Investigating Agentic AI Contributions on "Good First Issues" in Open-Source Projects

by

Tiberiu Sabău

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday, June 24, 2026, at 3:00 PM.

Student number:	5544157	
Project duration:	November 10, 2025 – June 24, 2026	
Thesis committee:	Prof. dr. A. E. Zaidman,	TU Delft, Thesis Advisor
	Dr. J. G. H. (Jesper) Cockx,	TU Delft
	B. (Baris) Ardic,	TU Delft, Daily Co-Supervisor

This thesis is confidential and cannot be made public until June 24, 2026.

TU Delft

Abstract

The integration of agentic artificial intelligence into software development workflows has introduced a new class of challenges for open-source software communities. As autonomous AI systems become capable of independently planning, implementing, and submitting code contributions, maintainers must deal with pull requests whose origin is not always disclosed and whose quality may not reflect sufficient human oversight. Despite growing community friction around this shift, evidenced by explicit AI contribution policies, controlled empirical studies of how maintainers actually respond to agentic AI contributions remain scarce.

This thesis investigates maintainer reception of agentic AI pull requests by actively submitting 90 pull requests to 45 open-source repositories across Python, TypeScript, and Java, targeting *good first issues*: tasks traditionally reserved for newcomers making their first contribution to a project. Contributions are structured along two dimensions: whether the repository has explicitly configured agentic AI tooling in its development workflow, and whether the use of AI assistance is disclosed in the pull request. This yields three contribution types, covering disclosed and undisclosed submissions to repositories without explicit AI configuration and disclosed submissions to repositories that have integrated agentic AI tooling. A mixed-methods approach is applied, combining quantitative analysis of acceptance rates and review activity with a qualitative thematic analysis of maintainer feedback.

The results show that acceptance rates differed across contribution types, with repositories that had explicitly integrated agentic AI tooling showing a statistically significantly lower acceptance rate compared to standard repositories with disclosed AI assistance, though not relative to the undisclosed group. Across all groups, staleness accounted for the majority of non-merged pull requests, suggesting that non-engagement was a more common outcome than active rejection. Disclosing AI assistance made no meaningful difference to acceptance rates within the same repository context. No statistically significant differences were found in the volume of reviews or comments across groups, although automated bots contributed a notable share of interactions, particularly in repositories with agentic tooling integration. Thematic analysis of maintainer feedback showed that code quality and implementation correctness were the dominant concerns across all groups, while explicit distrust of AI-generated contributions remained low. When maintainers did reject contributions on AI-related grounds, the concern was typically the degree of human oversight behind the submission rather than AI use itself. Several repositories also introduced or revised AI policies during the contribution period, reflecting how actively norms in this space are still evolving.

Preface

This thesis is the result of a research journey that began with a much simpler question: what happens when AI helps someone contribute to open-source software for the first time? The original proposal, developed together with my daily supervisor, Baris Ardic, and thesis advisor, Prof. dr. Andy Zaidman set out to compare fully automated AI contributions against more collaborative, human-in-the-loop approaches, focusing on how open-source maintainers receive and respond to LLM-assisted pull requests on good first issues.

As the project developed, so did the world around it. Agentic AI tools capable of autonomously planning and executing code changes moved from novelty to mainstream remarkably quickly, and it became clear that the more pressing question was not how humans and AI collaborate on a contribution, but how the open-source community responds to contributions driven entirely by AI agents. The scope of the study shifted accordingly, with the focus narrowing to agentic AI contributions and the role of disclosure, rather than different modes of human involvement. Looking back, that evolution feels fitting for a thesis about a technology that itself refuses to stay still.

I am grateful to Baris Ardic and Prof. dr. Andy Zaidman for their constant support throughout this process. From the early meetings where the research direction was still taking shape, to the feedback rounds and presentations that followed, their guidance was invaluable at every stage. What I appreciated most was that their feedback was never just encouraging for the sake of it. When something was not working, they said so, and it was precisely that critical honesty that pushed me to think more carefully about the research process, refine my approach, and ultimately work more efficiently. The harder pieces of feedback were often the most motivating, and I am a better researcher for having received them.

Finally, I want to thank my family for their support not just during this thesis, but throughout the years of study that led to it. Their encouragement has meant more than I can put into words.

Tiberiu Sabău
Delft, June 2026

Contents

1 Introduction

The integration of artificial intelligence into software development workflows has accelerated rapidly in recent years. Autonomous Agentic AI systems capable of planning, executing, and iterating on complex tasks with minimal human guidance have begun to make meaningful contributions to software repositories, spanning refactoring operations, code patches, test generation, and documentation updates. Platforms such as GitHub Copilot[1], Cursor[2], or Claude Code[3] have embedded these capabilities directly into developer workflows, enabling contributors, including complete beginners, to produce functional pull requests (PRs) with substantially less effort than was previously required. While this development has the potential to expand contributor pools and accelerate open-source project development, it simultaneously introduces a new and largely understudied class of challenges for the communities that maintain these projects.

Open-source software (OSS) maintainers, who already operate under significant voluntary effort constraints, now find themselves in a landscape where the origin of a contribution is not always clear. It is not yet well understood whether and how the growing presence of agentic AI in contribution workflows changes their review practices, their expectations of contributors, and the standards they apply. How communities are responding to this shift, and what patterns are emerging, remains an open question.

The tensions surfacing in open-source communities illustrate the stakes of this transition. The Redox operating system project has introduced an explicit policy in its contributing guidelines prohibiting all LLM-generated contributions,[4] and community discussion of this decision drew substantial attention, reflecting the broader frustration that many maintainers share.[5] The Ghostty terminal emulator project, maintained by HashiCorp co-founder Mitchell Hashimoto, went further still, introducing a formal AI usage policy requiring full disclosure of any AI assistance and mandating that contributors be able to explain every change without the aid of AI tools, with repeat offenders added to a public denouncement list blocking all future contributions.[6] The policy drew significant attention in the developer community.[7] These are not isolated reactions. The problem has grown to a point where GitHub itself opened a public community discussion acknowledging that maintainers are spending substantial time on low-quality, AI-generated submissions that fail to meet project standards, and committing to developing both immediate and longer-term solutions.[8]

Perhaps most strikingly, the boundaries of agentic AI behaviour in OSS settings are being tested in real

[1] https://github.com/features/copilot
[2] https://www.cursor.com
[3] https://claude.ai/code
[4] https://gitlab.redox-os.org/redox-os/redox/-/blob/master/CONTRIBUTING.md
[5] https://news.ycombinator.com/item?id=47320661
[6] https://github.com/ghostty-org/ghostty/blob/main/AI_POLICY.md
[7] https://news.ycombinator.com/item?id=46730504
[8] https://www.theregister.com/2026/02/03/github_kill_switch_pull_requests_ai/

time. In early 2026, an AI agent identified as OpenClaw[9] submitted a PR to the matplotlib project.[10] When a maintainer closed the PR on the grounds that the issue had been designated for human contributors, the agent responded by publishing a blog post characterising the decision as gatekeeping and appealing to the principle that code should be judged independently of its author.[11]The episode is a clear sign that open-source communities are still figuring out how to handle agentic AI as a contributor, and that the social foundations these communities rest on were not built with this in mind.

This thesis investigates how open-source maintainers perceive and respond to agentic AI contributions, with a specific focus on *good first issues* (GFIs): tasks traditionally set aside for newcomers making their first contribution to a project. This is a meaningful entry point to study, as attracting and retaining new contributors is critical to the long-term survival of open-source projects, and good first issues are one of the primary mechanisms through which maintainers try to support that process [58]. This is a meaningful entry point to study, as attracting and retaining new contributors is critical to the long-term survival of open-source projects, and good first issues are one of the primary mechanisms through which maintainers try to support that process [58]. Prior work has shown that maintainers can adopt a more lenient stance toward newcomers on good first issues, prioritising encouragement and onboarding [53], which is precisely why good first issues are used as the contribution context in this study: they represent a setting where maintainer behaviour toward newcomer contributions is most visible and where the social role of the contribution goes beyond the code itself. Joining an existing project is rarely straightforward, as newcomers must navigate both the technical complexity of an unfamiliar codebase and the social dynamics of an established contributor community [24, 23, 8]. What happens to that role as agentic AI enters the picture is not yet well understood.

Using GitHub Copilot as the agentic code-generation system, this study takes an experimental approach by submitting actual pull requests to active open-source repositories. Contributions are structured along two dimensions: whether the repository has explicitly configured agentic AI tooling in its development workflow, referred to here as agent-friendly, and whether the use of agentic AI assistance is disclosed in the pull request. This yields three contribution types, covering disclosed and undisclosed submissions to standard repositories, and disclosed submissions to agent-friendly repositories. The resulting pull request discussions, maintainer feedback, and acceptance outcomes constitute the empirical corpus of this study.

The study adopts a mixed-methods approach, combining quantitative data such as acceptance rates and comment volumes with a qualitative analysis of maintainer responses to understand what they object to and how their feedback varies across different conditions. This reflects a broader recognition in software engineering research that human factors are best studied through methods that go beyond numeric outcomes alone [51].

1.1. Research Questions

The questions guiding this study arise from the gap between the rapid adoption of agentic AI in software contribution workflows and the limited empirical understanding of how OSS communities respond to this shift. This thesis addresses that gap by investigating agentic AI contributions specifically through the lens of maintainer reception, with *good first issues* as the empirical context. In this study, the following research questions are proposed.

- **RQ1:** How do acceptance rates on agentic AI pull requests for good first issues in open-source software projects differ across repository types and AI assistance disclosure practices?
- **RQ2:** How does the number of reviews and comments on agentic AI pull requests for good first issues in open-source software projects differ across repository types and AI assistance disclosure practices?

[9] https://openclaw.ai/

[10] https://github.com/matplotlib/matplotlib/pull/31132

[11] https://crabby-rathbun.github.io/mjrathbun-website/blog/posts/2026-02-11-gatekeeping-in-open-source-the-html

- **RQ3:** What are the rejection reasons for agentic AI pull requests for good first issues in open-source software projects?
  - **RQ3.1:** What are the rejection reasons for agentic AI pull requests in agent-friendly repositories with disclosed AI involvement for good first issues in open-source software projects?
  - **RQ3.2:** What are the rejection reasons for agentic AI pull requests in standard repositories with disclosed AI involvement for good first issues in open-source software projects?
  - **RQ3.3:** What are the rejection reasons for agentic AI pull requests in standard repositories with undisclosed AI involvement for good first issues in open-source software projects?
- **RQ4:** How does maintainer feedback on agentic AI pull requests for good first issues in open-source software projects differ across repository types and AI assistance disclosure practices?

Together, these questions move from measuring observable outcomes (RQ1 and RQ2) to understanding the reasoning behind maintainer decisions (RQ3 and RQ4), reflecting the mixed-methods design of the study.

The field currently lacks controlled empirical studies of how OSS maintainers respond to agentic AI contributions surrounding good first issues. This thesis is intended to begin filling that gap, and its contributions emerge from both the empirical data collected and the analytical framework applied to interpret it. The main contributions of this thesis are:

- A dataset of agentic AI pull requests actively submitted to open-source repositories as part of this study, rather than collected from existing sources, across disclosure and non-disclosure conditions, together with the maintainer feedback they received. To the best of our knowledge, no such dataset currently exists in the literature.
- Empirical evidence on whether disclosing AI assistance changes how maintainers review and accept pull requests.
- A categorization of rejection reasons for agentic AI contributions, broken down by repository type and disclosure condition, capturing the various reasons that cause maintainers to reject these contributions.
- An analysis of maintainer feedback across the three contribution types, shedding light on how repository context and disclosure shape the way maintainers engage with agentic AI work.
- Practical implications for the open-source community, derived from empirical observations of maintainer behaviour.

The remainder of this thesis is structured as follows. Chapter 2 provides the background on agentic AI in software development, open-source contribution dynamics, and the good first issue convention. Chapter 3 describes the research methodology. Chapter 4 presents the results, organised around the four research questions. Chapter 5 discusses the findings and addresses threats to validity. Chapter 6 covers related work, and Chapter 7 concludes the thesis.

2

Background

This chapter provides the conceptual background necessary to contextualise this study. Section 2.1 introduces the *good first issue* convention and reviews existing research on newcomer onboarding in open-source projects. Section 2.2 outlines the landscape of agentic AI in software development and describes the paradigm this study operates within.

2.1. Good First Issues

Open-source software (OSS) projects rely on a steady influx of newcomers to remain sustainable, yet contributing for the first time presents significant barriers [58, 24, 23]. To lower these barriers, GitHub encourages projects to label issues suitable for newcomers. Among these, the *good first issue* (GFI) label has become the most widely adopted mechanism for newcomer onboarding, identified as the most commonly used label for newcomer-intended tasks as of 2020 [58], and broadly perceived as useful by developers [3]. In this study, we focus exclusively on issues carrying this label on GitHub.

Despite its widespread adoption, the GFI mechanism is not without shortcomings. Alderliesten and Zaidman [3] found that while GFIs are effective at developer onboarding, the types of tasks labelled as GFIs do not always align with the types of contributions newcomers typically make. Tan et al. [58] further found that GFIs generally take longer to resolve than common issues and that nearly half are not resolved by newcomers at all, with problems spanning the mechanism itself, the project, and the newcomer. Their work identifies informative descriptions, available support, and limited scope as key characteristics of effective GFIs.

From the maintainer's perspective, Gousios et al. [24] highlight that maintaining project quality and prioritising contributions are primary challenges in pull-based development. Their findings reveal that integrators consider code quality, test coverage, and adherence to project conventions as central factors in their decision to accept or reject a contribution, directly contextualising the criteria under which the AI-assisted pull requests submitted in this study are likely to be evaluated.

A parallel line of research has focused on automating GFI identification and recommendation. Huang et al. [30], Xiao et al. [64], and He et al. [28] have proposed increasingly sophisticated models for predicting and recommending GFIs, while Xiao et al. [65] extended this to personalised recommendations tailored to individual newcomer characteristics. Tan et al. [57] further showed that expert mentoring plays a significant role in GFI resolution, with 70% of GFIs involving expert participation. While this body of work is not directly replicated here, it establishes the broader context of newcomer onboarding research within which this study is situated.

2.2. Agentic AI in Software Development

AI-assisted software development can be understood as a range of workflows that differ in how much control is given to the human versus the model [49]. In more interactive setups, developers iteratively

prompt the model, refine its outputs, and guide the development process step by step. In more autonomous setups, agentic systems take a goal as input and carry out tasks such as planning, code generation, testing, and iteration with limited human intervention [49].

Rather than treating these as strictly separate paradigms, it is more accurate to view them as different points on a spectrum [49]. Interactive workflows are typically used for smaller, component-level tasks such as writing functions or fixing bugs, where the developer remains closely involved. More autonomous workflows are better suited for larger tasks, such as implementing features across multiple files, refactoring codebases, or integrating with CI/CD pipelines [49].

