ShodhAI: Journal of Artificial Intelligence
July-September 2026 3(2)
https://shodhai.org

Original Article

THE STATUS OF ARTIFICIAL INTELLIGENCE IN THE DEVELOPER COMMUNITY: A LITERATURE REVIEW OF STATISTICS, TRENDS, AND EXPECTATIONS CURRENT ADOPTION PATTERNS, EMERGING TRENDS, AND FUTURE OUTLOOK (2023–2026)

Stefano Colafranceschi [1*]
[1] ISAT, James Madison University, Virginia, US

ABSTRACT

Artificial intelligence (AI) has moved from an experimental novelty to a structural component of professional software development within roughly three years. This literature review synthesizes evidence from four large-sample industry surveys: the Stack Overflow. (2025) (n = 49,009), the Google Cloud. (2025) (n > 5,000), the JetBrains State of the Developer Ecosystem 2025 (n = 24,534), and Github. (2025) platform telemetry. This study presents a targeted set of peer-reviewed and preprint academic studies, including five randomized controlled trials, to assess the current status, dominant trends, and near-term outlook of AI adoption in the global developer community. Convergent survey evidence indicates that 84% of developers now use or plan to use AI tools (up from 76% in 2024), with 51% of professional developers using AI daily and adoption reaching 90% in DORA's sample. Despite this, trust in AI accuracy has fallen to 29–33%, down from roughly 40% in prior years, and positive favorability has declined from 72% to 60%, producing a well-documented adoption–trust paradox. Unlike prior industry-report syntheses, this review foregrounds contradictory causal evidence: while an early randomized controlled trial found AI assistance produced a 55.8% speed gain on a scoped, unfamiliar task, the most methodologically rigorous field experiment to date, a 2025 trial by METR, found that AI access slowed experienced open-source developers by 19% on real maintenance work, even though those same developers believed AI had made them faster both before and after the study. Additional peer-reviewed evidence links AI-assisted coding to reduced code security, rising code churn and duplication, and measurable deficits in skill formation among less experienced programmers. The review concludes that AI adoption is now unambiguous and structural, but that its net effect on software quality, security, and the profession's skill base remains contested, and appears to depend more on task complexity, codebase familiarity, and developer experience than on the technology in isolation.

Keywords: Artificial Intelligence, Developer Community, AI Coding Assistants, Generative AI, Software Engineering Productivity, Trust in AI, Agentic Coding, Code Quality, Security Vulnerabilities, Skill Formation, Randomized Controlled Trial, Stack Overflow Survey, Github Copilot, DORA Metrics

INTRODUCTION

The integration of artificial intelligence into software development has accelerated dramatically since large language models capable of code generation reached general availability. GitHub Copilot's public launch in 2022, the rapid mainstreaming of chat-

*Corresponding Author:
Email address: Stefano Colafranceschi (colafrsx@jmu.edu)
Received: 18 July 2026; **Accepted:** 20 August 2026; **Published** 07 September 2026
DOI: 10.29121/ShodhAI.v3.i2.2026.103
Page Number: 46-56
Journal Title: ShodhAI: Journal of Artificial Intelligence
Journal Abbreviation: ShodhAI J. Artif. Intell.
Online ISSN: 3048-9245, **Print ISSN:** 3108-1940
Publisher: Granthaalayah Publications and Printers, India
Conflict of Interests: The authors declare that they have no competing interests.
Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.
Authors' Contributions: Each author made an equal contribution to the conception and design of the study. All authors have reviewed and approved the final version of the manuscript for publication.
Transparency: The authors affirm that this manuscript presents an honest, accurate, and transparent account of the study. All essential aspects have been included, and any deviations from the original study plan have been clearly explained. The writing process strictly adhered to established ethical standards.

based assistants such as ChatGPT and Claude, and the subsequent emergence of IDE-native and agentic tools such as Cursor, Claude Code, and Windsurf have moved AI from an early-adopter curiosity to a default component of professional tooling within roughly four years. This trajectory is unusually well documented: because AI coding tools are themselves digital products, their usage leaves telemetry, commit-level, and survey traces at a scale rarely available for prior developer-tooling shifts, enabling a more quantitative and falsifiable assessment than was possible for earlier transitions such as the move to distributed version control or cloud-native deployment.

The stakes of this shift extend beyond convenience. Software developers number in the tens of millions globally and underpin nearly all digital infrastructure, so even modest per-developer productivity or quality effects aggregate to substantial economic and security consequences. At the same time, code is a domain in which correctness is verifiable and errors can be consequential, which makes it an informative test case for broader claims about generative AI's reliability, and a domain where the gap between perceived and measured benefit can be studied with precision.

Most existing syntheses of this topic rely on a single vendor-produced survey or platform telemetry report, present adoption and sentiment statistics without reference to controlled experimental evidence, and rarely reconcile self-reported productivity perceptions with the smaller but methodologically stronger body of randomized controlled trials, security audits, and code-quality analyses. This creates a risk of survivorship in the evidence base: figures that are easy to cite (adoption percentages, favorability scores) circulate widely, while harder-to-summarize but arguably more decision-relevant findings. For example, that the most rigorous randomized trial to date found AI usage slowed experienced developers down are comparatively underexposed.

This review addresses that gap. It has three objectives: first, to document current, verifiable statistics on AI tool adoption and usage intensity within the global developer community, drawn from the largest and most recent industry surveys; second, to identify dominant trends in tool selection, workflow integration, and productivity outcomes, explicitly distinguishing correlational survey findings from causal experimental evidence; and third, to examine stated expectations and anticipated trajectories for the near future in light of that evidence. Particular attention is paid to three tensions that recur throughout the literature: rising usage alongside declining trust; productivity gains that are robust on average but contested for complex, unfamiliar work; and adoption benefits that concentrate among already-experienced practitioners and technologically well-resourced regions, potentially widening rather than narrowing existing gaps.

