AI Coding Assistant Reliance, Verification Discipline, and Skill Development Among Developers

Deepti Anand[1], Maleeta Angelina Goddard[2], Harsh Tripathi[1], and Ashok Patel[1]

[1]IDCONS TECHNOVA PRIVATE LIMITED (ITPL), New Delhi, India

[2]IDC India — Innovation Development Council, New Delhi, India

Author Note

Deepti Anand https://orcid.org/0009-0003-7685-466X

Maleeta Angelina Goddard https://orcid.org/0009-0006-8540-9498

Harsh Tripathi https://orcid.org/0009-0006-5159-9705

Ashok Patel https://orcid.org/0009-0007-7211-7581

We have no conflicts of interest to disclose.

Author Contributions: D.A. (Director, ITPL) supervised the study and contributed the behavioural-economics framing. H.T. and A.P. (Software Developers, ITPL) identified the underlying problem, developed the survey instrument, and collected the data. M.A.G. (Editor,

IDC India Innovation Development Council) reviewed the manuscript for editorial and academic standards. All authors approved the final manuscript.

Correspondence concerning this article should be addressed to IDC India- Innovation Development Council, 5th Floor, Wing A, Statesman House, Barakhamba Road, Connaught Place, New Delhi – 110001, India. Email: editorial@idcijst.com

This manuscript is prepared for submission to IDCIJST — Innovation Development Council's International Journal of Science and Technology, ISSN (Online): 3139-7395, published by IDC India Innovation Development Council, New Delhi.

Abstract

AI coding assistants such as ChatGPT, Claude, GitHub Copilot, and Cursor have become standard infrastructure in software development, raising concern that developers, particularly early in their careers, may be trading verification discipline and fundamental skill development for short-term productivity. This study surveyed 24 developers across student, early-career, mid-level, and senior levels to examine the statistical relationship between AI reliance, verification discipline, and self-reported debugging skill trends. Composite scores were constructed for AI Reliance, Verification Discipline, and Skill Confidence, and analysed using Spearman rank correlation, Cronbach's alpha for internal consistency, and ordinary least squares (OLS) regression. Reliance intensity did not differ significantly across experience levels ($\rho = 0.24$, $p = .256$), whereas verification discipline rose sharply and significantly with experience ($\rho = 0.57$, $p = .003$). In a multivariate regression controlling for reliance and experience, verification discipline was the only significant predictor of a positive debugging-skill trend ($\beta = 0.84$, $p = .011$; model $R^2 = 0.50$). Developers who reported bypassing an independent attempt and asking AI immediately showed the weakest verification discipline and flattest skill trend of any behavioural group. These findings, read alongside existing evidence on automation bias, cognitive offloading, and causal experimental work on AI-induced skill impairment, suggest that the operative risk factor is the absence of a verification habit rather than AI usage volume itself. The paper closes with a conceptual framework, the statistical formulae used, and practical

implications for individual practice, education, and organisational policy, alongside the limitations of a small, self-selected, self-report sample.

Keywords: AI coding assistants, developer skill development, verification discipline, automation bias, cognitive offloading, bounded rationality, human-AI collaboration, software engineering education

Convenience or Competency? AI Coding Assistant Reliance, Verification Discipline, and Skill Development Among Developers

AI coding assistants have moved from novelty to default infrastructure in professional and academic software development within a few years. Adoption figures reflect this shift starkly: in a global survey of over 49,000 developers, 84% reported using or planning to use AI coding tools, up from 76% the previous year (Stack Overflow, 2025). In a controlled field experiment at Microsoft, Accenture, and a Fortune 100 company, developers with access to an AI pair programmer completed a defined task 55.8% faster than a control group, with the largest gains observed among less experienced developers (Peng, Kalliamvakou, Cihon, & Demirer, 2023). Yet the same Stack Overflow (2025) survey recorded a sharp decline in trust: only 29% of developers said they trusted the accuracy of AI output, down from 43% the year before, and experienced developers showed the lowest rate of high trust and the highest rate of high distrust in the sample. This combination, rising use alongside falling trust, is the empirical starting point for the present study.

The concern motivating this research is twofold. First, AI-generated code is not verified against ground truth; it is generated to be statistically plausible, which means it can be fluent, idiomatic, and confidently presented while still being wrong. Second, developers, especially those early in their careers, may increasingly route around the foundational work of understanding syntax, debugging logic, and system design by delegating this reasoning to an AI

assistant rather than building the underlying competency themselves. Where these two risks compound is where they matter most: a developer who has not built the competency to recognise flawed output is also the developer least equipped to catch it.