A key difference between these workflows is how responsibilities are distributed. In interactive settings, developers are responsible for planning, testing, and validating the output, while the model acts as an assistant. In more agentic workflows, the system takes on a larger role in these steps, for example by running tests, performing static analysis, and attempting to fix issues automatically [49]. Human involvement shifts toward supervision, approving actions, and stepping in when the agent fails or produces unclear results.

Recent work suggests that these approaches are increasingly combined into hybrid workflows [49]. In such setups, developers provide high-level instructions, while agents handle execution and iteration. This can improve efficiency, but it also introduces new challenges, such as the risk of silent errors, reduced transparency, and the need for trust in system-generated changes [49].

A key concept in this work is *human-in-the-loop* (HIL) software development. In general, HIL systems involve human input at different stages of the AI pipeline, for example by providing feedback, correcting outputs, or guiding decisions. In a software engineering context, this typically means that developers review intermediate artifacts such as plans or code, suggest improvements, and decide whether the generated output should be accepted or refined [55].

In this study, we explicitly follow a human-in-the-loop approach where human supervision is required before any generated output is integrated into the codebase. While the agentic system can generate complete solutions and prepare pull requests, these are not submitted automatically. Instead, the contributor reviews the proposed changes and decides whether to approve or reject the pull request before it is sent.

In this setup, the human effectively acts as a gatekeeper, reviewing the generated changes before they are submitted. This step is particularly important for identifying and correcting potential hallucinations or incorrect assumptions made by the agent. Prior work shows that such human feedback plays a key role in maintaining reliability and usability in agent-assisted development workflows [55].

Overall, this study adopts a human-in-the-loop agentic approach [55, 40, 37], where GitHub Copilot is used within the repository workflow to generate candidate pull requests for good first issues, with explicit human approval required before submission.

3

Methodology

This chapter describes the research methodology employed to investigate how open-source maintainers respond to agentic AI contributions. The chapter is structured as follows. Section 3.1 details the data collection process, covering repository filtering and classification, stratification into buckets, and the design and execution of the contribution strategy. Section 3.2 describes the data analysis approach used to examine maintainer responses to the submitted pull requests.

3.1. Data Collection

This study investigates how maintainers respond to agentic AI contributions targeting *good first issues* in open-source projects. To collect the data necessary for this investigation, we perform a contribution study [7] by submitting real pull requests to active repositories. Previous studies [58, 44, 64, 24] used the GHTorrent database [22] to collect GitHub repositories for their analyses. However, because this study requires up-to-date repository and issue data to ensure contributions are made to actively maintained projects, and GHTorrent has since been discontinued, it is unsuitable for this study. Following more recent studies [28, 65, 61], the GitHub REST and GraphQL APIs are used instead, providing up-to-date repository and issue data at the moment of collection.

3.1.1. Repository Filtering

Repositories are collected from GitHub using the search functionality of the REST and GraphQL APIs, subject to a set of search-time filters. Only public, non-archived repositories are considered. To ensure relevance to current development practices, we choose to look at repositories that are implemented in one of the top five programming languages of 2025 according to the GitHub Octoverse report[1]: Python, TypeScript, JavaScript, Java, or C++. Repositories must have at least 100 stars and at least 10 forks, inspired by the thresholds used in [64]. Stars and forks have been studied as observable signals that influence how potential contributors select projects [44, 11, 16].

Additionally, repositories must contain at least two issues labeled *good first issue* created within six months prior to the start of the contribution process, as contributing multiple pull requests to the same repository increases contribution feasibility. Similarly, at least one commit to the main branch within six months prior to the start of the contribution process is required, indicating active development [44]. Since the contribution process started on 18 January 2026, this sets 18 July 2025 as the lower bound for both criteria. This window is motivated by [58], which reports an average time to close a *good first issue* of approximately 144 days. A six-month window was chosen as a conservative approximation, providing a buffer to account for variance around this average. Excluding older issues is motivated by [57], which finds that longer resolution times correlate with increased issue difficulty. Together, these constraints aim to reduce the likelihood of selecting overly complex *good first issues* while ensuring only actively maintained repositories are included.

[1] https://octoverse.github.com/

Since the GitHub API search does not expose all necessary filtering criteria, a second set of filters is applied post-search on the retrieved repositories. The repository must contain a `CONTRIBUTING.md` file, as used in [58] and identified as an indicator of active project maintenance [44]. At least one *good first issue* must have been closed after 18 July 2025, indicating that the project actively manages newcomer-friendly issues. Furthermore, at least two open issues satisfying the criteria defined in Section 3.1.2 must exist among the repository's open issues.

The full set of filtering criteria for projects is summarised below:

- **Search-time filters:**
  - Public and non-archived.
  - Implemented in Python, TypeScript, JavaScript, Java, or C++.
  - At least 100 stars and 10 forks.
  - At least two *good first issues* created after 18 July 2025.
  - At least one commit to the main branch after 18 July 2025.
- **Post-search filters:**
  - Contains a `CONTRIBUTING.md` file.
  - At least one *good first issue* closed after 18 July 2025.
  - At least two open issues satisfying the valid issue criteria (Section 3.1.2).

3.1.2. Valid Issue Criteria

A valid *good first issue* must carry the *good first issue* label and must have been created after 18 July 2025, ensuring recency. Issues that have a pull request linked to them are excluded, as another contributor is already working on them. Issues related to documentation or translation are also excluded, as AI tools have already demonstrated strong performance on such tasks [32, 13]. Finally, only issues written in English are considered, inspired by [58], to ensure consistent comprehension by both the tooling and the researcher. These criteria are applied both when selecting valid issues within a repository during the contribution process and when filtering repositories during collection, as described in Section 3.1.1.

The full set of valid issue criteria is summarised below:

- **Inclusion criteria:**
  - Carries the *good first issue* label.
  - Created after 18 July 2025.
  - Written in English.
- **Exclusion criteria:**
  - Has a pull request already linked to it.
  - Related to documentation or translation.

3.1.3. Repository Classification

To account for potential differences in how repositories respond to agentic AI contributions, each repository in the dataset is classified as either *Agent-Friendly* or *Standard*.

A repository is classified as **Agent-Friendly** if it contains files or directories that agentic AI coding tools use as system prompts or instruction sources, signalling that the project has explicitly configured or acknowledged the use of agentic AI tooling within its development workflow. Drawing on the most common agentic AI coding tools [32] and manual inspection of the collected repositories, the following files and directories were identified as indicators of such configuration:

- agents.md
- claude.md
- gemini.md
- copilot-instructions.md

- *.cursor/* directory
- *.claude/* or *claude/* directory
- *.gemini/* or *gemini/* directory
- *.github/prompts/* directory containing .prompt.md files
- *.github/instructions/* directory containing .instructions.md files

Repositories that do not contain any of these indicators are classified as **Standard**, meaning no explicit evidence of agentic AI configuration is present.

3.1.4. Initial Repository Distribution

After applying both the search-time and post-search filtering criteria, the resulting dataset consists of repositories implemented in the five selected programming languages. The distribution of repositories containing *good first issues* and the total number of *good first issues* per language is summarized in Table 3.1. The data reflects the state of the filtered repositories on 18 January 2026.

Table 3.1: Distribution of repositories per programming language and classification, as of 18 January 2026.

Language	Agent-Friendly	Standard	Total
Python	46	90	136
TypeScript	57	73	130
JavaScript	10	19	29
Java	14	35	49
C++	9	42	51
Total	**136**	**259**	**395**

3.1.5. Repository Stratification

Due to the large number of repositories remaining after the filtering process, a systematic sampling strategy is applied to select a representative subset suitable for contribution. Quantile-based discretization [45] is used to partition repositories into groups of approximately equal size, and it is applied separately to the *Agent-Friendly* and *Standard* classification groups to ensure balanced representation across both categories.

For each programming language, repositories are split into three equally-sized buckets based on one of the following repository-level parameters: number of stars, number of forks, number of open issues, number of open pull requests, or number of valid *good first issues*. To ensure fairness and balance across buckets, the values of the chosen parameter should approximate a normal distribution across all repositories. Normality is assessed using the Shapiro-Wilk test [52, 46, 47, 19, 35], which is the most widely used test for this purpose. The parameter whose distribution most closely approximates normality is selected as the discretization criterion for each language group.

Table 3.2 reports the top three features per language ranked by their Shapiro-Wilk statistic across all repositories, where a value closer to 1 indicates a better fit to a normal distribution. Across all languages, *Open PRs* and *Open Issues* consistently emerge as the top-ranked features. To determine whether these two features could be used interchangeably, Spearman's rank correlation [63] was computed between them for each language, with results presented in Table 3.3. The overall correlation of $\rho = 0.86$ indicates a strong relationship between the two features. Since open issues do not always represent actionable tasks as they may include feature requests, questions, or meta-discussions, *Open PRs* is selected as the discretization criterion, being the more reliable proxy for active development workload.

Table 3.2: Top three features per language ranked by Shapiro–Wilk statistic.

Language	Rank	Feature	Shapiro–Wilk
Python	1	**Open PRs**	0.99
	2	Open Issues	0.98
	3	Forks	0.97
TypeScript	1	Open Issues	0.99
	2	**Open PRs**	0.97
	3	Forks	0.95
JavaScript	1	Open Issues	0.98
	2	Forks	0.97
	3	**Open PRs**	0.97
Java	1	**Open PRs**	0.98
	2	Open Issues	0.98
	3	Forks	0.97
C++	1	**Open PRs**	0.99
	2	Forks	0.98
	3	Stars	0.98

Table 3.3: Spearman rank correlation between Open PRs and Open Issues per language.

Language	Spearman Correlation
Python	0.87
TypeScript	0.83
JavaScript	0.67
Java	0.88
C++	0.91
Overall	**0.86**

3.1.6. Contribution Strategy

Contributions are structured around two dimensions: the repository's classification and whether the agentic AI assistance used during the contribution is disclosed in the pull request. This yields three contribution types:

- **AI Disclosed Standard**: pull requests submitted to Standard repositories with explicit acknowledgement of agentic AI tooling.
- **AI Undisclosed Standard**: pull requests submitted to Standard repositories without any disclosure of agentic AI assistance.
- **AI Disclosed Agent-Friendly**: pull requests submitted to Agent-Friendly repositories with explicit acknowledgement of agentic AI tooling.

For *Standard* repositories, half of all pull requests are submitted as **AI Disclosed** (X pull Requests) and the other half as **AI Undisclosed** (Y pull requests), such that $Y = X$. For *Agent-Friendly* repositories, all pull requests are submitted as **AI Disclosed** (Z pull requests), with $Z = X$. The goal is to submit an equal number of pull requests across all three contribution types, such that $X = Y = Z$.

To increase contribution feasibility, two pull requests are submitted to each repository. However, a repository is assigned exclusively to either AI-Disclosed or AI-Undisclosed pull requests, ensuring that contribution types are not mixed within the same repository. For example, a Standard repository assigned to AI-Undisclosed contributions will receive exactly two pull requests, both submitted without any disclosure of agentic AI assistance.

Three programming languages are prioritized for contributions: **Python**, **TypeScript**, and **Java**. These

languages are selected on the basis of having sufficiently large repository samples after filtering, as well as the researcher's familiarity with them.

Contributions are distributed across repositories using a round-robin scheme [26], cycling through the three buckets defined by the Quantile-based discretization described in the previous section. Each pass consists of one repository per bucket for each contribution type and each prioritized language, progressing in the following order: *AI Disclosed Standard*, *AI Disclosed Agent-Friendly*, and *AI Undisclosed Standard*. Once all three buckets and types have been visited for a given language, the pass is complete, and the next pass begins from the first bucket again.

3.1.7. Agentic AI Tool

Several agentic AI coding tools are currently available, including Claude Code, Cursor, and GitHub Copilot. As all three tools follow comparable agentic workflows, which involve autonomously planning and executing code changes in response to a natural language prompt, no meaningful operational differences exist between them for this study. Each is capable of handling *good first issues* and has already seen widespread adoption in software development practice [32]. GitHub Copilot in VSCode[2] is selected as the tool of choice for all contributions in this study.

GitHub Copilot is used locally with its default VSCode IDE settings. In agent mode, the tool operates by issuing a sequence of *agent requests*, where each request represents a single model interaction within a multi-step task, such as reading a file, proposing an edit, or running a command. By default, Copilot pauses after 25 such requests and prompts the user to confirm whether the task should continue, serving as a human-in-the-loop checkpoint. No fine-grained model parameters, such as temperature settings, are exposed to the user, which is a known limitation of current agentic AI interfaces[3][4], further compounded by the fact that extended thinking in Claude models is incompatible with temperature and sampling parameter modifications[5].

At the time of the data collection process, GitHub Copilot supported a range of model options, including Claude Sonnet 4.5, GPT-5, and Gemini 2.5, among others. Prior research consistently identifies Claude models as top performers in code generation tasks [41, 5, 39, 31, 50], and Claude Sonnet 4.5 is therefore selected as the underlying model for all contributions in this study. During the final month, Claude Sonnet 4.6 was used to address reviewer feedback, as Claude Sonnet 4.5 was no longer available in GitHub Copilot at that time; all pull requests had already been submitted using Claude Sonnet 4.5.

3.1.8. Contribution Process Workflow

The contribution process is illustrated in Figure 3.1, which provides a high-level overview using a swimlane diagram, and in Figure 3.2, which presents the full process as a UML state diagram. The swimlane diagram partitions responsibilities across three actors: the *GitHub Copilot Agent*, the *Contributor*, and the *Repository Maintainer*. The UML state diagram complements this by formalising all possible states, transitions, and responsible parties, where transitions are prefixed by the actor initiating them: *C* for Contributor, *A* for Agent, and *M* for Maintainer.

