

Does Working with AI Agents Change How Developers Code, Test, and Review?

Muhammad Hamza  

Department of Software Engineering, LUT University, Lahti, Finland

Dominik Siemon  

Department of Software Engineering, LUT University, Lahti, Finland

Wardah Naeem Awan  

Department of Software Engineering, LUT University, Lahti, Finland

Abstract

AI coding agents such as GitHub Copilot and Claude Code are increasingly contributing pull requests (PRs) autonomously across large-scale software repositories. While prior research has primarily investigated the quality and acceptance of agent-generated artifacts, less is known about how sustained exposure to AI agents influences developers' own practices. Drawing on automation complacency theory, we investigate whether agent adoption is associated with changes in coding effort, testing discipline, and code review behavior. We conduct a longitudinal study of 669 developers across 103 repositories using the AIDev dataset and analyze behavioral changes before and after agent adoption across 11 metrics. To distinguish agent-associated effects from broader ecosystem trends, we complement within-subject analyses with a difference-in-differences (DiD) design using 228 control developers from repositories without agent activity. We find one large agent-associated effect: a 76.8% increase in PR description length ($r = 0.561$), alongside consistent but smaller improvements in testing discipline across all three testing measures. Coding effort remains stable. Although review behavior changes noticeably – reviews become faster, less detailed, and more permissive – these patterns are also observed in control repositories and therefore appear to reflect broader ecosystem trends rather than agent adoption. Overall, we find no evidence that developers reduce productive effort when working with AI agents. Instead, developers increase documentation and testing activity, while apparent declines in review thoroughness are better explained by wider changes in software development practice. These findings not only clarify how developers adapt to AI coding agents but also demonstrate how comparative causal designs help distinguish agent-associated behavioral changes from broader ecosystem trends.

2012 ACM Subject Classification Software and its engineering → Software creation and management; Computing methodologies → Artificial intelligence

Keywords and phrases AI Coding Agents, Automation Complacency, Developer Behavior, Pull Requests, Difference-in-Differences, Empirical Software Engineering, Longitudinal Study

Digital Object Identifier 10.4230/LIPICs.ESEM.2026.67

Category Emerging Results, Vision & Reflection Track Paper

Funding This research was fully funded by the Managed AI-powered Software Innovations (MAISA) project (Project No. 23B3C109YT10).

1 Introduction

AI coding agents such as GitHub Copilot, Cursor, and Claude Code are increasingly becoming part of everyday software development workflows. Beyond generating code suggestions, these agents now act as semi-autonomous collaborators that submit pull requests (PRs), generate tests, and respond to developer feedback [11, 10]. Large-scale datasets such as AIDev report hundreds of thousands of agent-authored contributions across diverse repositories [11], suggesting that AI agents are no longer experimental tools but an emerging



© Muhammad Hamza, Dominik Siemon, and Wardah Naeem Awan;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;

Article No. 67; pp. 67:1–67:13



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

part of contemporary software engineering practice. As AI agents become increasingly integrated into development practice, an important question emerges: how does sustained collaboration with AI agents reshape developer behavior? Understanding this shift is important because software quality depends not only on generated artifacts, but also on how developers review, test, document, and maintain oversight during development.

Prior research has primarily focused on the outputs of AI agents rather than their influence on developers themselves. Existing studies have examined PR acceptance patterns [16, 22], code quality [9, 20], and testing practices in agent-generated contributions [25, 8]. Yet it remains unclear how sustained collaboration with AI agents influences developers' own practices over time. This question is methodologically challenging because developers who adopt AI agents may already differ from non-adopters, while behavioral changes over time may also reflect broader shifts in software development practices rather than the effects of AI agents themselves.

To understand this, we draw on automation complacency theory [14], which predicts reduced human monitoring and verification effort under reliable automation. In software engineering, this suggests that developers working alongside AI agents may change how they code, test, and review software over time. Accordingly, we investigate the following research question.

RQ: *How does the observable adoption of AI coding agents reshape developers' coding effort, testing practices, and code review behavior?*

To answer this question, we conduct a longitudinal study of 669 developers across 103 repositories using the AIDev dataset and complement within-subject analyses with a (DiD) design using 228 control developers from repositories without agent activity. Our findings reveal selective behavioral adaptation rather than broad behavioral degradation. Agent adoption is associated with increased documentation effort and improved testing discipline, while changes in review behavior largely reflect broader ecosystem-wide trends rather than agent-associated effects. These findings make two contributions. Empirically, they show that developers maintain coding effort while strengthening documentation and testing practices when working with AI agents. Methodologically, they demonstrate that relying on within-subject analyses alone can misattribute broader ecosystem trends to AI-agent effects, highlighting the importance of comparative causal designs when studying AI-assisted software development.

