



Research



Navigating Open Source Software Development in the Generative AI Era

Challenges and Opportunities for Project Communities



October 2026

Sam Boysel, *The Linux Foundation*

Anni Lai, *Generative AI Commons*

Foreword by Brian Proffitt, *Red Hat*

Navigating Open Source Software Development in the Generative AI Era

Prepare for genAI contribution at scale

59% of maintainers
(69% in large projects)
see rising contribution
volume from AI.



Filter low-quality PRs

1 in 4 incoming contributions
are AI slop. This rate increases
to 31% for larger projects.

Adopt AI for contribution triage

Half of all maintainers
leverage AI for bug checks
and code review. But 1 in 4
do not use it at all.



Streamline review cycles

Low quality contributions
and increased volume
compound to strain
maintainers' review time.

Build contribution scalable verification systems

Policy and tooling can shift
review burdens back to
contributors, increase
PR quality, and reduce
maintainer burnout risk.



Establish clear AI governance

Only 1 in 3 projects have
a formal policy that allows
AI contribution.



Invest in new contributor mentorship

The majority of maintainers
state AI demands more
contributor guidance, not less.



Standardize AI contribution attribution

Currently, only 3 in 10
always disclose AI assistance
in contributions and
2 in 10 never do.



Require pre-submission testing

3 of 5 OSS developers
already thoroughly test
AI-assisted contributions
before submitting.



Address AI governance risks first

While current barriers are
technical, improving model
provenance, training ethics,
and data privacy are more
important to AI holdouts.



Capture AI efficiency in OSS development

Harness up to 50% time
savings across coding,
reviewing, and issue
reporting. Maintainers
benefit from efficient
project discourse.



Reinvest AI gains back into OSS

Generative AI lets
contributors spend more
time addressing technical
debt, developing new
features, and upskilling.

Contents

Foreword	4	Productivity impacts of genAI	23
Executive summary	6	Generative AI makes OSS development faster across the board.....	23
Introduction	7	Time savings shrink as task complexity rises.....	25
The impact of genAI on project and maintainers	8	Saved time flows into technical debt and new features	26
Increased review times and risk of maintainer burnout greatest in large projects	8	Conclusion.....	27
AI slop now approaches one in four PRs.....	9	Methodology	29
Contribution volume keeps growing fastest at scale	11	Survey demographics.....	30
Maintainers use AI to help with AI	11	OSS development.....	30
Most projects still lack an AI contribution policy.....	12	Generative AI experience	34
Verification still relies on trust rather than tooling.....	13	Maintainer demographics	35
AI lowers barriers to entry but can erode community trust.....	14	About the authors.....	36
How genAI is used in OSS development	16	Acknowledgments.....	36
Generative AI reduces contribution barriers but project guardrails provide important moderation	16		
AI tools have become standard but model provenance remains top of mind	18		
Most contributors review AI code closely yet attribute it inconsistently.....	20		
Non-adopters see trust problems as fixable, technical problems as inherent	22		



Foreword

Over the last three decades, I have watched open source projects and communities absorb one disruption after another: the rise of the web, the arrival of the mobile phone and the commensurate rise of big data and cloud-native computing. In all of these technological waves, communities have managed to adapt in ways that have made them stronger and more efficient. Generative AI is the newest of these waves, and it may very well be the most consequential yet. What makes it different is where it lands. Open source has always depended as much on trust, collaboration, and shared labor as it does on the code. So when a technology like AI begins reshaping how contributions are written, submitted, and vetted, it lands right in the heart of how open source communities function.

For Red Hat, our conviction is simple: the future of AI should be open. It should be built in the open, governed transparently, and shaped by communities, rather than dictated by a single vendor. An AI ecosystem that is vendor-neutral and community-driven is the only durable path to models and tooling that people can inspect, trust, and improve.

Much of the discussion to date has been anecdotal: endless tales of “AI slop,” overwhelmed maintainers, and eroding trust. What has been missing from the conversation about AI and open source is rigorous data. This Survey on Generative AI and Open Source Software Development survey of 234 contributors and maintainers evolves those impressions into measurement, and it is exactly the kind of empirical work open source communities need to navigate what’s coming next.

Two of this survey’s findings deserve special attention because they don’t fall into easy narratives. The first is that generative AI does not affect all projects equally. Large and small projects are affected in fundamentally different ways. For larger projects, maintainer burnout displaces code quality as the primary concern, becoming the top issue for projects with more than 50 contributors. The largest projects report the longest review times, the highest share of low-quality “AI slop” contributions, and the fastest-growing contribution volume. Scale, in other words, amplifies the burden rather than disbursing it across more hands. The most visible, critical projects in our ecosystem are absorbing the brunt of this generative AI wave, and they are among the ones we cannot afford to see burn out.



The second interesting finding is potentially another pin to pop the hype balloon: the productivity dividend of generative AI drops as tasks grow more complex. The survey shows dramatic time savings on well-defined, task-oriented work like code authoring. But those gains decrease steadily as the work gets into coordination, judgement and open-ended discussion. On the hardest tasks, a significant share of developers report needing more time than before when using AI, not less.

This really shouldn't come as a surprise for anyone who understands how AI tools work. AI is a great accelerator of the routine, but the critical-thought, difficult work at the core of any project still relies on human expertise, and the data shows it. The data shows that when time is saved, it's being reinvested into projects by paying down technical debt, building new features, and onboarding new contributors.

Together, these and the other findings of this survey point to a conclusion that makes a lot of sense: that the communities managing this incoming generative AI wave are neither outright banning or wholeheartedly adopting AI. Community members are doing the harder, more valuable work of building the policy, verification, and mentorship infrastructure that lets the benefits of AI be realized without losing the most important part of an open source community: trust and quality.

My thanks go to Sam Boyssel and Anni Lai for their careful work, to Hilary Carter and her Linux Foundation Research team for this survey's rigor, and to every participant who shared their experience. The picture that emerges from this survey is neither the utopia or the apocalypse we hear bandied about in the ever-polarized media, but, rather, something more useful: a measured look at where the open source community and generative AI actually stands. I look forward to the discussions that this report will foster.

Brian Proffitt

Senior Principal Community Architect – Strategic Insights,
Red Hat Open Source and AI Program Office

Executive summary

Generative AI (genAI) is already having a profound impact on open source software (OSS) development, sparking an explosion of AI-assisted contributions that raises urgent concerns about code volume, quality, and the increasing burden on time-constrained maintainers. Despite these problematic trends, there are potential upsides including increased productivity and bringing new contributors into the ecosystem. To provide robust empirical evidence to the evolving discussion about genAI and OSS, we developed, fielded, and analyzed a survey of 234 OSS contributors and maintainers. This research explores exactly how genAI is being deployed in OSS workflows, how it alters contribution dynamics, and how project leaders are navigating the complex intersection of new opportunities and emerging challenges.

Our findings indicate that the impact of generative AI on projects and workflows most often depends on project scale. Larger projects with coordinated maintainer teams currently bear the heaviest total administrative burden. These teams face surges in contribution volume, with large projects reporting that up to 31% of incoming contributions are low quality “AI slop.” Consequently, 64% of maintainers on these projects report extended review periods and cite burnout as their primary concern. While project leaders generally agree that AI effectively lowers the barrier to contribution, this accessibility comes at the cost of decreased community trust, higher mentorship demands for new contributors, and a highly fragmented landscape for AI contribution policies and verification methods.

Despite these friction points, generative AI remains a powerful productivity accelerator across the open source ecosystem. Contributors and maintainers report that these tools reduce the time required for OSS development, with average time savings ranging from 19% to 52% depending on the task.

Maintainers specifically benefit from highly efficient technical discussions and proposal reviews, saving between 25% and 40% of their time — gains that 66% of respondents reinvest directly into addressing technical debt. Furthermore, genAI emboldens developers to explore unfamiliar technologies and contribute to new projects. Crucially, we find strong evidence of conscientious adoption: 61% of contributors conduct detailed, line-by-line reviews and full functional testing of AI-generated code before submission, and 80% emphasize the importance of understanding the underlying source model and training data lineage.

To resolve the core tensions surrounding the use of genAI in OSS development, projects must move from reactive management to intentional contribution policy and workflow design. Successful communities are establishing clear, transparent guidelines that, instead of forbidding genAI or naively accepting all genAI results, pass the review burden back to contributors by mandating responsible genAI usage, standardizing genAI attribution, and explicitly defining expectations for submitted code. Simultaneously, maintainers should consider using AI themselves to combat growing workloads, fighting fire with fire to reclaim valuable bandwidth. Crucially, project leaders must recognize that while AI lowers entry barriers and accelerates technology adoption, it cannot replace human knowledge and mentorship; sustaining long-term community health requires continued investment in hands-on guidance and contribution norms that welcome newcomers without sacrificing quality or eroding trust.



Introduction

The transformative impact that genAI has on software development is hard to overstate. By reducing the human capital and time to generate functional code, these new tools can significantly influence workflows for novice and veteran developers. But these forces now come against longstanding norms and practices in open source software (OSS) development, where the social aspect of collaboration often matters at least as much as technical skills. Reports of AI-assisted contributions flooding OSS projects have sparked concerns about code quality, maintainers' workload, and the erosion of trust within contributor communities. Project communities are searching for ways to continue creating high-quality OSS, and most acknowledge that AI-assisted development is here to stay.

We start by illustrating the impact of genAI using a broad sample of OSS projects and maintainer perspectives, rather than relying solely on anecdotal evidence. It's important to first establish the extent to which genAI changes contribution volume and quality, how this varies across different project scales, and what this means for maintainers' workload. The study also explores how maintainers set AI contribution policies, verify AI-assisted contributions, and use AI to assist with contribution review.

In the next phase of the study, we investigate how OSS developers are using genAI tools and how these tools have influenced contribution behavior. The survey asks respondents what specific OSS development tasks they use genAI for, what kind of projects they contribute to, and how they verify and attribute AI-assisted contributions. We also explore why some developers choose not to use genAI, and what factors might persuade them to adopt it in the future.

Finally, we examine the productivity impacts of genAI on OSS development. The survey asks respondents to estimate how much time they save on various tasks when using genAI, how these effects vary across tasks and contributor roles, and how they reinvest that saved time. This section provides insights into how genAI is reshaping the OSS development landscape, and what this means for maintainers, contributors, and the broader OSS community.

The impact of genAI on project and maintainers

Maintainers sit at the receiving end of the genAI contribution wave, and their experience of it is shaped almost entirely by project scale. Many large projects are under tremendous strain due to a massive influx of low-quality AI-generated contributions. Both elevated volume and increased verification burdens compound to erode maintainers' precious time and increase the risk of burnout. In this section, we present survey evidence confirming the toll genAI has on maintainer workload and contribution quality. We link these effects to rising contribution volume, project AI contribution policy, and the verification and review methods maintainers use for AI-assisted contributions.

Increased review times and risk of maintainer burnout greatest in large projects

To get a sense of the overall impact of genAI on OSS maintainers, we begin by identifying the top concerns AI-assisted contributions raise for maintainers. We take care to segment the results by project size, since scale is a major factor in how maintainers experience the AI contribution wave.

Maintainers cite degraded contribution quality as the overarching problem with AI-assisted contributions, but as projects grow in size¹, that concern is compounded by an even more pressing one: **maintainer burnout becomes the dominant issue in large projects** (those with more than 50 contributors). This suggests that rising contribution volume is translating directly into reviewer fatigue rather than being absorbed painlessly (Table 1).

The clearest sign of strain is time: **AI-assisted contributions are measurably lengthening the review process for the maintainers who have to evaluate them**. Lead and co-maintainers bear the heaviest burden, reporting the highest rates of increased review time (Figure 1). Solo maintainers stand out as the only role where reduced review time approaches or exceeds the share reporting an increase, likely a function of the smaller, more familiar contributor base they typically manage. Scale compounds the effect: large projects see a markedly higher increased-review share (64%) than small (46%) or medium (55%) projects, confirming that the largest, most visible projects are absorbing the brunt of the added workload.

