DATA 542 Project – Milestone 3: Final Report

HAOZHONG JI, University of British Columbia Okanagan, Canada
YIN-WEN TSAI, University of British Columbia Okanagan, Canada

1 Research Questions and Dataset

We use the AIDev dataset. It is about pull requests on GitHub that use AI coding agents. The dataset has information on pull requests, repositories, users, reviews, and commits. In this project, we only study pull requests that involve AI agents. We also keep only pull requests with valid times. Their final status must be clear. So they are either merged or closed.

We will answer three research questions (RQs). Each RQ uses more than one feature.

- **RQ1:** Do different AI coding agents get used in different types of repositories? We compare repositories with different popularity and activity.
- **RQ2:** How do the size and type of AI pull requests relate to review work and merge results, when we group pull requests into simple size and file-type buckets?
- **RQ3:** For AI-agent pull requests, how do simple human–AI collaboration patterns (based on follow-up commits) relate to merge rate and time-to-merge?

2 Methodology

For all RQs, we will load the pull request table and the repository table. We will join them by repository ID. We will keep only AI-related pull requests with clear outcomes. We will create features for our analysis. These include agent type, repository stars, and repository activity. We will also use commit-level tables and user information to detect human and agent commits inside each pull request.

RQ1 (adoption patterns). We will group repositories by star counts. For example, low, medium, and high stars. We will count how many AI pull requests each agent makes in each group. We will also look at changes over time, such as by month. We will show the results with line charts and bar charts.

RQ2 (characteristics and outcomes). For each pull request, we will measure its size. We use lines added, lines deleted, and files changed. We will label file types using file paths. For example, code, tests, docs, or config files. We will record review comments, merge outcome, and time-to-merge. We will then group pull requests into simple size buckets (for example, small, medium, large) and by main file type. For each group, we will summarize merge rate, median time-to-merge, and average number of review comments. We will compare these groups using tables, bar charts, and box plots. If time allows, we may run a very simple logistic regression as an exploratory check, but our main focus will be on data wrangling and descriptive analysis.

RQ3 (collaboration patterns). For RQ3, we will use commit-level tables joined with the user table to detect human and agent commits inside each pull request. In AIDev, every pull request is opened by an AI agent, so we study teamwork only through follow-up commits. For each pull request, we will count human and agent commits after the first agent commit and assign a simple pattern label, such as "agent-only" (no human commits) or "agent-then-human" (at least one human follow-up commit). For each pattern, we will summarize merge rate, median time-to-merge, and average review comments and compare the patterns with tables and a few bar or box plots.

Authors' Contact Information: Haozhong Ji, jhzzs@student.ubc.ca, University of British Columbia Okanagan, Kelowna, Canada; Yin-Wen Tsai, ywtsai@student.ubc.ca, University of British Columbia Okanagan, Kelowna, Canada.

3 Results

3.1 Results for RQ1

3.1.1 ***Data and grouping.*** For RQ1, we use the AIDev dataset and focus on pull requests (PRs) opened by AI coding agents. Each PR is linked to its repository, so we can attach repository-level features such as stars and activity. We only keep PRs with a clear final status (merged or closed) and valid timestamps.

To measure *popularity*, we use the number of stars on each repository. We rank repositories by star count and then cut the rank into three equal-sized groups: *Low*, *Medium*, and *High* popularity. This rank-based grouping helps reduce the effect of a few extremely famous projects.

To measure *activity*, we build a simple activity score from repository-level events (for example, commits, issues, and pull requests). We then split this score into three groups: *Low*, *Medium*, and *High* activity.

For both popularity and activity, we count how many AI-generated PRs each agent creates in each group. Then we convert these counts to proportions within each group. This gives two main tables:

- popularity group × agent (`pop_agent`),
- activity group × agent (`act_agent`).

We also visualize these distributions using grouped bar charts, with one panel for popularity and another for activity.

All percentages in the report are rounded to one decimal place. This means that very small but non-zero shares (for example, less than 0.05%) can appear as 0.0% in the tables and plots. In those cases, we describe them as "negligible but non-zero", since they exist in the raw counts but do not change the main patterns.

3.1.2 ***Popularity and agent usage.*** Overall, **OpenAI_Codex** dominates AI usage in all popularity groups, but the mix of agents changes with popularity.

In *low-popularity* repositories, OpenAI_Codex accounts for about 90.8% of AI PRs (260,202 PRs). *Copilot* contributes about 8.4% (24,193 PRs), and *Claude_Code* about 0.8% (2,247 PRs). Other agents such as *Cursor* and *Devin* only appear, if at all, at negligible levels in this group. Their shares are so small (below 0.1%) that they round to 0.0% in our summary tables, so they do not affect the overall picture.