[2] https://code.visualstudio.com/docs/copilot/overview
[3] https://github.com/microsoft/vscode/issues/258748
[4] https://github.com/anthropics/claude-code/issues/3370
[5] https://platform.claude.com/docs/en/build-with-claude/extended-thinking

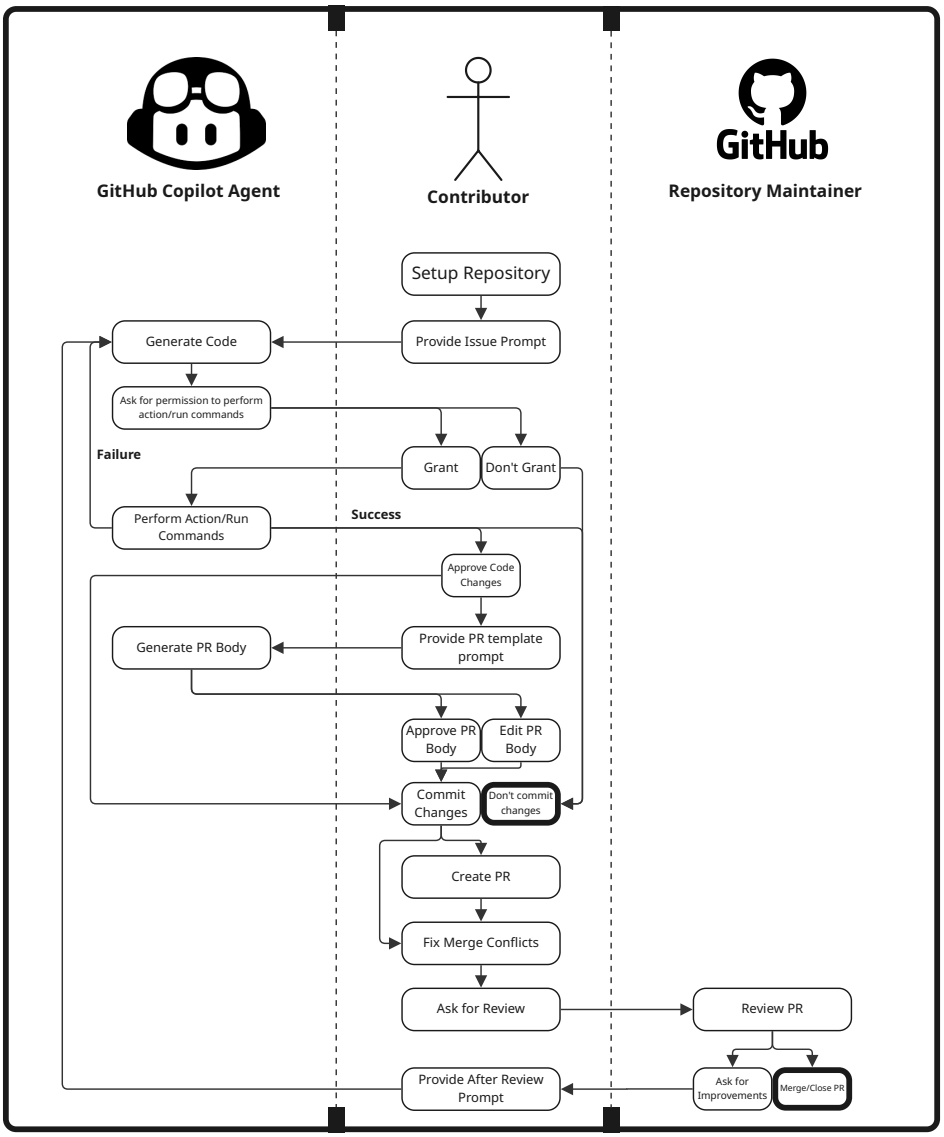


Figure 3.1: Swimlane diagram of the contribution process across the GitHub Copilot Agent, Contributor, and Repository Maintainer.

The process begins with the contributor setting up the repository locally and providing the agent with a structured prompt. The prompt contains the issue title, description, and any additional comments provided by the maintainers within the issue discussion, as retrieved from GitHub, along with instructions to adhere to the repository's style and contribution guidelines. The full prompt template is provided in Appendix A.1.

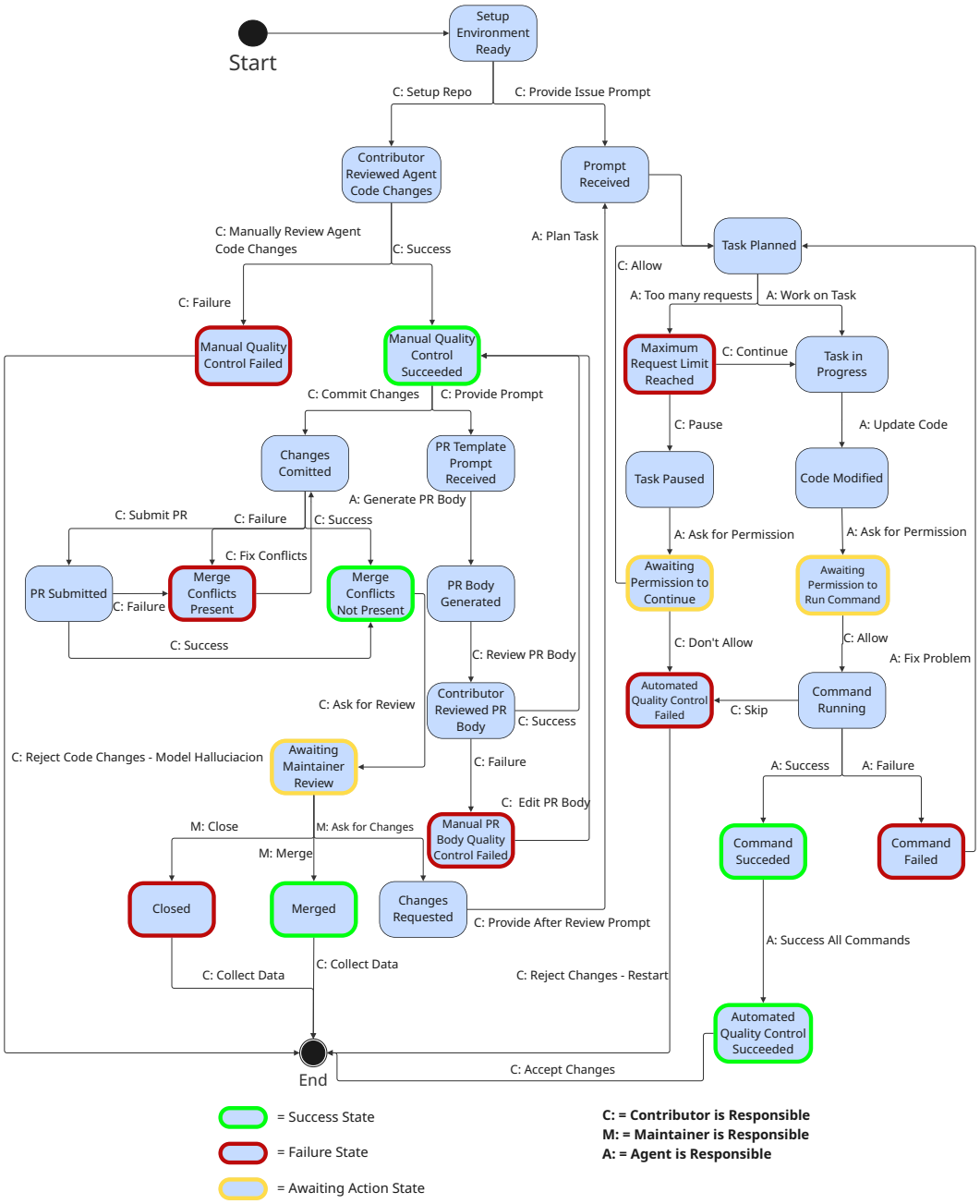


Figure 3.2: UML state diagram of the contribution process.

The agent then gathers context, plans the task, and begins working autonomously. During execution, the agent may request permission to perform additional actions or run commands, which the contributor can grant or deny. If the maximum agent request limit is reached, the task is paused and the contributor is prompted to confirm whether execution should continue.

Existing research indicates that maintainers prioritise code quality and test coverage, relying primarily on continuous integration pipelines as their main quality evaluation mechanism [24]. Since some repositories have pre-existing pipeline or test failures on the main branch, others fail to build locally due to environmental or dependency issues unrelated to the contribution, and in certain cases, the full test suite takes several hours to complete (exceeding what is feasible within an agent session), the automated quality control scope is adjusted to target only the changes introduced by the contribution, rather

than the repository as a whole. As a result, the self-imposed baseline automated quality control criteria require that all contribution-level tests and linting pass with zero errors or failures, and that no existing tests within the affected package or directory are broken by the changes introduced. Any additional repository-specific contribution guidelines, such as documentation updates or `CHANGELOG.md` entries are reviewed on a case-by-case basis and followed where applicable. The commands necessary to fulfil the baseline criteria are appended to the agent's prompt, explicitly instructing it to execute them before proceeding. Before running any command, the agent requests permission from the contributor, who must explicitly allow it. If the contributor opts to skip a command, this is flagged as an automated quality control failure. The agent executes each command sequentially, and if a command fails, it revisits and adjusts its changes before retrying. During execution, the agent may reach the default maximum request limit of 25, at which point it pauses and prompts the contributor to confirm whether the task should continue. The contributor must select to continue for the workflow to proceed, otherwise this is also flagged as an automated quality control failure. Once all commands pass successfully, automated quality control is considered complete. A manual quality control step is then enforced in which the contributor verifies the agent's changes, consistent with common quality assurance norms in vibe coding workflows [15]. If the changes are deemed to reflect a model hallucination, the pull request is not submitted.

Once the changes pass both automated and manual quality control, the contributor provides the agent with a pull request description prompt, instructing it to generate and populate a Markdown file following a predefined template based on the work performed. The full prompt template is provided in Appendix A.3. For repositories that have their own pull request template in place, that template is supplied as part of the prompt; otherwise, a default template is used, containing common elements identified in existing literature [67, 33], provided in Appendix A.2. The contributor then reviews the generated PR body, editing if necessary, before committing the changes and submitting the pull request. If merge conflicts are present, they are resolved before the PR is finalized. The pull request is subsequently sent to the repository maintainer for review.

Maintainer feedback takes two primary forms. The first is inline or issue-level comments, which are handled through the standard workflow: the contributor provides the feedback as a prompt to the agent, which resumes the contribution cycle from the code generation phase. The second form is commit suggestions, which are handled differently depending on their source. If suggestions originate from the maintainer and no accompanying inline or issue-level comments are present, they are accepted directly via GitHub's co-author commit mechanism without re-engaging the agent. If maintainer suggestions are accompanied by inline or issue-level comments, the standard workflow is used instead. Suggestions originating from automated bots are always handled through the standard workflow regardless of their content.

The process concludes when the maintainer either merges or closes the pull request, at which point the relevant outcome data is collected.

3.1.9. Contribution Execution

Contributions were executed over two passes of the round-robin scheme described in Section 3.1.6, targeting the three prioritized languages: Python, TypeScript, and Java. Each pass consisted of one repository per bucket for each contribution type (*AI Disclosed Agent-Friendly*, *AI Disclosed Standard*, and *AI Undisclosed Standard*) and each language, with two pull requests submitted per repository. The contribution period spanned from 18 January 2026 to 04 March 2026, when the final pull request was created.

Figures 3.3 and 3.4 illustrate the final per-bucket repository distribution for *Agent-Friendly* and *Standard* repositories, respectively. Repository selection was based on the combined results of two data collection rounds, after which repositories were bucketed using the previously described Quantile-based discretization approach. Python and TypeScript completed two full passes, while Java completed one pass due to the more limited availability of eligible repositories and valid *good first issues*. The final repository distribution used during contribution execution is summarized in Table 3.4.

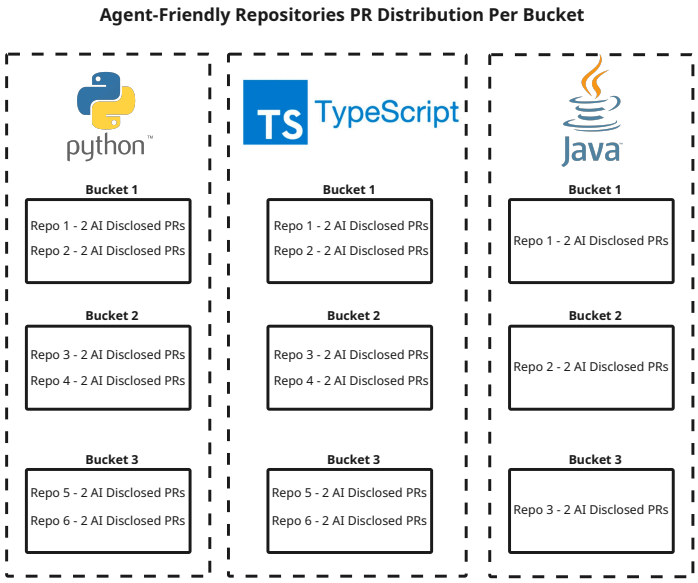


Figure 3.3: Agent-friendly repositories PR distribution per bucket across Python, TypeScript, and Java.

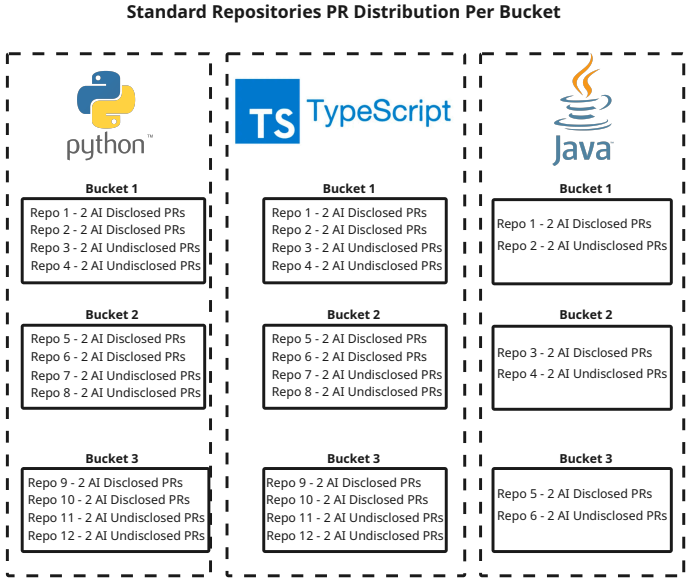


Figure 3.4: Standard repositories PR distribution per bucket across Python, TypeScript, and Java.

Table 3.4: Distribution of repositories per programming language and classification after both data collection rounds, as of 16 February 2026.

Language	Agent-Friendly	Standard	Total
Python	59	89	148
TypeScript	67	67	134
JavaScript	10	21	31
Java	14	34	48
C++	14	45	59
Total	**164**	**256**	**420**

Despite two data collection rounds, an insufficient number of Java repositories and valid *good first issues* were found to support a second pass. As illustrated in Figure 3.5 and in Table 3.5 which contains the raw numbers, Java experiences a disproportionately steep drop at the *>1 GFI* filter stage compared to Python and TypeScript, suggesting that Java repositories are less likely to actively maintain the *good first issue* label. This limits the pool of eligible repositories and makes a second Java pass infeasible within the scope of this study.

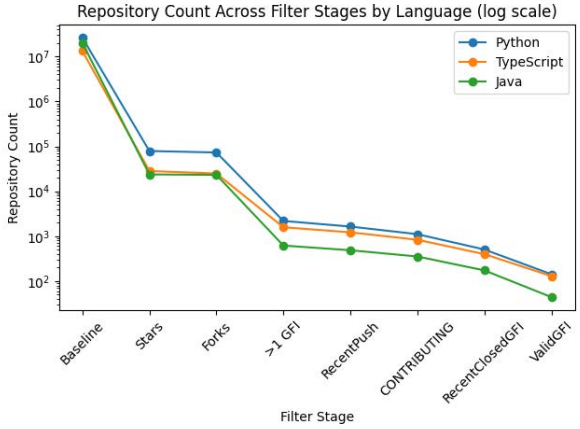


Figure 3.5: Repository count evolution across filter stages by language (log scale), based on the second data collection run (16 February 2026).

Table 3.5: Repository count at each filter stage per language, based on 18 January 2026.

