks, which is the difference between a funder who knows where to intervene and a funder who is merely aware that the risk exists. The data show where those frameworks are: the newer entrants at the high-concentration end of the distribution, not the mature frameworks that have already diversified. A grant that adds a second maintainer to a framework with a 90 percent top contributor buys more resilience than a grant that adds a marginal contributor to a framework with a 13 percent top contributor, and the present measurement is what lets a funder tell the two apart. This is the applied payoff of measuring concentration at the head of the category rather than reporting a category average.

The inverse relationship between popularity and concentration deserves separate emphasis, because it complicates the naive reading. The largest frameworks, LangChain, MetaGPT and AutoGen, have the most diversified contributor bases, with top-contributor shares in the teens, while the smaller and newer frameworks are the ones that rest on a single person. This means the risk is not uniformly distributed across the category: the frameworks that downstream work most heavily depends on are the least concentrated, and the concentrated ones are the newer entrants whose long-term survival is still unproven. The implication is that a framework's contributor concentration is best read alongside its maturity, not as a standalone badge of risk, and a rule of thumb that flags high concentration without considering age would misclassify every young framework, including the ones that will later diversify and succeed.

The strengths of the measurement are its transparency and its deliberate position at the head of the category. The top-contributor share is computable by anyone from a public endpoint, which makes the result independently checkable, and the sample is the set of frameworks that downstream work actually depends on, rather than a convenience sample of repositories that happen to be accessible. The measurement is therefore both reproducible and decision-relevant, which is the combination that an applied study of infrastructure risk should have. The concentration gradient that the measurement reveals, from a 13 percent top contributor in the largest frameworks to a 91 percent top contributor in the smallest, is a property of the category

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

that no average would have shown, and it is the property on which the practical recommendations rest.

The limitations of the measurement are the standard ones for GitHub-based contributor analysis, and they are worth naming precisely because each has a known direction of bias. The contributor count is capped at one hundred, which slightly overstates the top-contributor share by excluding the long tail of contributors; the remedy is paginated retrieval of the full contributor list, which the public rate limit makes expensive but which a re-run could apply. The commit count as the measure of contribution understates non-commit work such as issue triage and documentation; the remedy is to complement commits with a broader activity measure. The sample of twenty repositories, selected by stars, is the head of the category and not the category as a whole; the remedy is a larger sample that spans the long tail, where concentration is likely higher. These limitations are bounded in their effect: each would move the estimate in a known direction, and none would overturn the central finding that a meaningful minority of popular frameworks rest on a single dominant contributor.

The measurement of the bus factor has matured to the point where the concentration reported here could be monitored continuously rather than as a one-off snapshot, and the tools for that monitoring are no longer the bottleneck, so the remaining question is simply whether anyone in the ecosystem is actually watching. Tooling now exists to estimate the bus factor from commit history,[12] to predict the truck factor with machine learning before it becomes a crisis,[18] and to forecast core-contributor trends from project metrics,[19] and file-level significance weighting refines the estimate beyond a flat commit count.(11) The top-contributor share reported here is the rawest version of what these tools estimate, and the fact that a simple endpoint call already reveals five frameworks above a 50 percent share suggests that the more sophisticated tools would flag an even wider set of at-risk projects. The human factors that drive contributor retention, participation and turnover are themselves a mature object of study,[20] and the collaboration patterns and network positions that predict long-term contribution are measurable in the same data.(21) The identification of the key contributor in an open community has been formalised through organisational-learning and network-structural measures,[22] which is the deeper analysis that a single top-contributor share only approximates.

The sustainability consequences of concentration are equally well documented in the literature, and they bind the present finding to the wider open-source ecosystem rather than leaving it as a property of one category. Contributor turnover is a measured predictor of maintenance difficulty, and task assignment can be designed to counter it.(23) The structure of teams in social coding projects conditions how robustly a project survives the loss of any one member,[24] and one-time contributors differ systematically from core and peripheral developers in ways that matter for whether a project can renew its contributor base.(25) The death spiral of open-source projects, in which a thinning contributor base feeds on itself, is a documented post-mortem pattern,[26] and the multi-ecosystem modelling of project sustainability shows that the health of a project is a property of the ecosystems it spans, not just of its own repository.(27) The free-open-source-community literature has asked directly whether community sustainability makes a difference to software quality, and the answer is that it does,[28] which gives the concentration reported here a quality dimension, not just a risk dimension. The empirical study of community and coding patterns in open-source-for-social-good projects extends the same lens,[29] and the collaboration challenges of scientific open-source software show that the maintenance of a healthy contributor base is a general problem, not one specific to commercial agent frameworks.[30] The rise of artificial intelligence in software engineering itself, through the ai4se taxonomy and research-software-engineering frameworks, is the emerging context in which these frameworks and their maintainers now operate,[15,16] and it is a context that may either deepen or relieve the concentration reported here, a question the present snapshot cannot yet answer.