In *medium-popularity* repositories, the distribution becomes even more concentrated. Here, essentially all AI PRs (around 100%) are opened by **OpenAI_Codex** (286,642 PRs). Any other agents in this group, if present, contribute at most a very small number of PRs. Their shares are too small to be visible at one-decimal rounding and can be treated as negligible for this analysis.

In *high-popularity* repositories, the picture becomes more diverse. OpenAI_Codex still leads, but its share drops to about 74.7% (214,188 PRs). Copilot contributes about 5.6% (16,022 PRs), and Claude_Code about 0.9% (2,487 PRs). This is also where *Cursor* and *Devin* reach noticeable levels: about 9.0% (25,880 PRs) for Cursor and 9.8% (28,066 PRs) for Devin. There may be a few additional agents with very small shares (again below 0.1%), but they are negligible compared with these main tools.

These results suggest that new or alternative AI agents (such as Cursor and Devin) are mainly used in more popular projects. Less popular repositories rely almost entirely on OpenAI_Codex, with some additional usage of Copilot and Claude_Code, and only negligible use of other agents.

3.1.3 ***Activity and agent usage.*** The patterns by *activity* are similar, but not identical.

In *low-activity* repositories, OpenAI_Codex again dominates with about 88.7% of AI PRs (254,194 PRs). Copilot has about 10.3% (29,414 PRs), and Claude_Code around 1.1% (3,034 PRs). Other agents, including Cursor and Devin, only appear (if at all) at extremely small levels. Their shares are below our rounding threshold, so they show as 0.0% in the tables.

In *medium-activity* repositories, the distribution is again highly concentrated. OpenAI_Codex accounts for essentially all AI PRs (about 100%, 286,642 PRs). Any other agents in this group have either zero or a very small number of PRs. Their shares are negligible compared with OpenAI_Codex and do not change the main conclusion.

In *high-activity* repositories, usage becomes more mixed. OpenAI_Codex still has the largest share, at about 76.8% (220,196 PRs). Copilot accounts for about 3.8% (10,801 PRs), and Claude_Code about 0.6% (1,700 PRs). As with popularity, Cursor and Devin appear only in this high-activity group at substantial levels, with shares around 9.0% (25,880 PRs) and 9.8% (28,066 PRs), respectively. Again, any additional agents beyond these have very small shares and can be ignored at the scale of our plots.

So, repositories with higher activity levels are more likely to use a wider range of AI agents. Low- and medium-activity projects mostly depend on OpenAI_Codex, with some Copilot and Claude_Code and only negligible use of other tools.

3.1.4 ***Summary of RQ1 Findings.*** Across both popularity and activity, **OpenAI_Codex** is the main AI agent in the AIDev dataset. It accounts for almost all AI-generated PRs in medium-popularity and medium-activity repositories, and it is still the majority agent in the other groups.

However, we observe clear differences at the high end. Highly popular and highly active repositories use a more diverse set of agents. In those groups, Cursor and Devin appear with non-trivial shares (around 9–10%), and the relative share of OpenAI_Codex decreases, even though it remains dominant. Some other agents also appear at very small proportions, but their contributions are below 0.1% and do not change the main trends.

These patterns suggest that large or busy projects are more willing, or more able, to experiment with multiple AI tools. Smaller or less active repositories mostly adopt a single, established agent. This answers RQ1: different AI coding agents are indeed used in different types of repositories, and the diversity of AI tools increases in more popular and more active projects.

3.2 Results for RQ2

To investigate how PR size and dominant file type relate to merge outcomes and review effort, we analyzed merge rate, median time-to-merge, and the average number of review comments across three PR size buckets (Small, Medium, Large) and five dominant file types (code, config, docs, other, test). The results reveal consistent patterns across all metrics.

3.2.1 ***Merge Rate.*** According to Figure, small PRs show the highest merge rates across all file types, typically ranging from 85% to 100%. Test-dominated PRs exhibit the strongest merge likelihood, while code- and other-related PRs merge slightly less frequently but still maintain high overall acceptance.

Merge rates decline for Medium PRs and drop further for Large PRs. Large PRs dominated by test or other files show the lowest merge rates (around 60–73%), suggesting that reviewers are more cautious with large, complex contributions.

Our key observation is that: small PRs are consistently more mergeable. Merge rates decrease as PR size grows, regardless of file type.