TABLE 1: MAINTAINER BURNOUT BECOMES THE TOP CONCERN FOR LARGER PROJECTS

CONCERN RANK	SMALL PROJECTS	MEDIUM PROJECTS	LARGE PROJECTS
1st	Contribution quality [51%]	Contribution quality [50%]	Maintainer burnout [63%]
2nd	Security concerns [40%]	Increased technical debt [35%]	Contribution quality [60%]
3rd	Contribution volume [35%]	Maintainer burnout [35%]	Contribution volume [43%]

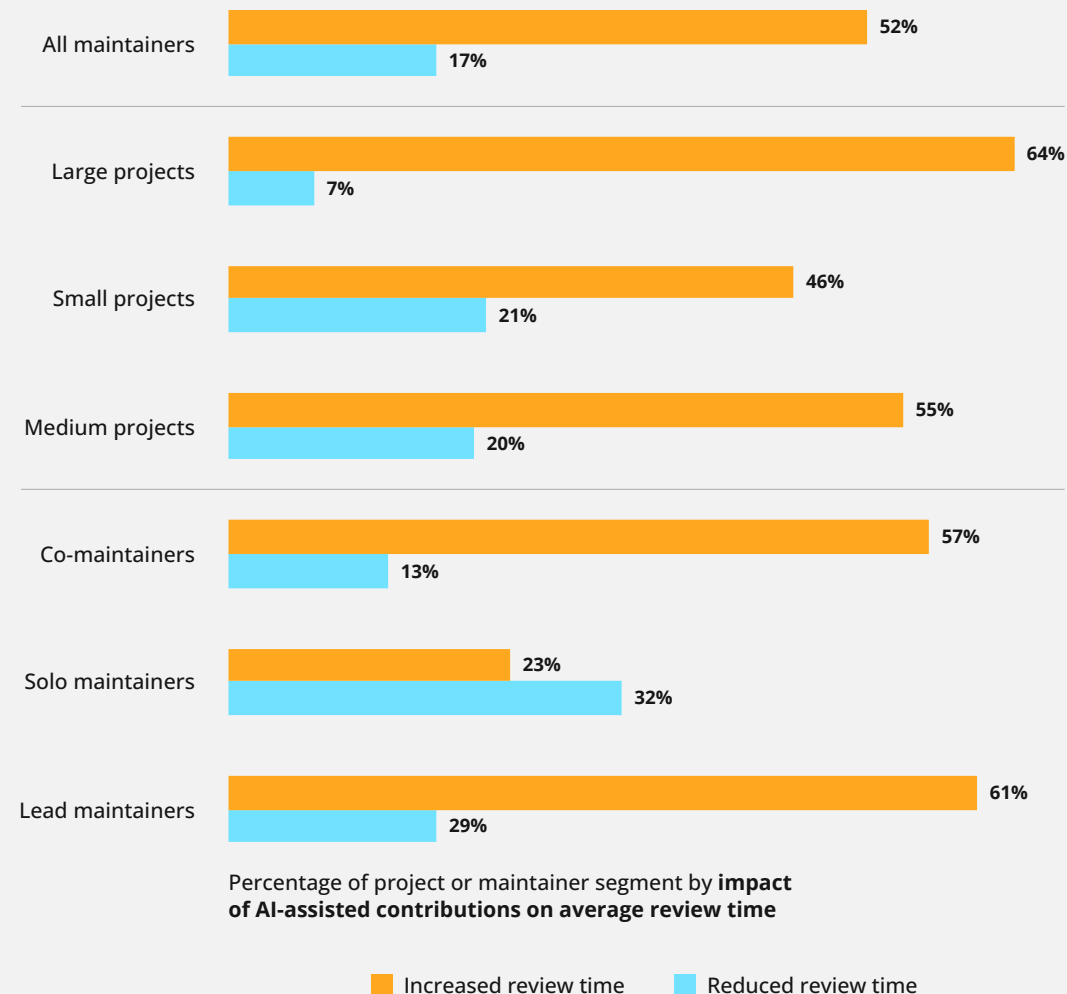
Source: Survey on Generative AI and Open Source Software Development. Q36 x Q29. Number of respondents = 133

1 Throughout this report, we use the following definitions for project size: Small projects = fewer than 10 contributors. Medium projects = 11 to 50 contributors. Large projects = more than 50 contributors.

FIGURE 1

AI-assisted contribution far more likely to increase review times for large projects and lead maintainers

Source: Survey on Generative AI and Open Source Software Development. Q39 x Q28 x Q29. Number of respondents = 174.



AI slop now approaches one in four PRs

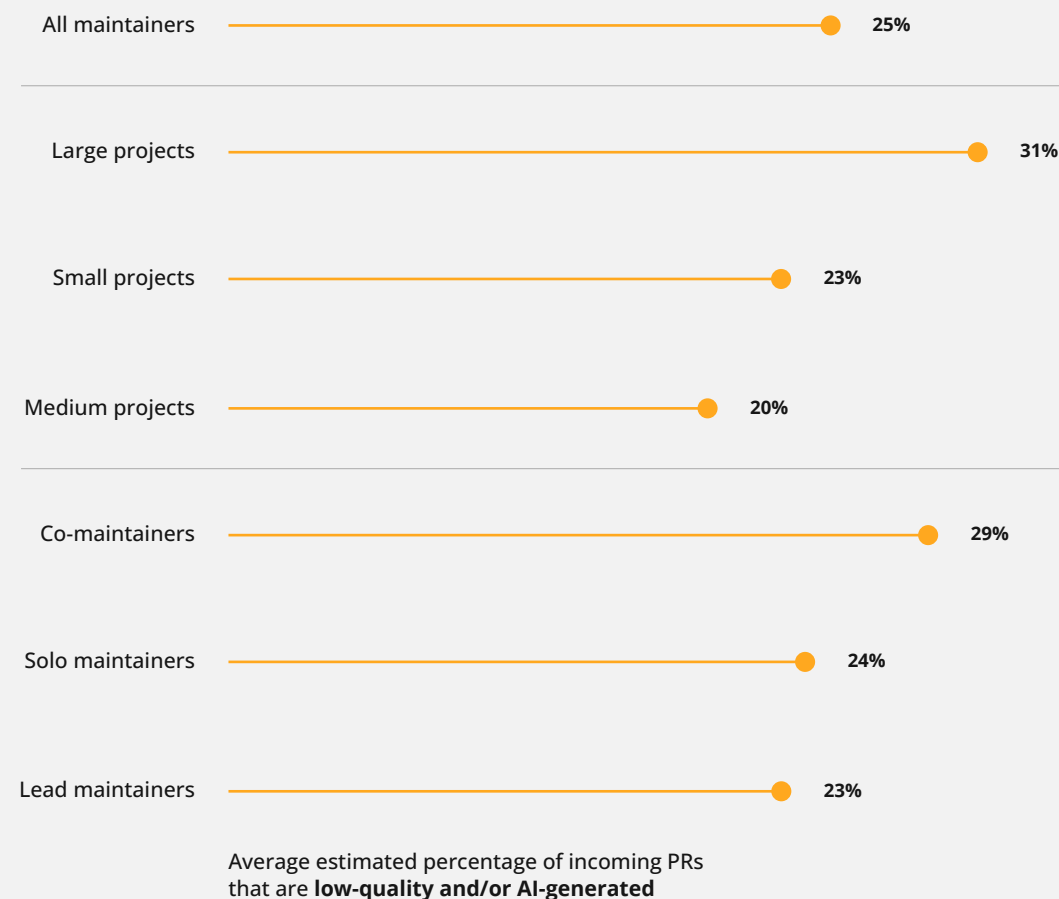
Behind that rising review burden lies a quality problem. “AI slop” are low quality AI-generated contributions that are unlikely to be worth the review time they demand. Large projects are focal points for increased contribution energy, and this is where maintainer concerns are most elevated.

On average, **25% of incoming PRs are estimated to be AI-generated low-quality contributions** (Figure 2). This burden again scales with size: large projects report a substantially higher mean slop share than small or medium projects (31% vs. 23% and 20%), confirming that scale amplifies exposure to low-effort AI submissions rather than diluting it across a larger reviewer pool. Role also matters independent of project size. Co-maintainers report the highest mean slop share among roles (29%), nearly 6 percentage points above lead and solo maintainers, possibly reflecting broader contributor networks with less direct oversight than a single lead maintainer can typically provide.

FIGURE 2

1 in 4 incoming PRs is AI slop. Large projects with co-maintainer teams receive the highest percentage

Source: Survey on Generative AI and Open Source Software Development. Q38 x Q28 x Q29. Number of respondents = 111. The top panel shows the sample average for all respondents. The middle panel segments maintainers by the size of their project. The bottom panel segments respondents by the type of maintainer.



Not all OSS projects receive slop or continue to receive slop. Curl project lead Daniel Stenberg reported in 2025 that the curl project was being overwhelmed by AI slop in vulnerability reports², yet by April 2026 he reported that the curl project had “stopped getting AI slop security reports” and were instead getting “an ever-increasing amount of really good security reports, almost all done with the help of AI”³. This suggests that as AI models get better, and as projects push back against AI slop, slop can be reduced over time. However, projects must make clear that AI slop is not acceptable, and that AI-generated work (such as vulnerability reports) must be carefully validated before being submitted to projects.

- 2 Daniel Stenberg, “Death by a Thousand Slops,” Daniel Stenberg’s Blog, July 14, 2025, <https://daniel.haxx.se/blog/2025/07/14/death-by-a-thousand-slops/>.
- 3 Daniel Stenberg, LinkedIn post, April 2, 2026. https://www.linkedin.com/posts/danielstenberg_hackerone-share-7446667043380076545-RX9b/

Contribution volume keeps growing fastest at scale

Underlying both the workload burden and the quality concerns just described is a basic shift in scale: the sheer amount of incoming contributions. The increased volume of low-quality contributions compounds the problem for already overworked maintainers.

Most projects have witnessed a growth in contribution volume since the rise of genAI: 59% of maintainers report an increase and only 4% a decrease (Figure 3). This is evidence against outright contribution collapse, but also the raw fuel behind the workload and quality pressures described above, since lowering the cost of participation draws in more submissions without necessarily improving them. That growth is not evenly distributed, however. It scales with project size: 69% of large (51+ contributor) projects report an increase, compared with 57% of small and 52% of medium projects. Aligning with elevated burnout risk in large projects, **higher-visibility projects are absorbing a disproportionate share of the new AI-assisted contribution traffic**. Small projects (less than 10 contributors) tell a messier story, mirroring solo maintainers with the widest variance of any segment — 40% report no change at all alongside 21% seeing growth of more than 50%, a split that likely reflects how much a single active AI-assisted contributor can skew the numbers at that scale.

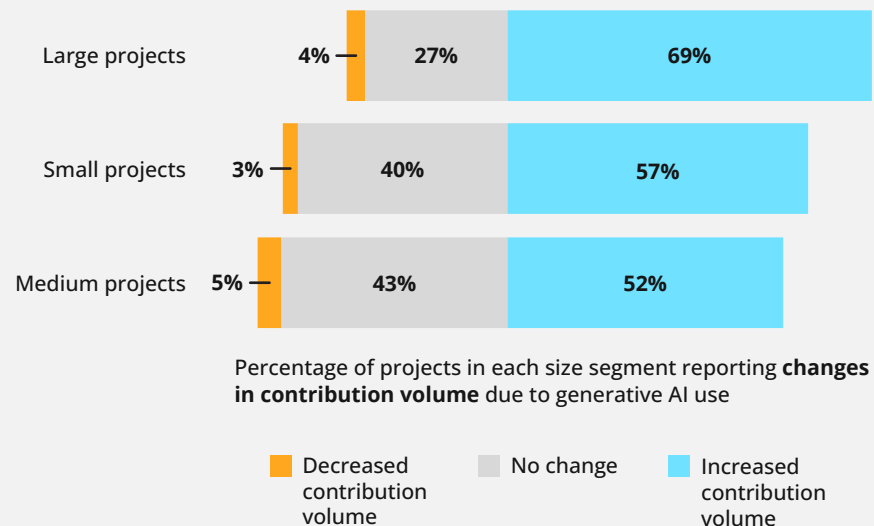
Maintainers use AI to help with AI

Maintainers need not face these rising challenges without assets of their own. Some are fighting fire with fire by using AI themselves to assist with maintenance workloads. While certainly not a replacement for maintainer expertise, skilled operators can leverage AI tools to more efficiently review the mounting backlog of contribution proposals as they grow in both volume and complexity.

FIGURE 3

AI-assisted contribution volume increases most for larger projects

Source: Survey on Generative AI and Open Source Software Development. Q37, segmented by Q29 (project size, collapsed to Small/Medium/Large). Number of respondents = 111.



Most maintainers use AI to understand incoming contributions [58%] or check for logic bugs [48%], turning the same technology that is driving up contribution volume into a partial counterweight against the added review burden. Still, 23% use no AI tools for contribution vetting at all (Figure 4), so a meaningful minority are choosing to absorb the growing review load with entirely manual methods.

FIGURE 4

3 out of 4 maintainers are actively using AI tools to manage AI-assisted contributions

Source: Survey on Generative AI and Open Source Software Development. Q33. Number of respondents = 92. Total responses = 193.

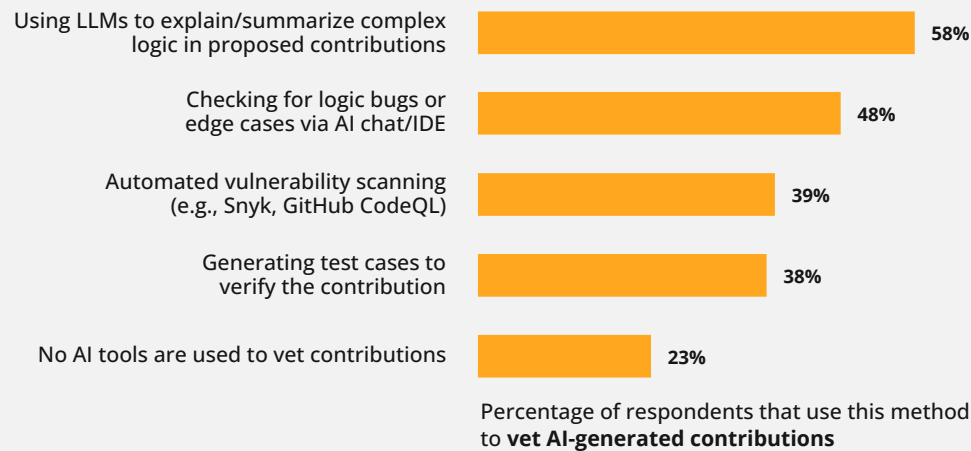
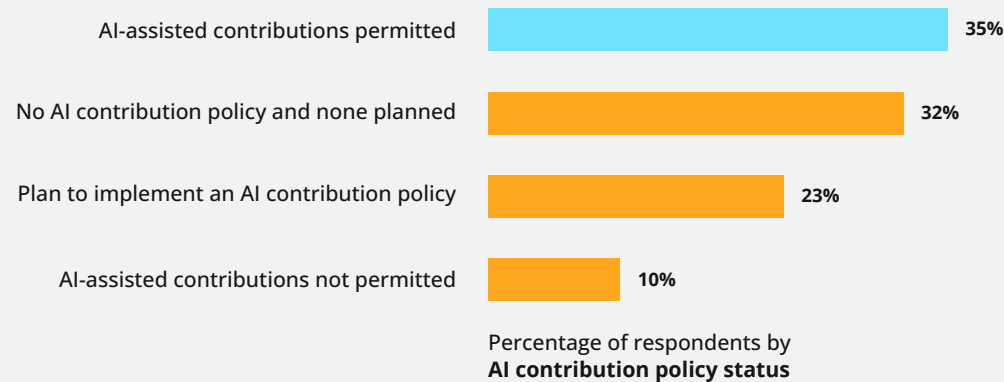


FIGURE 5

Only 1 in 3 OSS projects have a formal policy allowing AI contribution

Source: Survey on Generative AI and Open Source Software Development. Q30. Number of respondents = 119.