The remainder of the review proceeds as follows. Section 2 describes the search and source-selection approach. Section 3 reports current adoption statistics and tool-market data. Section 4 examines key trends, including agentic workflows, productivity patterns, community learning dynamics, and code quality and security implications. Section 5 details the trust paradox. Section 6 presents the contradictory experimental evidence on productivity in depth. Section 7 reviews emerging findings on cognitive and skill-formation effects. Section 8 offers a methodological critique of the cited literature. Section 9 discusses near-term expectations, Section 10 synthesizes the findings, Section 11 states the review's own limitations, and Section 12 concludes.

METHODOLOGY

This review follows a targeted narrative-synthesis approach rather than a full systematic review protocol; it does not claim PRISMA-level exhaustiveness, and this limitation is revisited explicitly in Section 11. Sources were identified between July and August 2026 through structured searches of arXiv, ACM Digital Library and Semantic Scholar listings, and the public-facing methodology and results pages of the major industry surveys.

Inclusion criteria: (a) large-sample developer or technology-professional surveys ($n > 1{,}000$) published by an identifiable organization with a stated methodology, covering 2023–2026; (b) peer-reviewed or arXiv-hosted empirical studies of AI-assisted software development, including randomized controlled trials, observational commit-log analyses, and mixed-methods workplace studies; (c) platform telemetry reports from major code-hosting or developer-tooling providers; and (d) macro-context sources (e.g., cross-country AI-adoption indices) used only to frame, not substitute for, developer-specific findings.

Exclusion criteria: marketing copy without an underlying methodology, single-anecdote case studies, and secondary aggregator articles were excluded as primary evidence; where only secondary reporting of a primary figure could be located (notably for some 2026 vendor revenue and market-share figures), this is flagged explicitly in-text rather than cited as if independently verified.

Priority was given to studies using random assignment for any causal claim about productivity, quality, or learning outcomes, consistent with standard evidence hierarchies in the social and computing sciences. Five randomized controlled trials meeting this bar were identified and are analyzed jointly in Section 6 specifically because their divergent findings are informative only when compared directly rather than cited in isolation, as is common in industry summaries. Reference lists of identified papers were used to locate closely related contemporaneous studies (a limited form of backward snowballing) but forward citation-chasing and database-level systematic screening were not performed, so the academic literature covered here should be read as illustrative of major findings and debates rather than exhaustive.

Throughout the review, sources are weighted according to their design: randomized controlled trials and pre-registered analyses are treated as the strongest evidence for causal claims; large-N observational studies (e.g., commit-log analyses across tens of millions of records) are treated as strong evidence for descriptive and correlational claims; and self-selected industry surveys, while informative about sentiment and self-reported behavior at scale, are treated as the weakest tier for causal inference and are explicitly flagged where their commercial provenance could plausibly bias reported findings. This tiering is revisited systematically in Section 8.

CURRENT ADOPTION STATISTICS

OVERALL USAGE RATES

The 2025 Stack Overflow Developer Survey, fielded from May 29 to June 23, 2025, received 49,009 responses from 177 countries, making it the largest single data source used in this review. Its methodology page discloses that respondents were recruited primarily through Stack Overflow's own channels: onsite messaging, blog posts, email/newsletter subscribers, banner ads, and social media with a small supplementary Reddit advertising campaign accounting for under 2% of responses; the platform notes that its most highly engaged existing users were likeliest to see the survey prompts. With that non-probability sampling frame in mind, the survey found that 84% of respondents are using or plan to use AI tools in their development process, up from 76% in 2024 and approximately 70% in 2023. Among professional developers, 51% report daily use of AI tools.

Parallel findings from JetBrains' State of the Developer Ecosystem 2025 survey (n = 24,534 across 194 countries, fielded April–June 2025) indicate that 85% of developers regularly use AI tools for coding and development, and 62% rely on at least one specialized AI coding assistant, agent, or code editor. Notably, only 44% report that AI is fully or partially integrated into their actual workflow, a substantially lower figure than the headline usage rate; this usage–integration gap suggests that a considerable share of "regular" AI use remains exploratory or piecemeal rather than embedded in standard practice, a nuance that headline adoption percentages alone can obscure.

Google Cloud. (2025) DORA Report, based on more than 5,000 technology-professional respondents and over 100 hours of supplementary qualitative interviews, records even higher penetration: AI adoption reached 90% (a 14-percentage-point rise year-over-year), with respondents reporting a median of two hours per day working with AI and 65% describing their use as "heavy." Thirty-nine percent report turning to AI reflexively, as a first resort, when encountering a problem or task, which DORA's authors interpret as a sign that AI has shifted from an optional aid to a default habit for a large minority of practitioners.

GitHub. (2025) report, published October 28, 2025, corroborates rapid mainstreaming from the platform-telemetry side: approximately 80% of new GitHub users tried Copilot within their first week on the platform, more than 1.1 million public repositories now import a large language model SDK (a 178% year-over-year increase, with 693,867 of those projects created within the trailing twelve months alone), and AI-related repositories overall exceed 4.3 million. Octoverse also reports that TypeScript overtook both JavaScript and Python to become GitHub's most-used language for the first time in August 2025 (the most significant language-ranking shift on the platform in over a decade) and links this partly to AI-assisted development, noting that typed languages allow compiler tooling to catch a class of error that GitHub's own analysis found accounts for a majority of LLM-generated compilation failures.

TOOL LANDSCAPE AND MARKET LEADERS

OpenAI's GPT models remain the most widely used large language models for development work (approximately 82% of Stack Overflow respondents). Among specialized coding assistants, GitHub Copilot holds a leading position, cited by around 68% of Stack Overflow respondents; Microsoft disclosed 4.7 million paid Copilot subscribers as of late January 2026 (75% year-over-year growth) and reported enterprise adoption across roughly 140,000 organizations, a tripling year-over-year, on its Q3 FY2026 earnings call, with deployment reported at roughly 90% of Fortune 100 companies. Claude (Anthropic) and Google Gemini have also achieved substantial shares of reported use (Claude Sonnet models cited by approximately 43–45% of professional respondents; Gemini Flash by approximately 35%).