3.2.2 ***Time-to-Merge.*** Small PRs in Figure merge rapidly, often within minutes or hours. Medium PRs require slightly more review time, particularly for config-heavy changes.

Large PRs demonstrate the most substantial delays. In particular, PRs dominated by configuration files exhibit the longest median time-to-merge (≈ 0.19 days). Large code PRs also face prolonged review, reflecting the higher verification burden associated with these changes.

We get the key observation: merge time increases with PR size, with config and code changes demanding the longest review effort for Large PRs.

3.2.3 ***Review Comments.***

Review activity follows a similar size-dependent trend from Figure. Small PRs receive roughly 2.7–2.9 comments on average, except test PRs, which receive significantly fewer (0.7 comments), indicating lower review friction.

Medium PRs receive more comments overall, with documentation PRs receiving the most discussion. For Large PRs, review intensity increases sharply. Config, code, and documentation PRs all receive between 3.8–4.3 comments on average, suggesting that reviewers allocate more effort to evaluating system-critical or widely visible changes.

The key observation of this part is: review effort scales with PR size, and config/code/docs PRs attract the most reviewer scrutiny.

3.2.4 ***Summary of RQ2 Findings.***

Across merge rate, merge time, and review comments, results align around two consistent insights:

(1) PR size is the strongest determinant of merge behavior. Smaller PRs merge more successfully, more quickly, and with fewer comments.
(2) Dominant file type modifies reviewer expectations and workload. Test and documentation changes are merged easily, while configuration and code changes—especially large ones—trigger slower merges and more review activity.

These findings suggest that contributing smaller, well-scoped PRs and isolating risky file types (such as config or core code) can substantially improve merge outcomes in AI-related repositories.

3.3 Results for RQ3

For RQ3, we study how simple human–AI collaboration patterns relate to merge outcomes. We restrict the analysis to pull requests (PRs) opened by AI agents and use commit-level information to detect whether humans make follow-up commits. We classify each PR into one of two collaboration patterns:

- **agent_only**: the PR has no human follow-up commits.
- **agent_then_human**: the PR has at least one human follow-up commit (non-bot committer different from the original author).

Using this rule, we obtain 29,887 PRs in the *agent_only* group and 3,709 PRs in the *agent_then_human* group. In other words, about 89% of AI PRs are handled only by AI commits, and about 11% involve human follow-up commits.

We compare these two patterns on four main outcomes: merge rate, time-to-merge, number of review comments, and number of human follow-up commits.

3.3.1 ***Merge rate.***

Merge rate is defined as the proportion of PRs that have a non-null `merged_at` timestamp. The *agent_only* group has a merge rate of about 71.7%, while the *agent_then_human* group has a slightly lower merge rate of about 69.6%. The difference between the two patterns is small.

This result suggests that, in this dataset, adding human follow-up commits does not lead to a clearly higher merge rate for AI-agent PRs. Both patterns have merge rates around 70%, and the *agent_only* group is actually marginally higher.

3.3.2 ***Time-to-merge.*** Time-to-merge is measured as the difference (in days) between `created_at` and `merged_at` for merged PRs. The median time-to-merge is very different between the two patterns:

- *agent_only*: median ≈ 0.0012 days (about 1–2 minutes),
- *agent_then_human*: median ≈ 0.59 days (about 14 hours).

So *agent_only* PRs tend to be merged almost immediately, while *agent_then_human* PRs take much longer to merge on average. This is consistent with the idea that once humans get involved, the PR is more complex, requires more discussion, or waits for human review and edits before merging.

3.3.3 ***Review comments.*** We also link PRs to the review comments table and count how many review comments each PR receives. In our extracted data, both collaboration patterns have an average number of review comments very close to zero.

This likely reflects a limitation of the available review comments table (for example, many review discussions may not be captured there), rather than the true amount of review discussion on GitHub. Because of this, we do not draw strong conclusions from the review comment counts and focus more on merge rate and time-to-merge.

3.3.4 ***Human follow-up commits.*** By construction, *agent_only* PRs have zero human follow-up commits (on average and by definition). For *agent_then_human* PRs, the average number of human follow-up commits is about 48.1 per PR.

This large value shows that once humans get involved, they often contribute many commits to the same PR, rather than making a single small change. In other words, human–AI collaboration in this dataset is not just a one-off human fix. It usually involves a substantial amount of human editing or extension on top of the original AI-generated changes.