Most projects still lack an AI contribution policy

Given rising volume, reduced quality, and other concerns about AI-assisted contribution, a natural question is whether projects have formalized any response. Contribution policies can take many forms and levels of formalization, from outright bans to detailed guidelines for responsible use. Moreover, projects simply vary in terms of their policy response to AI-assisted contributions, with no single approach dominating the landscape.

Most projects have yet to standardize policy for genAI contribution. **Only 35% of projects have a formal policy permitting genAI use, 32% have no policy and none planned, and 10% explicitly prohibit it** (Figure 5). This implies that a majority of projects navigate AI-assisted contributions on an ad hoc, case-by-case basis. But momentum is shifting towards formalization: 23% of projects plan to implement an AI contribution policy sometime in the future. Formal AI usage policies set clear expectations for contributors and reduce some review overhead for maintainers.

Where policies do exist, they tend to focus on making AI use visible rather than restricting it outright. Existing policies most commonly address transparency and disclosure requirements [76%] and contributor review responsibility [76%], while licensing and copyright is covered in only 39% of cases (Figure 6), a gap that leaves many projects confused about issues least amenable to informal, trust-based resolution.

GenAI policies can help resolve legal uncertainty: In many jurisdictions (including the EU and US), works generated entirely by AI without human creative input cannot be copyrighted. If it's a copy of something already copyrighted, it's still copyrighted. Court cases confirming this include *Bartz v. Anthropic PBC* and *Kadrey v. Meta Platforms, Inc.* The Linux Foundation has “Guidance Regarding Use of Generative AI Tools for Open Source Software Development”⁴ that discusses these issues. That said, such copying is relatively unlikely to occur in typical software development and vulnerability identification tasks when using AI. In addition, a work where some parts are under a license, and others are not copyrightable, is as a whole under that license. Policies that make this clear can help developers because most developers are not also lawyers.

A minority of projects have an explicit ban. In some cases, this ban takes the extreme form of rejecting all AI-generated reports including security vulnerability reports. Such policies put users at extreme risk. As Michael Catanzaro notes, “banning good vulnerability reports solely because some portion of the report was generated by AI is unacceptable. AI-assisted vulnerability reports are the new industry standard... Prohibiting issue reports reduces the quality and safety of your software, punishing your users”⁵

Verification still relies on trust rather than tooling

In the absence of comprehensive contribution policies, maintainers face new challenges for reviewing and evaluating the integrity of AI-assisted contributions. Maintainers are naturally concerned with issues in AI-generated code entering their codebases beyond simple technical quality, such as licensing, copyright, or data origin. Verification and vetting of AI-assisted contributions becomes a critical step in maintaining project quality and trust even where no written policy exists.

⁴ <https://www.linuxfoundation.org/legal/generative-ai>

⁵ Michael Catanzaro, “Please Do Not Ban AI-Assisted Issue Reports,” Michael Catanzaro’s Blog, June 8, 2026, <https://blogs.gnome.org/mcatanzaro/2026/06/08/please-do-not-ban-ai-assisted-issue-reports/>.

FIGURE 6

Most AI contribution policies shift the burden of review back to the contributor and require disclosure of AI

Source: Survey on Generative AI and Open Source Software Development. Q31. Number of respondents = 54. Total responses = 122.



Project maintainers employ a mix of verification strategies, and the pooled results suggest most of this work still rests on **informal judgment rather than formal process**: verification relies primarily on trusting the contributor [47%] and manual vetting via inspection [44%], while 26% have no current verification mechanism in place at all (Figure 7).

As with policy adoption, **verification formality tends to follow project scale**. Attestation requirements (explicit statements or provenance checks required of contributors) grow sharply with project size, from 20% for small projects to 41% for large ones, suggesting that formal provenance checks only become standard once contributor volume makes trust-based review impractical. Small projects have the highest “no verification” rate (33%), and at every scale, trust-based and manual vetting remain the dominant mechanisms. So even as projects grow, most maintainers are still leaning on judgment and relationships rather than tooling to catch problematic AI-assisted contributions.

AI lowers barriers to entry but can erode community trust

The study also examines the maintainer perspective on broader project and community health implications of using genAI for OSS development. We find that in addition to the immediate workload impacts within their own projects, maintainers are also concerned about the long-term consequences for community trust, mentorship practices for new contributors, and community cohesion.

FIGURE 7

Most projects rely on contributor trust and manual vetting for AI-assisted contribution as opposed to automated processes

Source: Survey on Generative AI and Open Source Software Development. Q32 x Q29. Number of respondents = 131.

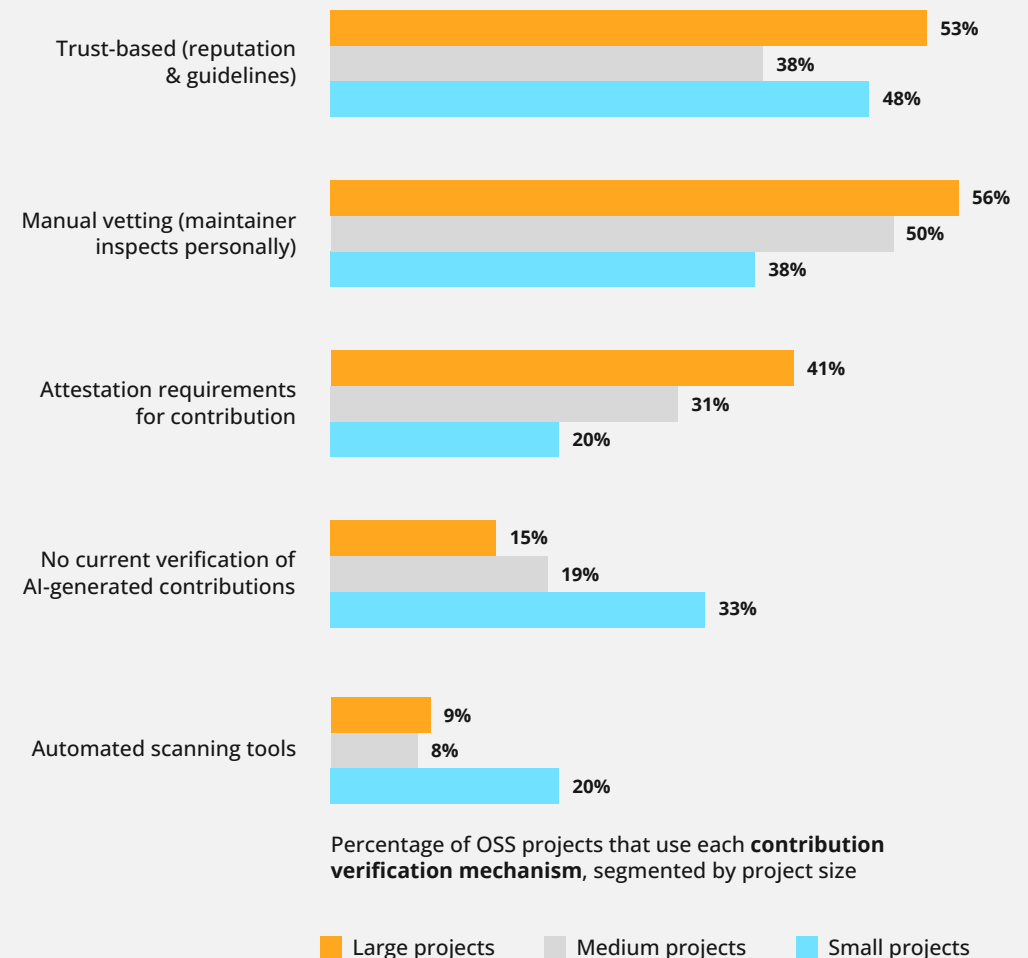
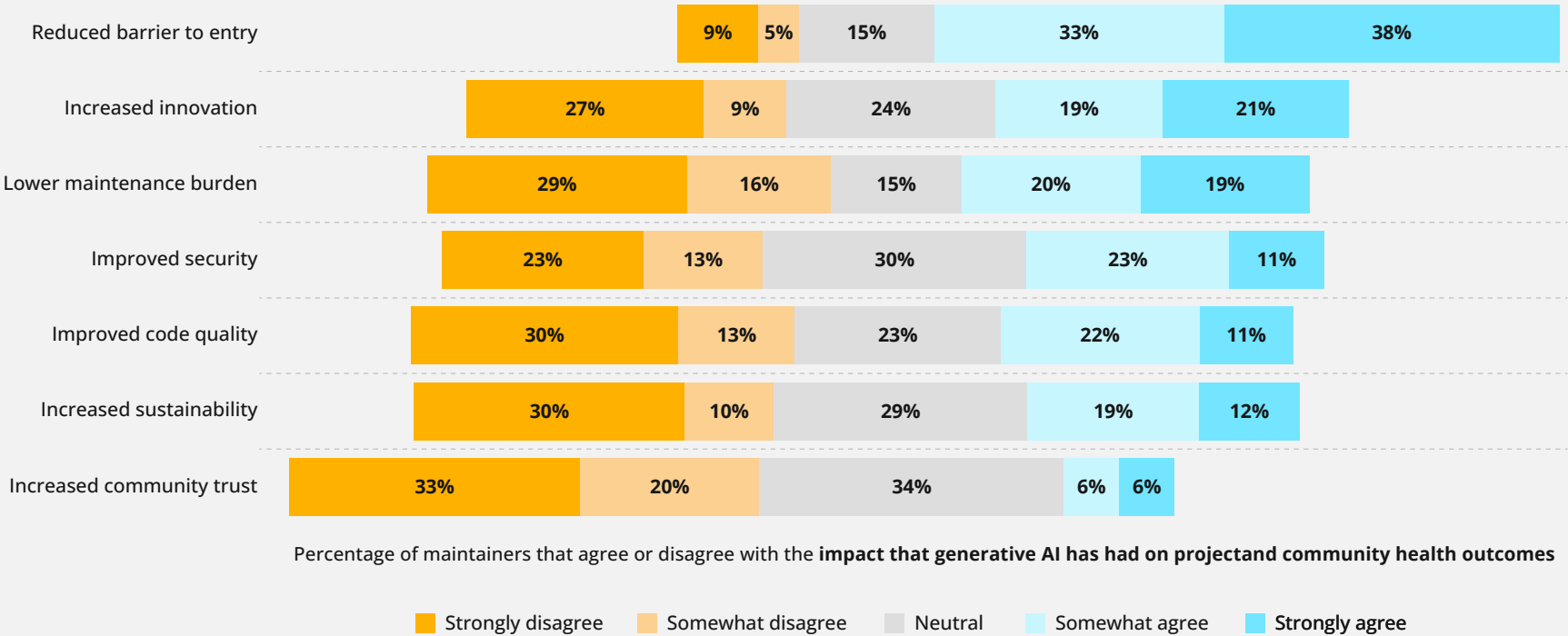


FIGURE 8

Maintainers typically agree that genAI reduces contribution barriers but at the expense of sustainability and community trust

Source: Survey on Generative AI and Open Source Software Development. Q35. Number of respondents = 118.



71% of maintainers agree that genAI reduces the barrier to entry for OSS (Figure 8). This strong consensus surfaces a key benefit of genAI for OSS contribution is an expansion of the potential contributor pool, bringing in new ideas and promoting inclusivity within projects. On the other hand, **community trust is the clearest casualty**: 53% disagree that it has improved, against just 13% who agree. The same accessibility gains that lower barriers to entry come at the direct expense of the trust that binds contributor communities together.

That tension extends to how new contributors are onboarded. Maintainers report that **the use of genAI in OSS development is more likely to increase (46%) than decrease (32%) the need for mentorship for new contributors** (Figure 9). Reading alongside the barrier-to-entry finding above, this suggests AI is drawing in more newcomers even as it makes those newcomers more, not less, dependent on human guidance to contribute successfully. This is an added mentorship load that falls on the same maintainers already absorbing higher contribution volume and longer review times.

How genAI is used in OSS development

Having examined how genAI is reshaping projects from the maintainer's chair, we now turn to the contributor's side of the equation: what developers actually do with these tools, which ones they reach for, and how carefully they vet the output. We also consider the minority of respondents who do not use genAI development, exploring the factors that drive this choice and what changes they would need to see to use it.

Generative AI reduces contribution barriers but project guardrails provide important moderation

What are the most common use cases for genAI in OSS development? The study investigates specifically how contributors use these AI-powered tools for common OSS tasks and the results show a broad range of applications across the development lifecycle.

The most common use cases for genAI in OSS development are developing new features [83%], debugging/testing [78%], refactoring [71%], and writing documentation [69%] (Figure 10). This implies that **OSS contributors view genAI tools as general-purpose complements across nearly every stage of the software development lifecycle (SDLC).**^{6,7} Furthermore, encouraging developers to use genAI for quality assurance tasks such as thorough testing and writing documentation may help encourage responsible use and reduce the risk of low-quality contributions being submitted to OSS projects.