Among AI-native, IDE-integrated tools, Stack Overflow's 2025 data place Cursor at 18% usage share, Claude Code at 10%, and Windsurf at 5%, reflecting a broader shift toward integrated, agentic development environments. Industry trackers and press reporting, sourced from vendor disclosures and analyst compilations rather than audited or peer-reviewed figures, and therefore held to a lower evidentiary standard than the survey data above, indicate that Cursor's developer, Anysphere, surpassed $2 billion in annualized revenue by March 2026 (up from roughly $1 billion in November 2025), that Cursor has more than 7 million monthly active users, and that Claude Code's reported workplace adoption rose from approximately 3% in April 2025 to roughly 18% by January 2026. Cognition's reported $3 billion acquisition of Windsurf during 2025 further illustrates the pace of consolidation in this segment. These figures should be read as directional indicators of a fast-moving, competitive market rather than precise, independently verified benchmarks.

REGIONAL AND DEMOGRAPHIC VARIATION

Developer-specific adoption figures should be read against the broader pattern of AI adoption globally, which is highly uneven. Stanford Institute for Human-Centered Artificial Intelligence. (2026) AI Index reports that generative AI reached approximately 53% adoption among the general population worldwide within three years of ChatGPT's release, a faster uptake curve than either the personal computer or the internet achieved, but with pronounced cross-country variation correlated with GDP per capita: Singapore leads at 61% adoption and the United Arab Emirates follows at 54%, while the United States, despite its dominance in AI development and investment, ranks 24th globally at 28.3%. These are general-population figures rather than developer-specific ones, but they provide important context: they indicate that raw national wealth is an imperfect predictor of adoption, and that the developer surveys synthesized in Section 3.1 are themselves geographically skewed, Stack Overflow's top ten responding countries in 2025 were the United States, Germany, India, the United Kingdom, France, Canada, Ukraine, Poland, the Netherlands, and Italy, none of which are in the Global South.

Qualitative and policy literature on AI adoption in lower-income regions points to compounding structural barriers largely absent from the wealthier markets that dominate developer-survey samples: unreliable electricity and limited broadband infrastructure, geographically concentrated compute capacity (predominantly in the United States and China), higher effective cloud-compute costs in many African markets due to smaller-scale local markets and currency instability, a documented adoption penalty in regions where low-resource languages predominate even after controlling for GDP and internet access, and constrained venture and public funding for local AI tooling. Because the major developer surveys cited in this review recruit disproportionately from North America and Europe, their headline adoption figures likely overstate global developer-population averages and should not be extrapolated uncritically to regions underrepresented in the underlying samples.

Table 1 consolidates the principal quantitative metrics referenced throughout this review, drawn from the sources described above, alongside their reference period. Because the underlying studies differ in design, population, and time frame, values are reported individually rather than forced into a single year-over-year comparison.

Metric	Value	Period	Source
Use or plan to use AI tools (developers)	76% → 84%	2024 → 2025	Stack Overflow
Daily AI use, professional developers	51%	2025	Stack Overflow
AI adoption, tech professionals	76% → 90%	2024 → 2025	Google DORA
Regular AI use for coding	85%	2025	JetBrains
Use ≥ 1 specialized AI coding assistant	62%	2025	JetBrains
AI fully/partially integrated in workflow	44%	2025	JetBrains
New GitHub users trying Copilot in week 1	~80%	Aug. 2025	GitHub Octoverse
Public repos importing an LLM SDK	>1.1M (+178% YoY)	Aug. 2025	GitHub Octoverse
AI-related GitHub repositories (total)	>4.3M	2025	GitHub Octoverse
Trust in AI accuracy	~40% → 29%	2023–24 → 2025	Stack Overflow
Positive favorability toward AI	72% → 60%	2024 → 2025	Stack Overflow
"Almost right" cited as top frustration	66%	2025	Stack Overflow
Agentic coding-tool adoption (GitHub projects)	22.2%–28.7%	Feb. 2026	Robbes et al. (2026)
AI-authored share of Python functions (US)	~29%	through 2025	Daniotti et al. (2025)
Code churn (revised within 2 weeks of commit)	3.1% → 7.1%	2020 → 2025	GitClear
GitHub Copilot paid subscribers	4.7M (+75% YoY)	Jan. 2026	Microsoft/GitHub

Generative AI adoption, general population (global)	~53%	2023–2026	Stanford Institute for Human-Centered Artificial Intelligence. (2026)

Source: Compiled by the author from Stack Overflow. (2025), Google Cloud. (2025), JetBrains. (2025), GitHub. (2025), GitClear. (2025), GitClear. (2026), Robbes et al. (2026), Daniotti et al. (2025), and Stanford Institute for Human-Centered Artificial Intelligence. (2026), see References.

KEY TRENDS

FROM AUTOCOMPLETE TO AGENTIC WORKFLOWS

A defining trend of 2025–2026 is the rapid emergence of agentic coding tools: systems capable of multi-step planning, code generation, testing, and iteration with reduced human intervention, as distinct from earlier autocomplete-style assistants. The most rigorous evidence on the spread of this category comes from Robbes et al. (2026), published in ACM Transactions on Software Engineering and Methodology, which analyzed 128,018 GitHub projects and estimated an agentic coding-tool adoption rate of 22.2–28.7% as of February 2026. The study found adoption spread across mature projects, established organizations, multiple programming languages, and diverse project topics, indicating that agentic coding is not confined to experimental or hobbyist repositories. At the same time, the same study found that adoption intensity remains highly concentrated: the median adopting repository generated only one to two agentic pull requests during a three-month observation window, indicating broad experimentation without, in most cases, deep workflow dependence. At least two independent, contemporaneous analyses using different sampling methodologies reported adoption rates in a similar range over the same period, lending the estimate reasonable convergent support.