2 Related Work

Prior research on AI-assisted software development has largely focused on short-term productivity gains and developer interactions with coding assistants. Controlled experiments show substantial improvements in task completion speed – for example, Peng et al. [15] report a 55.8% increase in task completion speed with GitHub Copilot, while Ziegler et al. [26] find similar perceived productivity benefits. However, complementary usability studies reveal a more nuanced picture: developers may accept AI-generated suggestions without thorough verification [21], exhibit varied interaction patterns [3], and face inefficiencies when evaluating generated code [13]. This literature is largely limited to short-term interactions and self-reported perceptions, leaving longer-term behavioral adaptation comparatively underexplored.

Beyond productivity, prior work has examined core development practices such as code review and testing, both of which are central to software quality. Code review research shows that practices converge across projects [17], serve broader functions such as knowledge transfer [2], and are strongly associated with software quality [12]. Additional work highlights the importance of factors such as comment specificity and reviewer familiarity [5].

Recent studies extend this line of inquiry to AI-generated contributions, showing differences in how developers review agent-authored PR [7]. Existing work, however, primarily examines the evaluation of agent-generated artifacts rather than broader changes in developers' review practices following sustained agent exposure. Similarly, testing practices have been studied through the lens of test–production co-evolution. Prior work shows that co-change between production and test files is a meaningful indicator of testing discipline and relates to downstream quality outcomes [19]. More recent studies investigate test generation by AI agents [25, 8], demonstrating variation across tools.

As with code review, this literature primarily focuses on agent-generated artifacts rather than longer-term developer adaptation. It remains unclear whether developers change their own testing behavior when working alongside agents. We therefore examine whether test inclusion and co-change patterns shift following agent adoption, and whether such changes are agent-associated or reflect broader ecosystem trends.

3 Research Method

We investigate the effects of AI agent adoption on developer behavior using a longitudinal within-subject quasi-experimental design, complemented with a DiD analysis [6] to better distinguish agent-related changes from broader ecosystem trends. For each treatment repository, we define the adoption point as the timestamp of the first observable agent-authored pull request, representing the earliest repository-level evidence of autonomous AI-agent participation. Behavioral changes are analyzed at the developer level, allowing each developer to serve as their own baseline while controlling for stable differences such as coding style, experience, and project context. To distinguish agent-related changes from broader temporal trends, we compare 103 treatment repositories with 148 control repositories that contain no agent activity. Because control repositories do not have a natural adoption event, we use the median pull request creation date as a pseudo-adoption point to partition repository history into balanced pre- and post-periods. This provides a consistent temporal reference for estimating developer-level behavioral changes while avoiding arbitrary calendar cutoffs and ensuring comparable pre/post partitions across repositories. We further assess the suitability of this construction by examining pre-adoption parallel trends between treatment and control groups (Figure 1).

Dataset and Sampling. Our analysis is based on the AIDev dataset [11]. We retain repositories that exhibit sustained agent activity spanning at least 90 calendar days, defined as a minimum of 90 days between the first and last agent-authored pull request, contain human-authored merged PRs in both pre- and post-adoption periods, and include at least one qualifying developer, yielding 103 treatment repositories. We retain developers who contributed at least five merged PRs in both periods within the same repository, a threshold chosen to balance statistical stability with sample retention. For the control group, we identify 148 repositories with no agent activity and apply the same developer criteria, yielding 228 qualified developers. To ensure data reliability, we remove invalid timestamps, exclude review turnaround times exceeding 720 hours (30 days), and retain only reviewers active in both periods. Coding-related metrics are filtered to reduce the influence of extreme values, and when developer-period groups exceed 20 PRs, we randomly subsample 20 to balance statistical stability and computational cost.

Metric Operationalization. We operationalize developer behavior using 11 metrics across three dimensions, aggregating all measures per developer per period using medians for volume-based metrics and means for proportions. Coding effort is captured through PR size, files changed, commits per PR, and PR description length. Testing discipline is measured via test file ratio, co-change rate, and production-conditioned co-change rate; test files are identified using standard filename heuristics (e.g., `test_`, `_test`, `.spec`, `/tests/`), which likely undercount modern co-located or implicit testing practices and therefore yield conservative estimates. Finally, review behavior is assessed through inline comments, review turnaround time, changes-requested ratio, and approval rate.