The comparison with the formal truck-factor estimates is the last point on which the result should be checked, because the two measures ought to agree with each other if both are capturing the same underlying risk. The truck-factor algorithms that the literature compares[3,14] estimate the minimum number of developers whose loss would stall a project, and they typically find that the number is small even in large projects. The top-contributor share reported here is consistent with that finding: a project whose top contributor writes 37.7 percent of the commits is, in the language of the truck-factor literature, a project with a truck factor near one, because the loss of that single contributor would remove a large share of the project's working knowledge. The two measures converge on the same risk from different directions, which strengthens confidence that the risk is real rather than an artefact of the chosen metric. Where the top-contributor share adds value is in its immediacy, because it can be read directly from a single number without the

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

algorithmic machinery that the formal truck factor requires, and it is that immediacy that makes it useful to an adopter scanning a list of frameworks.

The generalisability of the result is bounded by its snapshot, its platform and its metric, and each of those bounds is a place where a future extension could widen the claim. The sample is the head of one category on one platform at one time, and a different query, a different date or a different platform could return a different ordering, so the precise percentages should be read as a measurement of this category at this time rather than as a universal property of agent frameworks. The mechanism that generalises, however, is the relationship between maturity and diversification, in which a project starts concentrated and diversifies only if it survives long enough, and that mechanism is portable to any young, fast-moving open-source category. The commit count as a measure of contribution also bounds the result, because it understates the non-commit work of maintenance, and a framework whose top contributor writes all the code but whose community handles the documentation and issues would look more concentrated here than it is in practice.

The broader implication is that the open-source agent ecosystem is, at its head, a story of concentrated stewardship in the newer projects and diversified stewardship in the older ones, and the two states coexist in the same category at the same time rather than following one another in a simple sequence. The one-person-infrastructure problem, familiar from the wider open-source world, is present here in a form the field has not yet acknowledged, because the agent frameworks are young enough that their bus-factor risk has not yet materialised into failures, and a risk that has not yet materialised is easy to discount. The measurement reported here is intended to make that discounting harder to sustain.

The one-person-infrastructure problem, familiar from the wider open-source world, is present here in a form the field has not yet acknowledged, because the agent frameworks are young enough that their bus-factor risk has not yet materialised into failures. A funder or foundation that wants to harden this infrastructure would get the most value from targeting the concentrated, high-dependency frameworks before their single maintainer leaves, rather than waiting for the abandonment that the literature predicts.(7)

CONCLUSIONS

Contribution to the twenty most-starred open-source artificial-intelligence agent frameworks is highly concentrated, with the top contributor accounting for a median of 37.7 percent of commits and the top three for a median of 57.6 percent, and five of the twenty frameworks exceed a single-contributor majority of their commits. The concentration is greatest in the newer entrants and lowest in the mature frameworks, which means the field’s dependence on one-person-maintained agent infrastructure is a present risk that is unevenly distributed across the category, and one that adopters and funders can see directly in the top-contributor share before it materialises into abandonment.