While a majority of developers still rely primarily on simpler assistants or chat interfaces, the direction of travel is consistent across every source reviewed: Stack Overflow. (2025) data already shows meaningful usage shares for IDE-native agentic tools (Cursor 18%, Claude Code 10%, Windsurf 5%), and vendor-reported adoption trajectories for these same tools (Section 3.2) show some of the fastest growth curves in the developer-tooling market since the original release of Copilot.

PRODUCTIVITY GAINS AND THE EXPERIENCE PREMIUM

Self-reported productivity benefits are substantial in the aggregate: more than 80% of DORA 2025 respondents indicated that AI enhanced their productivity, and JetBrains data show that nearly nine in ten developers save at least one hour per week, with one in five saving eight hours or more. DORA's telemetry-based analysis found individual-level associations between AI adoption and both 21% more tasks completed and 98% more pull requests merged, though the same report cautions that throughput gains co-occur with increased software-delivery instability, and that time saved during code generation is frequently reallocated to auditing and verification rather than banked as net time savings, a caveat directly relevant to the trust-driven "almost right" frustration discussed in Section 5.

The most methodologically direct evidence on the distribution of these gains comes from Daniotti et al. (2025), who trained a classifier to detect AI-generated Python functions across more than 30 million GitHub commits from 170,000 developers. The study estimates that AI now writes approximately 29% of Python functions among US-based developers, a lead over other countries that is modest and narrowing, and that AI tool use has increased aggregate quarterly coding output by roughly 3.6%. Critically, the same analysis finds that experienced programmers capture nearly all of these productivity and domain-exploration gains: while AI assistance is associated with programmers venturing into unfamiliar areas of software development, this benefit accrues overwhelmingly to already-experienced practitioners, suggesting that generative coding tools may widen rather than narrow existing skill and income gaps within the profession, a finding that recurs, from a different methodological angle, in the skill-formation literature reviewed in Section 7.

These average, largely correlational findings sit in tension with the smaller but more tightly controlled experimental literature reviewed in depth in Section 6, which finds that average productivity effects mask sharply divergent outcomes depending on task type, codebase familiarity, and developer experience. Readers are cautioned against treating the aggregate percentages in this subsection as evidence that AI assistance reliably accelerates all categories of development work; Section 6 shows this is demonstrably not the case for at least one important category, complex maintenance work on unfamiliar, mature codebases.

LEARNING AND COMMUNITY DYNAMICS

AI has become a significant learning resource: 44% of developers who learned new techniques or languages in the past year did so with AI assistance (up from 37%), and 36% specifically learned AI-related coding skills. Simultaneously, continued high reliance on community platforms, Stack Overflow (84%), GitHub (67%), and YouTube (61%), underscores that human-curated knowledge and peer interaction remain essential, particularly given that AI outputs require verification. This coexistence is notable: rather than displacing community knowledge resources, AI adoption appears so far to have layered onto them, consistent with Butler et al.

(2024) workplace finding, discussed in Section 6, that developers have begun using AI assistants as informal substitutes for web search rather than as replacements for human community engagement.

CODE QUALITY, SECURITY, AND TECHNICAL DEBT

A growing body of evidence indicates that the productivity benefits documented above carry measurable quality and security costs. In a controlled study predating the current wave of agentic tools but frequently replicated in spirit since, Perry et al. (2023) found that participants with access to an AI coding assistant (33 treatment participants versus 14 controls, across five security-relevant programming tasks in Python, JavaScript, and C) wrote significantly less secure code than participants without access, with particularly pronounced effects for string encryption and SQL injection tasks. Strikingly, the same participants who wrote less secure code were simultaneously more likely to believe their code was secure than the control group, a calibration failure with direct relevance to the trust paradox discussed in Section 5, since it suggests AI assistance can inflate developer confidence precisely where accuracy is weakest.

At the codebase level, GitClear's analysis of 211 million changed lines of code from 2020 through 2024, drawn from repositories owned by large technology companies and enterprise clients, found that the proportion of changed code that was refactored or moved fell from approximately 25% in 2021 to under 10% in 2024, while copy-pasted code rose over the same period; 2024 was the first year in GitClear's dataset in which copy-pasted code exceeded moved/refactored code, alongside an eightfold increase in duplicated code blocks. GitClear also reports that code churn, the share of new code revised within two weeks of its initial commit, a common proxy for premature or low-quality commits, rose from approximately 3.1% in 2020 to 5.7% in 2024 and 7.1% in 2025, roughly doubling over two years coinciding with the mainstreaming of AI coding assistants. Prior software-engineering literature associates cloned code blocks with 15–50% more defects than non-duplicated code, suggesting a plausible quality-cost pathway, although GitClear's own commercial interest in code-review tooling warrants treating its findings as directionally informative rather than independently audited (see Section 8).

Taken together, the self-reported "almost right" frustration documented by Stack Overflow (Section 5), DORA's finding that generation-time savings are partly offset by increased auditing effort, GitClear's commit-level evidence of rising churn and duplication, and Perry et al.'s experimental evidence of reduced security and inflated confidence together constitute a convergent picture across four independent methodologies, self-report survey, organizational telemetry, large-scale commit mining, and laboratory experiment, that AI-assisted code frequently requires more downstream verification than its generation speed alone would suggest.