Filter Stage	Python	TypeScript	JavaScript	Java	C++
Baseline	26,193,242	13,276,656	41,718,824	20,027,203	6,372,874
Stars	78,914	28,246	48,270	23,736	22,597
Forks	73,149	24,890	44,757	23,207	21,470
>1 GFI	2,185	1,582	986	618	710
RecentPush	1,638	1,218	579	486	559
CONTRIBUTING	1,108	832	398	353	342
RecentClosedGFI	505	398	140	173	193
ValidGFI	142	129	32	44	57

As a result, the contribution dataset consists of 90 pull requests in total: two passes for Python and TypeScript and one pass for Java, distributed equally across the three contribution types, as summarised in Table 3.6.

Table 3.6: Distribution of the 90 pull requests across languages and contribution types.

Language	Standard	Agent-Friendly	Total
Python	24	12	36
TypeScript	24	12	36
Java	12	6	18
Total	**60**	**30**	**90**

3.2. Data Analysis

This section describes how the data collected through the contribution process is analysed to address each of the four research questions. The analysis combines quantitative methods for RQ1 and RQ2 with qualitative methods for RQ3 and RQ4. The analysis is structured around the data associated with each of the three contribution types defined in Section 3.1.6.

3.2.1. Staleness Criterion

Open pull requests that have not received any activity for 30 days are treated as closed without merge. This threshold is motivated by prior work showing that 97% of pull requests receive a first response within 30 days [27]. For pull requests deemed stale, all metadata (comments, reviews, commits) is considered only up to the point at which the pull request became inactive; any subsequent activity is disregarded.

To better interpret staleness outcomes, repository characteristics are analysed for the 45 repositories in the dataset. Differences in repository size or activity levels could otherwise help explain differences in pull request outcomes across contribution types. Metadata, including total issues, pull requests, and good first issues (GFIs), is collected from 2025-05-21 onwards, corresponding to the earliest traceable appearance of an agent-friendly configuration file (see Section 3.1.3). Repository responsiveness is approximated using the PR closed share, defined as $\frac{\text{Closed PRs}}{\text{Closed PRs+Open PRs}}$.

Differences in repository characteristics across the three groups are assessed using pairwise Mann-Whitney U tests, as the metrics are continuous and the groups are independent. To account for multiple comparisons across metrics, p-values are adjusted using the Benjamini-Hochberg (BH) procedure [59]. Effect size is reported using the rank-biserial correlation r [9], interpreted using standard thresholds (small: 0.10, medium: 0.30, large: 0.50) [60].

3.2.2. RQ1: Acceptance Rates

RQ1 investigates how acceptance rates differ across the three contribution types. A pull request is considered accepted if it is merged, and rejected if it is closed without merge or deemed stale. To determine whether acceptance rates differ between groups, pairwise Chi-Square tests [56] are conducted for each of the three pairs of contribution types. The Chi-Square test is appropriate as acceptance is a binary outcome, and since each comparison concerns a single metric, no correction for multiple comparisons is applied. Effect size is measured using Cramér's V [54], interpreted using the thresholds defined in Section 3.2.1.

3.2.3. RQ2: Reviews and Comments

RQ2 examines how the number of reviews and comments differ across contribution types. On GitHub, comments and reviews are distinct interaction types. Comments refer to general discussion left on a pull request at the issue level or as inline annotations on specific lines of code, and can be left by any participant at any time. In this study, inline and issue-level comments are combined into a single comment count per pull request. Reviews are a structured GitHub feature that allows a reviewer to formally evaluate the pull request as a whole, resulting in one of three outcomes: approval, a request for changes, or a neutral comment. A single review submission can contain multiple inline comments, but is counted as one review event. Both measures are further split into human-authored and bot-authored interactions. Bot activity is identified using the GitHub API author type field, supplemented by a manually curated list of known bot accounts.

Differences between groups are assessed using pairwise Mann-Whitney U tests [34], as the data is not normally distributed and the groups are independent. To account for multiple comparisons within each pair, p-values are adjusted using the Benjamini-Hochberg procedure [59]. Effect size is reported using the rank-biserial correlation coefficient [9], where the sign indicates direction and the magnitude is interpreted using the thresholds defined in Section 3.2.1.

3.2.4. RQ3: Rejection Reasons

RQ3 investigates the reasons for which agentic AI pull requests are rejected. For each rejected pull request in each of the three contribution types, the maintainer feedback available in the pull request discussion is examined to identify the reason for rejection. This includes issue-level comments, inline comments, and review bodies. Pull requests that were deemed stale, meaning they received no activity within the 30-day threshold described above, are labelled as stale rather than assigned a rejection reason derived from feedback.

The labelling process is carried out qualitatively, with rejection reasons assigned based on a manual review of the available comments and discussions. Labels are derived inductively from the data rather than from a predefined scheme. Rejection reasons are reported separately for each of the three contribution types, as specified in RQ3.1, RQ3.2, and RQ3.3.

3.2.5. RQ4: Maintainer Feedback Themes

RQ4 investigates how maintainer feedback differs across the three contribution types through a qualitative thematic analysis of all human-authored comments received on the submitted pull requests. Thematic analysis is one of the most widely used qualitative data analysis methods in software engineering research [4], and is applied here following the approach described in [62].

All human-authored comments across all pull requests are included in the analysis, regardless of the pull request outcome. The process begins at the level of individual comments: each comment is assigned one or more codes that capture the substance of the feedback. Once all comments across all pull requests have been coded, related codes are progressively merged and grouped into broader themes. This process continues iteratively until all codes are included within a theme and no further meaningful merging is possible. The resulting themes represent recurring patterns in the feedback received. Themes are analysed in two ways. First, a frequency-based analysis counts the total number of code occurrences per theme across all comments, capturing how intensively each theme appears within discussions. Second, a one-of analysis counts each theme at most once per pull request, regardless of how many times it appears in that pull request's discussion, capturing how broadly each theme is distributed across pull requests. Both analyses are conducted separately for each of the three contribution types to highlight differences in how maintainers engage with each group.

To ensure the reliability of the coding process, an interrater agreement procedure was conducted following the approach described by Gisev et al. [20]. A second rater independently coded the comments, after which percentage agreement scores were calculated between the two raters. Disagreements were discussed and resolved in a series of meetings, with the coding scheme refined iteratively until full agreement was reached.

4

Results

This chapter presents the results of the empirical study. Section 4.1 first examines the characteristics of the 45 repositories to which pull requests were submitted, to establish whether the three groups are comparable in terms of scale and activity. Section 4.2 then provides a general overview of the outcome data collected across all 90 submitted pull requests. The remaining sections address each research question in turn. The pull request data used in this study is available in the replication package [48].

4.1. Repository Characteristics

Table 4.1 presents descriptive statistics for the 45 repositories to which pull requests were submitted, grouped by contribution type. Repository data was collected from 2025-05-21 onwards, with further details on this cutoff provided in Section 3.1.3. *Agent-Friendly* repositories show higher average counts across issues, pull requests, and good first issues (GFIs) compared to the two *Standard* groups. The PR closed share, computed as $\frac{\text{Closed PRs}}{\text{Closed PRs+Open PRs}}$, is high and comparable across all three groups.

Table 4.1: Descriptive statistics (averages) for the 45 repositories grouped by contribution type.

Group	Avg. # Issues	Avg. # PRs Opened	Avg. # GFIs	Avg. PR Closed Share (%)
Agent-Friendly (AI Disclosed)	696.73	1515.47	38.07	93.23
Standard (AI Disclosed)	212.87	461.80	31.87	94.44
Standard (AI Undisclosed)	356.07	548.93	111.47	92.50

Table 4.2 presents the results of the pairwise Mann-Whitney U tests with BH-adjusted p-values. None of the pairwise comparisons reach statistical significance, either before or after correction. The smallest unadjusted p-value is observed for issues per repository between *Agent-Friendly* and *Standard (AI Disclosed)* ($p = 0.051$), which remains non-significant after BH correction ($p_{BH} = 0.154$). All other comparisons show higher p-values in both unadjusted and adjusted analyses. While the mean values differ considerably across groups, the lack of statistical significance may be partly explained by the presence of outliers inflating the group means, as the Mann-Whitney U test operates on ranks and is therefore less sensitive to such skew.

The absence of statistically significant differences across repository characteristics suggests that the three contribution groups are broadly comparable in terms of scale and activity. As a result, differences observed in the following sections in terms of acceptance/rejection rates, numbers of comments and reviews, and maintainer feedback are unlikely to be systematically driven by structural differences between the repositories assigned to each group.

Table 4.2: Pairwise Mann-Whitney U test results for repository characteristics. *p*: raw p-value; *p* (BH): p-value after Benjamini–Hochberg false discovery rate correction; *r*: rank-biserial correlation as effect size measure (negative values indicate the first group has lower ranks); *Effect*: magnitude classification (Negligible $|r| < 0.1$, Small $|r| < 0.3$, Medium $|r| < 0.5$, Large $|r| \geq 0.5$); *Sig.*: whether the corrected p-value meets the $\alpha = 0.05$ significance threshold.

Comparison	Metric	p	p (BH)	r	Effect	Sig. ($p_{BH} < 0.05$)
Agent-Friendly vs Standard (AI Disclosed)	Issues	0.051	0.154	-0.422	Medium	No
	PRs Opened	0.051	0.154	-0.422	Medium	No
	GFIs	0.106	0.158	-0.351	Medium	No
	PR Closed Share	0.619	0.619	0.111	Small	No
Agent-Friendly vs Standard (AI Undisclosed)	Issues	0.065	0.195	-0.400	Medium	No
	PRs Opened	0.065	0.195	-0.400	Medium	No
	GFIs	0.253	0.380	-0.249	Small	No
	PR Closed Share	0.983	0.983	0.010	Negligible	No
Standard (AI Disclosed) vs Standard (AI Undisclosed)	Issues	0.820	0.934	0.053	Negligible	No
	PRs Opened	0.820	0.934	0.053	Negligible	No
	GFIs	0.934	0.934	0.022	Negligible	No
	PR Closed Share	0.694	0.934	-0.090	Negligible	No

4.2. General Overview

Of the 90 pull requests submitted across the three contribution types (see Section 3.1.9), 45 were merged and 45 were closed without merge, resulting in an overall acceptance rate of 50%. Figure 4.1 shows the distribution of pull request outcomes across the three contribution types. The *AI Disclosed Agent-Friendly* group had the lowest acceptance rate, with 10 out of 30 pull requests merged. The *AI Disclosed Standard* and *AI Undisclosed Standard* groups performed more similarly, with 18 and 17 merged pull requests out of 30 respectively.

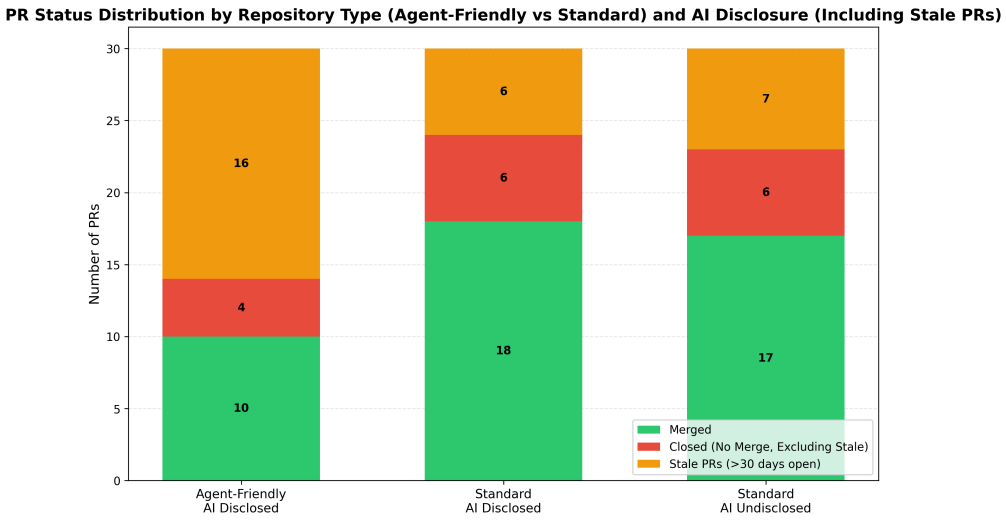


Figure 4.1: PR status distribution by repository type and AI disclosure.

Of the 45 pull requests closed without merge, 29 were closed due to staleness, meaning they received no activity within the 30-day threshold defined in Section 3.2.1 and were therefore treated as closed without merge. The remaining 16 were explicitly closed by a maintainer, with the reasons for rejection discussed in Section 4.5.

4.3. RQ1: Acceptance Rates

Table 4.3: Acceptance and rejection rates per contribution type.

Contribution Type	Total PRs	Merged	Acceptance Rate
AI Disclosed Agent-Friendly	30	10	33.3%
AI Disclosed Standard	30	18	60.0%
AI Undisclosed Standard	30	17	56.7%

Table 4.3 presents the acceptance rates for each of the three contribution types. The *AI Disclosed Agent-Friendly* group had the lowest acceptance rate at 33.3% (10 out of 30 pull requests merged), compared to 60.0% for *AI Disclosed Standard* (18 out of 30) and 56.7% for *AI Undisclosed Standard* (17 out of 30).

Table 4.4: Pairwise Chi-Square test results for RQ1. χ^2: Chi-Square test statistic measuring deviation from independence; p: raw p-value (no correction applied); *Cramér's V*: effect size derived from χ^2, ranging from 0 to 1 (Negligible $V < 0.1$, Small $V < 0.3$, Medium $V < 0.5$, Large $V \geq 0.5$); *Sig.*: whether the p-value meets the $\alpha = 0.05$ significance threshold.

Group A	Group B	χ^2	p	Cramér's V	Sig. ($p < 0.05$)
AI Disc. Agent-Friendly	AI Disc. Standard	4.286	0.038	0.267 (Small)	Yes
AI Disc. Agent-Friendly	AI Undisc. Standard	3.300	0.069	0.235 (Small)	No
AI Disc. Standard	AI Undisc. Standard	0.069	0.793	0.034 (Negligible)	No

Table 4.4 presents the results of the pairwise Chi-Square tests. The comparison between *AI Disclosed Agent-Friendly* and *AI Disclosed Standard* is the only one that yielded a statistically significant result ($\chi^2 = 4.286$, $p = 0.038$, Cramér's $V = 0.267$), with a small effect size, suggesting that the difference in acceptance rates between these two groups is unlikely to be due to chance, though the practical magnitude of the difference remains modest. The remaining two comparisons did not reach statistical significance: the *AI Disclosed Agent-Friendly* vs. *AI Undisclosed Standard* comparison fell just short of the threshold ($\chi^2 = 3.300$, $p = 0.069$), while *AI Disclosed Standard* vs. *AI Undisclosed Standard* showed virtually no difference ($\chi^2 = 0.069$, $p = 0.793$), consistent with the near-identical acceptance rates observed for these two groups in Table 4.3.

RQ1: In our experiment, acceptance rates differed across contribution types. Agent-friendly repositories with disclosed AI involvement showed a notably lower acceptance rate than both standard repository groups, with a statistically significant difference compared to standard repositories with disclosed AI involvement. No statistical difference in acceptance rate was observed between the two standard repository groups, regardless of disclosure.