6 Addy Osmani, Shubham Saboo, and Sokratis Kartakis, "The New SDLC With Vibe Coding," white paper, Kaggle, May 2026, <https://www.kaggle.com/whitepaper-the-new-SDLC-with-vibe-coding>.

7 Louis Claxton, "The AI-Native SDLC Playbook," Claude Blog, August 21, 2026, <https://claude.com/blog/the-ai-native-sdlc-playbook>.

FIGURE 9

Maintainers are more likely to find that genAI increases the need for mentorship of new contributors rather than decrease it

Source: Survey on Generative AI and Open Source Software Development. Q34. Number of respondents = 114.

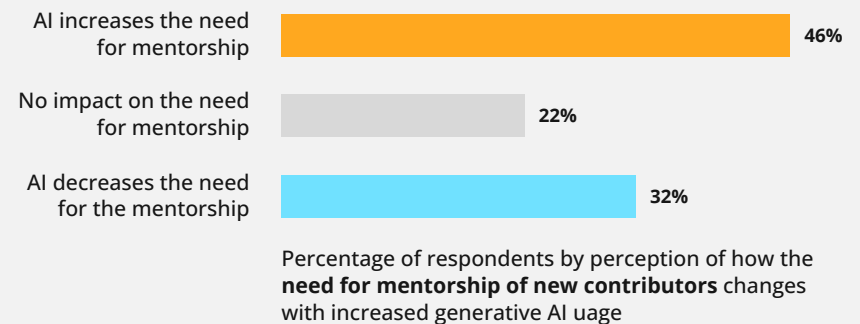
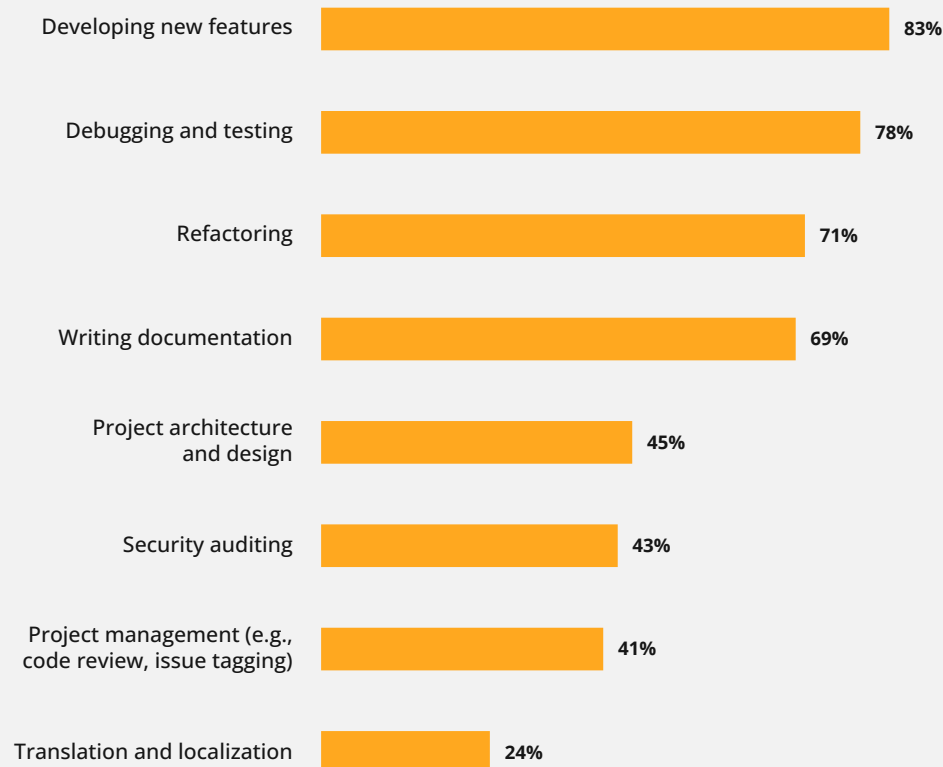


FIGURE 10

Generative AI is being used in virtually every layer of OSS development

Source: Survey on Generative AI and Open Source Software Development.
Q13. Number of respondents = 139. Total responses = 644.



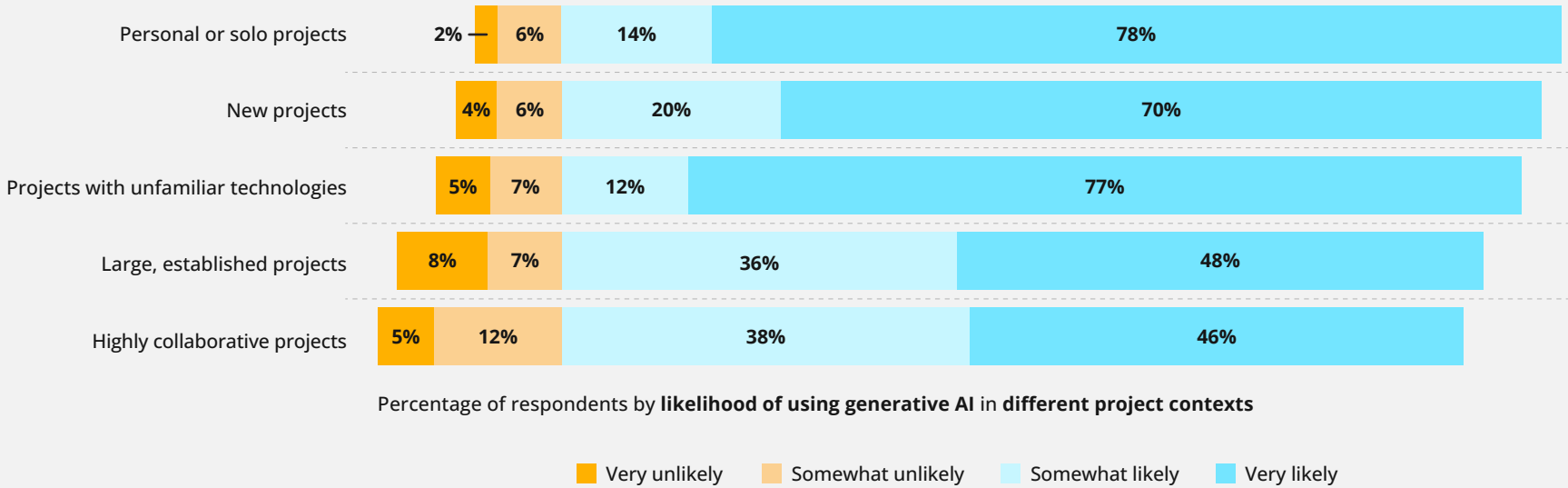
“OSS contributors view genAI tools as general-purpose complements across nearly every stage of the software development lifecycle (SDLC).”

Willingness to use genAI for OSS development depends heavily on project context. Large, established projects feature broad contributor bases, ingrained community norms, and structural guardrails to filter out low quality contributions. These collaboration frictions can frustrate new contributors but are important for balancing contribution velocity with quality. On the other hand, contributors to new or solo projects face fewer institutional barriers and prioritize rapid prototyping over strict governance. Consequently, the likelihood of using generative AI is highest for personal or solo projects (78% “very likely”) and tasks involving unfamiliar technologies (77% “very likely”), compared to a notably lower adoption likelihood for large, established projects (48% “very likely”) (Figure 11). **OSS developers are most likely to use genAI in contexts where collaborative frictions are lower.** It’s also worth noting that a particularly useful application of genAI in OSS development are unfamiliar technologies. Newcomers can more easily make effective contributions using new languages or frameworks.

FIGURE 11

Contributors more likely to use genAI in contexts with low collaborative frictions

Source: Survey on Generative AI and Open Source Software Development. Q14. Number of respondents = 128.



That comfort with unfamiliar territory extends to a concrete behavioral shift: **genAI substantially increases willingness to contribute to unfamiliar OSS projects**, with 71% reporting increased likelihood (43% significant, 28% slight) and only 3% reporting a decrease (Figure 12). OSS developers who use genAI are encouraged to branch out beyond their comfort zone. In effect, genAI is functioning as an onboarding accelerant, giving contributors enough context and confidence to cross into codebases or technologies they would otherwise have avoided.

AI tools have become standard but model provenance remains top of mind

Generative AI tools come in different forms, from autocomplete functionality to chatbots to agentic frameworks and beyond. The study asked respondents which tools they use for OSS development, how frequently they use them, and the level of scrutiny they apply to the underlying models that power those tools.

FIGURE 12

Generative AI makes developers far more likely to contribute to unfamiliar projects

Source: Survey on Generative AI and Open Source Software Development. Q17. Number of respondents = 136.

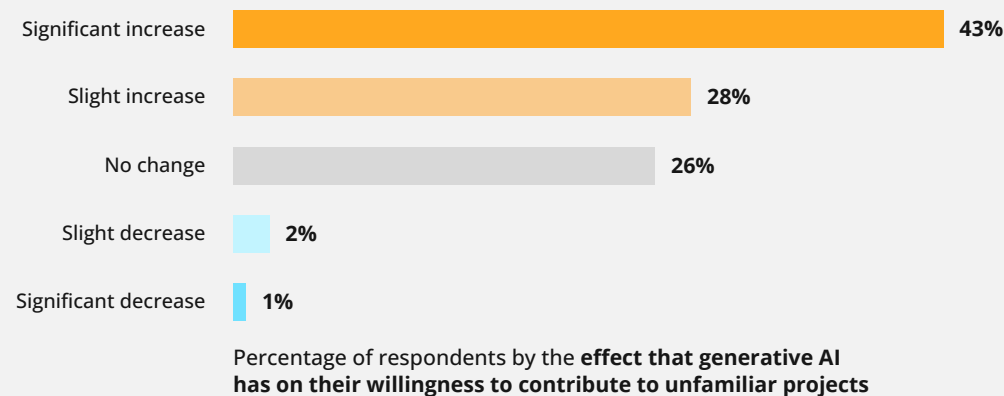
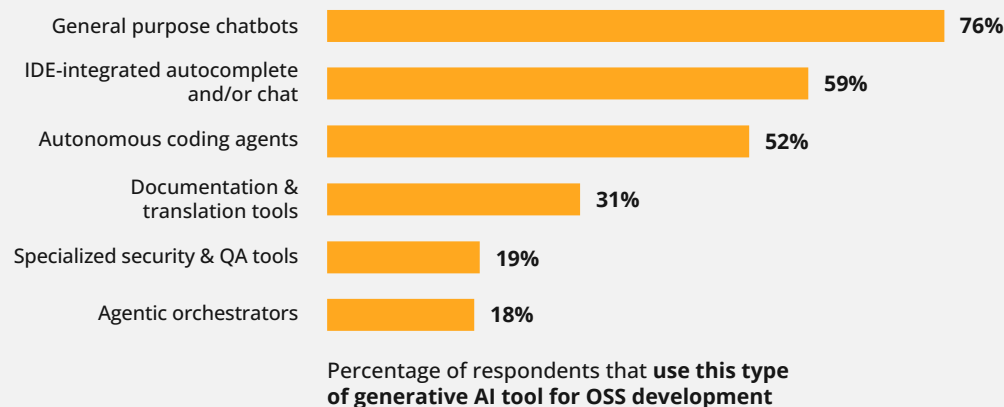


FIGURE 13

Chatbots lead autocomplete and autonomous agents for OSS development

Source: Survey on Generative AI and Open Source Software Development. Q15. Number of respondents = 139. Total responses = 367.



General-purpose chatbots [76%] are the dominant genAI tool used by OSS developers, followed by IDE-integrated tools [59%] and autonomous coding agents [52%] (Figure 13). This hierarchy suggests that **most contributors are still primarily using genAI as a conversational aid rather than as an embedded, autonomous part of their workflow**, even though more integrated options are already in widespread use.

Usage frequency reinforces that picture. When OSS developers use genAI, 76% use interactive chat daily or nearly every day (Figure 14). genAI chatbots are general purpose tools and complement software developers' own expertise on a wide range of different tasks. Developers still use heavier tools like code completion (65%) and agentic frameworks (53%) quite often, but somewhat less frequently on average, demonstrating that the **genAI tool adoption curve is inversely related to the degree of control the developer gives up**.

That wariness is not limited to agentic tools; it extends to how contributors think about the models themselves. **80% of OSS contributors consider knowing the source model and training data lineage of genAI coding tools to be important** (44% extremely, 37% moderately) (Figure 15). Provenance and trust concerns are top of mind even for contributors who use genAI tools daily.

FIGURE 14

Contributors use genAI on a daily basis

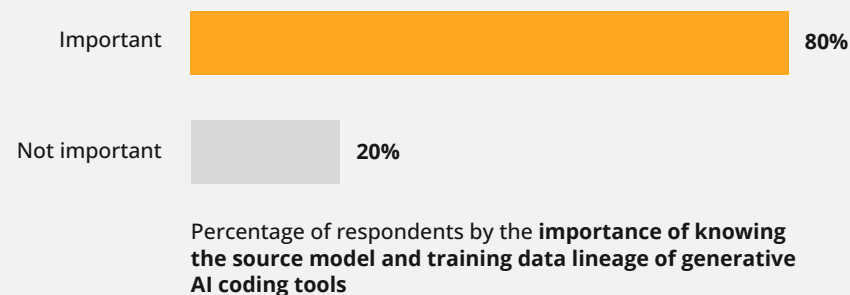
Source: Survey on Generative AI and Open Source Software Development. Q12. Number of respondents = 139.