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

REFERENCES

Jabrayilzade E, Evtikhiev M, Tuzun E, Kovalenko V. Bus Factor in Practice. 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). 2022:97-106. doi: 10.1109/icse-seip55303.2022.9793985
Avelino G, Passos L, Hora A, Valente MT. A novel approach for estimating Truck Factors. 2016 IEEE 24th International Conference on Program Comprehension (ICPC). 2016:1-10. doi: 10.1109/icpc.2016.7503718
Ferreira M, Mombach T, Valente MT, Ferreira K. Algorithms for estimating truck factors: a comparative study. Software Qual J. 2019;27(4):1583-1617. doi: 10.1007/s11219-019-09457-2
Wang Z, Feng Y, Wang Y, Jones JA, Redmiles D. Unveiling Elite Developers' Activities in Open Source Projects. ACM Trans Softw Eng Methodol. 2020;29(3):1-35. doi: 10.1145/3387111
Liu X, Bai J, Liu L, Ouyang H, Zhou H, Xu L. Mining Core Contributors in Open-Source Projects. In: Lecture Notes in Computer Science. Springer International Publishing; 2019. p. 690-703. doi: 10.1007/978-3-030-30952-7_70
NOURRY O, KONDO M, SAITO S, IIMURA Y, UBAYASHI N, KAMEI Y. Abandonment and Resilience: Understanding Core Developer Turnover in Open-Source Software. IEICE Trans Inf & Syst. 2025;E108.D(11):1412-1415. doi: 10.1587/transinf.2025edl8005
Avelino G, Constantinou E, Valente MT, Serebrenik A. On the abandonment and survival of open source projects: An empirical investigation. 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2019;2019-Septemer:1-12. doi: 10.1109/esem.2019.8870181
Sebetci Ö. Popularity, maintenance burden, and ownership structure in highly starred GitHub repositories: Evidence across software and AI/ML-related domains. Information and Software Technology. 2026;200:108298. doi: 10.1016/j.infsof.2026.108298
Osborne C, Daneshyan F, He R, Ye H, Zhang Y, Zhou M. Characterising Open Source Co-opetition in Company-hosted Open Source Software Projects: The Cases of PyTorch, TensorFlow, and Transformers. Proc ACM Hum-Comput Interact. 2025;9(2):1-30. doi: 10.1145/3710944
Lumbard K, Afzali FM, Link G. Before and After the Switch: Exploring Contributor Levels when Vendor-Controlled Open Source Projects Go Proprietary. Proceedings of the 2026 ACM Southeast Conference. 2026:2-9. doi: 10.1145/3746467.3801504
Haratian V, Evtikhiev M, Derakhshanfar P, Tüzün E, Kovalenko V. BFSig: Leveraging File Significance in Bus Factor Estimation. Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2023:1926-1936. doi: 10.1145/3611643.3613877
Kazimoto DM, Choetkiertikul M, Ragkhitwetsagul C, Sunetnanta T. DEV-EYE: A Tool for Monitoring Bus Factor Using Commit History. 2024 31st Asia-Pacific Software Engineering Conference (APSEC). 2024:462-466. doi: 10.1109/apsec65559.2024.00060
Ray B, Posnett D, Devanbu P, Filkov V. A large-scale study of programming languages and code quality in GitHub. Commun ACM. 2017;60(10):91-100. doi: 10.1145/3126905
Ferreira M, Valente MT, Ferreira K. A Comparison of Three Algorithms for Computing Truck Factors. 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC). 2017;0:207-217. doi: 10.1109/icpc.2017.35
Schieferdecker IK. Augmenting software engineering with AI. The ai4se taxonomy and its use. Innovations Syst Softw Eng. 2026;22(2):29. doi: 10.1007/s11334-026-00652-6
Farshidi S, Bennin KE, Babur Ö, Sallou J, Kassahun A, Tekinerdogan B. Advancing research software engineering with AI: a research framework. Autom Softw Eng. 2026;33(3):79. doi: 10.1007/s10515-026-00621-0
Chen Y, Stol KJ, Santos F, German DM, Trinkenreich B. How Does Core Contributor Disengagement Impact Open Source Project Activity? A Quasi-Experiment. Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering. 2026:2350-2362. doi: 10.1145/3744916.3773135
El Cheikh Ammar A, Eraslan S, Yesilada Y. Predicting the truck factor in a software repository using machine learning. Information and Software Technology. 2025;184:107765. doi: 10.1016/j.infsof.2025.107765
Zhu Z, Chen T, Wu L, Song Q. CoreCast: Leveraging Project Metrics to Predict Core Contributor Trends in Open Source C57. International Conferences on Software Engineering and Knowledge Engineering. 2024;2024:351-356. doi: 10.18293/seke2024-010
Ruangwan S, Thongtanunam P, Ihara A, Matsumoto K. The impact of human factors on the participation decision of reviewers in modern code review. Empir Software Eng. 2019;24(2):973-1016. doi: 10.1007/s10664-018-9646-1
Jo S, Kwon G. The Impact of Collaboration Patterns and Network Centrality on Long-Term Contribution in GitHub Project. Applied Sciences. 2025;15(1):352. doi: 10.3390/app15010352
Lee UJ, Lee S, Jahng J. Who is the key contributor in an open community? Based on organizational learning theory with structural holes and strength of ties,. Information and Software Technology. 2026;198:108203. doi: 10.1016/j.infsof.2026.108203
Etemadi V, Bushehrian O, Robles G. Task assignment to counter the effect of developer turnover in software maintenance: A knowledge diffusion model. Information and Software Technology. 2022;143:106786. doi: 10.1016/j.infsof.2021.106786
El Mezouar M, Zhang F, Zou Y. An empirical study on the teams structures in social coding using GitHub projects. Empir Software Eng. 2019;24(6):3790-3823. doi: 10.1007/s10664-019-09700-1
Lee A, Carver JC. Are One-Time Contributors Different? A Comparison to Core and Periphery Developers in FLOSS Repositories. 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2017;2017-November:1-10. doi: 10.1109/esem.2017.7

https://doi.org/10.62486/978-9915-9966-3-9.Ch06

Kaushik M, Chahal KK. The death spiral of open source projects: A post-mortem analysis of pull request workflow dynamics. Journal of Systems and Software. 2026;240:112942. doi: 10.1016/j.jss.2026.112942
Ashok A, Khan NI, Singhvi S, Stanciulescu S, Wang Z, Filkov V. Multi-ecosystem modeling of OSS project sustainability. Journal of Systems and Software. 2026;241:112985. doi: 10.1016/j.jss.2026.112985
Alami A, Pardo R, Linåker J. Free open source communities sustainability: Does it make a difference in software quality?. Empir Software Eng. 2024;29(5):114. doi: 10.1007/s10664-024-10529-6
Ouf M, Noei S, Van Iterson Z, Guizani M, Zou Y. An empirical Analysis of Community and Coding Patterns in OSS4SG vs. Conventional OSS. Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering. 2026:2389-2401. doi: 10.1145/3744916.3787782
Sun J, Patil A, Li Y, Guo JLC, Zhou S. Collaboration Challenges and Opportunities in Developing Scientific Open-Source Software Ecosystem: A Case Study on Astropy. Proc ACM Hum-Comput Interact. 2025;9(7):1-33. doi: 10.1145/3757462

https://doi.org/10.62486/978-9915-9966-3-9.Ch06