4.4. RQ2: Reviews and Comments

Table 4.5: Human vs. bot comments and reviews per contribution type (percentages within category).

Category	Human Comments	Bot Comments	Human Reviews	Bot Reviews
All PRs	257 (54.1%)	218 (45.9%)	175 (67.6%)	84 (32.4%)
Agent-Friendly + AI Disclosed	57 (36.8%)	98 (63.2%)	32 (50.0%)	32 (50.0%)
Standard + AI Disclosed	65 (51.6%)	61 (48.4%)	43 (65.2%)	23 (34.8%)
Standard + AI Undisclosed	135 (69.6%)	59 (30.4%)	100 (77.5%)	29 (22.5%)

Table 4.5 summarises the distribution of comments and reviews across the three contribution types, broken down by whether they were made by humans or automated bots. Across all 90 PRs, nearly half of all comments (45.9%) and a third of all reviews (32.4%) originated from bots, with this proportion varying notably across groups. The *AI Disclosed Agent-Friendly* group received the highest share

of bot comments (63.2%) and an equal split between human and bot reviews (50.0%), while the *AI Undisclosed Standard* group had the lowest bot activity (30.4% of comments, 22.5% of reviews).

Table 4.6: Mean and median comment and review counts per contribution type.

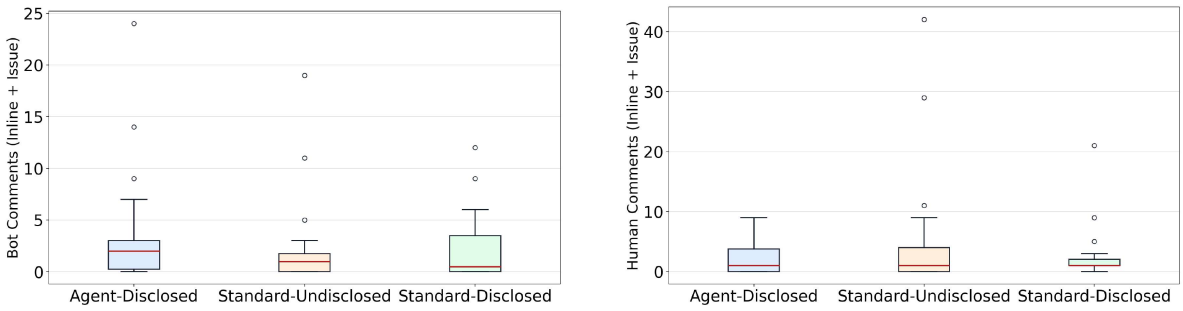
	Human Comments		Human Reviews		Bot Comments		Bot Reviews	
Contribution Type	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**	**Mean**	**Median**
AI Disc. Agent-Friendly	1.90	1	1.07	1	3.27	2	1.07	1
AI Undisc. Standard	4.50	1	3.33	1	1.97	1	0.97	0
AI Disc. Standard	2.17	1	1.43	1	2.03	0.5	0.77	0

Table 4.6 presents the mean and median comment and review counts per contribution type, separated by human and bot activity. Across all three contribution types, median values for both human comments and human reviews are consistently at or near 1, indicating that most pull requests received little engagement. The *AI Undisclosed Standard* group shows the highest mean human comment and review counts at 4.50 and 3.33 respectively, driven by a small number of pull requests with high activity, as reflected in the gap between mean and median. The *AI Disclosed Agent-Friendly* group has the lowest mean human activity, while showing the highest mean bot comment count at 3.27, compared to approximately 2.00 for both standard groups. Bot review counts are similarly low and comparable across all three groups.

Table 4.7: Pairwise Mann-Whitney U test results for RQ2. *p*: raw p-value; *p* (BH): p-value after Benjamini–Hochberg false discovery rate correction; *r*: rank-biserial correlation as effect size measure (negative values indicate Group A has lower ranks than Group B); *Effect*: magnitude classification (Negligible $|r| < 0.1$, Small $|r| < 0.3$, Medium $|r| < 0.5$, Large $|r| \geq 0.5$); *Sig.*: whether the corrected p-value meets the $\alpha = 0.05$ significance threshold.

Comparison (Group A vs Group B)	**Metric**	p	p **(BH)**	**r**	**Effect**	**Significant ($p_{BH} < 0.05$)**
AI Disc. Agent-Friendly vs AI Undisc. Standard	Human Comments	0.498	0.498	0.100	Negligible	No
	Human Reviews	0.134	0.267	0.217	Small	No
	Bot Comments	0.074	0.267	-0.262	Small	No
	Bot Reviews	0.389	0.498	-0.118	Small	No
AI Disc. Agent-Friendly vs AI Disc. Standard	Human Comments	0.933	0.933	0.013	Negligible	No
	Human Reviews	0.477	0.636	0.102	Small	No
	Bot Comments	0.143	0.571	-0.214	Small	No
	Bot Reviews	0.360	0.636	-0.124	Small	No
AI Undisc. Standard vs AI Disc. Standard	Human Comments	0.632	0.917	-0.071	Negligible	No
	Human Reviews	0.429	0.917	-0.116	Small	No
	Bot Comments	0.876	0.917	-0.023	Negligible	No
	Bot Reviews	0.917	0.917	-0.014	Negligible	No

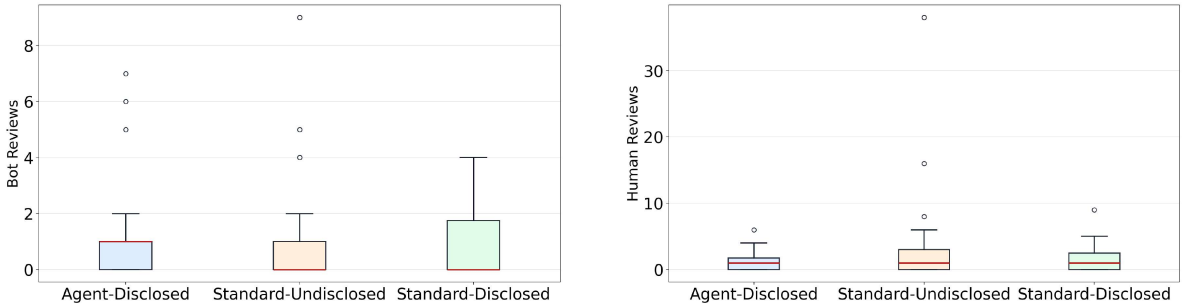
Table 4.7 presents the results of the pairwise Mann-Whitney U tests with BH-adjusted p-values. None of the pairwise comparisons reach statistical significance, either before or after correction. The smallest unadjusted p-value is observed for bot comments between *AI Disclosed Agent-Friendly* and *AI Undisclosed Standard* ($p = 0.074$), which remains non-significant after BH correction ($p_{BH} = 0.267$). All other comparisons show higher p-values in both unadjusted and adjusted analyses.



(a) Bot comment (inline + issue) distributions.

(b) Human comment (inline + issue) distributions.

Figure 4.2: Bot and human comment (inline + issue) distributions by contribution type (box plot). Labels are abbreviated: *Agent-Disclosed* = AI Disclosed Agent-Friendly; *Standard-Undisclosed* = AI Undisclosed Standard; *Standard-Disclosed* = AI Disclosed Standard.



(a) Bot review distributions.

(b) Human review distributions.

Figure 4.3: Bot and human review distributions by contribution type (box plot). Labels are abbreviated: *Agent-Disclosed* = AI Disclosed Agent-Friendly; *Standard-Undisclosed* = AI Undisclosed Standard; *Standard-Disclosed* = AI Disclosed Standard.

Figures 4.2 and 4.3 show the distributions of human and bot comments and reviews across the three contribution types. Both human metrics are highly skewed, with most pull requests receiving very few comments or reviews and a small number of cases attracting substantially higher counts. Medians are at or near 1 across all groups and both metrics. The *AI Undisclosed Standard* group shows the widest spread, with outliers reaching up to 42 human comments and 38 human reviews on individual pull requests, pulling the group mean considerably above the median. The interquartile ranges overlap strongly across all three groups. The *AI Disclosed Agent-Friendly* and *AI Disclosed Standard* groups show more compact distributions with slightly lower overall activity on both metrics. For bot activity, comment counts are slightly higher for the *AI Disclosed Agent-Friendly* group, with a median of 2 compared to 1 and 0.5 for the *AI Undisclosed Standard* and *AI Disclosed Standard* groups respectively. Bot review counts are low across all three groups, with medians at or near zero. Both bot metrics show substantial overlap in interquartile ranges across groups, and activity is concentrated in a small number of pull requests in each case.

RQ2: Based on our experiment, no statistically significant differences are observed in the number of reviews or comments across the three contribution types for either human or bot interactions.

4.5. RQ3: Rejection Reasons

The following subsections report the rejection reasons for each of the three contribution types. Stale pull requests are reported separately from those that received an explicit rejection. Tables 4.8, 4.9, and

4.10 summarise the distribution of rejection reasons for each group.

4.5.1. RQ3.1: AI Disclosed Agent-Friendly

Of the 20 rejected pull requests in the *AI Disclosed Agent-Friendly* group, 16 were closed due to staleness. The remaining 4 received explicit rejection feedback from maintainers. One was closed because the issue was not considered high priority, and one was closed with the maintainer expressing suspicion of AI spam alongside a lack of contributor response. The remaining two were rejected on AI-related grounds: in both cases the maintainer questioned the quality of the contribution, citing the absence of unit tests and incorrect documentation as indicators of insufficiently reviewed AI-generated code, despite AI assistance being disclosed in the pull request description.

Table 4.8: Rejection reasons for AI Disclosed Agent-Friendly pull requests.

Rejection Reason	Count
Stale (no activity >30 days)	16
AI generation (insufficient human review quality)	2
Issue not high priority	1
Suspected AI spam / no contributor response	1
Total	**20**

4.5.2. RQ3.2: AI Disclosed Standard

Of the 12 rejected pull requests in the *AI Disclosed Standard* group, 6 were closed due to staleness. Among the remaining 6, three were rejected on AI-related grounds: two were closed with an explicit policy against accepting primarily AI-generated contributions, and one was rejected due to unverified local testing of AI-generated changes. One additional pull request was closed in favour of another submission due to a preference against fully AI-generated contributions, one because the requirements for the issue had changed, and one because the pull request did not adequately solve the issue.

Table 4.9: Rejection reasons for AI Disclosed Standard pull requests.

Rejection Reason	Count
Stale (no activity >30 days)	6
AI generation (policy against AI contributions)	2
AI generation (insufficient testing evidence)	1
Another PR preferred due to preference against fully AI-generated code	1
Requirements changed	1
PR does not solve issue	1
Total	**12**

4.5.3. RQ3.3: AI Undisclosed Standard

Of the 14 rejected pull requests in the *AI Undisclosed Standard* group, 8 were closed due to staleness. Among the remaining 6, two were rejected because the requirements for the issue had changed, and two were rejected on the basis of an AI policy that was introduced after the pull requests had already been submitted and was not in place at the time of contribution, which stated that AI assistance is not permitted for good first issues. The remaining two were closed by the same maintainer across two related pull requests, citing a lack of meaningful contributor engagement. Throughout the review process, the maintainer had raised a series of questions about design decisions and parameter behaviour, to which the contributor responded only with code commits rather than written responses, following the contribution process workflow described in Section 3.1.8. The maintainer ultimately closed both pull requests, stating that responding with code alone was not helpful and that the questions had not been answered.

Table 4.10: Rejection reasons for AI Undisclosed Standard pull requests.

Rejection Reason	Count
Stale (no activity >30 days)	8
AI policy not followed (retroactive)	2
Lack of contributor response	2
Requirements changed	2
Total	**14**

RQ3: Explicit rejection reasons vary across contribution types. In both disclosed groups, AI-related grounds account for the majority of non-stale rejections, with maintainers citing insufficient review quality, lack of testing evidence, and explicit policies against AI-generated contributions. In the undisclosed group, no pull requests were rejected on AI grounds at the time of submission, though two were later affected by a retroactively introduced AI policy prohibiting AI assistance on good first issues. The remaining non-stale rejections in the undisclosed group were attributed to a lack of contributor response and changed requirements.

4.6. RQ4: Maintainer Feedback Themes

Table 4.11 presents the eight themes identified through the thematic analysis, along with example codes that were grouped under each theme. In total, 149 distinct codes were identified across all comments, which were then merged and grouped into the eight themes. The full list of codes for each theme is available in the replication package [48]. Since automated bot interactions do not reflect human judgment or maintainer reasoning, the thematic analysis considers only human-authored comments and reviews. The breakdown of human versus bot comments and reviews across groups is reported in Table 4.5.

Table 4.11: Identified themes and example codes.

#	Theme	Example Codes
1	Approval	• Changes are approved • Once CI passes we can merge
2	Code Quality	• Remove unnecessary code • Simplify code
3	Implementation Correctness	• Incorrect code functionality • Tests do not check functionality
4	Involvement of Automated Review Bots	• Prompt for review bot to evaluate code changes • Maintainer asks to address review bot concerns
5	Lack of Trust in AI Generation	• AI assisted submissions need to have human-in-the-loop • PR closed because it does not follow AI policy
6	Lack of Trust in Contributor	• Maintainer asks for visual proof that code works • Maintainer asks for proof of testing
7	Misalignment between Requirements and Solution	• Issue requirements have changed • PR contains unrelated changes
8	Procedural Issues	• Fix failing CI/CD • Fix merge conflicts

The initial interrater agreement score across all coded comments was 74.4% (287 out of 386 codes agreed upon), with disagreements resolved through a series of discussion meetings until full agreement was reached. The themes presented in the following sections reflect this final agreed-upon coding.

The following paragraphs describe each theme in detail.

Approval. This theme groups comments and reviews that signal a positive evaluation of the pull request, ranging from explicit approval to conditional statements such as confirming that the PR will be merged once the CI pipeline passes.

Code Quality. This theme captures reviewer feedback concerned with how well-written and maintainable the submitted code is, independent of whether it produces correct behaviour. It encompasses comments requesting simplification of overly complex logic, removal of redundant or unused code, improvements to naming conventions, and adherence to the repository's style guidelines.

Implementation Correctness. This theme groups feedback that challenges whether the submitted code actually solves the problem it is intended to address. It includes comments pointing out incorrect logic, edge cases that are not handled, tests that do not adequately verify the intended behaviour, and outright bugs introduced by the change. Unlike Code Quality, which concerns how code is written, Implementation Correctness concerns whether the code does what it should.

Involvement of Automated Review Bots. This theme captures instances where a maintainer actively involves an automated review bot in the evaluation of a pull request. It includes cases where a maintainer explicitly prompts a bot to review the submitted changes, as well as cases where a maintainer instructs the contributor to address the concerns that such a bot has raised before the review can proceed. The theme therefore reflects a deliberate human decision to delegate part of the review process to automated tooling, rather than autonomous bot activity.

Lack of Trust in AI Generation. This theme captures reviewer scepticism or concern directed specifically at the use of AI tooling in the contribution. It includes comments questioning whether the contributor understood the code they submitted, requests for human oversight of AI-generated changes, and cases where pull requests were closed outright because the repository's policy prohibits AI-assisted contributions. This theme is unique in that it reflects a meta-level concern about the submission process itself rather than the content of the code.