THE TRUST PARADOX AND PRIMARY FRUSTRATIONS

Despite accelerating adoption, trust metrics have moved in the opposite direction. On Stack Overflow's direct trust question, confidence in the accuracy of AI tools declined from approximately 40% in prior years to 29% in 2025. A more granular breakdown using the survey's five-point scale shows the gap even more starkly: 46% of respondents reported actively distrusting AI accuracy compared with 33% who reported trusting it, with only 3% reporting that they "highly trust" AI output. Experience moderates this relationship in a counterintuitive direction: experienced developers report the lowest "highly trust" rate (2.6%) and the highest "highly distrust" rate (20%) of any group, suggesting that familiarity with AI tools in professional, accountability-bearing contexts breeds more skepticism, not less, even as those same experienced developers capture most of the measured productivity gains described in Section 4.2.

Positive favorability toward AI fell from 72% (or higher in some prior-year framings) to around 60% over the same period. The dominant frustration, cited by 66% of respondents, is "AI solutions that are almost right, but not quite," which in turn drives the second-most-cited issue: increased time spent debugging AI-generated code, cited by 45% of respondents. Consistent with this, roughly three-quarters of respondents indicate they would still rather consult a human colleague than rely on an AI answer they distrust, and majorities separately report ethical or security concerns about AI-generated code and a desire to fully understand code before shipping it.

This trust gap does not appear to be deterring usage; rather, it is reshaping practice toward "trust but verify" workflows, consistent with DORA's finding that generation-time savings are partly reallocated to verification effort. Daily users tend to report somewhat higher trust than infrequent users, suggesting that familiarity and selective, task-appropriate application can mitigate some concerns even as it does not eliminate them. Nevertheless, the persistence of this gap, and its concentration among the most experienced developers, who are best positioned to judge AI output quality, remains a central and, on current evidence, unresolved challenge for tool designers and engineering leaders.

CONTRADICTORY EVIDENCE: THE PRODUCTIVITY DEBATE

The productivity claims summarized in Section 4.2 rest overwhelmingly on self-report and observational data. A smaller set of randomized controlled trials, the design best suited to isolating AI's causal effect on developer speed, produces a contradictory picture that industry summaries rarely present side by side. juxtaposes the four principal trials identified in this review.

Study	Sample / Design	Task Context	Measured Effect
Peng et al. (2023)	n = 70 professional developers, RCT	Scoped greenfield task (build an HTTP server) in an unfamiliar codebase	−55.8% completion time (faster)
Paradis et al. (2024)	n = 96 Google software engineers, RCT	Complex enterprise task; three AI-assist features (completion, Smart Paste, NL-to-code)	~−21% completion time (faster; wide confidence interval)
Butler et al. (2024)	Multinational company; mixed-methods RCT plus 3-week diary study	Real daily work over a multi-week window	Perceived usefulness/enjoyment increased; trust in AI-generated code unchanged; speed not isolated as a discrete outcome
Becker et al. (2025)	n = 16 experienced open-source developers, 246 tasks, RCT	Complex maintenance work on developers' own mature, familiar codebases	+19% completion time (slower)

Source: Peng et al. (2023), Paradis et al. (2024), Butler et al. (2024), Becker et al. (2025)

The METR trial merits particular attention because it is, on standard evidence-hierarchy grounds, the strongest design in this set applied to the most ecologically realistic task: experienced developers (averaging five years of prior experience on the specific codebases involved) completed real maintenance tasks on projects they already knew, with random assignment to AI access using contemporary tools (primarily Cursor Pro and Claude 3.5/3.7 Sonnet) between February and June 2025. Rather than the anticipated speedup, allowing AI access increased completion time by 19%. The most striking feature of the result is not the direction of the effect but the size of the perception gap surrounding it: before starting, participating developers forecast that AI access would reduce completion time by 24%; after finishing, and despite having just been slower with AI, they estimated that AI had reduced their completion time by 20%. The slowdown persisted across alternative outcome measures, estimators, and data subsets, which the study's authors and independent commentators have both characterized as evidence against the result being a statistical artifact.

This is not straightforwardly reconcilable with Peng et al. (2023) 55.8% speedup or Paradis et al. (2024) 21% speedup by dismissing either result; all three are randomized experiments conducted by credible research teams. The more defensible reading, advanced in preprint form by Farrag (2026) under the label of a "productivity-reliability paradox," is that task type and codebase context function as moderators rather than noise: Peng et al.'s task was a short, well-specified, greenfield exercise in an unfamiliar codebase where AI's knowledge advantage was largest and human context was least valuable; Paradis et al.'s task, while more complex, was still bounded and supported by purpose-built enterprise AI features; whereas METR's task, real maintenance work on codebases the developers already knew intimately, was precisely the setting in which a general-purpose AI assistant's lack of deep, tacit project context would be most costly, and in which experienced developers' own knowledge was most valuable and hardest for AI to substitute or usefully augment. Farrag's synthesis further cites organizational telemetry showing AI adoption associated with 98% more pull requests but substantially longer review times and flat downstream delivery metrics, consistent with DORA's finding (Section 4.2) that generation-time gains are partly consumed by increased verification effort. This reconciling account is plausible and consistent with the full pattern of evidence reviewed here, but it should be treated as an emerging interpretive framework rather than an established finding, since Farrag's paper is a single-author theoretical synthesis that has not, at the time of writing, been independently replicated or peer-reviewed.

A practical implication follows regardless of which theoretical account proves correct: aggregate, cross-task productivity statistics of the kind reported in Section 4.2 and in most industry surveys should not be extrapolated to predict AI's effect on any specific category of work, and claims of uniform productivity benefit are not supported by the strongest available experimental evidence once task complexity and codebase familiarity are taken into account.