4 Interpretation

4.1 RQ1 Interpretation

For RQ1, our main takeaway is that AI agents are adopted very differently across projects. OpenAI_Codex appears almost everywhere, but its share is not constant.

In low- and medium-popularity or activity groups, nearly every AI PR comes from OpenAI_Codex. Copilot and Claude_Code do show up, but only in small proportions, and Cursor or Devin are almost invisible at the scale of our tables. These projects look like they picked one trusted AI tool and stayed with it.

In the high-popularity and high-activity groups, the mix of tools changes. OpenAI_Codex still opens most PRs, but its share drops, and Cursor and Devin each reach around 9–10% of AI PRs. In other words, the busiest and most visible projects seem more willing to try several AI agents and keep them in parallel.

Our analysis is descriptive: we look at usage patterns, not code quality or long term impact. Very small shares are rounded away, so we focus on the main agents that drive the plots. Overall, RQ1 suggests that project context matters a lot. Larger and more active repositories use a more diverse set of AI tools, while smaller or quieter repositories mostly rely on a single dominant agent.

4.2 RQ2 Interpretation

For RQ2, the patterns match what many developers would expect, but now we see them clearly for AI-generated PRs.

First, PR size matters a lot. As PRs move from Small to Medium to Large, merge rates go down, merge times go up, and reviewers leave more comments. This holds across all file types, which means that the sheer amount of change is already a strong signal for reviewers.

Second, the dominant file type changes how careful reviewers are. Large config and code PRs take the longest to merge and attract the most comments, which fits their higher risk: a bad configuration or a big code change can easily break the system. Documentation and small test PRs, on the other hand, move through the process more quickly and with less discussion.

In short, AI-generated PRs are treated in a very similar way to human-written ones:

(1) small and focused changes are easier to merge,
(2) some file types, especially configuration and core code, naturally get more scrutiny,
(3) review effort grows together with PR size.

This suggests that how we package AI changes (for example, splitting them into smaller PRs) can be just as important as which AI agent produced them.

4.3 RQ3 Interpretation

For RQ3, we looked at human–AI collaboration inside PRs and found a strong imbalance. Most AI-agent PRs belong to the *agent_only* pattern, and only a minority have any human follow-up commits. So in this dataset, the default seems to be letting the AI open and complete the PR without further human commits.

When humans do step in, the pattern is very different. The *agent_then_human* group has on average about 48 human follow-up commits per PR, which is a surprisingly large number. This looks less like a quick fix, and more like humans taking over and heavily editing or extending the AI's initial work.

Merge rates for the two patterns are quite similar, so we do not see a big gain in merge probability from adding human commits. What changes more clearly is time-to-merge: *agent_only* PRs are merged almost immediately, while *agent_then_human* PRs stay open much longer. This is consistent with PRs that involve more discussion, more edits, and more coordination.

A limitation of our RQ3 analysis is the review comments data, which appears very sparse. Since both patterns show almost zero review comments on average, we treat this field as incomplete and rely more on merge outcomes and commit counts.

Overall, RQ3 suggests that AI agents usually work "solo" in this dataset, and that explicit human collaboration is reserved for a smaller set of harder cases. When humans do join, they tend to invest heavily in the PR, which leads to more work and longer review time, but not necessarily a dramatically higher merge rate.

5 Link to GitHub Repository

data 542 project

A Appendix

A.1 Group Member Roles

We discussed the overall research plan together and jointly reviewed each other's work. We also worked together on editing the report text, adjusting the formatting, and discussing the results and figures.

- **Haozhong Ji**
  - Wrote the RQ1 report section (results and interpretation).
  - Helped debug the RQ1 code and performed a second check of the RQ1 data and summaries.

  - Wrote the RQ3 report section (results and interpretation).
  - Helped debug the RQ3 code and performed a second check of the RQ3 data and summaries.
- **Yin-Wen Tsai**
  - Wrote and debugged the RQ1 analysis code.
  - Wrote and debugged the RQ2 analysis code, and performed a second check of the RQ2 data and summaries.
  - Wrote the RQ2 report section (results and interpretation).
  - Wrote and debugged the RQ3 analysis code.

A.2 GenAI Disclosure

We used GenAI tools in this project.

In particular, we used ChatGPT to:

- Get ideas for how to structure some parts of the analysis code for RQ1, RQ2, and RQ3.
- Receive suggestions for debugging issues such as table joins and group-by summaries.
- Help polish the English writing style of the report (for example, in the methods, results, and interpretation sections).

All code was run, checked, and modified by us. All final analysis choices and interpretations in this report are our own.