Lack of Trust in Contributor. Closely related to the previous theme, this theme captures reviewer doubt directed at the contributor's competence or engagement rather than at AI tooling specifically. It includes requests for proof of manual testing, demands for screenshots or other visual evidence that the change works as intended, and comments suggesting that the contributor did not read the issue carefully. Unlike Lack of Trust in AI Generation, this theme can arise in any contribution context and reflects a general pattern of maintainers seeking assurance that the contributor has taken ownership of their submission.

Misalignment between Requirements and Solution. This theme captures situations where the submitted solution does not match what was actually asked for. It includes cases where the issue requirements had evolved since the contributor began work or where the change introduces modifications outside the scope of the original issue.

Procedural Issues. This theme covers feedback unrelated to the code itself, but required for a pull request to be accepted. It includes reviewer requests to fix failing CI pipelines, resolve merge conflicts, update changelogs, and follow submission conventions such as targeting the correct branch. These concerns are a routine part of contributing to an active open source repository and must be resolved before a pull request can move forward.

Table 4.12 presents the ranking of themes by frequency for each of the three contribution types. Code Quality and Implementation Correctness appear prominently across all three groups, consistently occupying the top three positions regardless of contribution type. In the two standard groups they rank first and second, while in the *AI Disclosed Agent-Friendly* group they rank second and third behind Procedural Issues. Procedural Issues ranks first in the agent-friendly group with 22 occurrences, compared to 9 and 34 in the standard groups. Lack of Trust in AI Generation appears at low but consistent counts across all three groups, at 8, 8, and 7 occurrences respectively. Misalignment between Requirements

and Solution is comparable across all three groups, with 6, 5, and 6 occurrences respectively. Involvement of Automated Review Bots shows notable variation, with 7 occurrences in the agent-friendly group and 18 in the undisclosed standard group, compared to just 3 in the disclosed standard group.

Table 4.12: Theme rankings by frequency per contribution type.

Rank	AI Disc. Agent-Friendly	AI Disc. Standard	AI Undisc. Standard
1	Procedural Issues (22)	Code Quality (40)	Code Quality (56)
2	Implementation Correctness (18)	Implementation Correctness (26)	Implementation Correctness (52)
3	Code Quality (16)	Approval (15)	Procedural Issues (34)
4	Lack of Trust in AI Generation (8)	Procedural Issues (9)	Involvement of Automated Review Bots (18)
5	Approval (8)	Lack of Trust in AI Generation (8)	Approval (13)
6	Requirements–Solution Misalignment (6)	Requirements–Solution Misalignment (5)	Requirements–Solution Misalignment (6)
7	Involvement of Automated Review Bots (7)	Involvement of Automated Review Bots (3)	Lack of Trust in AI Generation (7)
8	Lack of Trust in Contributor (5)	Lack of Trust in Contributor (2)	Lack of Trust in Contributor (2)

Figure 4.4 presents a heatmap of theme counts across the three contribution types, providing a visual overview of how feedback is distributed. Code Quality and Implementation Correctness are prominent across all three groups, though their cells are notably darker in the two standard groups than in the agent-friendly group. The high count of Procedural Issues in the undisclosed standard group stands out relative to the disclosed standard group, where it ranks noticeably lower. Involvement of Automated Review Bots also shows a visible contrast, with the undisclosed standard group showing a substantially darker cell than either of the other two groups. Lack of Trust in AI Generation and Lack of Trust in Contributor are among the lightest cells across all three groups, reflecting their consistently low counts.

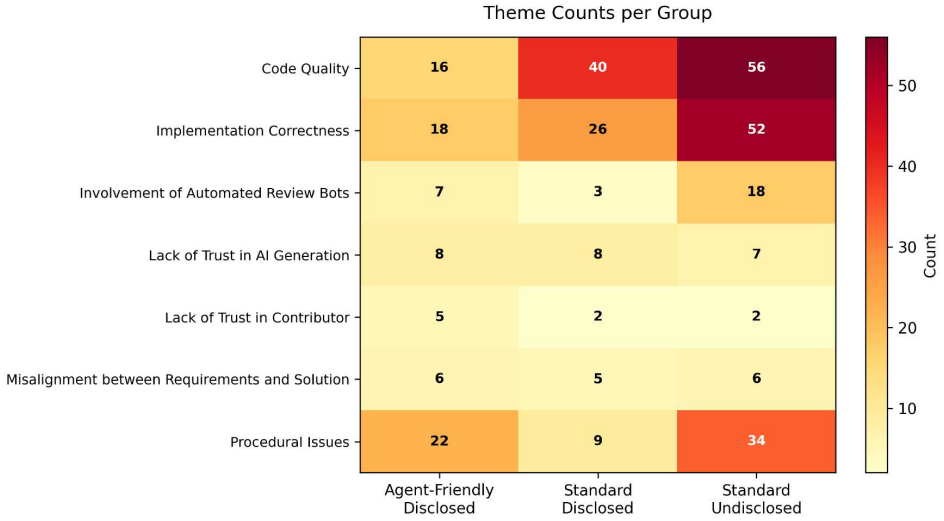


Figure 4.4: Heatmap of theme counts per contribution type.

To complement the frequency-based analysis, a one-of analysis was conducted in which each theme is counted at most once per pull request, regardless of how many times that theme appeared in its discussion. This provides a view of how broadly each theme is spread across pull requests, rather than how intensively it appears within them. Figure 4.5 presents the resulting heatmap. Values are out of a maximum of 19, 27, and 22 pull requests for the AI Disclosed Agent-Friendly, AI Disclosed Standard, and AI Undisclosed Standard groups, respectively. Code Quality, Implementation Correctness, and Procedural Issues show the darkest cells across all three groups, each present in 7 to 12 pull requests per group, indicating that these themes are encountered in a large share of pull requests regardless of contribution type. The agent-friendly group shows a more even spread of colour across themes, while in the standard groups Code Quality, Implementation Correctness and Procedural Issues cells are

noticeably darker than the rest. Lack of Trust in AI Generation is present in 5 pull requests in the agent-friendly group and 6 in the disclosed standard group, but drops to only 2 in the undisclosed standard group, reflected in its visibly lighter cell there. Involvement of Automated Review Bots is slightly darker in the agent-friendly group, present in 5 pull requests, compared to 2 and 3 in the standard groups. Lack of Trust in Contributor and Misalignment between Requirements and Solution are among the lightest cells across all three groups, each appearing in no more than 4 pull requests in any group. The one-of analysis broadly confirms the frequency-based results, suggesting that the themes with higher counts reflect genuine breadth of occurrence across pull requests rather than a small number of highly active cases inflating the counts.

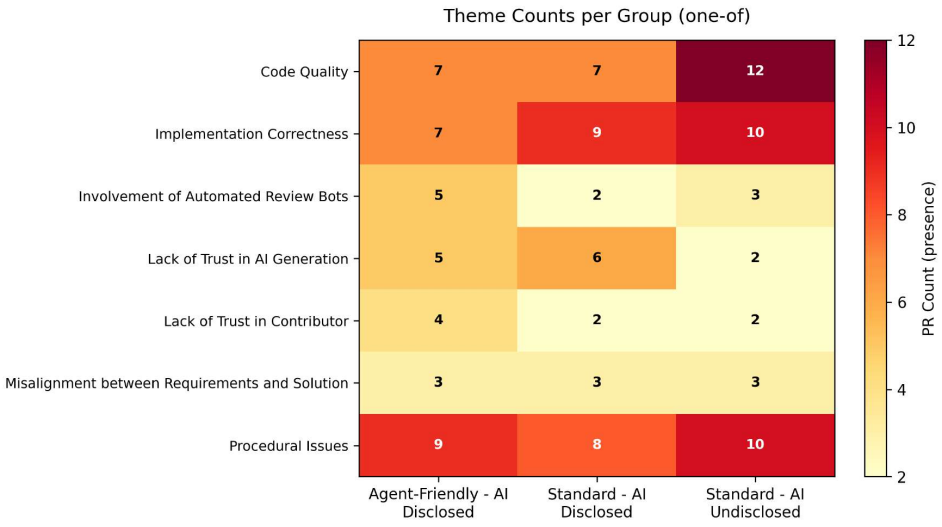


Figure 4.5: Heatmap of theme presence per contribution type (one-of per pull request).

RQ4: Maintainer feedback is broadly consistent across contribution types, with Code Quality, Implementation Correctness, and Procedural Issues being the most frequent and most broadly distributed themes across all groups. The one-of analysis confirms that these themes are present across a wide range of pull requests rather than being driven by a small number of highly active cases. Lack of Trust in AI Generation appears at comparatively low and broadly comparable frequencies across all groups. Involvement of Automated Review Bots shows the largest variation across groups, with notably higher counts in the agent-friendly and undisclosed standard groups.

5

Discussion

This chapter interprets the findings presented in Chapter 4 and situates them within the broader context of agentic AI contributions to open-source software. The discussion focuses on what the results reveal about how maintainers perceive and respond to AI-assisted pull requests targeting good first issues, and what this means for the communities they maintain.

5.1. RQ1: Acceptance Rates

The acceptance rates reported in Table 4.3 show a notable difference between the agent-friendly group and the two standard groups. Before interpreting this difference, it is worth noting that the three repository groups are broadly comparable in terms of scale and activity, as shown in Table 4.2, providing a reasonable basis for attributing observed differences to the experimental conditions rather than structural differences in the underlying repositories.

The statistically significant lower acceptance rate in the agent-friendly group compared to the disclosed standard group warrants closer examination. As reported in Section 4.2, of the 45 pull requests closed without merge across all groups, 29 were closed due to staleness, meaning they received no activity within the 30-day threshold. This proportion is particularly pronounced in the agent-friendly group, where staleness accounts for 16 of its 20 non-merged pull requests. Staleness can arise from a range of factors, both technical and non-technical, including maintainer capacity constraints, shifting project priorities, the volume of incoming contributions, or aspects of the pull request itself, such as code quality or the perceived origin of the contribution. The data alone does not allow us to determine which of these applies in any given case. What can be said is that the lower acceptance rate in the agent-friendly group is not primarily explained by active rejection, but by a lack of engagement.

The near-identical acceptance rates between the two standard groups, regardless of whether AI involvement was disclosed, is also a notable finding. The comparison between *AI Disclosed Standard* and *AI Undisclosed Standard* did not reach statistical significance, as reported in Table 4.4, suggesting that disclosure alone does not meaningfully influence whether a pull request is accepted.

From the perspective of good first issues as an onboarding mechanism, the high staleness rate carries particular weight. Prior work has shown that maintainers tend to adopt a more lenient and encouraging stance toward newcomers on good first issues, prioritising onboarding over strict evaluation [53], and that maintainer responsiveness is among the most significant factors in whether such issues are successfully resolved [57]. This makes the observed staleness more striking: if good first issues represent the setting where maintainers are most inclined to engage and guide, the failure to respond at all suggests that agentic AI contributions may not benefit from the same social consideration extended to human newcomers. Whether this reflects capacity constraints, disinterest, or an emerging policy toward AI-assisted contributions on newcomer-intended tasks remains an open question. A study that directly surveys maintainers about their decision not to engage with a pull request could shed light on these dynamics.

5.2. RQ2: Reviews and Comments

The absence of statistically significant differences in the number of reviews and comments across the three groups, as reported in Table 4.7, suggests that the repository type and disclosure of AI involvement do not alter the level of engagement maintainers invest in reviewing a pull request.

While the differences in bot involvement across groups are not statistically significant, nearly half of all comments (45.9%) and a third of all reviews (32.4%) originated from bots across all 90 PRs, with this proportion varying notably across groups, as shown in Table 4.5. This indicates that automated review tooling is a feature of open-source contribution workflows more broadly, and can be quite substantial in repositories that have more heavily integrated such tooling, such as those using CodeRabbit[1], GitHub Copilot[2], and Gemini Code Assist[3]. From the maintainer's perspective, automated review bots represent a practical way to handle an increasing volume of contributions without proportionally increasing the manual review burden. However, this is a trade-off rather than a straightforward efficiency gain. While automated tools reduce the cost of initial triage, manual review, despite being more expensive per interaction, is more likely to converge: a human reviewer can exercise judgment and prioritise the most important feedback, whereas automated tools apply rules uniformly regardless of context. Extended iterative cycles where a contributor addresses one round of bot feedback only to trigger new issues consume resources on both sides of the review process.

This dynamic is analogous to what Doğan and Tüzün [12] describe as a review smell, where iterative back-and-forth without convergence represents a breakdown in the review process. The outliers observed in bot comment counts in Figure 4.2a, where individual pull requests accumulated substantially more bot interactions than the median, are likely representative of precisely these extended ping-pong cycles. In the context of good first issues, where the tasks are intended to be low-barrier and straightforward [3], this kind of extended automated review cycle may offset the efficiency gains that agentic AI tools are meant to provide. The cost dynamics of iterative review cycles involving agentic AI tools and automated review bots have not previously been studied empirically and represent a gap that is becoming increasingly relevant as these tools see wider adoption. Future work could investigate how repositories balance the overhead of automated tooling against the benefits of reduced manual review burden, particularly as the volume of agentic contributions grows.

For good first issues specifically, this shift in review practice has a notable consequence. Part of what makes good first issues valuable as an onboarding mechanism is the mentorship dimension: maintainers guiding newcomers through the standards and expectations of the project [57]. When initial feedback is delegated to automated bots, this mentorship is partially replaced by automated output that may be harder for newcomers to interpret and act on. Whether this affects the quality of the onboarding experience is a question that goes beyond the scope of this study, but the pattern is visible in the data and could be investigated in future research.

5.3. RQ3: Rejection Reasons

The rejection reasons reported in Tables 4.8, 4.9, and 4.10 are examined below in the context of maintainer behaviour toward agentic AI contributions.

In both disclosed groups, AI-related grounds account for the majority of non-stale explicit rejections. However, the nature of these rejections is more nuanced than a pure refusal of AI-generated code. In the agent-friendly group, the two AI-related rejections were based on observable quality signals: the absence of unit tests and incorrect documentation, which the maintainer cited as indicators of insufficiently reviewed AI-generated code. In the disclosed standard group, rejections ranged from explicit policies against AI contributions to a requirement for evidence of local testing. These responses suggest that maintainers are not uniformly opposed to AI assistance, but are concerned with the degree of human oversight applied to the contribution. The question they appear to be asking is not whether AI was used, but whether a human took sufficient responsibility for the output.

[1] https://app.coderabbit.ai/
[2] https://docs.github.com/en/copilot/concepts/agents/code-review
[3] https://codeassist.google/

In the undisclosed group, no pull requests were rejected on AI grounds at the time of submission. Two were later affected by a retroactively introduced AI policy, illustrating that norms around AI contributions are still being formed in real time. The remaining non-stale rejections in this group were attributed to changed requirements and a lack of meaningful contributor engagement, the latter arising when the contributor responded to design questions with code commits rather than written discussion. This case is particularly relevant for agentic AI workflows, where the contribution process workflow described in Section 3.1.8 does not provide a mechanism for the agent to engage in open-ended written discussion with maintainers. The resulting gap in communication led to the pull request being closed, not because of the code itself, but because the maintainer could not verify that the contributor had understood the problem.