Existing discussion of AI coding assistants, however, tends to treat these two threads separately. One line of work documents reliability failures in AI-generated code. Spracklen et al. (2025) analysed 2.23 million code samples across sixteen commercial and open-source code-generating models and found that 19.7% of recommended packages did not exist, a phenomenon since termed “slopsquatting” because attackers can register the hallucinated package names to distribute malware. A separate line of work documents productivity gains (Peng et al., 2023) and adoption trends (Stack Overflow, 2025). Comparatively little structured, quantitative attention has examined the intersection of the two constructs within a working developer population, across experience levels, using formal statistical methods. This paper reports such a study.

The primary research question is: how can developers and organisations use AI coding assistants to improve productivity while preserving fundamental programming skills, independent problem-solving, debugging ability, and real-world developer experience? Rather than assume the answer, this paper treats it as an empirical question, testable through composite psychometric scores, correlation, and regression, and reports what a working sample of developers actually demonstrated about their own habits, confidence, and skill trajectories.

Literature Review

Reliability of AI-Generated Code

Spracklen et al. (2025), in a study presented at the USENIX Security Symposium, generated 2.23 million code samples across sixteen widely used code-generating models and found that 19.7% contained at least one hallucinated, non-existent package reference. Open-source models hallucinated at more than four times the rate of commercial models (21.7% versus 5.2%). Critically, when identical prompts were rerun ten times, 43% of hallucinated package names reappeared in every run, indicating the failure mode is persistent and exploitable rather than random noise. This finding establishes, independent of any individual developer's skill or diligence, that AI-generated code carries a measurable and non-trivial reliability gap.

Developer Trust and Verification Behaviour

Stack Overflow's (2025) Developer Survey (n ≈ 49,000) found that despite record adoption (84%), only 33% of developers reported trusting AI output's accuracy, and 46% actively distrusted it. The top-cited frustration (66%) was AI output that is “almost right, but not quite,” and 45% reported that debugging AI-generated code takes longer than writing it independently would have. Notably, the survey found experienced developers were the most cautious group, the lowest share of “highly trust” responses and the highest share of “highly distrust” responses, suggesting that scepticism toward AI output tracks experience rather than disappearing with it.

Productivity Evidence

Peng et al. (2023) conducted a randomised controlled experiment across Microsoft, Accenture, and an anonymous Fortune 100 company, finding that developers using GitHub Copilot completed a defined task 55.8% faster than a control group, with less experienced developers showing higher adoption rates and greater productivity gains. This establishes that the productivity case for AI coding assistants is not merely anecdotal; it is measured and substantial, which is precisely why the accompanying reliability and skill-development risks warrant equally rigorous study rather than dismissal.

Automation Bias and Complacency

Parasuraman and Manzey (2010), reviewing decades of human-factors research, developed an integrated model of automation-induced complacency and automation bias, defining complacency as reduced monitoring of an automated system under multiple-task load, and automation bias as the tendency to accept an automated recommendation without adequately verifying it against other available information. Their review found automation complacency occurs in both novice and expert operators and is not reliably eliminated by practice alone, which frames a testable expectation for the present study: verification behaviour around AI output should not be assumed to develop automatically with mere exposure, but instead requires

distinct, deliberate practice or organisational enforcement. Lee and See (2004) extended this line of work with the concept of trust calibration, arguing that appropriate reliance on an automated system requires a user's trust to track the system's actual, rather than perceived, capability; miscalibration in either direction, overtrust or undertrust, produces misuse or disuse of the tool respectively.

Cognitive Offloading and Skill Persistence

Risko and Gilbert (2016) defined cognitive offloading as the use of an external tool or action to reduce the information-processing demands of a task, and reviewed evidence that increased offloading is associated with degraded unaided performance once the external aid is withdrawn. Recent causal experimental evidence extends this concern directly to AI assistance. Liu, Christian, Dumbalska, Bakker, and Dubey (2026), across a series of randomised controlled trials (N = 1,222) spanning mathematical reasoning and reading comprehension tasks, found that although AI assistance improved performance during assisted sessions, participants who had used the AI performed significantly worse than controls once the assistance was withdrawn, and gave up more quickly. These effects emerged after as little as ten to fifteen minutes of AI-assisted work, which the authors attribute to reduced persistence: AI conditions users to expect an immediate, complete answer, denying them the experience of productively struggling with a problem, a process understood in the learning sciences to be foundational to durable skill acquisition.