Statistical Analysis. We analyze behavioral changes using both non-parametric and parametric methods. Within-subject differences are tested using the Wilcoxon signed-rank test [23], which is robust to non-normal paired distributions and extreme values, with rank-biserial correlation (r) reported as the effect size. To distinguish agent-related changes from broader ecosystem trends, we evaluate per-developer DiD deltas using the Mann-Whitney U test. As an additional robustness check, we fit two-way fixed-effects models [24] of the form $y_{idt} = \alpha_i + \lambda_t + \beta(\text{Post}_{rt} \times \text{Treat}_r) + \epsilon_{idt}$ with repository-clustered standard errors, allowing us to account for stable unobserved differences across repositories and time periods. Because several outcomes are right-skewed, these models are used to validate the consistency of the findings rather than as the primary basis for interpretation. To further reduce selection bias between treatment and control repositories, we perform one-to-one nearest-neighbor propensity-score matching [18] based on programming language and log star count (caliper 0.2 standard deviations), achieving 83.2% language overlap and a median log star-distance of 0.52. We also conduct a dose-response analysis by dividing repositories according to the median level of agent activity (14 PRs) to examine whether stronger agent exposure is associated with larger behavioral changes. All p -values are adjusted using the Benjamini–Hochberg [4] procedure across 11 tests ($\alpha = 0.05$). Compared with within-subject analyses alone, the DiD design provides stronger evidence for agent-associated behavioral changes. However, the seven-month observation window captures only short-term behavioral changes; longer-term dynamics, including the gradual emergence of complacency effects predicted by classical theory, remain an important direction for future work.

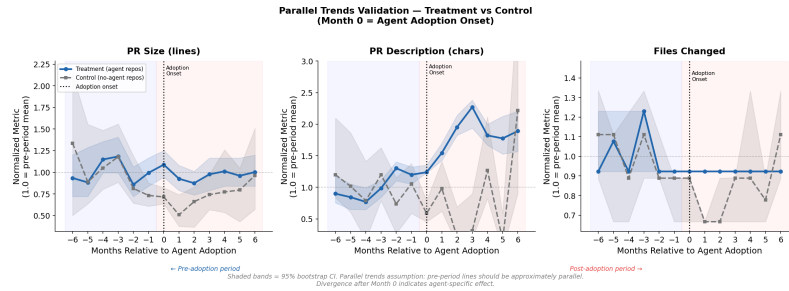
4 Results

Our final dataset comprises 669 developer–repository pairs from 103 treatment repositories and 228 pairs from 148 control repositories. Table 1 summarizes the dataset.

■ **Table 1** Dataset summary for treatment and control groups.

Property	Treatment	Control
Repositories	103	148
Developer–repo pairs	669	228
Enriched PRs	22,792	6,656
Reviews analyzed	47,468	7,168
Qualified reviewers	532	–
Median before/after (days)	349 / 344	–
Observation window	7 months before/after first agent PR	Full history
Split criterion	First agent PR	Median PR date

Figure 1 shows approximately parallel pre-adoption trends between treatment and control groups, supporting the validity of the DiD analysis.



■ **Figure 1** Parallel pre-adoption trends for treatment repositories and median-split control repositories used in the Difference-in-Differences analysis.

Across our analyses, seven of ten within-subject significant findings do not survive DiD validation, corresponding to a potential 70% misattribution rate for within-subject-only designs in this setting. This suggests that within-subject analyses alone may incorrectly attribute broader ecosystem trends to agent adoption.

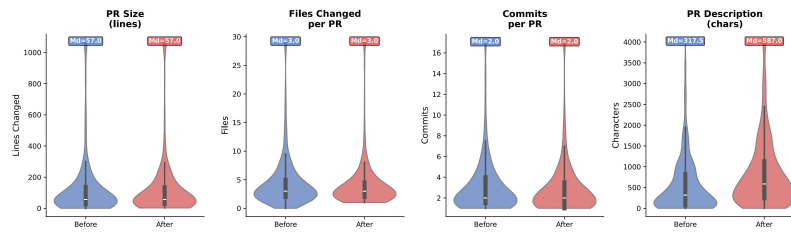
4.1 Coding Effort

We next examine how developers' coding effort changes after agent adoption. Figure 2 depicts the distributional shifts across all four coding effort metrics. Overall, coding effort remains largely stable, with one notable exception: a substantial increase in PR documentation. Detailed within-subject statistical results are available in the replication package [1].

PR Size. shows no significant change ($p_{adj} = 0.805$, $r = 0.011$, negligible). The near-equal split between developers who increase versus decrease PR size (49.8% vs. 50.2%) indicates the absence of any systematic directional shift. Despite agents generating substantial code in the same repositories, developers do not reduce the volume of code they personally write following the first observable agent-authored pull request. Similarly, **Files Changed and Commits per PR** shows statistically significant within-subject differences ($p_{adj} = 0.004$ and 0.022 , small effects) despite identical medians. These effects reflect compression in the upper tails of the distribution: developers who previously submitted broader, multi-file PRs narrow their scope, while the majority remain stable. However, this same pattern is observed among control developers in the DiD analysis (Table 2), indicating that it reflects broader ecosystem trends rather than agent-associated effects.

