

# 1 DATA 542 Project – Milestone 2: Mid-Project Report

## 2

3 HAOZHONG JI, University of British Columbia Okanagan, Canada

4 YIN-WEN TSAI, University of British Columbia Okanagan, Canada

## 7 1 Research Questions and Dataset

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

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

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

## 20 2 Methodology

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

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

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

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

45  
46  
47 Authors' Contact Information: Haozhong Ji, jhzzs@student.ubc.ca, University of British Columbia Okanagan, Kelowna,  
48 Canada; Yin-Wen Tsai, ywtsai@student.ubc.ca, University of British Columbia Okanagan, Kelowna, Canada.

### 50      3 Results

#### 51      3.1 Results for RQ1

52      **3.1.1 Data and grouping.** For RQ1, we use the AI Dev dataset and focus on pull requests (PRs)  
 53      opened by AI coding agents. Each PR is linked to its repository, so we can attach repository-level  
 54      features such as stars and activity. We only keep PRs with a clear final status (merged or closed)  
 55      and valid timestamps.  
 56

57      To measure *popularity*, we use the number of stars on each repository. We rank repositories by  
 58      star count and then cut the rank into three equal-sized groups: *Low*, *Medium*, and *High* popularity.  
 59      This rank-based grouping helps reduce the effect of a few extremely famous projects.  
 60

61      To measure *activity*, we build a simple activity score from repository-level events (for example,  
 62      commits, issues, and pull requests). We then split this score into three groups: *Low*, *Medium*, and  
 63      *High* activity.  
 64

65      For both popularity and activity, we count how many AI-generated PRs each agent creates in  
 66      each group. Then we convert these counts to proportions within each group. This gives two main  
 67      tables:  
 68

- 66      • popularity group × agent (pop\_agent),  
 67      • activity group × agent (act\_agent).

68      We also visualize these distributions using grouped bar charts, with one panel for popularity  
 69      and another for activity.  
 70

71      All percentages in the report are rounded to one decimal place. This means that very small but  
 72      non-zero shares (for example, less than 0.05%) can appear as 0.0% in the tables and plots. In those  
 73      cases, we describe them as “negligible but non-zero”, since they exist in the raw counts but do not  
 74      change the main patterns.  
 75

76      **3.1.2 Popularity and agent usage.** Overall, **OpenAI\_Codex** dominates AI usage in all popularity  
 77      groups, but the mix of agents changes with popularity.  
 78

79      In *low-popularity* repositories, OpenAI\_Codex accounts for about 90.8% of AI PRs (260,202 PRs).  
 80      Copilot contributes about 8.4% (24,193 PRs), and Claude\_Code about 0.8% (2,247 PRs). Other agents  
 81      such as Cursor and Devin only appear, if at all, at negligible levels in this group. Their shares are so  
 82      small (below 0.1%) that they round to 0.0% in our summary tables, so they do not affect the overall  
 83      picture.  
 84

85      In *medium-popularity* repositories, the distribution becomes even more concentrated. Here,  
 86      essentially all AI PRs (around 100%) are opened by **OpenAI\_Codex** (286,642 PRs). Any other agents  
 87      in this group, if present, contribute at most a very small number of PRs. Their shares are too small  
 88      to be visible at one-decimal rounding and can be treated as negligible for this analysis.  
 89

90      In *high-popularity* repositories, the picture becomes more diverse. OpenAI\_Codex still leads,  
 91      but its share drops to about 74.7% (214,188 PRs). Copilot contributes about 5.6% (16,022 PRs), and  
 92      Claude\_Code about 0.9% (2,487 PRs). This is also where Cursor and Devin reach noticeable levels:  
 93      about 9.0% (25,880 PRs) for Cursor and 9.8% (28,066 PRs) for Devin. There may be a few additional  
 94      agents with very small shares (again below 0.1%), but they are negligible compared with these main  
 95      tools.  
 96