Evidence from Computing Education

Parallel concerns have emerged directly within programming education. Zhai, Wibowo, and Li (2024), in a systematic review, found consistent evidence that over-reliance on AI dialogue systems is associated with diminished critical thinking and problem-solving among students. Marouf, Kotorov, Krasylnykova, and Aslam (2025) surveyed students and instructors in computer science higher education and reported persistent concern, from both groups, that AI tool use complicates the verification of genuine student learning. Jeilani and Yusuf (2025) found that over-reliance on AI in educational settings was associated with negative learning outcomes, an effect moderated by institutional policy and students' own ethical concern about the practice, a finding directly relevant to this study's later discussion of organisational policy as an intervention point. Xue, Chen, Bai, Tairas, and Huang (2024) experimentally examined introductory programming students using ChatGPT and found genuine short-term completion benefits alongside evidence that certain forms of unstructured use bypassed the productive struggle understood to build debugging competency.

A Behavioural Economics Lens: Bounded Rationality and Satisficing

The pattern described above can also be framed through the economics of decision-making under limited cognitive resources. Simon (1955) proposed that decision-makers are boundedly rational: rather than exhaustively optimising, they satisfice, accepting the first alternative that clears a good-enough threshold, because complete evaluation of every alternative exceeds the time, attention, and information available to a real decision-maker. Verifying AI-generated code against documentation and edge cases is, in this framing, a costly information-search activity; a developer under time pressure who has already received an output that appears syntactically correct has, in Simon's terms, met a satisficing threshold and has an economic incentive to stop searching, whether or not the output is actually correct. This reframes the reliability gap documented above not simply as a technical failure of the AI model, but as a predictable outcome of a rational, boundedly-informed developer operating under time constraints, absent an external reason (organisational policy, training, or reputational cost) to raise the verification threshold. This economic lens complements, rather than replaces, the automation bias and cognitive offloading explanations above; the three together generate the hypotheses tested later in this paper.

Synthesis and Gap

Taken together, the literature establishes three separate facts firmly: AI-generated code carries a measurable and persistent reliability risk (Spracklen et al., 2025); developer trust in that output is falling even as usage rises (Stack Overflow, 2025); and controlled experimental evidence shows AI assistance can cause measurable, if short-term, impairment of independent

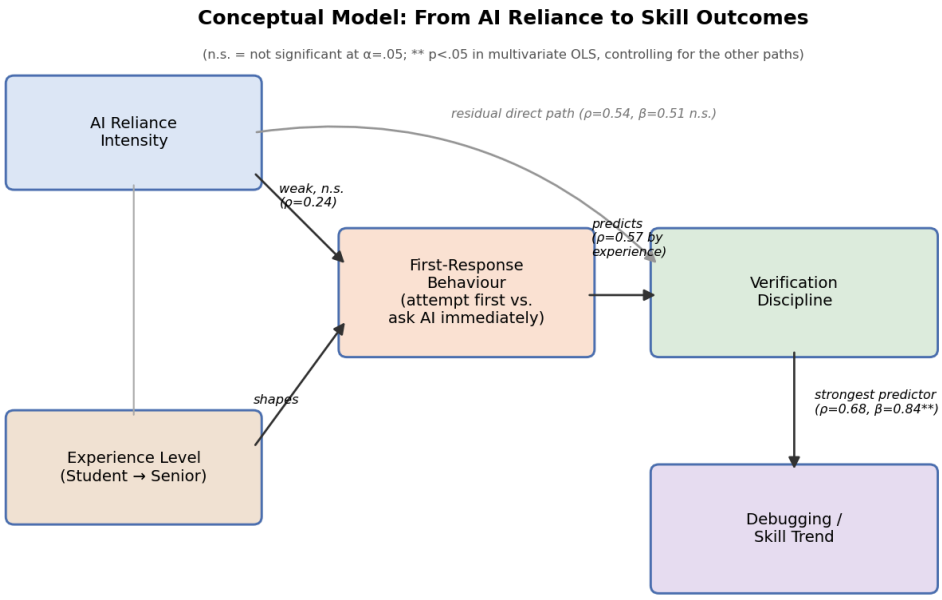
performance and persistence (Liu et al., 2026). What remains comparatively underexamined is whether these patterns hold, and in what form, among working developers across a real range of experience levels, using composite, formally validated (or explicitly flagged as exploratory) measurement scales, and whether verification discipline, rather than reliance volume, is the variable that actually predicts skill outcomes. This is the gap the present study's survey and statistical analysis addresses.