COGNITIVE AND SKILL-FORMATION EFFECTS

A further body of recent evidence, largely absent from industry adoption surveys, examines AI assistance's effect on learning and long-term skill formation rather than immediate task speed. Shen and Tamkin (2026), in a randomized study also released as Anthropic research, had 52 mostly junior developers (each with more than one year of Python experience) learn an unfamiliar asynchronous-programming library, either with or without AI assistance, and then complete a 14-question comprehension quiz with

no AI assistance permitted for either group. The hand-coding group averaged 67% on the quiz versus 50% for the AI-assisted group, a 17-percentage-point gap, roughly two letter grades, despite AI assistance producing only a small, not statistically significant, speed advantage during the learning task itself. The same study identified six distinct patterns of AI interaction, three of which involved active cognitive engagement (for example, asking follow-up questions or requesting explanations) and were associated with preserved learning outcomes, while the remaining three involved more passive delegation and were associated with the observed comprehension deficit, indicating that the effect depends on how AI is used, not merely whether it is used.

This finding is consistent with Liu, Christian, Dumbalska, Bakker, and Dubey (2026), whose randomized trials (N = 1,222, spanning mathematical reasoning and reading-comprehension tasks rather than coding specifically, a scope limitation worth noting explicitly) found that AI assistance improved short-term performance but reduced participants' subsequent persistence and impaired their later unassisted performance. The authors characterize current AI systems as "short-sighted collaborators" optimized to provide complete answers immediately, in contrast to human mentorship models that deliberately scaffold learning at the cost of short-term speed. Gerlich (2025), in a mixed-methods study of 666 general-population participants (again not developer-specific), reported a negative correlation between frequent AI tool use and critical-thinking performance, mediated by self-reported cognitive offloading and more pronounced among younger participants, with higher educational attainment associated with better critical-thinking scores regardless of AI use.

These findings complicate two claims made elsewhere in this review. First, they sharpen the experience-premium finding of Section 4.2: if AI assistance both preferentially benefits experienced developers and measurably impairs unfamiliar-skill acquisition among less experienced ones, the two effects are plausibly mutually reinforcing rather than independent, with the least experienced practitioners simultaneously capturing the smallest productivity gains and bearing the largest learning costs. Second, they introduce tension with the onboarding narrative discussed in Section 9, in which new developers increasingly encounter AI tools as default infrastructure from their first days in the profession; the skill-formation evidence reviewed here suggests this default exposure carries a nontrivial risk to foundational skill development unless it is deliberately structured to encourage the more cognitively engaged interaction patterns identified by Shen and Tamkin (2026).

METHODOLOGICAL CONSIDERATIONS ACROSS CITED STUDIES

A review aiming for scientific soundness must evaluate not only what the cited literature finds but how confidently it can be said to find it. Several recurring limitations affect the evidence base synthesized above.

Self-selection in survey samples: Stack Overflow, JetBrains, and DORA all recruit respondents through each organization's own channels rather than through probability sampling of the global developer population. Stack Overflow's own methodology documentation acknowledges that its most highly engaged existing users are likeliest to see and respond to survey prompts. This limits generalizability and may bias sentiment measures (such as trust and favorability) in either direction relative to the broader, less engaged developer population.

Geographic and linguistic skew: the top responding countries across the major surveys are concentrated in North America and Europe, while independent evidence reviewed in Section 3.3 documents substantial infrastructure-, cost-, and language-related barriers to AI adoption in large parts of the Global South. "Global" adoption figures in this literature should be understood as skewed toward wealthier, higher-connectivity markets.

Public-repository bias in platform and commit-mining data: GitHub Octoverse telemetry and academic commit-mining studies such as Daniotti et al. (2025) and Robbes et al. (2026) can only observe public repositories. Private, enterprise, and proprietary codebases, plausibly the majority of commercial software development, are invisible to these methods, and enterprise adoption patterns may differ substantially from open-source patterns because of intellectual-property and compliance constraints on AI tool use.

Correlational versus causal evidence: most adoption, output, and sentiment figures in this review (all four major surveys, GitHub telemetry, and Daniotti et al.'s commit analysis) are observational. Only six studies cited here: Peng et al. (2023), Perry et al. (2023), Paradis et al. (2024), Becker et al. (2025), Shen and Tamkin (2026), and Liu et al. (2026), use random assignment to support causal claims, and even among these, sample sizes are frequently small (ranging from n = 16 to n = 96, with Liu et al.'s N = 1,222 a notable exception), which limits statistical power, and task designs vary widely between toy exercises and real production work, plausibly explaining much of the cross-study heterogeneity documented in Section 6.

Commercial provenance: several key sources: GitHub/Microsoft on Copilot usage, GitClear on code quality (a company that sells code-review analytics), and press-reported figures for Cursor, Claude Code, and Windsurf are published or disclosed by parties with a direct commercial stake in the reported outcome. These sources are retained because no independent alternative exists at comparable scale, but their figures are flagged in-text and weighted below the peer-reviewed and randomized-trial evidence throughout this review.

Field velocity and recency risk: reported adoption figures for individual tools have moved by an order of magnitude within a single year in this literature (for example, Claude Code's reported workplace share rising roughly sixfold between April 2025 and

January 2026), and a separate 2026 survey of web developers specifically found nearly half expressing concern about AI-driven job displacement, a notably more pessimistic figure than Stack Overflow. (2025) sample (36% viewing AI as a threat, discussed in Section 9). Any point-in-time statistic in this domain should be treated as a snapshot subject to rapid change rather than a stable parameter, and figures drawn from different survey populations and question framings are not always directly comparable.

EXPECTATIONS AND NEAR-TERM OUTLOOK

Industry and research forecasts for 2026–2027 emphasize several converging developments, tempered by the contradictory evidence reviewed in Sections 6 through 8. First, agentic systems are expected to expand from the 22–29% early-adopter footprint documented by Robbes et al. (2026) into broader production use, with multi-agent coordination and automated review becoming more common; however, the same study's finding of shallow adoption intensity at the median adopting project suggests this expansion is likely to be gradual and organizationally uneven rather than uniform. Second, verification, security-scanning, and quality-assurance tooling tailored to AI-generated code are anticipated to mature in direct response to the "almost-right" problem documented in Section 5 and the code-quality findings of Section 4.4; DORA's finding that generation-time savings are already being reallocated to auditing suggests this shift is underway rather than merely anticipated. Third, AI is increasingly discussed as infrastructure rather than optional assistance, with new developers encountering AI tools as part of the default onboarding experience, a trend that the skill-formation evidence reviewed in Section 7 suggests warrants deliberate pedagogical design rather than unstructured exposure.