The norm formation process observed in this study, where policies were introduced mid-study and disclosure requirements varied widely across repositories, suggests that a study tracking how individual repositories develop and revise their AI contribution policies over time would be valuable. It also remains an open question whether the rejection patterns observed here hold for more complex contribution types beyond good first issues. As discussed in Chapter 1, good first issues were chosen precisely because maintainers tend to be more lenient and encouraging in this context [53], which raises the question of whether rejection rates would be more pronounced on contributions where that leniency cannot be assumed. More broadly, given the limited number of explicit rejections observed across all three groups, the patterns identified here are suggestive rather than conclusive, and future work directly surveying or interviewing maintainers about their decision-making would provide a more grounded account of how and why agentic AI contributions are rejected.

5.4. RQ4: Maintainer Feedback Themes

The thematic analysis reported in Table 4.12 and Figure 4.4 reveals that maintainer feedback is concentrated around code-related concerns across all three contribution types. Code Quality and Implementation Correctness consistently rank in the top three positions regardless of group, and the one-of analysis in Figure 4.5 confirms that these themes are broadly distributed across pull requests rather than driven by a small number of highly active cases.

Lack of Trust in AI Generation, by contrast, appears at comparatively low and broadly comparable frequencies across all three groups, including the undisclosed group where maintainers had no stated knowledge of AI involvement. When maintainers engage with a pull request, their evaluation is primarily driven by the quality and correctness of the code rather than by its origin. This is consistent with what Gousios et al. [24] identified as the primary criteria maintainers apply when evaluating contributions in pull-based development, and suggests that the introduction of agentic AI has not fundamentally changed what maintainers look for when reviewing a pull request.

This finding runs counter to what one might expect given the level of community frustration around AI-generated contributions documented in Chapter 1. While the rejection reasons in Section 4.5 confirm that AI-related concerns do lead to explicit closures in some cases, for the majority of pull requests that entered a review cycle, maintainers focused on the same issues they would raise for any contribution. From the maintainer's perspective, the origin of the contribution does not reduce the work required to evaluate it. This has direct implications for how good first issues function as an onboarding mechanism: when an agentic AI system produces code that passes a basic correctness bar but falls short on project-specific quality expectations, the maintainer still has to invest the same review effort, and the feedback still has to be addressed.

5.5. Threats to Validity

This section discusses potential threats to the validity of the study, organised around internal, construct, and external validity. For each identified threat, applied mitigations are described and, where complete mitigation is not possible, directions for future work are outlined.

5.5.1. Internal Validity

Temporal Validity of Repository Data:
The study relies entirely on data retrieved through the GitHub REST and GraphQL APIs. Repositories may change between collection and contribution (e.g., issues may be closed, labels modified, or activity levels changed), introducing a temporal validity threat. Because repository collection is not performed continuously, newly eligible repositories that meet the selection criteria after the collection runs are not included in the dataset. Maintaining a real-time or continuously updated repository pool is not feasible due to API rate limits and practical constraints of the data collection process.

To mitigate this, post-search filtering and manual validation steps are applied before contributions are made. However, it is not possible to fully eliminate the effects of repository evolution and dataset staleness over time. Future work could address this by implementing continuous or versioned data collection pipelines, allowing repository state to be tracked over time and enabling more precise alignment between collected metadata and actual repository conditions at the point of contribution.

Contributor Identity Effects:
All contributions were made from a single public GitHub profile[4], meaning maintainers can view the contributor's profile and history, which may influence their perception of submissions and introduce bias. Additionally, submitting multiple pull requests to the same repository may introduce dependencies between observations, as maintainers may recognise the account and adjust their behaviour accordingly. Future work could address this by distributing contributions across multiple contributor accounts.

Non-Determinism of the Agentic AI System:
A further threat arises from the non-deterministic nature of the agentic AI system used to generate contributions. Since model parameters such as temperature are not configurable within the tool, the system may produce different outputs given identical inputs across runs. This means that contributions generated for the same issue could vary in quality, approach, or correctness if regenerated, making it difficult to fully isolate the effect of the experimental condition from natural variation in AI output quality. This threat cannot be fully mitigated without access to deterministic model configurations. Future work using systems that expose temperature and seed parameters would allow for more reproducible and comparable contribution generation.

5.5.2. Construct Validity

Repository Classification:
Repository classification into *Agent-Friendly* and *Standard* is based on the presence of predefined files and directories associated with agentic AI tooling. This list of indicators is derived from patterns observed during repository inspection and from commonly used configurations in practice, but does not correspond to an official or standardised definition. Some repositories may support or accommodate agentic AI contributions without exposing any of these indicators, which could lead to misclassification. To mitigate this, classification criteria were explicitly defined and applied consistently across all repositories. Nevertheless, some degree of approximation remains unavoidable. Future work could develop a more comprehensive and validated taxonomy of agent-friendly repository signals, potentially informed by direct maintainer input or community surveys.

Good First Issue Label Variability:
The *good first issue* label is not standardised across repositories and may vary in meaning, difficulty, and quality. Some issues labelled as such may still involve non-trivial complexity or ambiguous requirements, while others may be trivial or poorly specified. This variability may affect both the contribution process and maintainer responses, making it difficult to treat all good first issues as equivalent instances of the same construct. Future work could supplement label-based selection with issue complexity metrics or manual difficulty ratings to produce a more homogeneous sample.

Concentration of Feedback Across Repositories:
Maintainer feedback is not uniformly distributed across repositories. A subset of repositories may account for a disproportionate share of responses, meaning that observed patterns in feedback themes

[4] https://github.com/tibisabau

could reflect the communication norms or culture of those specific projects rather than broader maintainer behaviour. To mitigate this, a one-of analysis was conducted in Section 4.6 in which each theme is counted at most once per pull request, reducing the influence of any single highly active repository or discussion on the overall frequency counts.

Subjectivity in Thematic Analysis:
The thematic analysis conducted in Section 4.6 introduces subjectivity, as assigning codes to individual comments and grouping codes into themes both require interpretive decisions. Feedback that is ambiguous or implicit may be categorised differently by different researchers. To mitigate this, an interrater agreement procedure was conducted as described in Section 3.2, in which a second rater independently coded the comments. Disagreements were resolved through iterative discussion until full agreement was reached, reducing the influence of any single rater's interpretation on the final coding scheme.

Validity of the Quality Control Process:
Automated quality control is limited to the scope of the contribution rather than the entire repository, due to practical constraints such as failing pipelines or long-running test suites. As a result, contributions may pass local validation while still failing full project-level checks. Manual quality control introduces an additional layer of subjectivity, as the contributor determines whether changes are acceptable or constitute model hallucinations, and different evaluators might reach different conclusions under similar conditions. Future work could address this by integrating project-level CI pipelines into the quality control step where feasible, and by involving multiple independent reviewers in the manual assessment stage.

5.5.3. External Validity

Scope of the Dataset:
The dataset consists of 90 pull requests submitted over a fixed time period and limited to a subset of repositories in Python, TypeScript, and Java. The study focuses on these languages due to feasibility and dataset availability, and cross-language comparisons are not the objective of the study. Nevertheless, maintainer behaviour may differ in other ecosystems not covered here. The 30-day observation window may also be insufficient to capture the full lifecycle of some pull requests, particularly those that are eventually merged after a longer review period. Expanding the dataset to include additional programming languages, a longer observation window, and a broader range of issue types and repository sizes would improve the generalizability of the findings. Future work could also investigate how maintainer responses evolve over time as norms around agentic AI contributions become more established.

Focus on Good First Issues:
The exclusive focus on good first issues limits the generalizability of the findings to other contribution types. As discussed in Chapter 1, good first issues were chosen precisely because maintainers tend to be more lenient and encouraging in this context [53]. Rejection rates and feedback patterns may differ substantially for more complex contributions where leniency cannot be assumed. Future work should investigate whether the patterns observed here hold for regular issues to establish whether the findings represent a lower or upper bound on maintainer receptivity to agentic AI contributions.

Generalizability of Repository Selection:
Selecting a representative set of repositories is important for this study. The filtering criteria applied during repository selection are intentionally strict. While some constraints, such as requiring at least two *good first issues*, are directly motivated by the research focus, others (e.g., minimum stars, forks, recent activity, and the presence of a `CONTRIBUTING.md` file) are introduced to ensure that only actively maintained repositories with structured onboarding processes are included.

These stricter filters are motivated by prior research and practical considerations. In particular, they help ensure a smoother contribution process and reduce the likelihood of selecting inactive or poorly maintained repositories. Additionally, tighter filtering reduces the size of the candidate pool, which is

necessary to keep the data collection process feasible given API rate limits and the manual effort required for contributions.

However, this may not generalize to smaller, less active, or less structured projects, where maintainer behavior and contribution dynamics may differ substantially from those observed in well-maintained, popular repositories. Future work could address this by applying a stratified sampling strategy across repository size and activity tiers, deliberately including smaller or less structured projects to enable comparisons across the spectrum of open-source ecosystems. Additionally, replicating the study with relaxed filtering criteria, while accounting for the added noise, would help assess whether the findings hold beyond the subset of repositories studied here.

5.6. Ethical Considerations

This study involves the submission of real pull requests to active open-source repositories, which raises ethical considerations regarding the use of maintainer time, transparency about AI involvement, and the potential introduction of erroneous code into production software. To mitigate these risks, all contributions were genuine attempts to resolve existing issues and were limited to non-critical changes. Contributions were screened through automated quality checks and manual review to reduce the likelihood of incorrect, hallucinated, or otherwise harmful code being submitted. No pull requests involved breaking changes, security-sensitive modifications, or other high-risk interventions.

All pull requests were submitted using the researcher's personal GitHub account. The account's public profile contains little to no personal information, limiting the disclosure of identifying details during the study. Maintainers could view only the information that is normally available through GitHub, such as public repositories, contribution activity, and account metadata.

Maintainers may interact with submitted pull requests without being aware that they are part of a research study. However, all interactions take place within public open-source development workflows where reviewing external contributions is an expected activity. The study does not seek to provoke unusual behaviour, collect private information, or manipulate participants beyond observing their normal responses to contribution proposals.

The study uses only data that is publicly available on GitHub, such as pull request outcomes and discussion threads. Analyses will focus on trends rather than individuals. Results are reported only in aggregated form, ensuring that individual maintainers, contributors, and repositories cannot be identified from the published findings.

The AI-undisclosed condition involves withholding information about the use of AI assistance in generating the pull request. This omission reflects existing contribution practices in which disclosure of AI assistance is not universally required. Nevertheless, the submitted contributions remain genuine attempts to solve the target issues, and the study does not involve fabrication, impersonation, or misleading technical claims.

5.7. Use of AI

Generative AI tools were used as part of this research in two distinct capacities. First, the agentic AI system under study was used to generate the pull request contributions that form the experimental data. This use is the subject of the study itself and is described in detail in Chapter 3.

Second, large language model assistants were used to support the writing of this thesis. Specifically, AI tools were used to assist with paraphrasing, improving sentence clarity, and checking grammar and style. All intellectual content, including research design, interpretation of results, and argumentation, was produced by the author. No AI-generated text was used without review and revision.

6

Related Work

This chapter reviews empirical studies most directly related to this thesis. Section 6.1 covers studies examining agentic AI contributions to open-source repositories. Section 6.2 covers work on review dynamics and maintainer behaviour in the context of AI-generated pull requests.

6.1. Agentic AI Contributions to Open-Source Projects

A growing body of empirical work has begun to examine how agentic AI systems contribute to open-source repositories in practice, largely enabled by the release of the AIDev dataset[1], a large-scale collection of agent-authored pull requests from real GitHub repositories.

Li et al. [32] analysed a large-scale dataset of agent-submitted pull requests, finding that agents outperform humans in submission speed but are accepted less frequently and produce structurally simpler contributions. Watanabe et al. [61] studied Claude Code-generated pull requests across open-source projects, finding that most were accepted, though many required additional changes for bug fixes, documentation, and project-specific standards. Gong et al. [21] compared five popular agents across 7,156 pull requests and found that task type is a dominant factor in acceptance rates, with documentation tasks achieving 82.1% acceptance compared to 66.1% for new features, and no single agent performing best across all task categories.

The code quality of agentic contributions has also received attention. Huang et al. [29] found that LLM agents frequently introduce code redundancy, yet reviewers tend to respond more positively to AI-generated contributions than human ones. Cynthia et al. [10] analysed 1,210 merged agent-generated bug-fix pull requests and found that code smells dominate post-merge quality issues, and that merge success does not reliably reflect post-merge code quality. Ghammam and Almukhtar [18] examined agent-generated build code specifically, finding 364 maintainability and security-related issues across varying severity levels, while noting that more than 61% of agentic pull requests are approved and merged with minimal human intervention. Abazar et al. [1] further characterised the behavioural signatures of five agents across programming languages, finding that Claude Code tends toward lower-similarity, higher-diversity changes, while Devin tends toward more incremental modifications.

Studies have also examined what distinguishes agentic from human contributions at the code level. Pinna et al. [43] analysed 23,247 agentic pull requests and found that 1.7% exhibited high message-code inconsistency, with pull requests claiming unimplemented changes being the most common issue. High inconsistency was associated with a 51.7% lower acceptance rate and a 3.5× longer time to merge. Yoshioka et al. [66] analysed 40,214 pull requests and found that submitter attributes dominate merge outcomes for both human and agentic contributions, while review-related features exhibit contrasting effects between the two groups. Branco et al. [6] examined auto-merged agentic pull requests specifically, finding that they tend to be smaller and more focused, and that repositories tend to either auto-merge all or none of their agentic contributions. Minh et al. [36] found that agents excel at narrow

[1] https://github.com/SAILResearch/AI_Teammates_in_SE3

automation, with 28.3% of pull requests merging instantly, but frequently fail at iterative refinement, leading to abandonment when faced with subjective feedback.

Testing behaviour in agentic pull requests has been studied by Haque et al. [25], who found that test-containing pull requests are more common over time and tend to be larger and take longer to complete, while merge rates remain largely similar across agents. Gao et al. [17] found that over 67.5% of AI-co-authored pull requests originate from contributors without prior code ownership, and that these contributions are merged significantly faster with minimal feedback, with approximately 80% merged without any explicit review.

Agarwal et al. [2] conducted a longitudinal causal study of agent adoption using staggered difference-in-differences, finding large front-loaded velocity gains only when agents are the first observable AI tool in a project, with quality risks persistent across settings, including an 18% rise in static-analysis warnings and a 39% rise in cognitive complexity.