Conceptual Framework

Building on the constructs reviewed above, this study models AI Reliance Intensity and Experience Level as antecedents of First-Response Behaviour (whether a developer attempts a problem, consults documentation, or asks a colleague before turning to AI, versus asking AI immediately), which in turn is expected to shape Verification Discipline, which is hypothesised to be the proximate predictor of a developer's Debugging/Skill Trend. Figure 1 presents this framework together with the strength and direction of each path as estimated later in this paper.

Figure 1

Conceptual Model — From AI Reliance to Skill Outcomes



*Note. n.s. = not significant at α = .05; ** p < .05 in the multivariate OLS regression reported in the Results section, controlling for the other paths.*

Method

Design and Sample

This study used a structured, self-administered online survey distributed to a convenience sample of developers, comprising 21 items covering demographic and experience background, AI usage intensity, verification behaviour, self-assessed skill confidence and debugging trend, and openness to structured workflow interventions. Twenty-four respondents completed the survey: six students, ten early-career developers (0–2 years), four mid-level developers (2–5

years), and four senior developers (5+ years). All respondents reported using AI coding assistants in their regular workflow.

Composite Measures and Internal Consistency

Three composite scores were constructed from the ordinal survey items to summarise the underlying constructs of interest. For a composite built from k items, each rescaled to a common 0–4 range, the composite score for respondent i is the item mean:

$$C_i = \frac{1}{k}\sum_{j=1}^{k} x_{ij}$$

where x_{ij} is respondent i's rescaled response to item j, and k is the number of items in that scale.

The AI Reliance Score (range 0–4) was built from three items: frequency of AI use, proportion of shipped code originating from AI suggestions, and change in AI usage over the past year. The Verification Discipline Score (range 0–4) was built from three items: frequency of checking AI output against documentation, frequency of testing against edge cases, and whether AI's bug explanations are treated as confirmed fact or as a hypothesis to verify. The Skill Confidence Score (range 1–5) was built from two items: confidence solving a recent non-trivial problem without AI, and confidence explaining AI-generated code line by line. A separate single item, Debugging Trend (coded −2 to +2), captured each respondent's self-assessed change in independent debugging ability since adopting AI regularly.

Internal consistency of each multi-item composite was assessed with Cronbach's alpha:

$$\alpha = \frac{k}{k-1}\left(1 - \frac{\sum \sigma^2_{item}}{\sigma^2_{total}}\right)$$

where k is the number of items, σ^2 item is the variance of an individual item across respondents, and σ^2 total is the variance of the summed composite score. Reliability was acceptable for the Verification Discipline Score ($\alpha = 0.57$) and the Skill Confidence Score ($\alpha = 0.58$), though both fall below the conventional $\alpha \geq 0.70$ threshold for a fully validated scale, and was poor for the AI Reliance Score ($\alpha = 0.04$). This is reported plainly as a limitation: the Reliance Score's three items (frequency, proportion of AI-originated code, and year-over-year change) evidently measure related but distinct facets of AI use rather than a single unidimensional construct at this sample size, and findings involving that score should be read as more provisional than those involving Verification Discipline or Skill Confidence.

Statistical Approach

Given the small, non-random sample, the primary bivariate analysis used Spearman's rank correlation coefficient, appropriate for ordinal and non-normally distributed composite scores:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

where d_i is the difference between the ranks of the two variables for respondent i, and n is the sample size.

To test whether Verification Discipline predicted the Debugging Trend independently of AI Reliance and Experience Level, a multivariate ordinary least squares regression was estimated:

$$DebugTrend_i = \beta_0 + \beta_1 \cdot Reliance_i + \beta_2 \cdot Verification_i + \beta_3 \cdot Experience_i + \varepsilon_i$$

where β_0 is the intercept, β_1, β_2, and β_3 are the estimated coefficients for AI Reliance Score, Verification Discipline Score, and Experience Level (coded 0 = Student to 3 = Senior) respectively, and ε_i is the residual error term for respondent i. The model was fitted by ordinary least squares with n = 23 (one respondent's Debugging Trend was coded “not sure” and excluded listwise).

Results

Table 1 presents descriptive statistics for the three composite scores and the Debugging Trend variable across the full sample.