FIGURE 15

4 in 5 OSS contributors consider knowing AI model provenance important

Source: Survey on Generative AI and Open Source Software Development. Q18. Number of respondents = 126.



Most contributors review AI code closely yet attribute it inconsistently

A primary concern with AI slop is that the contributor makes an insufficient effort to independently verify that their patch is compatible with an upstream codebase and conforms to codebase standards. This creates even more work for already time-constrained maintainers. While in many cases contributions can be clearly identified as being generated with AI assistance, an even more fundamental issue is that contributors may not even acknowledge they used AI tools in the contribution's PR or commit message. Contributor due diligence and transparency can go a long way in helping maintainers efficiently process increased contribution volume.

We asked contributors about the extent to which they verify and attribute AI-assisted contributions prior to submitting them to OSS projects.

Encouragingly, the dominant pattern is caution rather than complacency:

61% conduct detailed line-by-line review and full functional testing of AI-generated code before submitting, and only 5% apply minimal or no review (Figure 16). This is evidence that most contributors are not

simply forwarding model output, but treating it as a draft that still requires their own scrutiny.

Disclosure practices are less consistent than review rigor, however.

Attribution practices are mixed: 52% mark AI-assisted work “most of the time” or “always,” while 19% never do (Figure 17). Combined with the thin policy coverage on this topic noted earlier, this split points to attribution as an area where **individual norms, not project policy, are currently doing most of the work**, with predictably uneven results.

Suggestions for attributing genAI

contribution Part of the problem is the lack of a single widely-adopted machine-readable convention for attribution, especially in commit messages where such proposals would typically be made. The Apache Software Foundation (ASF) suggests “Generated-by:”.⁸ But this phrase is misleading when parts were created or edited by a human. Others use “Co-authored-by:”, but AI-generated code cannot be copyrighted in most jurisdictions, so this phrase is misleading.

The Linux kernel project requires attribution in commits using the form “Assisted-by: <agent>:<model_version> [tool1] [tool2]” where tools are specialized tools, e.g., “Assisted-by: Claude:claude-4.5-sonnet coccinelle sparse”. This format provides detailed attribution information, including for specialized tools, without a misleading phrase. For more information, see <https://docs.kernel.org/process/coding-assistants.html>.

8 <https://www.apache.org/legal/generative-tooling.html>

FIGURE 16

3 out of 5 developers thoroughly test AI-assisted contributions before submitting

Source: Survey on Generative AI and Open Source Software Development. Q20. Number of respondents = 137.

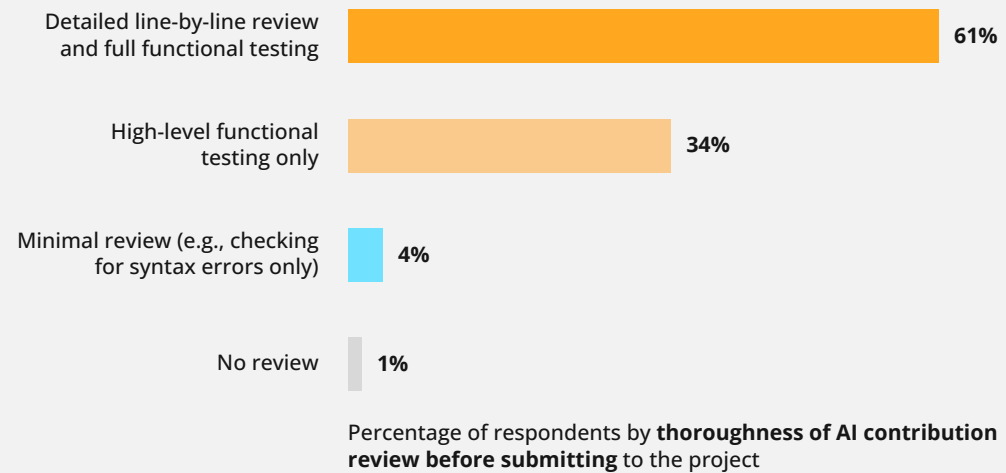
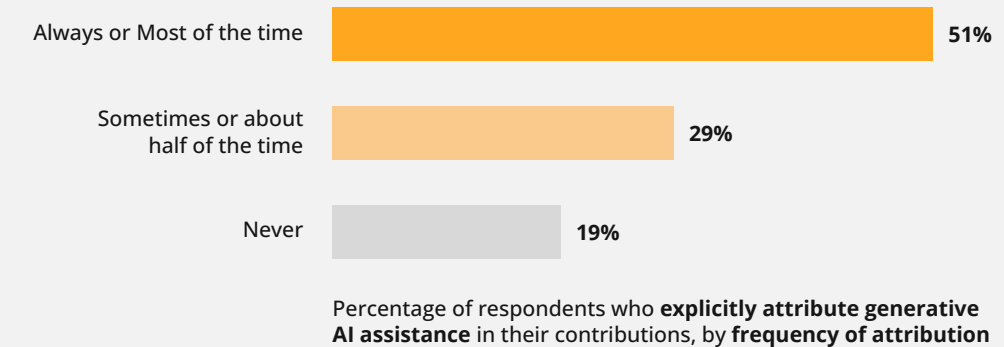


FIGURE 17

Half of contributors always attribute AI-assistance. 1 in 5 never do

Source: Survey on Generative AI and Open Source Software Development. Q19. Number of respondents = 134.



Non-adopters see trust problems as fixable, technical problems as inherent

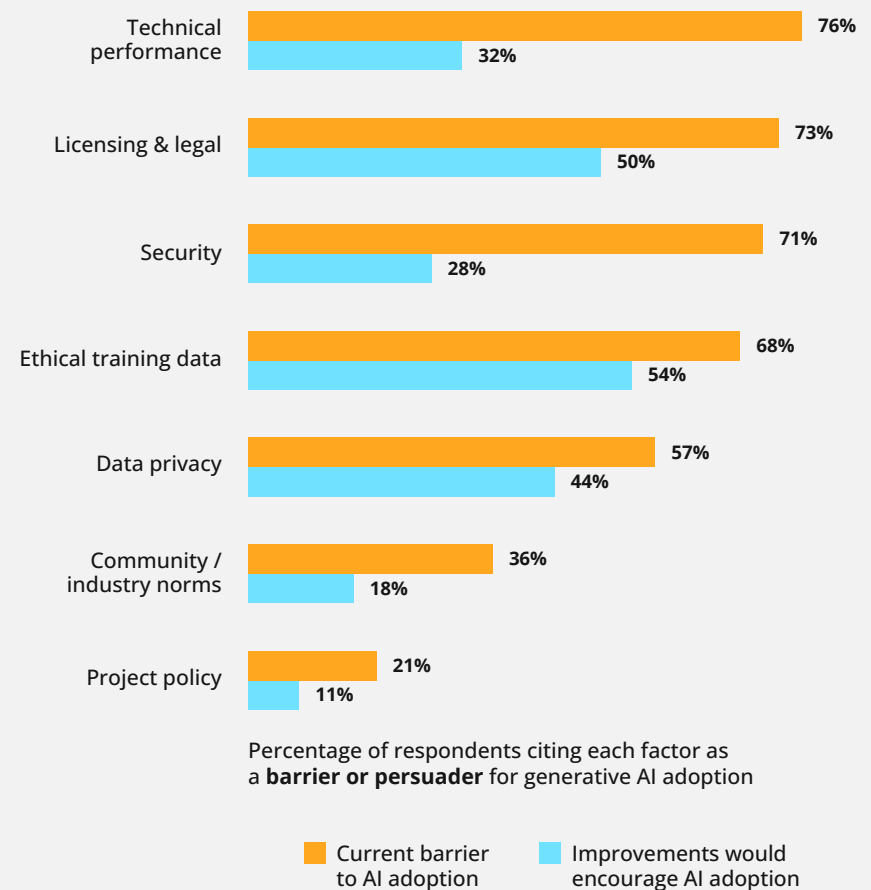
Not every contributor has embraced genAI, and understanding why they have does not matters as much as tracking adoption itself. The study asks non-adopters to identify the reasons they avoid using genAI and what might persuade them to change their minds.

In Figure 18, we compare what factors are current barriers and what factors would need to see improvements before non-adopters would consider using genAI for OSS development. Trust and governance concerns, including data training ethics and privacy, convert into a persuader at roughly the same rate they're cited as a barrier (around three-quarters as likely), while technical performance and security convert at less than half that rate. In other words, **non-adopters don't necessarily see reliability and security as more important than trust and governance**. They cite them just as often, but **they view them as harder to fix**, and are far more skeptical that improvements on those fronts would actually change their minds. "Personal philosophy" [68%] has no comparable persuader at all, and 25% say "none of the above" would move them — together marking a committed non-adopter segment that no amount of added transparency, security, or performance is likely to reach.

FIGURE 18

Improved trust and governance for genAI are viewed as top priorities for non-adopters

Source: Survey on Generative AI and Open Source Software Development. Q26 x Q27. Number of respondents = 96.



Productivity impacts of genAI

Beyond contribution volume and adoption patterns, the most direct way to measure genAI's value to OSS work is time: how much of it the tools save, who benefits most, and where that reclaimed time goes.

Generative AI makes OSS development faster across the board

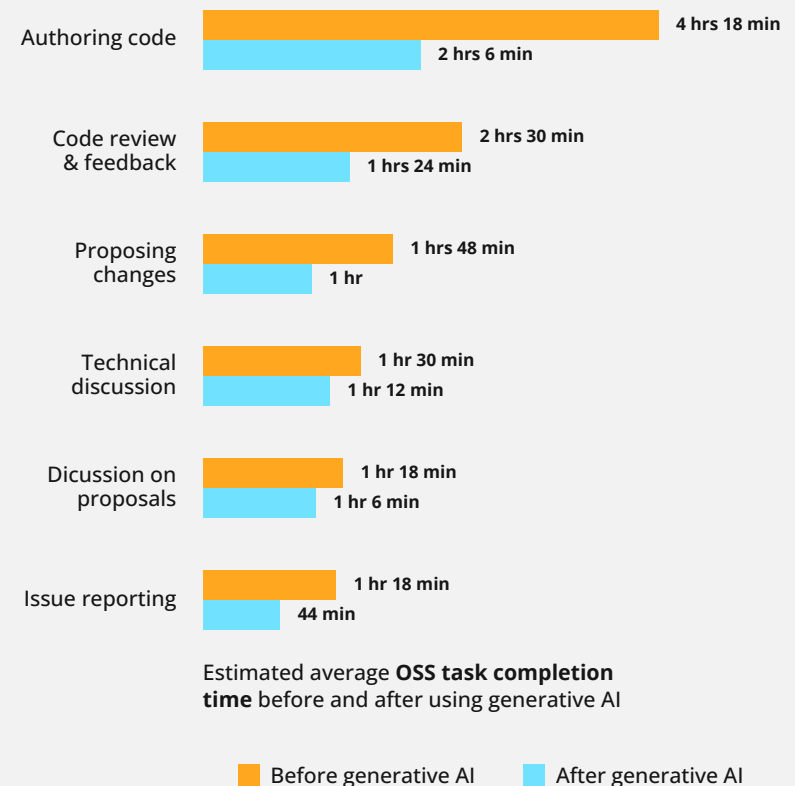
As we have seen, genAI tools are being used in virtually every layer of the software development workflow. Not only do they accelerate writing lines of code, these tools are also used to distill complex codebases, debug problems, identify vulnerabilities, and provide architectural guidance. But how exactly do the productivity effects of genAI propagate across different OSS development tasks? The OSS collaboration paradigm is not purely technical. Many interactions require social interactions such as contribution proposals, bug or issue reporting, and technical discussions. We ask survey respondents who use genAI how long, on average, the typical OSS development task would take them before and after they began using genAI.

Across the board, **genAI reduces estimated task time across all six measured activities** in the pooled sample, with the largest gains in code authoring (–52%) and code review & feedback (–44%), and smaller but still meaningful reductions in technical discussion (–20%) and discussion on proposals (–19%) (Figure 19). This pattern is consistent with genAI's strengths being most pronounced on well-defined, mechanical tasks and weakest on open-ended, judgment-heavy communication.

FIGURE 19

Generative AI improves average productivity across the OSS task spectrum

Source: Survey on Generative AI and Open Source Software Development. Q23. Number of respondents = 96.



Splitting the reductions by open source development role reveals two different experiences (Figure 20). **Maintainers see much larger savings on discussion-heavy tasks** specifically: discussion on proposals (–26% vs. –7%) and technical discussion (–25% vs. –12%), consistent with their heavier review and communication workload and suggesting genAI is easing exactly the coordination burden that scales with project size. Contributors, by contrast, benefit most on code authoring (–58% vs. –49%), where their time is more concentrated — meaning the pooled –52% figure is largely contributor-driven rather than an even split across roles.

“The data surfaced in this report puts numbers behind what we’ve been seeing in the ecosystem. Generative AI increases developer productivity, but it strains the shared trust and process that hold open source projects together. Closing that gap isn’t something any one maintainer can do alone. It takes cross-project coordination, the kind LF AI & Data and the Linux Foundation exist to provide.”

MARK COLLIER GM OF AI & INFRASTRUCTURE, LINUX FOUNDATION

FIGURE 20

Generative AI is particularly beneficial for maintainers in project discussions

Source: Survey on Generative AI and Open Source Software Development. Q2, Q23. Number of respondents = 96.

