Milestone 2

GROUP 4: ANDRE GNANDT AND BINGZHENG JIN*

1 Progress on Research Questions

1.1 RQ1

Our first research question is to explore certain statistics related to AI agents contributing code tests to their respective repositories or code base. We want to explore the frequency in which tests are contributed and the type of tests contributed.

For Mile Stone 2, we have generally decided to skip this research question and leave its exploration for Mile Stone 3.

1.2 RQ2 - Andre Gnandt

Our second research question is to explore the frequency in which AI agents make contributions (pull requests in this case) that contain code errors or code quality issues. We also want to explore the proportion of different types of errors that occur (i.e. legitimate bugs vs errors vs code style problems vs code quality issues, etc.).

For the most part, this research question has been covered in this Milestone, (although we will likely seek to improve the quality of our findings/results in Milestone 3).

1.3 RQ3 - Bingzheng Jin

We want to explore the rate at which AI agents produce code contributions. It would be interesting to examine just how efficient at producing correct and reliable code these agents are, and the overall time it takes for an agent to complete its task and for its PR to be approved.

For the most part, this research question has been covered in this Milestone, (although we will likely seek to improve the quality of our findings/results in Milestone 3).

2 Data wrangling techniques applied so far

2.1 RQ2 - Andre Gnandt

Data Frame indexing, sub-setting and filtering was frequently used in the exploration of this RQ. In the code, you will see this used in several cases. Including: Getting count of PR's that have PR reviews, getting count of PR's that have PR reviews in a commented state, getting count of PR's that have PR reviews in a changes requested state, getting count of PR's that are imperfect (not initially approved or dismissed) and much more. All of these involved the indexing, filtering, sub-setting, and querying (.query() method)

*Both authors contributed equally to this research. Andre Gnandt contributed most of RQ2 and Bingzheng Jin contributed most of RQ3

Author's Contact Information: Group 4: Andre Gnandt and Bingzheng Jin.

operations between the pr_reviews and pr_review comment tables to identify PR's with code issues of different severities (significant issues, minor issues, possible issues, and "all imperfect PR's" - which includes have any PR review in a state of commented or changes requested).

Text Processing using dictionaries was utilized along with aggregation and transformation (using .apply() method) to match all PR's with possible code issues (as described earlier) to certain categories of code issues based on matching keywords in the texts of their body fields. This was also used to identify PR reviews in a "commented state" that have PR review comments or bodies with texts that match code issues identifying keywords, to better identify PR reviews in commented state which actually have code issues.

These techniques were used to obtain the values for: count of all PR's that are imperfect (either had a review with commented or changes requested state), count of all PR's with possible code issues (had a PR review with a commented state), count of all PR's with minor code issues (had a PR review with a commented state and text matched comments or body that identify code issues), count of all PR's with significant code issues (had a PR review with a "changes requested" state, and the number of PR's that have code issues belonging to a certain "code-issue" category".

These 4 counts of issue-identifying PR's were compared to the total number of PR's that have reviews to obtain the metrics on the frequency of code issues in AI Agent PR's and severity of such, and plotted in bar charts for analysis. The number of PR's that have code issues belonging to a certain category was plotted on a pie chart (for each category) for analysis.

2.2 RQ3 - Bingzheng Jin

The analysis uses the AIDev-pop dataset, publicly hosted on Hugging Face. Four data tables are relevant:

(1) pull_request.parquet — main PR metadata including creation time, closure time, merge state, and agent attribution.
(2) issue.parquet — repository issue metadata, including initial creation time.
(3) related_issue.parquet — mapping between PRs and their corresponding issues.
(4) pr_commit_details.parquet — commit-level code changes including additions and deletions.

We restrict our analysis to:

- PRs that are closed
- PRs with non-null agent fields

This ensures that only valid agent-generated software contributions are studied.

Temporal Metrics

For each PR, we compute:

- Issue → PR creation time
- Issue → PR closure time

- Issue → PR merge time (if available)

Time durations are converted into days using timestamp differences. These intervals help quantify how long it takes for issues to be addressed and PRs to be reviewed and completed.