97      These results suggest that new or alternative AI agents (such as Cursor and Devin) are mainly  
 98      used in more popular projects. Less popular repositories rely almost entirely on OpenAI\_Codex,  
 99      with some additional usage of Copilot and Claude\_Code, and only negligible use of other agents.  
 100

101     **3.1.3 Activity and agent usage.** The patterns by *activity* are similar, but not identical.  
 102

103     In *low-activity* repositories, OpenAI\_Codex again dominates with about 88.7% of AI PRs (254,194  
 104     PRs). Copilot has about 10.3% (29,414 PRs), and Claude\_Code around 1.1% (3,034 PRs). Other agents,  
 105

99 including Cursor and Devin, only appear (if at all) at extremely small levels. Their shares are below  
 100 our rounding threshold, so they show as 0.0% in the tables.

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

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

111 So, repositories with higher activity levels are more likely to use a wider range of AI agents.  
 112 Low- and medium-activity projects mostly depend on OpenAI\_Codex, with some Copilot and  
 113 Claude\_Code and only negligible use of other tools.

114  
 115 **3.1.4 Summary of RQ1 Findings.** Across both popularity and activity, **OpenAI\_Codex** is  
 116 the main AI agent in the AIDev dataset. It accounts for almost all AI-generated PRs in medium-  
 117 popularity and medium-activity repositories, and it is still the majority agent in the other groups.

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

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

## 128 **3.2 Results for RQ2**

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

133  
 134 **3.2.1 Merge Rate.** According to Figure 1, small PRs show the highest merge rates across all  
 135 file types, typically ranging from 85% to 100%. Test-dominated PRs exhibit the strongest merge  
 136 likelihood, while code- and other-related PRs merge slightly less frequently but still maintain high  
 137 overall acceptance.

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

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

143  
 144 **3.2.2 Time-to-Merge.** Small PRs in Figure 2 merge rapidly, often within minutes or hours.  
 145 Medium PRs require slightly more review time, particularly for config-heavy changes.

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

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

153  
 154 **3.2.3 Review Comments.** Review activity follows a similar size-dependent trend from Figure 3.  
 155 Small PRs receive roughly 2.7–2.9 comments on average, except test PRs, which receive significantly  
 156 fewer ( 0.7 comments), indicating lower review friction.

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

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

163  
 164 **3.2.4 Summary of RQ2 Findings.** Across merge rate, merge time, and review comments, results  
 165 align around two consistent insights:

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

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

## 173 4 Interpretation

### 174 4.1 RQ1 Interpretation

### 175 4.2 RQ2 Interpretation

176 The results indicate that both PR size and dominant file type meaningfully influence merge behavior.  
 177 Across all file types, merge rates consistently decrease as PR size increases, suggesting that reviewers  
 178 are more hesitant to approve large AI-generated changes. This pattern aligns with general software  
 179 engineering practices, where larger PRs carry higher review cost and perceived risk.

180 File type further differentiates review outcomes. Configuration-dominant PRs show the slowest  
 181 time-to-merge and the highest review activity, likely because configuration errors can have system-  
 182 wide impact and therefore require more careful validation. In contrast, documentation PRs are  
 183 merged more quickly and with fewer comments, consistent with their lower risk profile. Small test  
 184 PRs also show very high merge rates, indicating low resistance to accepting additional tests.

185 Overall, AI-generated PRs do not bypass typical reviewer behaviors. Instead, they reinforce  
 186 existing patterns:

- 187 (1) small, focused changes are easier to merge,
- 188 (2) certain file types (e.g., configuration) trigger more scrutiny,
- 189 (3) review workload scales with PR size.

190 These observations suggest that the effectiveness of AI-generated contributions depends not  
 191 only on content quality but also on how the changes are structured and scoped.