Table 1

Descriptive Statistics for Composite Measures (n = 24, except Debugging Trend n = 23)

Measure	Range	M	SD	Min	Max
AI Reliance	0–4	2.85	0.69	0.89	3.70
Verification	0–4	2.42	0.87	1.00	4.00

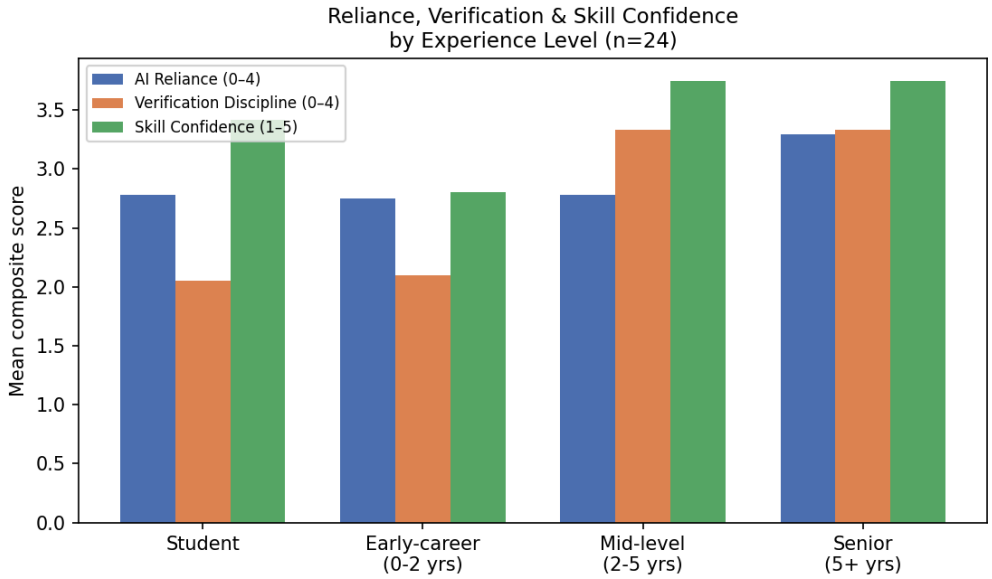
Confidence	1–5	3.35	0.71	1.50	4.50
Debug Trend	−2 to +2	0.70	1.29	−2.00	2.00

AI reliance is high and does not differ significantly by experience level. Mean AI Reliance Scores were broadly similar across students (2.78), early-career developers (2.75), mid-level developers (2.78), and senior developers (3.30); the correlation between experience level and reliance was not statistically significant ($\rho = 0.24$, $p = .256$, $n = 24$). In this sample, heavy AI use was not confined to junior developers; it was close to universal.

Verification discipline scales clearly with experience. Mean Verification Discipline Scores were markedly lower among students (2.06) and early-career developers (2.10) than among mid-level (3.33) and senior developers (3.33), a statistically significant relationship ($\rho = 0.57$, $p = .003$). Figure 2 presents these composite means by experience level.

Figure 2

Reliance, Verification, and Skill Confidence by Experience Level



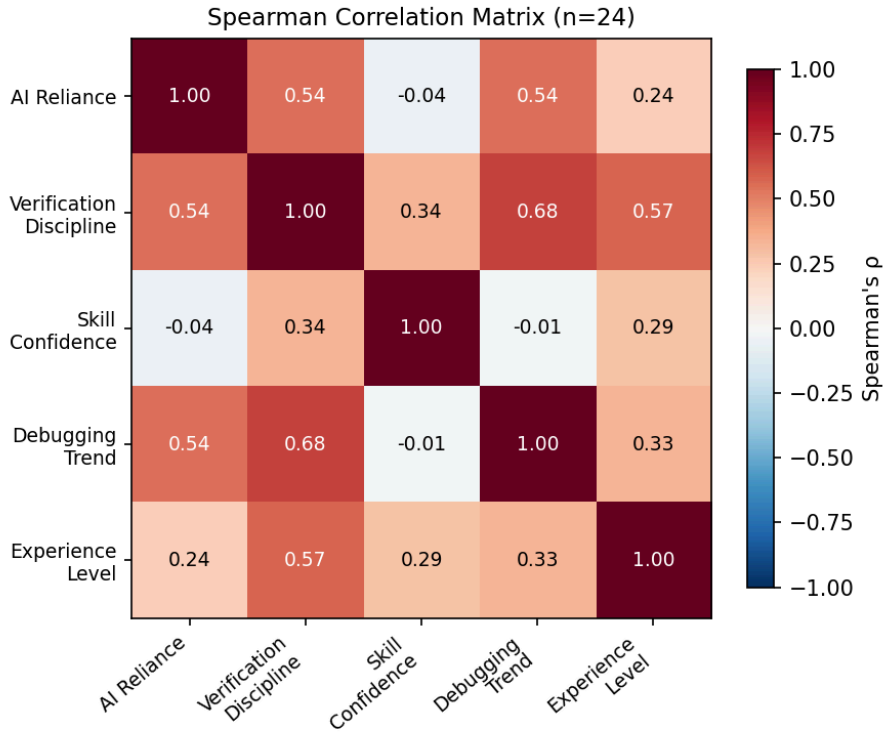
Note. N = 24. Bars show mean composite scores by self-reported experience level.