Code-Change Metrics
We measure modification complexity by aggregating commit-level changes:

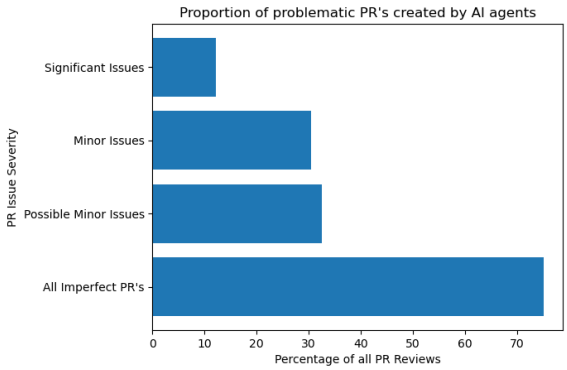
- Total additions
- Total deletions
- Total changes = additions + deletions

Aggregating changes across all commits linked to each PR helps determine whether agent-generated edits tend to be small or large in scale.

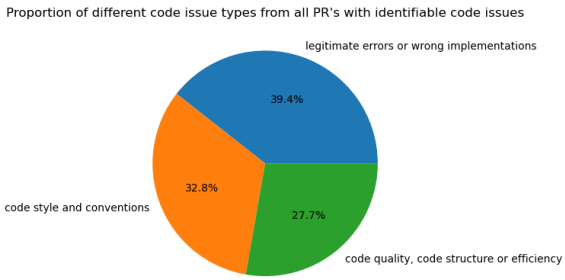
We calculate descriptive statistics for both temporal and code metrics, and visualize distributions using histograms and boxplots. The objective is to observe skewness, spread, and presence of extreme values.

3 Preliminary findings/results

3.1 RQ2 - Andre Gnandt



From the results we see that approximately 12.14% of AI Agents PR's contain significant code issues, 30.37% contain minor issues, 32.51% possibly contain minor issues, and 75.04% of their PR's are not perfect and required some form of reconsideration before merging (includes all other 3 categories). The first 3 categories are disjoint while the last one is the union of the first 3. From this, we know that 12.14+30.37% = 42.51% of PR's contain some form of code issues (mostly minor).



39.4% of code-issues are "legitimate errors or wrong implementations", 32.8% are issues with "code style and conventions", and 27.7% are issues with "code quality, code structure or efficiency".

3.2 RQ3 - Bingzheng Jin

Time Efficiency

Durations from issue creation to PR closure are heavily skewed to the right and exhibit long-tail characteristics. Most PRs complete within a moderate time range, but some take significantly longer. This implies that agent-generated PRs do not uniformly guarantee rapid turnaround time. Factors such as repository policy, review workload, and change complexity may contribute to long delays.

Code Modification Complexity

Code modifications vary widely:

- Many PRs involve relatively small edits
- Some PRs involve significant additions and deletions, resulting in high total change volume
- The boxplot shows strong asymmetry with large upper outliers

This suggests that agent-driven contributions are not consistently lightweight. Some edits resemble substantial refactoring work or larger feature patches, potentially increasing review cost.

4 Link to your GitHub repository

GitHub repo link: https://github.com/andre-gnandt/DATA542-Project

Please see the markdown file for instructions and folders RQ1, RQ2 and RQ3 for the appropriate code for each research question.

5 Remaining work to be completed

5.1 RQ1

We must focus mostly on RQ1 for Milestone 3, as this is the remaining research question that we have left for this part of the project. Although we started a bit on this code, we will have to complete this entire RQ for the most part. It will involve querying the pull_request, pr_commits and pr_commit_details tables to match records to PR's that contribute code tests and the types of tests. These metrics will be used to plot charts to analyze the frequency of test contributions and test types. Any further improvements to text processing in RQ2 (as mentioned below) will also be applied here for texts related to code tests.

5.2 RQ2 - Andre Gnandt

Although RQ2 is mostly complete, there are some improvements that may need to be made if we have enough time to do so. This will involve improving the quality and accuracy of the text processing techniques used, either by:

1. Increasing the size and accuracy of the keywords used to match code issue related texts, and adding more code issue categories.

and/or

2. Using NLP techniques such as Word2Vec to more accurately identify text with context related to code issues.

5.3 RQ3 - Bingzheng Jin

RQ3 is basically completed in this Milestone. However, we may seek to make a few improvements in Mile stone 3. If so, then these improvements will be discussed in that Milestone 3's report.