197    **5 Link to GitHub Repository**

198    [data 542 project](#)

199

200    **6 Remaining Work to Be Completed**

201

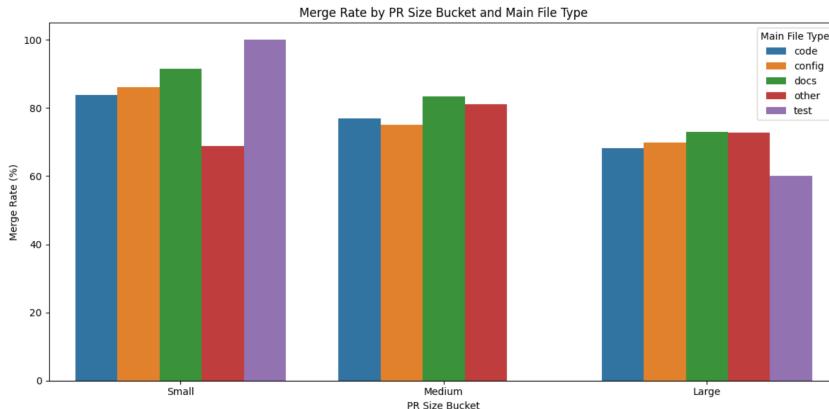
We will add the interpretation section for RQ1, complete the full analysis and reporting for RQ3, and finalize the code repository. The final project report will also be refined to ensure cohesion and consistency across all research questions.

204

205    **A Appendix**

206

207

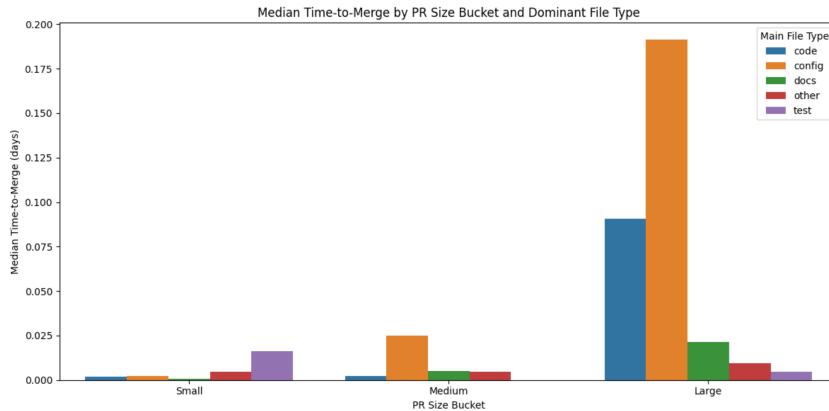


221    Fig. 1. Merge Rate by PR Size Bucket and Main File Type

222

223

224



238    Fig. 2. Median Time-to-Merge by PR Size Bucket and Dominant File Type

239

240

241

242

243

244

245

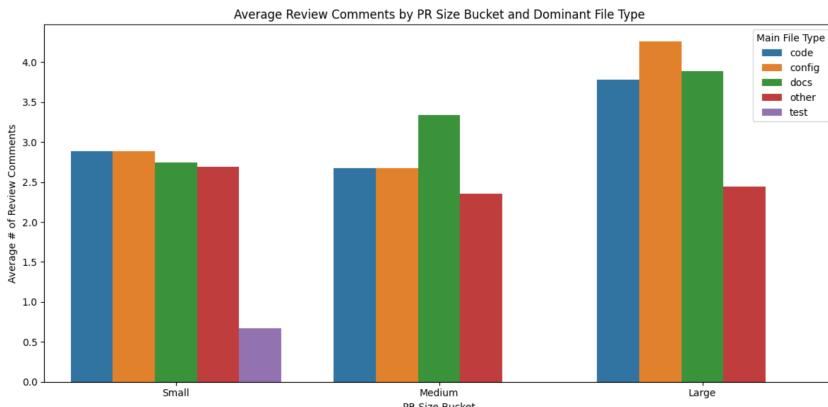


Fig. 3. Average Review Comments by PR Size Bucket and Dominant File Type