Additionally, **PR Description Length** shows the largest effect among all coding effort metrics. The median increases from 304.0 to 537.5 characters (+76.8%, $p_{adj} < 0.001$, $r = 0.561$, large), with 65.3% of developers increasing their descriptions and a median paired difference of +117 characters. This represents a broad-based behavioral shift rather than an outlier-driven effect. Notably, this pattern runs counter to automation complacency: communication effort increases, rather than decreases, following agent adoption.

DiD Validation. The DiD analysis refines the within-subject findings: only PR description length remains significant when compared against the control group. Treatment developers increase descriptions by +121.5 characters, while control developers show no change ($p_{adj} < 0.001$, $r = -0.323$, medium effect). This effect is consistently supported across



■ **Figure 2** Distribution of coding effort metrics before and after agent adoption.

■ **Table 2** DiD results for coding effort (BH-adjusted).

Metric	Treat. Δ	Ctrl. Δ	DiD	p_{adj}	r	Agent-Sp.
PR Size	+0.0	−10.5	+10.5	0.053	−0.101	No
Files Changed	+0.0	+0.0	+0.0	0.877	0.007	No
Commits/PR	+0.0	+0.0	+0.0	0.836	−0.018	No
PR Description	+121.5	+0.0	+121.5	<0.001	−0.323	Yes

analytical methods. To assess whether this increase reflects behavioral adaptation rather than mechanical adoption of agent-generated templates, we compare repositories with lower versus higher levels of agent-authored PR activity (low exposure: ≤ 14 agent PRs; high exposure: > 14) to examine whether stronger agent exposure is associated with larger behavioral changes. Low-exposure developers increase description length by +67.5% ($p < 0.001$, $r = 0.601$), while high-exposure developers increase by +107.4% ($p < 0.001$, $r = 0.545$), a difference of +39.8 percentage points. The significant difference between low- and high-exposure repositories indicates that the effect scales with agent exposure, supporting the interpretation of behavioral adaptation rather than incidental template reuse.

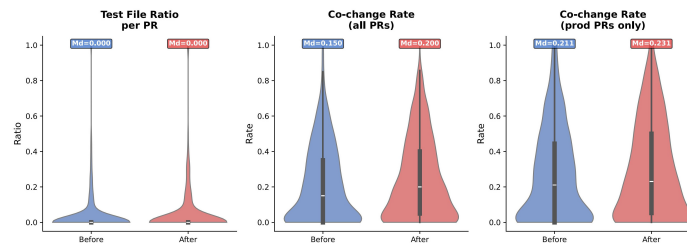
Key Findings

Agent adoption primarily affects how developers communicate their work rather than the amount of code they produce. Developers write more detailed PR descriptions, while coding effort remains largely unchanged and other apparent changes in coding activity are explained by broader ecosystem trends rather than agent adoption.

4.2 Testing Discipline

We next investigate how developers' testing practices change following agent adoption. Figure 3 illustrates the distributional changes. In contrast to the expectations of automation complacency, all three testing metrics show consistent improvements, indicating increased rather than reduced testing discipline. Detailed within-subject statistical results are available in the replication package [1].

Test File Ratio. The median test file ratio is zero in both periods, indicating that most developers do not include test files in the majority of their PRs. Nevertheless, a statistically significant difference is observed ($p_{adj} = 0.021$, $r = 0.203$, small). This reflects a distributional shift above the median: among developers who include tests, inclusion frequency increases after adoption. Although only 16.9% of developers increase their test file ratio, the magnitude of those increases is sufficient to produce a meaningful rank-based effect, suggesting that testing improvements are concentrated within a subset of developers rather than uniformly



■ **Figure 3** Distribution of testing discipline metrics before and after agent adoption.

across the population. Similarly, **Co-change Rate**: The co-change rate increases from a median of 0.150 to 0.200, corresponding to a 33.3% relative increase ($p_{adj} = 0.008$, $r = 0.144$, small). The proportion of PRs that include both production and test changes rises, indicating that developers are more likely to couple code changes with corresponding test updates after agent adoption. Together, these findings indicate improved integration between development and testing activities. Additionally, **Production-Conditioned Co-change Rate**: When restricted to PRs that modify production code, the co-change rate increases from 0.211 to 0.231 (+9.6%, $p_{adj} = 0.010$, $r = 0.137$, small). This provides a more conservative estimate by focusing specifically on scenarios where testing is most relevant. The consistent increase across both unconditional and conditioned measures strengthens confidence that developers are more likely to include tests when modifying production code after agent adoption. The consistent improvement across all three metrics strengthens confidence in the observed testing-related changes. Because test-file detection relies on filename heuristics, the reported effects likely represent conservative estimates.