Sentiment regarding employment impact is mixed and, on the most recent evidence, softening. In Stack Overflow. (2025) survey, 64% of developers did not view AI as a threat to their employment, down from 68% in 2024, indicating declining but still majority confidence. A separate, more recent 2026 survey focused specifically on web developers found a substantially less reassured picture, with nearly half of respondents expressing concern about AI-driven displacement; because this survey targets a narrower, potentially more exposed subpopulation and uses different question framing, the discrepancy illustrates how sensitive employment-sentiment figures are to sampling frame and cannot be resolved from the evidence reviewed here. Where forecasts do converge is on the expected nature of role change rather than headcount: greater emphasis on architecture, human oversight, prompt and steering skill, domain expertise, and the ability to evaluate and integrate AI output. Consistent with this, JetBrains found that 68% of developers expect AI proficiency to become a formal job requirement. Headcount reduction is not the dominant forecast in the sources reviewed; instead, most analyses project sustained or growing demand for developers who can effectively govern AI systems while remaining accountable for correctness, security, and maintainability, a framing that aligns closely with DORA's characterization of AI as a "mirror and multiplier" of existing organizational strengths and weaknesses rather than a uniform substitute for engineering judgment.

DISCUSSION

The evidence surveyed here does not support a single-axis narrative of AI as either an unambiguous productivity boon or an overhyped disappointment; it supports a multi-axis picture in which different outcomes carry different levels of evidentiary confidence. Adoption itself is the most strongly established finding in this review: four independent large-sample surveys using different recruitment methods, different populations, and different survey instruments (Stack Overflow, DORA, JetBrains, and GitHub's platform telemetry) converge on adoption rates in the 84–90% range among professional developers, which constitutes unusually strong convergent validity for a social-survey finding and should be treated as close to settled within the limits of the sampling caveats discussed in Section 8.

Short-task, well-specified productivity gains are also reasonably well established: two independent randomized trials Peng et al. (2023), Paradis et al. (2024) found substantial speedups on bounded tasks, consistent with the large self-reported gains in DORA and JetBrains data. Complex-task and maintenance-work productivity is, by contrast, contested: the single most methodologically rigorous trial available Becker et al. (2025) found a significant slowdown on precisely this category of work, and the gap between developers' perceived and measured performance in that trial is large enough to raise a broader caution about relying on self-reported productivity data anywhere in this literature, including in the survey findings reported in Sections 3 and 4.

Code quality and security effects are concerning and moderately well established, with convergent signals across a laboratory experiment Perry et al. (2023), large-scale commit-log analysis (GitClear), and organizational telemetry (DORA's auditing-reallocation finding) despite each individual source carrying its own limitations. Skill-formation effects are concerning but earlier-stage: the relevant studies Shen and Tamkin (2026), Liu et al. (2026), Gerlich (2025) are recent, and two of the three extend beyond coding specifically, so this should be read as an emerging and biologically/cognitively plausible concern rather than a settled finding, pending replication specifically within programming contexts. Finally, the equitable distribution of benefit is a recurring theme across otherwise unrelated methodologies, Daniotti et al.'s commit-level analysis, the trust-by-experience breakdown in Stack Overflow's data, and the regional-adoption evidence in Section 3.3 all independently point toward benefit concentration among already-advantaged practitioners and markets, which is arguably the most consistent cross-cutting finding in the entire literature reviewed here, more consistent even than the productivity findings themselves.

Read together, these strands suggest that AI's effect on software engineering as a practice is not fixed by the technology alone but is substantially conditional on how organizations deploy, govern, and train developers to use it, a conclusion reached independently by DORA's qualitative "mirror and multiplier" framing and by Farrag (2026) more formal productivity-reliability paradox model, despite the two sources using unrelated methods and vocabularies. That convergence, from a commercial telemetry report and a theoretical preprint arriving at structurally similar conclusions, is itself a modest additional piece of evidence in the model's favor, though it does not substitute for the direct empirical replication the model still requires.

LIMITATIONS OF THIS REVIEW

This review is a targeted narrative synthesis, not a formal systematic review or meta-analysis: no PRISMA flow diagram was produced, no dual independent screening or formal risk-of-bias scoring was performed, and no pooled or meta-analytic effect size was computed across the divergent productivity trials in Section 6, since their task designs are too heterogeneous to support meaningful pooling. Source identification relied on targeted search and limited backward citation-following rather than exhaustive, pre-registered database screening across all software-engineering and human-computer-interaction venues, so the academic studies discussed should be understood as illustrative of major findings and live debates rather than a complete census of the relevant literature.

Only English-language sources were reviewed, which may omit relevant regional survey data, particularly from markets outside North America and Europe. A substantial share of the academic sources cited, including the METR trial, Farrag's synthesis, both companion agentic-adoption analyses, and the skill-formation studies, are arXiv preprints that had not, at the time of writing, completed formal peer review; their findings are reported as currently available evidence but could be revised under review. Finally, given the roughly two-year telescoping of the underlying technology documented throughout this review, specific point-in-time figures are likely to date quickly, and the review's own synthesis carries the same recency risk it identifies in the literature it cites (Section 8). No independent statistical reanalysis of any study's underlying microdata was undertaken; all figures are reported as published by the original source.

CONCLUSION

The literature and survey data reviewed here, spanning 2023 to 2026, portray a developer community that has adopted AI rapidly and largely irreversibly, with convergent evidence from four independent large-sample surveys placing usage in the 84–90% range among professional developers and daily use above 50%. This adoption finding is among the most strongly evidenced claims in the social-scientific literature on software engineering practice to date.