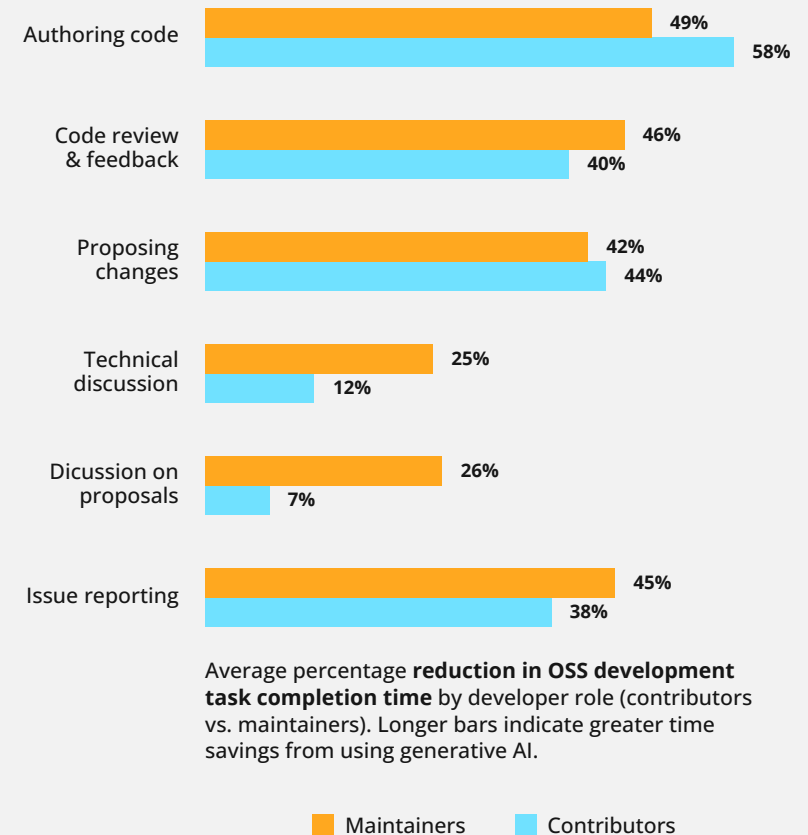
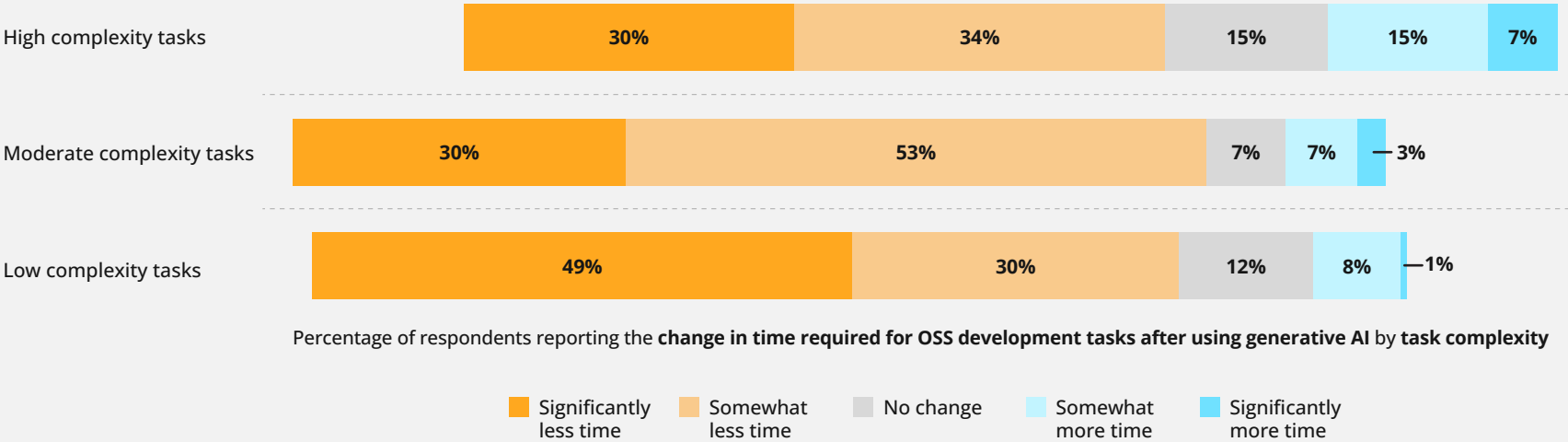


FIGURE 21

Generative AI improves efficiency on average – but not always for more complex tasks

Source: Survey on Generative AI and Open Source Software Development. Q24. Number of respondents = 136.



Time savings shrink as task complexity rises

The time-saving effect is not uniform across complexity levels. Generative AI saves time most reliably for low [79% save time] and moderate [83% save time] complexity tasks (Figure 21), while high-complexity tasks show more mixed results: 64% still save time, but 21% report needing more time than before. **GenAI’s productivity dividend narrows, and can even reverse, as task difficulty rises.**

The finding that AI tends to be more effective at lower complexity tasks is consistent with other studies. Kwa et al. 2025 propose measuring AI systems as a “50%-task-completion time horizon” (the time humans typically take to complete tasks that AI models can complete with 50% success rate). They found that as of 2025 frontier AI models had a 50% time horizon of around 50 minutes. However, it would be unwise to presume that AI systems will not improve. The same study found that frontier AI time horizon has been doubling approximately every seven months since 2019.⁹

9 Thomas Kwa et al., “Measuring AI Ability to Complete Long Software Tasks,” preprint, submitted March 18, 2025, <https://arxiv.org/abs/2503.14499>.

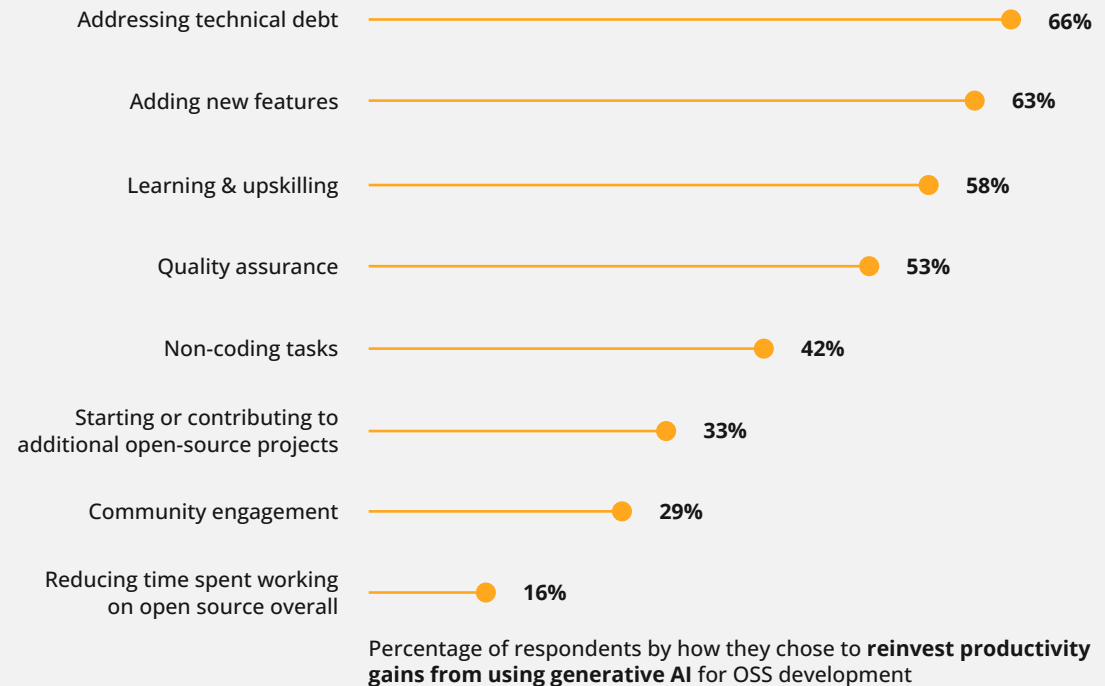
Saved time flows into technical debt and new features

Time savings alone would not matter much if they simply vanished; instead, they are visibly being redirected. **Productivity gains are most often reinvested in addressing technical debt [66%], adding new features [63%], and learning/upskilling [58%]** (Figure 22). Since they are far less likely to use this time outside of OSS development [16%], **genAI productivity is invested right back into OSS projects. In other words, genAI is not shrinking the overall OSS workload so much as reallocating it toward higher-value and longer-deferred work**, from debt paydown to skill-building.

FIGURE 22

OSS developers reinvest genAI productivity gains back into project and human capital

Source: Survey on Generative AI and Open Source Software Development.
Q25. Number of respondents = 139. Total responses = 513.





Conclusion

Generative AI's impact on general software development is hard to overstate. OSS communities are experiencing that impact first-hand and at scale. Because OSS depends as much on social trust, collaboration norms, and shared review labor as it does on code, an influx of AI-assisted contributions cuts closer to the core of how these projects function than it would in other development contexts. Generative AI is not a substitute for best practices, collaboration etiquette, and longstanding community norms. It is a tool that can improve productivity, but it also introduces new challenges and risks that must be managed. Rising concerns about contribution volume, quality, and maintainer burnout are neither abstract or inconsequential — they bear on the sustainability of contributor labor that OSS depends on. To keep open source healthy, communities and project leaders must understand how genAI is altering contribution dynamics and identify optimal adaptation strategies as the technology becomes more tightly integrated into software development.

Much of the public discussion of genAI's effect on OSS to date has been anecdotal, with high-profile examples citing a deluge of "AI slop," mounting review queues, and increased maintainer burnout. This study validates these concerns with direct evidence based on a broad sample of the OSS community. By surveying 234 OSS contributors and maintainers — spanning solo projects to organizations with hundreds of contributors, genAI adopters and non-adopters, and every major governance model — we've been able to confirm which anecdotal concerns hold up at scale (increased review time, quality dilution, disproportionate burden on large projects) and which are more nuanced than the loudest narratives suggest, including the productivity gains, adoption patterns, and largely conscientious verification practices that counterbalance those risks. While prior discussions offered impressions, this report offers measurement.

This study surfaces a number of concrete recommendations for OSS communities and project leaders to consider as they navigate the genAI transition:

- Establish a clear, written policy on AI-assisted contributions and vulnerability reporting rather than leaving practice to informal, ad hoc norms.
- Prioritize transparency and disclosure requirements in that policy. Contributors should send clear and consistent signals to maintainers about what genAI tooling they've used.
- Shift the burden of review onto contributors by requiring self-attestation, testing, and documentation of AI-assisted changes before submission.
- Standardize AI contribution attribution practices so disclosure doesn't remain a matter of individual discretion. One solution would be to adopt and require using the Linux kernel's "Assisted-by:" format in commit messages.
- Use AI as a maintenance asset. AI tools can be used to triage, summarize, and flag issues in incoming contributions to ease review burden and reduce maintainer burnout.
- Continue investing in human mentorship for new contributors — genAI lowers the barrier to entry but increases, rather than decreases, the need for guidance once contributors arrive.
- Weigh the accessibility gains of genAI against its cost to community trust, and design contribution norms that welcome newcomers without sacrificing quality or trust.
- Quality genAI governance and trust are more important than improved technical performance for AI holdouts.

This study is a first look, not a final word. Future research¹⁰ should track these dynamics longitudinally as AI contribution policies and verification tooling mature, dig deeper into how outcomes vary by technical domain and governance model, and give more attention to the roughly one in five OSS developers who have not adopted genAI at all — understanding what would change their minds matters as much as tracking those who have already adopted.

As genAI continues to transform the software development lifecycle (SDLC) in general¹¹, the OSS development paradigm will also adapt. Even though genAI should not be forced upon communities that do not want it, the projects and communities managing this transition most successfully are neither banning genAI nor adopting it uncritically; they are building the policy, verification, and mentorship infrastructure to let its benefits — faster onboarding, reduced toil, and reinvested time — accrue without sacrificing the trust and quality that have always been open source's real competitive advantage.

¹⁰ Related to this work, LF Research has studied how open source program offices (OSPOs) are preparing for the rise of genAI in OSS development. See <https://www.linuxfoundation.org/research/ospo-2025> for additional insights.

¹¹ <https://claude.com/blog/the-ai-native-sdlc-playbook>



Methodology

DATA COLLECTION METHOD	Web survey
SURVEY LENGTH	45 questions
SURVEY TARGET POPULATION	OSS contributors and project maintainers that may or may not use generative AI for open source software development
SURVEY DISTRIBUTION	Linux Foundation newsletter subscribers, Linux Foundation social media channels, research partner email/social channels (Red Hat, OpenSSF, Kaiyuanshe, GOSIM, LF AI & Data)
SURVEY INSTRUMENT DEVELOPMENT PERIOD	February - April 2026
SURVEY FIELDING PERIOD	April - June 2026
ANALYTICAL SAMPLE SIZE	234 qualified respondents
MARGIN OF ERROR (95% CONFIDENCE LEVEL)	+/- 6.41%
DATA ACCESS	https://zenodo.org/communities/linux-foundation-research
DECLARATION OF AI IN THE RESEARCH PROCESS	LF Research uses generative AI and AI-assisted technologies in varying degrees for research-related tasks including aggregating secondary sources, synthesizing data, supporting report drafting, copyediting, and translation. All AI-generated content is reviewed and revised as needed.