DiD Validation. To assess whether these improvements are associated with agent adoption, we perform a DiD analysis (Table 3). All three testing metrics remain significant after DiD validation, suggesting that the observed improvements are more strongly associated with agent adoption than broader ecosystem trends. Although the median change is zero for both treatment and control groups, the distributions of developer-level changes differ significantly. Treatment developers exhibit systematically more positive shifts than controls across all three testing metrics, indicating that the effect is distributional rather than a shift in the median. This consistent pattern strengthens confidence that testing improvements are associated with agent adoption rather than broader ecosystem evolution.

■ **Table 3** DiD results for testing discipline (BH-adjusted). All three effects remained significant after comparative validation.

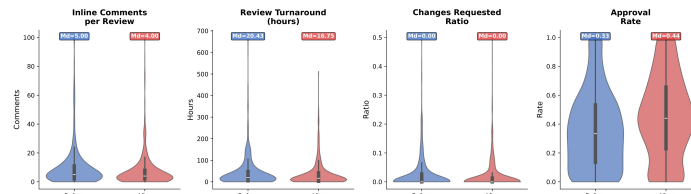
Metric	Treat. Δ	Ctrl. Δ	DiD	p_{adj}	r	Agent-Sp.
Test File Ratio	0.0	0.0	0.0	<0.001	-0.158	Yes
Co-change Rate	0.0	0.0	0.0	<0.001	-0.157	Yes
Prod. Co-change	0.0	0.0	0.0	0.006	-0.139	Yes

Key Findings

Agent adoption is associated with stronger testing discipline. Developers become more likely to integrate testing into their development workflow, suggesting that AI agents reinforce verification-oriented behaviors rather than reducing attention to quality.

4.3 Review Behavior

We next examine how developers' review behavior changes following agent adoption. Figure 4 illustrates the distributional changes. In contrast to the stability and improvements observed in coding and testing practices, all four review metrics show consistent changes: reviews become faster, less detailed, and more permissive after adoption. Detailed within-subject statistical results are available in the replication package [1].



■ **Figure 4** Distribution of review behavior metrics before and after agent adoption.

Inline Review Comments. The median number of inline comments per review decreases from 5.0 to 4.0 (-20.0% , $p_{adj} < 0.001$, $r = 0.244$). Only 34.8% of reviewers increase their commenting activity, indicating a broad reduction in review detail after adoption.

Review Turnaround Time: Median turnaround time decreases from 20.43 to 16.75 hours (-18.0% , $p_{adj} < 0.001$, $r = 0.192$), with 57.0% of reviewers completing reviews more quickly. Together with reduced commenting activity, this suggests less thorough review behavior rather than purely improved efficiency.

Changes Requested: The proportion of reviews requesting changes declines ($3.5\% \rightarrow 2.6\%$, $p_{adj} < 0.001$, $r = 0.259$). Only 18.2% of reviewers increase their rate of requesting changes, indicating that most reviewers become less strict in enforcing revisions.

Approval Rate: The approval rate increases from 0.333 to 0.441 ($+32.2\%$, $p_{adj} < 0.001$, $r = 0.411$), representing the strongest within-subject effect in review behavior. A majority of reviewers (60.2%) approve a higher proportion of PRs after adoption, reinforcing the shift toward more permissive review practices. Taken together, these results initially appear consistent with automation complacency, where monitoring effort declines while productive effort remains stable.

DiD Validation: To determine whether these changes are attributable to agent adoption, we perform a DiD analysis (Table 4). However, the DiD results change this interpretation.

■ **Table 4** DiD results for review behavior (BH-adjusted). None remained significant after comparative validation.

Metric	Treat. Δ	Ctrl. Δ	DiD	p_{adj}	r	Agent-Sp.
Inline Comments	-1.00	+0.00	-1.00	0.649	-0.039	No
Turnaround (h)	-1.59	-0.05	-1.54	0.090	0.094	No
Changes Req'd	+0.00	+0.00	+0.00	0.828	0.022	No
Approval Rate	+0.05	+0.00	+0.05	0.877	-0.009	No

None of the four metrics remains significant after DiD validation. Control developers exhibit the same directional patterns as treatment developers, including fewer comments, faster reviews, and higher approval rates. All DiD effect sizes are negligible ($|r| \leq 0.039$ for three metrics). Turnaround time shows the largest treatment-control difference (-1.54

hours) but does not survive multiple testing correction ($p_{adj} = 0.090$). These findings suggest that the observed changes reflect broader ecosystem-wide trends rather than agent-associated effects. Without a control group, the within-subject results would likely have been incorrectly attributed to agent adoption. Because our analysis includes only reviewers active in both periods ($n = 532$), the findings likely represent conservative estimates of changes in review behavior. Longer-term dynamics may also differ beyond the seven-month observation window.