What is considerably less settled, despite frequently being reported with similar confidence in industry summaries, is AI's net effect on the profession's core outputs. The strongest available experimental evidence indicates that productivity benefits are real but conditional on task type and codebase familiarity, that code security and maintainability carry measurable costs that are not visible in adoption or favorability statistics, and that skill formation among less experienced developers may be actively harmed by exactly the kind of unstructured, default AI exposure that current onboarding trends are moving toward. The declining trust documented across every major 2025 survey is therefore not merely a lag effect awaiting familiarity, as some industry commentary suggests, but a plausibly rational response to a mixed evidence base, one in which the developers most qualified to judge AI output, the most experienced, express the least confidence in it even as they capture most of its measurable benefit.

Future research should prioritize replication of the small but pivotal randomized trials identified in Section 6, particularly the METR result, across a wider range of organizations, task types, and tools; longitudinal tracking of the skill-formation effects identified in Section 7 as today's AI-assisted novices become tomorrow's senior developers; and closer integration between the industry-survey and peer-reviewed academic literatures, which at present are synthesized together far too rarely given how differently they weight self-report against controlled measurement. For practitioners and organizations, the evidence reviewed here points toward a specific and actionable imperative: treat AI as a capability that must be governed rather than as a uniformly beneficial default. AI has become part of the developer's toolkit; on the evidence assembled here, skilled, verifying human judgment has become more, not less, essential to using it well.

ACKNOWLEDGMENTS

None.

REFERENCES

Anthropic. (2026, February). How AI Assistance Impacts the Formation of Coding Skills. Anthropic Research.

Becker, J., Rush, N., Barnes, E., and Rein, D. (2025). Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity [Preprint]. Arxiv.
Butler, J., Suh, J., Haniyur, S., and Hadley, C. (2024). Dear Diary: A Randomized Controlled Trial of Generative AI Coding Tools in the Workplace [Preprint]. Arxiv.
Center for Strategic and International Studies. (2025). An Open Door: AI Innovation in the Global South Amid Geostrategic Competition.
Daniotti, S., Wachs, J., Feng, X., and Neffke, F. (2025). Who is Using AI to Code? Global Diffusion and Impact of Generative AI [Preprint]. Arxiv.
Farrag, S. E. (2026). The Productivity-Reliability Paradox: Specification-Driven Governance for AI-Augmented Software Development [Preprint]. Arxiv.
Gerlich, M. (2025). AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking. Societies, 15(1), Article 6. https://doi.org/10.3390/soc15010006
GitClear. (2025). AI Copilot Code Quality: 2025 Data Suggests 4x Growth in Code Clones.
GitClear. (2026). The Maintainability Gap: 2026 AI Code Quality Research.
GitHub. (2025). How AI is Reshaping Developer Choice (and Octoverse Data Proves It). The GitHub Blog.
GitHub. (2025, October 28). Octoverse: A New Developer Joins GitHub Every Second as AI Leads Typescript to #1. The GitHub Blog.
Google Cloud DORA. (2025). Balancing AI Tensions: Moving from AI Adoption to Effective SDLC use.
Google Cloud. (2025). How Are Developers Using AI? Inside Google's 2025 DORA report.
JetBrains. (2025). Artificial Intelligence—The State of Developer Ecosystem in 2025.
JetBrains. (2025, October). The State of Developer Ecosystem 2025: Coding in the Age of AI, New Productivity Metrics, and Changing Realities. The JetBrains Blog.
Liu, G., Christian, B., Dumbalska, T., Bakker, M. A., and Dubey, R. (2026). AI Assistance Reduces Persistence and Hurts Independent Performance [Preprint]. Arxiv.
New America. (2025). Closing the Adoption Gap: How AI is Being Built in the Global South.
Paradis, E., Grey, K., Madison, Q., Nam, D., Macvean, A., Meimand, V., Zhang, N., Ferrari-Church, B., and Chandra, S. (2024). How Much Does AI Impact Development Speed? An Enterprise-Based Randomized Controlled Trial [Preprint]. Arxiv.
Peng, S., Kalliamvakou, E., Cihon, P., and Demirer, M. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot [Preprint]. arXiv.
Perry, N., Srivastava, M., Kumar, D., and Boneh, D. (2023). Do Users Write More Insecure Code with AI Assistants? In Proceedings of the 2023 ACM Sigsac Conference on Computer and Communications Security (2785–2799). Association for Computing Machinery. https://doi.org/10.1145/3576915.3623157
Robbes, R., Matricon, T., Degueule, T., Hora, A., and Zacchiroli, S. (2026). Agentic Much? Adoption of Coding Agents on GitHub. ACM Transactions on Software Engineering and Methodology. Advance online publication. https://doi.org/10.1145/3822180
Shen, J. H., and Tamkin, A. (2026). How AI Impacts Skill Formation [Preprint]. arXiv.
Stack Overflow. (2025a). 2025 Stack Overflow Developer Survey.
Stack Overflow. (2025b). AI | 2025 Stack Overflow Developer Survey.
Stack Overflow. (2025c, December 29). Developers Remain Willing But Reluctant to use AI: The 2025 Developer Survey Results Are Here. Stack Overflow Blog.
Stack Overflow. (2025d). Methodology | 2025 Stack Overflow Developer Survey.
Stack Overflow. (2025e, October 23). What Leaders Need to Know from the 2025 Stack Overflow Developer Survey. Stack Overflow Blog.
Stack Overflow. (2026). Stack Overflow's 2025 Developer Survey Reveals Trust in AI At An All-Time Low [Press Release].
Stanford Institute for Human-Centered Artificial Intelligence. (2026). The 2026 AI Index Report: Economy. Stanford HAI.
The Register. (2026, May 21). Web Devs Sleeping with the Enemy: AI is Doing Their Job and they Worry It's After Their Desk Too.