Survey demographics

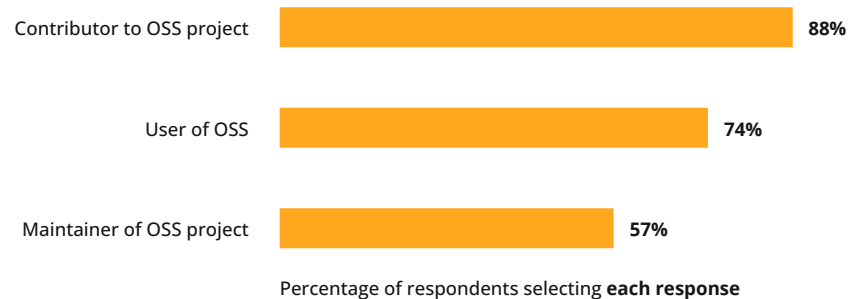
OSS development

Respondents bring deep and varied experience to their OSS work. Most respondents wear both hats: of the 234 contributors and/or maintainers surveyed, 206 identify as contributors and 134 as maintainers (Figure 23), a substantial overlap reflecting how common it is for maintainers to remain active contributors to their own or other projects.

FIGURE 23

Which of the following roles best describe your engagement with OSS? (select all that apply)

Source: Survey on Generative AI and Open Source Software Development.
Q2. Number of respondents = 234. Total responses = 512.

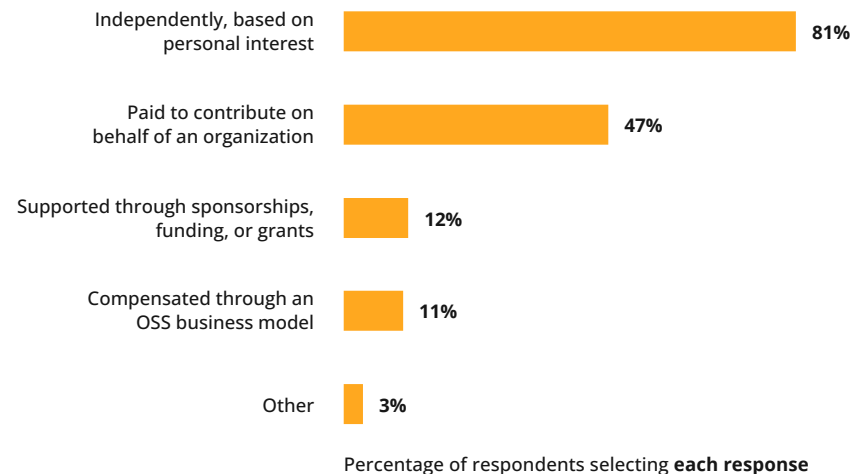


Personal interest is the dominant driver of participation: 81% of contributors participate independently out of personal interest, and nearly half [47%] are also compensated by an employer to contribute (Figure 24) — figures that are not mutually exclusive, underscoring that intrinsic motivation and paid work frequently coexist rather than compete.

FIGURE 24

In what capacity do you contribute to OSS projects? (select all that apply)

Source: Survey on Generative AI and Open Source Software Development.
Q6. Number of respondents = 234. Total responses = 361.



That commitment shows up in tenure as well: contributors are highly experienced, with roughly 52% reporting 11 or more years as a project contributor (Figure 25). Maintainer experience is more evenly distributed across tenure bands, and a full quarter of respondents report no maintainer experience at all — a reminder that this sample captures both long-tenured project leaders and contributors who have not yet stepped into a maintainer role.

The work itself is concentrated but not narrow. Code and development [89%] dominates the types of contributions respondents make, with documentation [60%] and testing/QA [59%] rounding out the top three (Figure 26) — evidence

that most respondents are hands-on technical contributors first, even as many also take on the surrounding work that keeps a project healthy. The projects they support skew toward infrastructure and systems software: operating systems/Linux distributions [37%], networking/infrastructure [28%], and programming languages/runtimes [27%] are the most common domains (Figure 27), typically organized as personal/solo [63%] or community-led [58%] efforts, though foundation-led [49%] and corporate-backed [39%] projects are also well represented (Figure 28) — a governance mix spanning the full spectrum from individually run repositories to formally structured foundations.

FIGURE 25

Please indicate your years of experience in the following OSS roles. (select one response per row)

Source: Survey on Generative AI and Open Source Software Development. Q4. Number of respondents = 234.

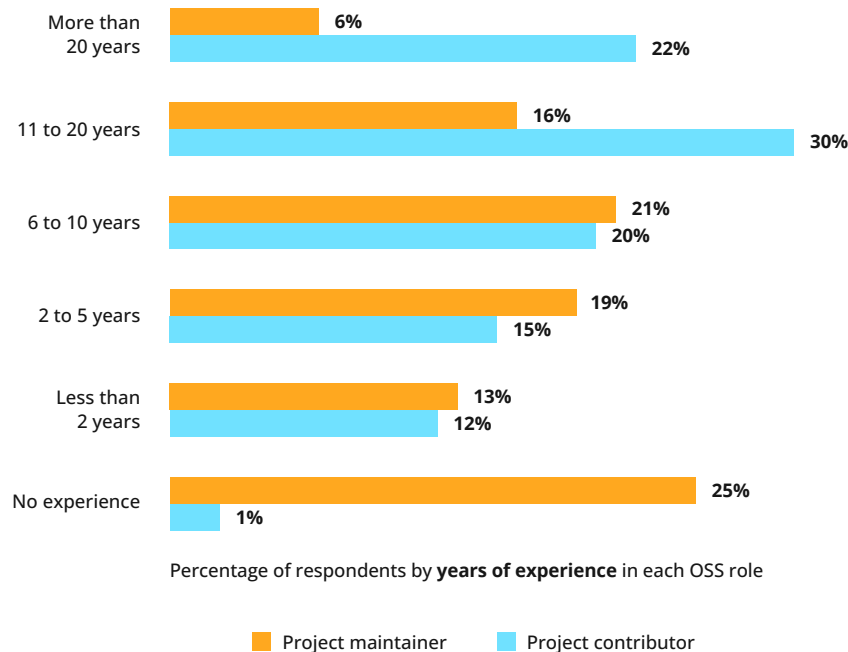


FIGURE 26

In what ways do you contribute to the OSS ecosystem? (select all that apply)

Source: Survey on Generative AI and Open Source Software Development. Q5. Number of respondents = 234. Total responses = 967.

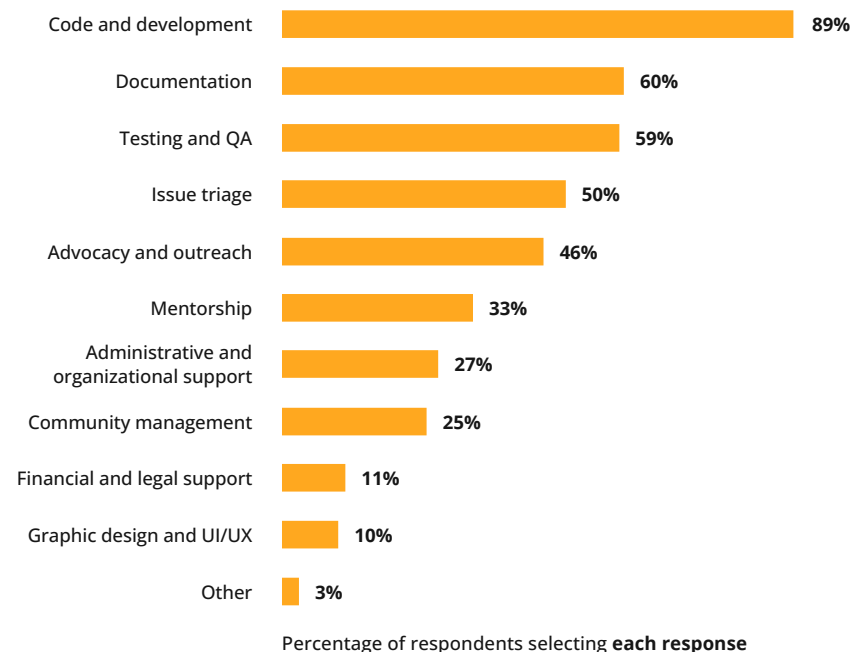
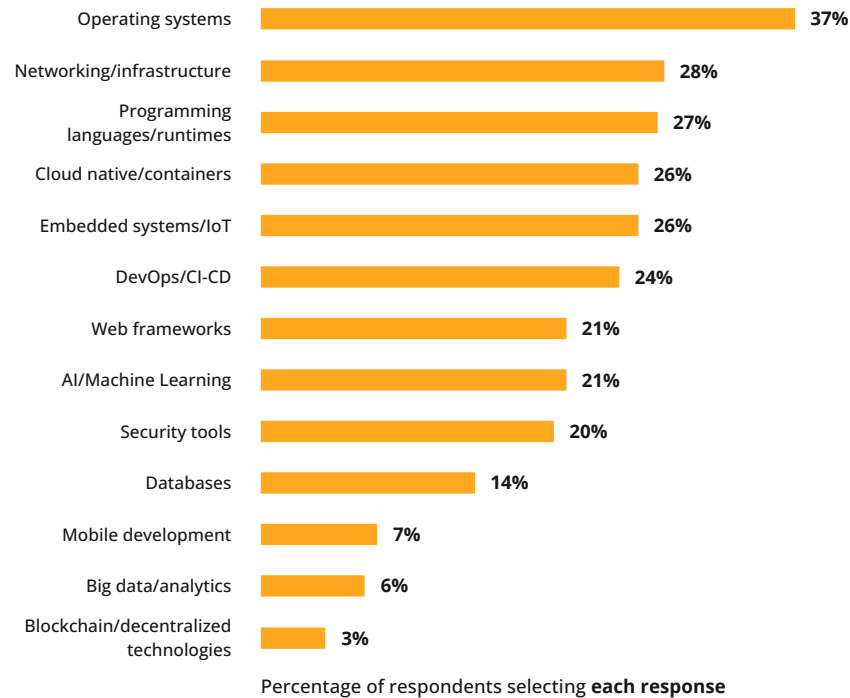


FIGURE 27

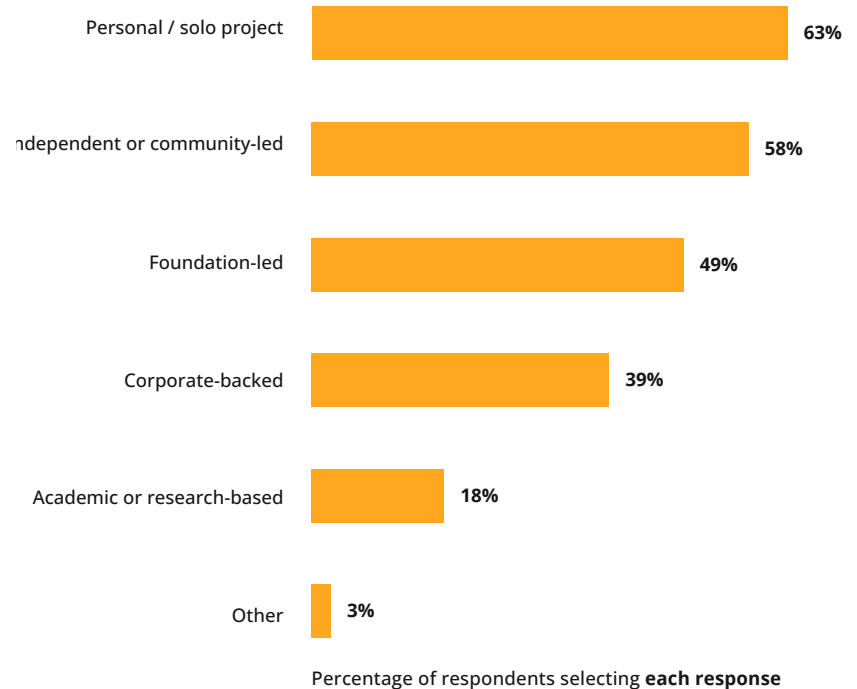
Which of the following best describes the primary domain of the OSS project(s) you contribute to? (select all that apply)

Source: Survey on Generative AI and Open Source Software Development.
Q7. Number of respondents = 213. Total responses = 553.

**FIGURE 28**

Which of the following governance models best describes the OSS project(s) you contribute to? (select all that apply)

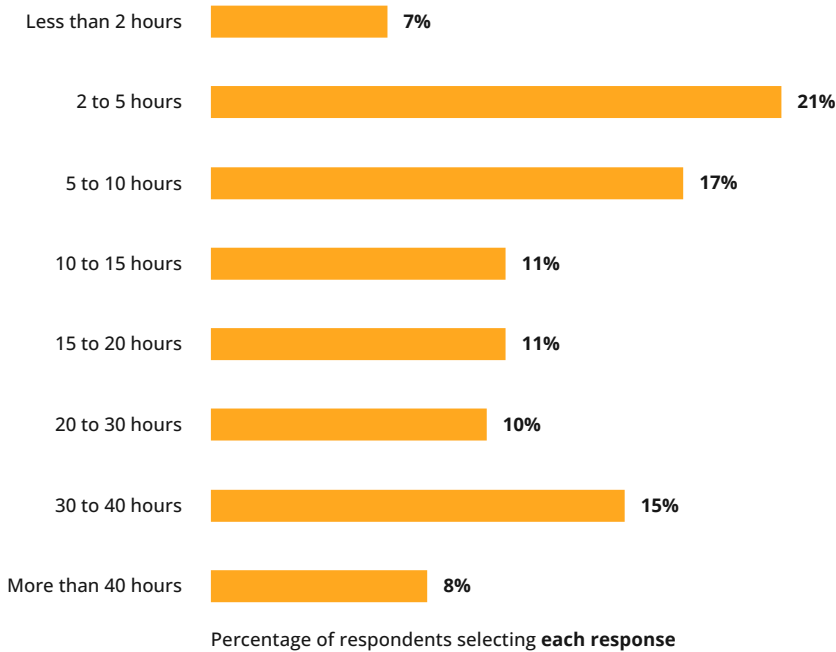
Source: Survey on Generative AI and Open Source Software Development.
Q8. Number of respondents = 234. Total responses = 536.