Key Findings

Changes in review behavior are not uniquely associated with agent adoption. Although reviews become faster, less detailed, and more permissive over time, similar patterns are observed in repositories without agents, suggesting that these shifts reflect broader changes in software development practice rather than agent-associated adaptation.

4.4 Synthesis of Findings

The results across all 11 metrics reveal a clear pattern: agent-associated changes are concentrated in productive development activities, whereas most review-related changes align with broader ecosystem trends. Table 5 summarizes the findings across all analyses. Only four metrics are consistently supported across analytical methods (bold rows in Table 5): PR description length and all three testing discipline measures. Across these dimensions, developers produce more detailed documentation and show stronger coupling between production and test changes following agent adoption. In contrast, the remaining seven metrics are not supported as agent-associated effects. PR size remains stable under both within-subject and DiD analyses, while files changed and commits per PR show within-subject differences that disappear under DiD validation, suggesting broader ecosystem trends rather than agent-associated effects. Most notably, all four review metrics remain highly significant in within-subject analyses but lose significance under DiD validation. Although overall directional patterns are clear, developers differ substantially in both the magnitude and direction of behavioral change. For example, 65.3% of developers increase PR description length after adoption, whereas 34.7% decrease it; similarly, 34.8% increase inline review comments while 65.2% decrease them. This heterogeneity suggests that responses to agent adoption are shaped by contextual factors such as repository norms, development practices, and developer roles. The exposure analysis strengthens the interpretation of the documentation effect. Developers in high-exposure repositories increase PR description length by +107.4%, compared with +67.5% in low-exposure repositories, indicating that the magnitude of the effect scales with agent usage. This gradient is consistent with behavioral adaptation rather than incidental template reuse. Overall, the findings support a pattern of selective behavioral adaptation. Agent adoption is associated with measurable changes in development behaviors, particularly documentation and testing practices, while evaluative review behaviors evolve alongside broader ecosystem dynamics independent of agent presence. Automation complacency, as classically defined, is therefore not supported by the observed comparative evidence.

5 Discussion

Across coding, testing, and review activities, our findings reveal a clear asymmetry: agent adoption is associated with changes in productive development behaviors, whereas most apparent changes in review behavior are better explained by broader ecosystem trends than by agent adoption itself.

■ **Table 5** Summary of within-subject and DiD findings across all metrics (BH-adjusted).

RQ	Metric	p_{WS}	r	p_{DiD}	Conf.	Conclusion
RQ1	PR Size	0.805	0.011	0.053	0/4	No effect
RQ1	Files Changed	0.004	0.154	0.877	1/4	Ecosystem trend
RQ1	Commits/PR	0.022	0.121	0.836	1/4	Ecosystem trend
RQ1	PR Description	<.001	0.561	<.001	4/4	agent-associated (large)
RQ2	Test File Ratio	0.021	0.203	<.001	3/4	agent-associated (small)
RQ2	Co-change Rate	0.008	0.144	<.001	4/4	agent-associated (small)
RQ2	Prod. Co-change	0.010	0.137	0.006	3/4	agent-associated (small)
RQ3	Inline Comments	<.001	0.244	0.649	1/4	Ecosystem trend
RQ3	Turnaround	<.001	0.192	0.090	2/4	Ecosystem trend
RQ3	Changes Req'd	<.001	0.259	0.828	1/4	Ecosystem trend
RQ3	Approval Rate	<.001	0.411	0.877	1/4	Ecosystem trend

Revisiting the complacency hypothesis. Automation complacency theory predicts reduced monitoring and verification effort when humans interact with reliable automated systems [14]. Our findings both challenge and refine this expectation in the context of AI-assisted software development. Developers do not appear to disengage from productive engineering activities following agent adoption. Instead, the observed improvements in documentation and testing practices suggest active behavioral adaptation, where developers invest additional effort in communication and verification when collaborating with AI agents. At the same time, the review-related changes initially resemble classical complacency patterns: reviews become faster, less detailed, and more permissive. However, these patterns are also present in repositories without agent adoption, suggesting that they reflect broader shifts in software development practice rather than direct effects of AI agents themselves. Our findings therefore do not support a classical automation complacency interpretation of code review behavior. Instead, they point toward a wider ecosystem shift in review practices, potentially driven by increasing development velocity, growing reliance on automated quality assurance tools, and evolving collaboration norms in modern software engineering.