Reliance and verification are positively, not negatively, correlated. AI Reliance Score and Verification Discipline Score were positively correlated across the whole sample ($\rho = 0.54$, $p = .006$); developers who used AI more intensively were also, on average, somewhat more likely to verify it, contrary to a simple displacement hypothesis in which more AI use directly substitutes for checking behaviour.

Verification discipline is the strongest bivariate predictor of a positive skill trend. Verification Discipline Score correlated strongly with the self-reported Debugging Trend ($\rho = 0.68$, $p < .001$). AI Reliance Score also correlated positively with Debugging Trend ($\rho = 0.54$, $p = .007$). Figure 3 presents the correlation matrix across all study variables, and Figure 4 presents the bivariate relationship between Verification Discipline and Debugging Trend directly.

Figure 3

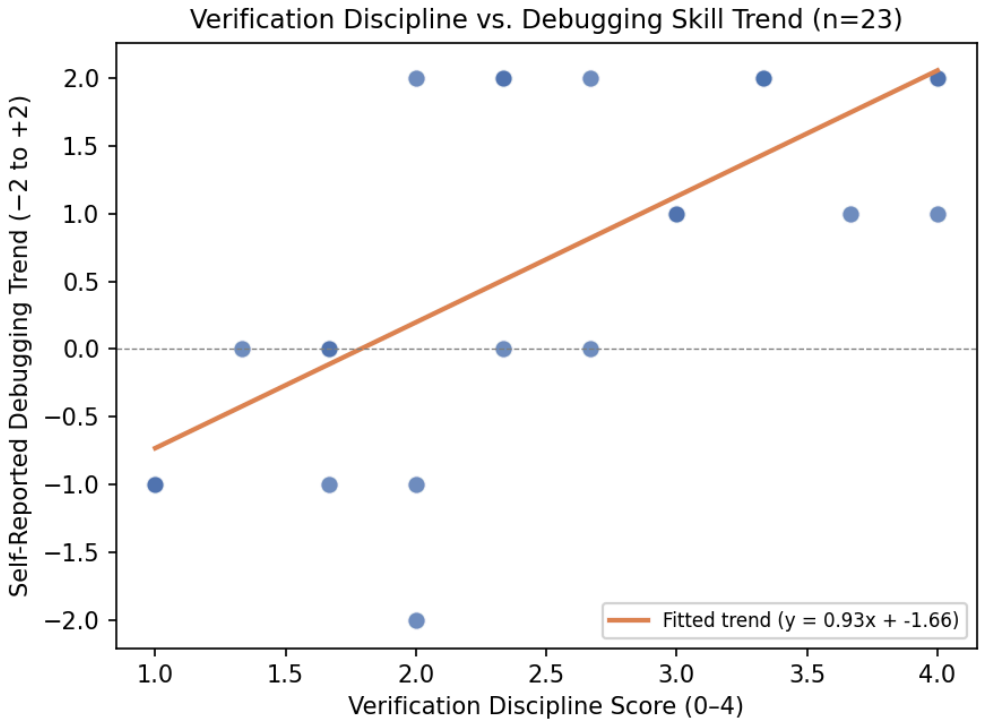
Spearman Correlation Matrix



Note. N = 24. Values are Spearman's rho.

Figure 4

Verification Discipline versus Debugging Skill Trend



Note. N = 23. Line shows the unweighted linear fit.

In a multivariate model, verification discipline remains the only significant predictor. The OLS regression specified above (n = 23) returned a significant overall model, $R^2 = 0.50$, $F(3, 19) = 6.29$, $p = .004$. Table 2 reports the coefficient estimates.

Table 2

OLS Regression Predicting Debugging Trend (n = 23)

Predictor	**β**	**SE**	**t**	**p**
Intercept	−2.82	1.11	−2.53	.020
AI Reliance	0.51	0.42	1.21	.242
Verification	0.84	0.30	2.81	.011

Experience Level	−0.09	0.24	−0.39	.705

Once Verification Discipline is included in the model, neither AI Reliance Score nor Experience Level remains an independent significant predictor of Debugging Trend; the effect of experience on skill outcomes appears to operate through its effect on verification discipline, rather than directly.