How much of that work translates into hours tells a bimodal story: the single most common response is 2–5 hours/week [21%], but 15% allocate 30–40 hours, pointing to a mix of casual and deeply committed contributors rather than one typical profile (Figure 29).

FIGURE 29
How many hours per week do you allocate to OSS development? (select one)

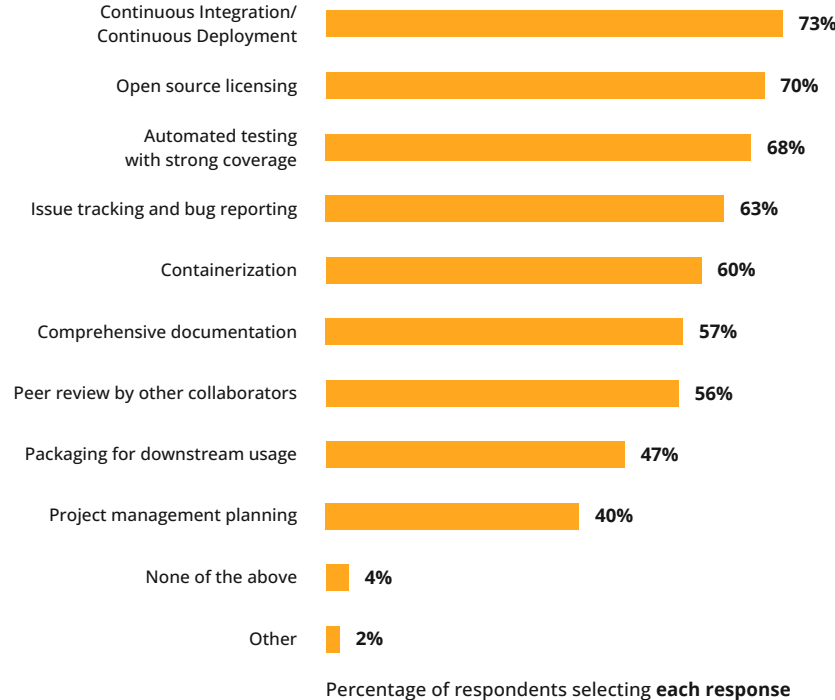
Source: Survey on Generative AI and Open Source Software Development. Q21. Number of respondents = 138.



This is also a population with mature engineering habits: CI/CD [73%], open source licensing [70%], and automated testing [68%] are the most widely adopted best practices (Figure 30), suggesting that the genAI workflows described earlier in this report are being layered onto projects that already follow established engineering discipline, not bolted onto ad hoc process.

FIGURE 30
Which of the following best practices do you utilize in your OSS development workflow? (select all that apply)

Source: Survey on Generative AI and Open Source Software Development. Q16. Number of respondents = 139.



Generative AI experience

Generative AI use is now the norm rather than the exception across this population: only 20% report no usage at all, and adoption is spread nearly evenly across OSS development [59%], other tasks [58%], and non-OSS software development [49%] (Figure 31) — confirming that the dynamics explored earlier in this report describe a population for whom genAI is already a routine part of the toolkit, not a novelty.

Usage is heavily concentrated at high frequencies: 52% use genAI daily and 23% nearly every day, meaning three-quarters use it at least near-daily (Figure 32). That intensity is matched by tenure — a majority [57%] have been using genAI for more than a year (Figure 33) — indicating that the views captured throughout this report largely come from experienced, habitual users rather than recent or occasional adopters.

FIGURE 31
Do you use generative AI? (select all that apply)

Source: Survey on Generative AI and Open Source Software Development. Q1. Number of respondents = 234. Total responses = 437.

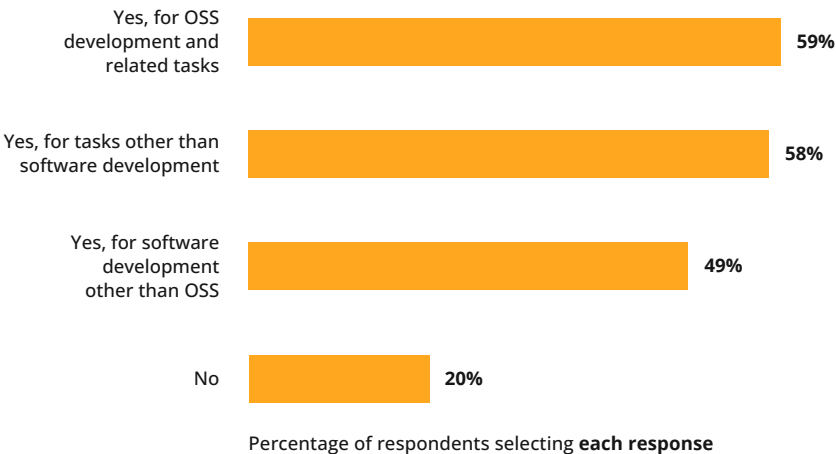


FIGURE 32
How frequently do you use generative AI? (select one)

Source: Survey on Generative AI and Open Source Software Development. Q10. Number of respondents = 187.

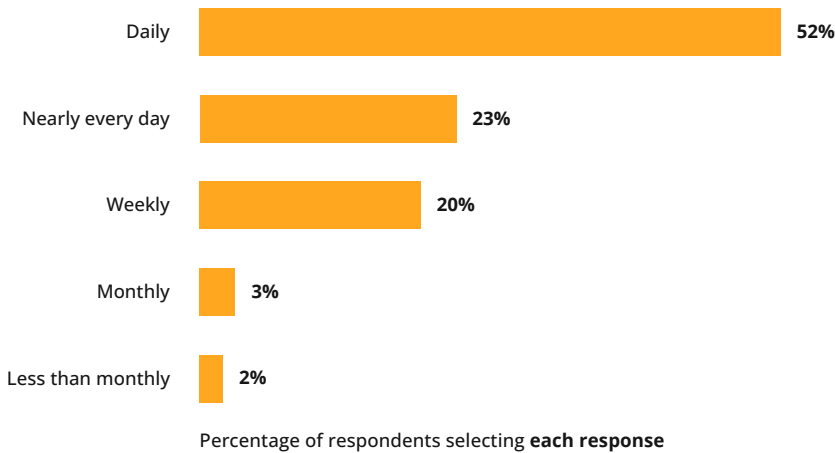
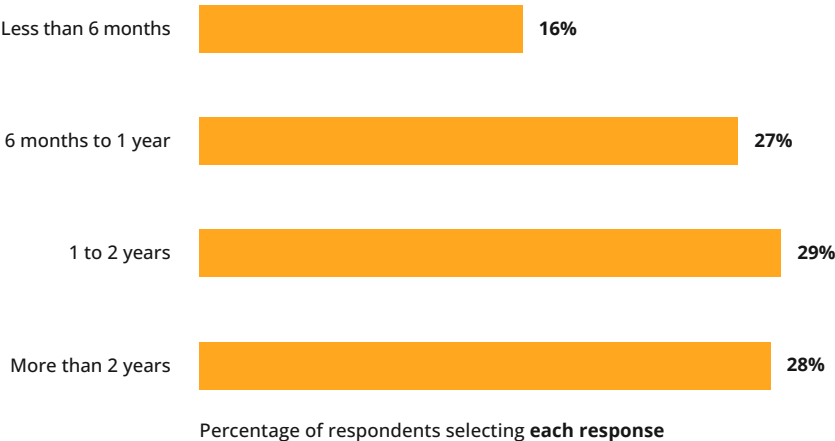


FIGURE 33
How long have you been using generative AI? (select one)

Source: Survey on Generative AI and Open Source Software Development. Q11. Number of respondents = 187.



Maintainer demographics

The maintainer subsample deserves a closer look, since project size and role are the two lenses used to segment maintainer findings throughout this report.

Most maintainers work on small projects [2–10 contributors: 42%], while micro (solo) and very large (500+ contributor) projects are equally represented at the margins [12% each] (Figure 34). Their responsibilities are distributed roughly evenly across lead [32%], co-maintainer [25%], and solo [24%] roles (Figure 35) — a balanced mix that gives the project-size and role-based findings elsewhere in this report broad coverage across the maintainer experience, from solo stewards of small repositories to lead maintainers coordinating large, distributed teams.

FIGURE 34
What is the scale of the project you maintain in terms of contributor base? (select one)

Source: Survey on Generative AI and Open Source Software Development. Q29. Number of respondents = 134.

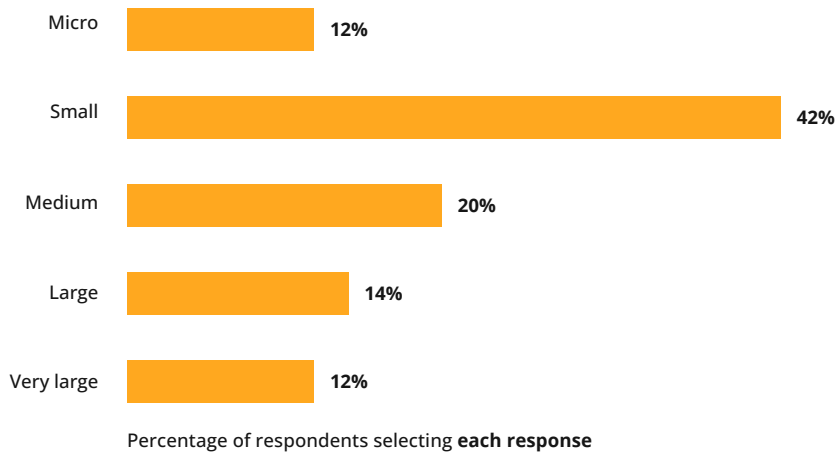
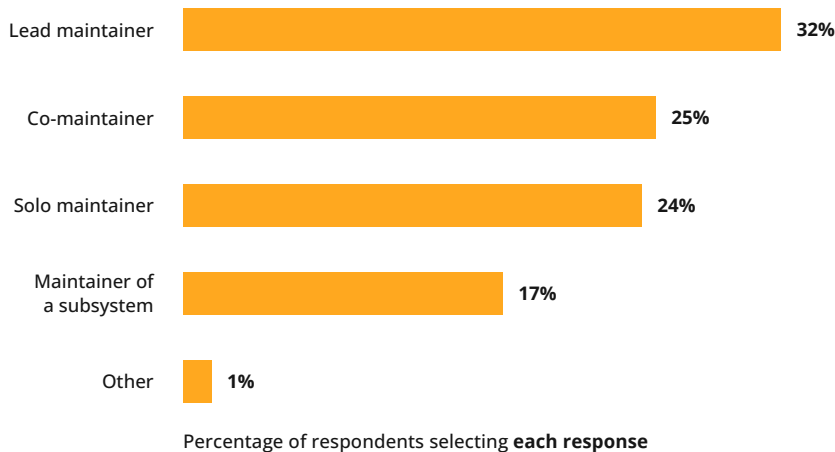


FIGURE 35
Describe your share of duties on the project you help maintain (select one)

Source: Survey on Generative AI and Open Source Software Development. Q28. Number of respondents = 135.



About the authors



SAM BOYSEL is a Data Scientist at the Linux Foundation. In this role, he contributes to a variety of empirical initiatives to uncover new insights into open source communities. His work often leverages microeconomic theory to analyze behaviors, incentives, and community dynamics across open source ecosystems. Before joining the Linux Foundation, he was a postdoctoral researcher with the Laboratory for Innovation Science at Harvard. He holds a B.A., M.S., and Ph.D. in Economics.



ANNI LAI is a longtime open source community builder, advocate, and evangelist. With nearly a decade of active involvement in the global open source ecosystem, she has supported developer communities across more than 40 countries and five continents. She currently serves as Co-Chair of the Generative AI Commons and an elected board member of LF AI & Data, and she also advises LF Europe, GOSIM, and KAIYUANSHE and leads the Open Source Operations at Futurewei. Anni is a passionate believer that AI should serve the global population—rooted in ethics, trust, and openness. Anni holds a B.A. and an M.S. in Computer Science from Rutgers University and San José State University, respectively.

Acknowledgments

We thank all the people who participated in the survey. Special thanks to Linux Foundation colleagues and collaborators for their involvement in the various stages of the research process: Kimberly Craven, Anni Lai, Adrienn Lawson, Frank Nagle, Brian Proffitt, Christopher “CRob” Robinson, David A. Wheeler, and Laura Guazzelli.

The Linux Foundation thanks the following organizations who sponsored this study:





Red Hat is an open hybrid cloud technology leader, delivering a consistent, comprehensive foundation for transformative IT and artificial intelligence (AI) applications in the enterprise. As a trusted adviser to the Fortune 500, Red Hat offers cloud, developer, Linux, automation, and application platform technologies, as well as award-winning services.



Founded in 2021, **Linux Foundation Research** explores the growing scale of open source collaboration, providing insight into emerging technology trends, best practices, and the global impact of open source projects. By leveraging project databases and networks and committing to best practices in quantitative and qualitative methodologies, Linux Foundation Research is creating the go-to library for open source insights for the benefit of organizations worldwide.

 x.com/linuxfoundation

 facebook.com/TheLinuxFoundation

 linkedin.com/company/the-linux-foundation

 youtube.com/user/TheLinuxFoundation

 github.com/linuxfoundation



Copyright © 2026 **The Linux Foundation**

This report is licensed under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License**.

To reference this work, please cite as follows: Sam Boysel and Anni Lai, "Navigating Open Source Software Development in the Generative AI Era: Challenges and Opportunities for Project Communities," foreword by Brian Proffitt, The Linux Foundation, October 2026.