Explaining the observed effects. Several mechanisms may explain the increases in documentation and testing effort. One possibility is social adaptation: developers repeatedly exposed to structured agent-authored contributions may gradually adopt similar practices themselves. Another possibility is compensatory behavior, where developers invest more effort in explanation and verification to reduce uncertainty around AI-assisted code changes. The stronger effects observed in high-exposure repositories are consistent with this interpretation. An alternative explanation concerns the timing of the intervention itself. Our adoption point corresponds to the first observable agent-authored pull request rather than the first use of AI-assisted programming tools within a repository. Developers may already have been using AI coding assistants such as GitHub Copilot, ChatGPT, or Cursor before autonomous agent-authored pull requests became visible. Consequently, our findings should be interpreted as behavioral changes associated with the transition to agent-based development workflows rather than the initial adoption of AI-assisted software development. Earlier exposure to AI coding assistants may also attenuate observable changes in coding-related metrics, such as pull request size, if behavioral adaptation had already begun before the observation window. Another possible explanation is that repositories adopting autonomous coding agents may simultaneously introduce new contribution templates, documentation practices, or workflow conventions that encourage more detailed pull request descriptions

and stronger testing practices. Although the observed exposure gradient is more consistent with behavioral adaptation than simple template reuse, these explanations cannot be fully disentangled using observational repository data. We also note that PR description length is an operationalization of communication effort rather than communication quality. Longer descriptions do not necessarily imply clearer or more useful documentation, and future work should complement quantitative measures with qualitative assessment of PR content. More broadly, the consistency of the testing improvements across multiple operationalizations suggests that the observed pattern is unlikely to be explained by measurement artifacts alone. In contrast, the review-related changes appear to reflect broader ecosystem evolution rather than agent-associated adaptation. Similarly, inline review comments, approval rates, and review turnaround time capture observable review behavior rather than review quality itself. These measures may also be influenced by factors such as pull request quality, team conventions, or increased use of automated quality assurance tools. Increasing development velocity, widespread adoption of CI/CD pipelines and automated quality tools, and shifting expectations around lightweight review processes may all contribute to faster and less detailed reviews across software projects generally. From this perspective, AI agents may coexist with these trends without necessarily driving them directly.

Methodological implications. Our findings highlight the importance of comparative research designs when studying AI-assisted software development. Several effects that appeared significant under within-subject analyses disappeared after comparison with control repositories, demonstrating that broader ecosystem changes can easily be misinterpreted as agent-associated behavioral effects when comparative controls are absent. Longitudinal within-subject analyses remain valuable for controlling developer-level heterogeneity, but our results suggest that they may be insufficient on their own for distinguishing agent-associated changes from concurrent ecosystem trends. Furthermore, because control repositories lack a natural intervention point, the pseudo-adoption date is based on the median pull request creation date. Although this enables balanced longitudinal comparisons, future work should investigate alternative constructions of comparable control windows (e.g., symmetric pseudo-observation windows or calendar-matched controls) and incorporate richer repository-level matching variables, such as project activity, contributor structure, CI practices, or review culture.

Practical implications. We find no evidence that agent adoption reduces productive engineering effort. Instead, developers maintain coding effort while improving testing and documentation practices. In contrast, declining review thoroughness appears to be an ecosystem-wide challenge rather than a challenge uniquely associated with agent adoption, suggesting that improving review quality will require broader process interventions rather than restricting agent use. At the same time, the observed increase in documentation practices may represent a valuable opportunity for improving knowledge transfer, onboarding, and maintainability in collaborative development environments.

6 Conclusion and Future Work

This study examined whether AI coding agent adoption changes how developers code, test, and review using a longitudinal within-subject design with comparative validation across 669 developer–repository pairs. Our findings reveal a pattern of selective behavioral adaptation. Agent adoption is associated with stronger documentation and testing practices,

while coding effort remains stable. In contrast, changes in review behavior reflect broader ecosystem-wide trends rather than agent-associated effects. More broadly, our findings challenge straightforward applications of automation complacency theory to AI-assisted software development. Developers do not appear to disengage from productive engineering activities when working with agents. Instead, the strongest agent-associated effects involve increased communication and verification effort. Methodologically, the study also highlights the importance of comparative research designs for distinguishing agent-associated behavioral changes from concurrent ecosystem evolution.

Several directions can extend this work. Future studies should strengthen causal identification under heterogeneous treatment timing, investigate the qualitative mechanisms underlying documentation and testing changes, and examine whether these behavioral patterns persist across longer time horizons, different agent types, and enterprise development settings.