The clearest risk marker is a behavioural one, not a volume one. When grouped by stated first response to a coding problem, the six respondents who reported asking AI immediately, skipping an independent attempt, showed the lowest mean Verification Discipline Score (2.00) and the flattest mean Debugging Trend (+0.17) of any behavioural group, lower than the "ask a colleague first" group (+2.00, n = 2), the "check documentation first" group (+1.50, n = 4), and the much larger "try myself first, then ask AI" group (+0.45, n = 12).

Self-reported negative outcomes were present but not concentrated among juniors. Eight of 24 respondents (33%) reported having shipped AI-generated code that later proved incorrect, insecure, or broken, and six (25%) reported being unable to fix a bug or explain their own code in a review or interview because it originated from AI. These incidents were distributed across experience levels rather than concentrated among students or early-career developers, though the sample is too small to draw a firm conclusion on this point.

Openness to structural intervention was high. Nineteen of 24 respondents (79%) said they would follow a structured "attempt first, check documentation, ask AI, review and test"

workflow if their team required it, or said it would depend on the situation. Twenty-one of 24 (88%) supported or were open to a mandatory "AI-assisted code" flag in code review for additional scrutiny.

Discussion

The most striking finding in this sample is a reframing of the original hypothesis rather than a confirmation of it. Both AI Reliance and Verification Discipline correlated positively, not negatively, with self-reported skill improvement, and reliance intensity did not differ meaningfully by experience level. What did differ sharply by experience, and what emerged as the only significant predictor once all three variables were modelled jointly, was verification discipline itself.

This pattern is broadly consistent with the automation bias and trust-calibration literature reviewed earlier. Parasuraman and Manzey's (2010) finding that automation complacency is not reliably resolved by mere practice, and Lee and See's (2004) argument that appropriate reliance depends on deliberately calibrating trust to actual system capability, together predict exactly the pattern observed here: exposure to AI (reliance) alone does not produce the safeguard; a distinct, separately cultivated verification habit does, and that habit appears to be built through experience, training, or organisational requirement rather than through usage volume alone. Viewed through Simon's (1955) bounded-rationality lens, this is unsurprising: verification is a

costly search activity, and a developer under time pressure has a rational, if myopic, incentive to satisfice once an output looks plausible, unless an external structure (a colleague's example, a code-review policy, or explicit training) raises the cost of not verifying.

A genuine tension exists, however, between this study's correlational finding and the causal experimental evidence reviewed earlier. Liu et al. (2026) found that even brief (ten- to fifteen-minute) AI-assisted sessions caused measurable impairment of independent performance once assistance was withdrawn. This study's cross-sectional design cannot establish whether the developers who report high reliance and high verification simply have more experience overall (a selection effect, consistent with the significant experience-verification correlation reported above), whether real-world, sustained workplace practice differs meaningfully from the brief, single-session laboratory tasks Liu et al. studied, or whether a genuine short-term skill cost coexists with a longer-term adaptive response in which developers who remain in the field learn to compensate through verification. This study's data cannot adjudicate between these explanations, and this is presented as an open question rather than resolved.

The behavioural finding, that respondents who skip an independent attempt and ask AI immediately show the weakest verification discipline and the flattest skill trend, is the most directly actionable result. It suggests interventions should target the sequence of the workflow rather than AI usage volume: whether a developer attempts the problem, or at least consults documentation, before turning to AI, and whether they verify what AI returns before shipping it. This is consistent both with the reliability risk documented independently by Spracklen et al. (2025), roughly one in five AI-suggested packages did not exist in their study, and with the

computing-education evidence (Zhai et al., 2024; Jeilani & Yusuf, 2025) that institutional policy and enforced structure, not developer goodwill alone, moderate whether over-reliance translates into a negative outcome.

The high openness to structured workflows and mandatory review flags (79% and 88% respectively) suggests this is not a population resistant to guardrails; several respondents linked their own willingness to verify more directly to "team or company guidelines" and "code-review requirements" in their open-text responses, echoing Jeilani and Yusuf's (2025) finding that institutional policy moderates the over-reliance effect.

These findings carry several limitations that temper their interpretation. The sample is small (n = 24), self-selected, and self-reported; the composite scores were constructed for this study and are not drawn from previously validated psychometric instruments, and the AI Reliance Score in particular showed poor internal consistency ($\alpha = 0.04$). The design is cross-sectional, so the direction of causality between AI reliance, verification discipline, and skill trend cannot be established from these data alone; it remains entirely possible that AI reliance impairs skill in the short term even where this study's real-world sample of working developers shows a positive association, as the tension with Liu et al. (2026) discussed above illustrates. These findings should therefore be read as hypothesis-generating for a larger, ideally task-based, follow-up study rather than as confirmatory evidence of any causal claim.

A natural direction for future research is a controlled follow-up study directly modelled on Liu et al.'s (2026) design: a within-subject task experiment comparing an independent-attempt-first condition against an AI-first condition, measuring error-catch rate and

post-task explanation ability, which would allow the causal direction between verification discipline and skill development to be tested directly rather than inferred from cross-sectional correlation. A larger sample would also permit the AI Reliance construct to be re-specified or re-measured with items that cohere more strongly as a single scale.

Conclusion

This exploratory, mixed-concept study of 24 developers found that AI reliance is now widespread and largely uniform across experience levels, but verification discipline is not: it is markedly lower among students and early-career developers, and it is verification discipline, not AI usage volume, that remains the sole significant predictor of a positive debugging-skill trend

once modelled jointly in a multivariate regression ($\beta = 0.84$, $p = .011$). The clearest behavioural risk signal identified was bypassing an independent attempt altogether: developers who reported asking AI immediately showed both the weakest verification habits and the flattest skill trends in the sample. Read alongside established theory on automation bias (Parasuraman & Manzey, 2010), trust calibration (Lee & See, 2004), cognitive offloading (Risko & Gilbert, 2016), and bounded rationality (Simon, 1955), and against causal experimental evidence that brief AI assistance can impair independent performance (Liu et al., 2026), this reframes the practical question from how to reduce AI use to how to build and enforce a verification habit around AI use, a distinction with direct implications for onboarding, code-review policy, and training, and one the surveyed developers themselves appear largely willing to accept. Given the small, self-selected sample and the modest reliability of the AI Reliance Score in particular, these findings should be treated as hypothesis-generating; a larger, task-based, ideally randomised follow-up study is needed to test the causal direction between verification discipline and skill development directly.

References

Jeilani, A., & Yusuf, M. (2025). The role of over-reliance on AI in the negative consequences of student learning: The moderating effects of ethical concerns and institutional policies. *Cogent Education.* https://doi.org/10.1080/2331186X.2025.2591503

Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors, 46*(1), 50–80. https://doi.org/10.1518/hfes.46.1.50_30392

Liu, G., Christian, B., Dumbalska, T., Bakker, M. A., & Dubey, R. (2026). AI assistance reduces persistence and hurts independent performance. *arXiv preprint arXiv:2604.04721.*

Marouf, R., Kotorov, I., Krasylnykova, Y., & Aslam, H. (2025). Evidence of learning in the presence of AI tools in computer science higher education. In G. Jezic et al. (Eds.), *Agents and multi-agent systems: Technologies and applications 2024* (Smart Innovation, Systems and Technologies, Vol. 406). Springer. https://doi.org/10.1007/978-981-97-6469-3_28

Parasuraman, R., & Manzey, D. H. (2010). Complacency and bias in human use of automation: An attentional integration. *Human Factors, 52*(3), 381–410. https://doi.org/10.1177/0018720810376055

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of AI on developer productivity: Evidence from GitHub Copilot. *arXiv preprint arXiv:2302.06590.*

Risko, E. F., & Gilbert, S. J. (2016). Cognitive offloading. *Trends in Cognitive Sciences, 20*(9), 676–688. https://doi.org/10.1016/j.tics.2016.07.002

Simon, H. A. (1955). A behavioral model of rational choice. *The Quarterly Journal of Economics, 69*(1), 99–118.

Spracklen, J., Wijewickrama, R., Sakib, A. H. M. N., Maiti, A., Viswanath, B., & Jadliwala, M. (2025). We have a package for you! A comprehensive analysis of package hallucinations by code-generating LLMs. *Proceedings of the 34th USENIX Security Symposium.*

Stack Overflow. (2025). *2025 Stack Overflow Developer Survey.* https://survey.stackoverflow.co/2025/

Xue, Y., Chen, H., Bai, G. R., Tairas, R., & Huang, Y. (2024). Does ChatGPT help with introductory programming? An experiment of students using ChatGPT in CS1. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training* (pp. 331–341). https://doi.org/10.1145/3639474.3640076

Zhai, C., Wibowo, S., & Li, L. D. (2024). The effects of over-reliance on AI dialogue systems on students' cognitive abilities: A systematic review. *Smart Learning Environments, 11*, Article 28. https://doi.org/10.1186/s40561-024-00316-7