Data Availability Statement

The replication package for this study, including the processed datasets, analysis scripts, supplementary statistical tables, figure-generation materials, and documentation required to reproduce the reported results, is publicly available at: <https://doi.org/10.5281/zenodo.20442228>

References

- 1 Anonymous. Replication package for “does working with ai agents change how developers code, test, and review?”, 2026. doi:10.5281/zenodo.20442228.
- 2 Alberto Bacchelli and Christian Bird. Expectations, outcomes, and challenges of modern code review. In *2013 35th international conference on software engineering (ICSE)*, pages 712–721. IEEE, 2013. doi:10.1109/ICSE.2013.6606617.
- 3 Shraddha Barke, Michael B James, and Nadia Polikarpova. Grounded copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA1):85–111, 2023. doi:10.1145/3586030.
- 4 Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.
- 5 Amiangshu Bosu, Michaela Greiler, and Christian Bird. Characteristics of useful code reviews: An empirical study at microsoft. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 146–156. IEEE, 2015. doi:10.1109/MSR.2015.21.
- 6 Brantly Callaway and Pedro HC Sant’Anna. Difference-in-differences with multiple time periods. *Journal of econometrics*, 225(2):200–230, 2021.
- 7 Haoyu Gao, Peerachai Banyongrakul, Hao Guan, Mansooreh Zahedi, and Christoph Treude. On autopilot? an empirical study of human-ai teaming and review practices in open source. *arXiv preprint arXiv:2601.13754*, 2026. doi:10.48550/arXiv.2601.13754.
- 8 Sabrina Haque, Sarvesh Ingale, and Christoph Csallner. An empirical study of tests in agentic pull requests. <https://ranger.uta.edu/csallner/papers/Haque26Empirical.pdf>, 2026.
- 9 Haoming Huang, Pongchai Jaisri, Shota Shimizu, Lingfeng Chen, Sota Nakashima, and Gema Rodríguez-Pérez. More code, less reuse: Investigating code quality and reviewer sentiment towards ai-generated pull requests. *arXiv preprint arXiv:2601.21276*, 2026. doi:10.48550/arXiv.2601.21276.
- 10 Hao Li, Haoxiang Zhang, and Ahmed E Hassan. The rise of ai teammates in software engineering (se) 3.0: How autonomous coding agents are reshaping software engineering. corr abs/2507.15003 (2025). *arXiv preprint*, 2025. arXiv:2507.15003.

- 11 Hao Li, Haoxiang Zhang, and Ahmed E Hassan. Aidev: studying ai coding agents on github. *arXiv preprint arXiv:2602.09185*, 2026. doi:10.48550/arXiv.2602.09185.
- 12 Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E Hassan. An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering*, 21(5):2146–2189, 2016. doi:10.1007/S10664-015-9381-9.
- 13 Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. Reading between the lines: Modeling user behavior and costs in ai-assisted programming. In *Proceedings of the 2024 CHI conference on human factors in computing systems*, pages 1–16, 2024. doi:10.1145/3613904.3641936.
- 14 Raja Parasuraman and Victor Riley. Humans and automation: Use, misuse, disuse, abuse. *Human factors*, 39(2):230–253, 1997. doi:10.1518/001872097778543886.
- 15 Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*, 2023. doi:10.48550/arXiv.2302.06590.
- 16 Giovanni Pinna, Jingzhi Gong, David Williams, and Federica Sarro. Comparing ai coding agents: A task-stratified analysis of pull request acceptance. *arXiv preprint arXiv:2602.08915*, 2026. doi:10.48550/arXiv.2602.08915.
- 17 Peter C Rigby and Christian Bird. Convergent contemporary software peer review practices. In *Proceedings of the 2013 9th joint meeting on foundations of software engineering*, pages 202–212, 2013. doi:10.1145/2491411.2491444.
- 18 Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.
- 19 Davide Spadini, Maurício Aniche, Margaret-Anne Storey, Magiel Bruntink, and Alberto Bacchelli. When testing meets code review: Why and how developers review tests. In *Proceedings of the 40th International Conference on Software Engineering*, pages 677–687, 2018.
- 20 Nanthit Temkulkiat, Chaiyong Ragkhitwetsagul, Morakot Choetkiertikul, Ruksit Rojpaisarnkit, and Raula Gaikovina Kula. When is generated code difficult to comprehend? assessing ai agent python code proficiency in the wild. *arXiv preprint arXiv:2604.00299*, 2026. doi:10.48550/arXiv.2604.00299.
- 21 Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, pages 1–7, 2022. doi:10.1145/3491101.3519665.
- 22 Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E Hassan. On the use of agentic coding: An empirical study of pull requests on github. *ACM Transactions on Software Engineering and Methodology*, 2025.
- 23 Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.
- 24 Jeffrey M Wooldridge. *Econometric analysis of cross section and panel data*. MIT press, 2010.
- 25 Suzuka Yoshimoto, Shun Fujita, Kosei Horikawa, Daniel Feitosa, Yutaro Kashiwa, and Hajimu Iida. Testing with ai agents: An empirical study of test generation frequency, quality, and coverage. *arXiv preprint arXiv:2603.13724*, 2026. doi:10.48550/arXiv.2603.13724.
- 26 Albert Ziegler, Eirini Kalliamvakou, X Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. Measuring github copilot’s impact on productivity. *Communications of the ACM*, 67(3):54–63, 2024. doi:10.1145/3633453.