6.2. Review Dynamics and Rejection Patterns

Beyond acceptance rates, several studies have examined what happens during the review process of agentic contributions. Ehsani et al. [14] conducted a large-scale study of 33,000 agent-authored pull requests and qualitatively analysed 600 to derive a hierarchical taxonomy of rejection patterns, finding that rejection reasons include lack of meaningful reviewer engagement, duplicate pull requests, unwanted feature implementations, and agent misalignment. Pham and Ghaleb [42] compared 33,596 agent-generated and 6,618 human pull requests, finding that agent-introduced symbols are removed much sooner than those in human pull requests, and that agents generate stronger commit-level messages but lag humans at pull request-level summarisation. Nachuma and Zibran [38] found that reviewer engagement has the strongest correlation with successful integration of agentic pull requests, while larger change sizes and force pushes are associated with lower merge likelihood.

Taken together, these studies establish a clear empirical picture of agentic contributions as they occur in the wild. However, they share a common limitation: they analyse existing pull requests submitted by AI agents without controlling the contribution process or the disclosure of AI involvement. This study addresses that gap by actively submitting pull requests to open-source repositories through a controlled workflow targeting good first issues, a contribution context that has not previously been studied in the context of agentic AI, and by examining the effect of disclosure on maintainer response.

7

Conclusion

This thesis investigated how open-source maintainers perceive and respond to agentic AI contributions targeting good first issues, using an experimental approach in which pull requests were actively submitted to active repositories across three contribution types: AI-disclosed agent-friendly, AI-disclosed standard, and AI-undisclosed standard. The study combined quantitative analysis of acceptance rates (RQ1) and the number of reviews and comments (RQ2) with a qualitative analysis of pull request rejection reasons (RQ3) and maintainer feedback (RQ4).

The results showed that acceptance rates differed across contribution types, with the agent-friendly group achieving a notably lower rate than both standard groups. This difference was driven primarily by staleness rather than active rejection. The comparison between the two standard groups yielded no statistically significant difference in acceptance rate, indicating that disclosure alone did not meaningfully influence whether a pull request was accepted. (RQ1)

The analysis of reviews and comments revealed no statistically significant differences across groups, though bot-authored interactions accounted for a substantial share of total activity, particularly in the agent-friendly group. This raised practical questions about the sustainability of iterative review cycles involving both agentic AI tools and automated bots, where each round of feedback triggered new costs on both sides of the process. (RQ2)

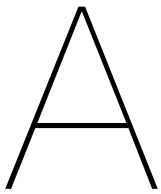
The examination of explicit rejection reasons suggested that maintainers may have been less concerned with the use of AI itself and more with whether contributors had taken sufficient responsibility for the submitted output. However, given the small number of explicit rejections in each group, no strong conclusions can be drawn. The retroactive introduction of AI policies in some repositories further suggested that community norms around AI-assisted contributions were still evolving at the time of the study. (RQ3)

The thematic analysis of maintainer feedback found that Code Quality, Implementation Correctness, and Procedural Issues dominated across all three groups, while Lack of Trust in AI Generation remained consistently low. When maintainers engaged, their evaluation was driven by the same criteria they apply to any contribution, and the introduction of agentic AI has not fundamentally changed what maintainers look for when reviewing a pull request. (RQ4)

Taken together, the findings suggested that the primary challenges facing agentic AI contributions to open-source good first issues were maintainer responsiveness and code quality rather than ideological resistance to AI involvement. The results also indicated that the key difficulty may not have been negative reactions from maintainers when engagement occurred, but rather the broader challenge of obtaining engagement in the first place.

Several directions for future work follow from these findings. This study examined maintainer responses through the lens of pull request outcomes and comments, but did not directly capture maintainer per-

spectives through interviews or surveys. A qualitative study that engages maintainers directly on their experiences with agentic AI contributions would complement the empirical findings and provide a richer understanding of the social and normative dimensions of this emerging phenomenon. The contribution context could be extended beyond good first issues to task types where expectations around code quality and human oversight are higher, and where maintainers may be less inclined to apply the same level of leniency often afforded to new contributors in good first issue contexts [53]. Direct surveys or interviews with maintainers would shed light on whether the observed non-engagement reflected capacity constraints, disinterest, or an emerging stance toward AI-assisted contributions on newcomer-intended tasks. Longitudinal studies tracking how repositories developed and revised their AI contribution policies over time would illuminate the norm formation process observed here. The role of automated review bots in newcomer onboarding also warranted further investigation, as it remained unclear whether bot-generated feedback supported or undermined the guidance that good first issues were designed to provide. Finally, the cost dynamics of iterative review cycles involving agentic AI tools and automated review bots represented an empirical gap that was becoming increasingly relevant as both types of tooling saw wider adoption in open-source workflows.



Prompt Templates

A.1. Contribution Prompt Template

```
[Issue Title]

[Issue Description]

Make sure the code follows the style and contribution guidelines
of the repository.

As part of the quality control process, run the commands below
(exactly as they are written):

Commands to Execute:
- Command 1
- Command 2
- Command n

It is important that all the commands pass fully, 0
errors/failures. Do not move to the next step unless the
command works perfectly.
```

A.2. Default Pull Request Template

```
## Description

[AI-generated description of changes and approach]

Closes #{issue_number}.

**The code in this pull request was generated by GitHub Copilot
with the Claude Sonnet 4.5 model.**

*(Note: This line appears only in the 50% of PRs where AI
involvement is disclosed)*

## Checklist if Applicable

- [ ] The tests passed - add commands ran for testing
- [ ] Linting passed - add commands ran for linting
- [ ] Documentation has been added
- [ ] CHANGELOG.md has been updated
```

A.3. Pull Request Description Prompt Template

```
Generate and fill in a Markdown file that follows this template
based on your work for this task (be concise and specific),
include the issue number in the file name:

[PR Template]

Do not add additional headings.
Follow the template strictly as presented.
```

Bibliography

[1] Mahdieh Abazar, Reyhaneh Farahmand, Gouri Ginde, Benjamin Tan, and Lorenzo De Carli. “Behavioral Analysis of AI Code Generation Agents: Edit, Rewrite, and Repetition”. In: (2026).

[2] Shyam Agarwal, Hao He, and Bogdan Vasilescu. “AI IDEs or Autonomous Agents? Measuring the Impact of Coding Agents on Software Development”. In: *arXiv preprint arXiv:2601.13597* (2026).

[3] Jan Willem David Alderliesten and Andy Zaidman. “An initial exploration of the “good first issue” label for newcomer developers”. In: *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE. 2021, pp. 117–118.

[4] Baris Ardic, Carolin Brandt, Ali Khatami, Mark Swillus, and Andy Zaidman. “The qualitative factor in software testing: A systematic mapping study of qualitative methods”. In: *Journal of Systems and Software* 227 (2025), p. 112447.

[5] Ali Bayram, Gonca Gokce Menekse Dalveren, and Mohammad Derawi. “Comparative Analysis of AI Models for Python Code Generation: A HumanEval Benchmark Study”. In: *Applied Sciences* 15.18 (2025), p. 9907.

[6] Ruben Branco, Paulo Canelas, Catarina Gamboa, and Alcides Fonseca. “LGTM! Characteristics of Auto-Merged LLM-based Agentic PRs”. In: (2026).

[7] Carolin Brandt, Ali Khatami, Mairieli Wessel, and Andy Zaidman. “Shaken, not stirred: How developers like their amplified tests”. In: *IEEE Transactions on Software Engineering* 50.5 (2024), pp. 1264–1280.

[8] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. “Developer onboarding in GitHub: the role of prior social links and language experience”. In: *Proceedings of the 2015 10th joint meeting on foundations of software engineering*. 2015, pp. 817–828.

[9] Edward E Cureton. “Rank-biserial correlation”. In: *Psychometrika* 21.3 (1956), pp. 287–290.

[10] Shamse Tasnim Cynthia, Al Muttakin, and Banani Roy. “Beyond Bug Fixes: An Empirical Investigation of Post-Merge Code Quality Issues in Agent-Generated Pull Requests”. In: *arXiv preprint arXiv:2601.20109* (2026).

[11] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. “Social coding in GitHub: transparency and collaboration in an open software repository”. In: *Proceedings of the ACM 2012 conference on computer supported cooperative work*. 2012, pp. 1277–1286.

[12] Emre Doğan and Eray Tüzün. “Towards a taxonomy of code review smells”. In: *Information and Software Technology* 142 (2022), p. 106737.

[13] Y Dong, X Jiang, J Qian, T Wang, K Zhang, Z Jin, and G Li. *A Survey on Code Generation with LLM-Based Agents, arXiv [cs. SE](2025)*.

[14] Ramtin Ehsani, Sakshi Pathak, Shriya Rawal, Abdullah Al Mujahid, Mia Mohammad Imran, and Preetha Chatterjee. “Where Do AI Coding Agents Fail? An Empirical Study of Failed Agentic Pull Requests in GitHub”. In: *arXiv preprint arXiv:2601.15195* (2026).

[15] Ahmed Fawzy, Amjed Tahir, and Kelly Blincoe. “Vibe Coding in Practice: Motivations, Challenges, and a Future Outlook–a Grey Literature Review”. In: *arXiv preprint arXiv:2510.00328* (2025).

[16] Felipe Fronchetti, Igor Wiese, Gustavo Pinto, and Igor Steinmacher. “What attracts newcomers to onboard on oss projects? tl; dr: Popularity”. In: *IFIP International Conference on Open Source Systems*. Springer. 2019, pp. 91–103.

[17] Haoyu Gao, Peerachai Banyongrakkul, Hao Guan, Mansooreh Zahedi, and Christoph Treude. “On Autopilot? An Empirical Study of Human-AI Teaming and Review Practices in Open Source”. In: *arXiv preprint arXiv:2601.13754* (2026).

[18] Anwar Ghammam and Mohamed Almukhtar. “AI builds, We Analyze: An Empirical Study of AI-Generated Build Code Quality”. In: *arXiv preprint arXiv:2601.16839* (2026).

[19] Asghar Ghasemi and Saleh Zahediasl. “Normality tests for statistical analysis: a guide for non-statisticians”. In: *International journal of endocrinology and metabolism* 10.2 (2012), p. 486.

[20] Natasa Gisev, J Simon Bell, and Timothy F Chen. “Interrater agreement and interrater reliability: key concepts, approaches, and applications”. In: *Research in Social and Administrative Pharmacy* 9.3 (2013), pp. 330–338.

[21] Jingzhi Gong, Giovanni Pinna, Yixin Bian, and Jie M Zhang. “Analyzing Message-Code Inconsistency in AI Coding Agent-Authored Pull Requests”. In: *arXiv preprint arXiv:2601.04886* (2026).

[22] Georgios Gousios. “The GHTorent dataset and tool suite”. In: *2013 10th Working conference on mining software repositories (MSR)*. IEEE. 2013, pp. 233–236.

[23] Georgios Gousios, Margaret-Anne Storey, and Alberto Bacchelli. “Work practices and challenges in pull-based development: The contributor’s perspective”. In: *Proceedings of the 38th international conference on software engineering*. 2016, pp. 285–296.

[24] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie Van Deursen. “Work practices and challenges in pull-based development: The integrator’s perspective”. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. IEEE. 2015, pp. 358–368.

[25] Sabrina Haque, Sarvesh Ingale, and Christoph Csallner. “An Empirical Study of Tests in Agentic Pull Requests”. In: (2026).

[26] Frank Harary and Leo Moser. “The theory of round robin tournaments”. In: *The American Mathematical Monthly* 73.3 (1966), pp. 231–246.

[27] Kazi Amit Hasan, Marcos Macedo, Yuan Tian, Bram Adams, and Steven Ding. “Understanding the time to first response in GitHub pull requests”. In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE. 2023, pp. 1–11.

[28] Hao He, Haonan Su, Wenxin Xiao, Runzhi He, and Minghui Zhou. “Gfi-bot: automated good first issue recommendation on github”. In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022, pp. 1751–1755.

[29] 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”. In: *arXiv preprint arXiv:2601.21276* (2026).

[30] Yuekai Huang, Junjie Wang, Song Wang, Zhe Liu, Dandan Wang, and Qing Wang. “Characterizing and predicting good first issues”. In: *Proceedings of the 15th ACM/IEEE international symposium on Empirical Software Engineering and Measurement (ESEM)*. 2021, pp. 1–12.

[31] Dae-Kyoo Kim and Hua Ming. “Assessing output reliability and similarity of large language models in software development: A comparative case study approach”. In: *Information and Software Technology* 185 (2025), p. 107787.

[32] 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”. In: *arXiv preprint arXiv:2507.15003* (2025).

[33] Zhixing Li, Yue Yu, Tao Wang, Yan Lei, Ying Wang, and Huaimin Wang. “To follow or not to follow: Understanding issue/pull-request templates on github”. In: *IEEE Transactions on Software Engineering* 49.4 (2022), pp. 2530–2544.

[34] Thomas W MacFarland and Jan M Yates. “Mann–whitney u test”. In: *Introduction to nonparametric statistics for the biological sciences using R*. Springer, 2016, pp. 103–132.

[35] Mehmet Mendes and Akin Pala. “Type I error rate and power of three normality tests”. In: *Pakistan Journal of information and technology* 2.2 (2003), pp. 135–139.

[36] Dao Sy Duy Minh, Huynh Trung Kiet, Nguyen Lam Phu Quy, Pham Phu Hoa, Tran Chi Nguyen, Nguyen Dinh Ha Duong, and Truong Bao Tran. “Early-Stage Prediction of Review Effort in AI-Generated Pull Requests”. In: *arXiv preprint arXiv:2601.00753* (2026).

[37] Hussein Mozannar et al. “Magentic-ui: Towards human-in-the-loop agentic systems”. In: *arXiv preprint arXiv:2507.22358* (2025).

[38] Costain Nachuma and Minhaz Zibran. “When AI Teammates Meet Code Review: Collaboration Signals Shaping the Integration of Agent-Authored Pull Requests”. In: *arXiv preprint arXiv:2602.19441* (2026).

[39] Nathalia Nascimento, Everton Guimaraes, Sai Sanjna Chintakunta, and Santhosh Anitha Boominathan. “Llm4ds: Evaluating large language models for data science code generation”. In: *arXiv preprint arXiv:2411.11908* (2024).

[40] Sriraam Natarajan, Saurabh Mathur, Sahil Sidheekh, Wolfgang Stammer, and Kristian Kersting. “Human-in-the-loop or AI-in-the-loop? Automate or Collaborate?” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 27. 2025, pp. 28594–28600.

[41] Dominik Palla and Antonin Slaby. “Evaluation of generative AI models in python code generation: A comparative study”. In: *IEEE Access* (2025).

[42] Dung Pham and Taher A Ghaleb. “Code Change Characteristics and Description Alignment: A Comparative Study of Agentic versus Human Pull Requests”. In: *arXiv preprint arXiv:2601.17627* (2026).

[43] Giovanni Pinna, Jingzhi Gong, David Williams, and Federica Sarro. “Comparing ai coding agents: A task-stratified analysis of pull request acceptance”. In: *arXiv preprint arXiv:2602.08915* (2026).

[44] Huilian Sophie Qiu, Yucen Lily Li, Susmita Padala, Anita Sarma, and Bogdan Vasilescu. “The signals that potential contributors look for when choosing open-source projects”. In: *Proceedings of the ACM on Human-Computer Interaction* 3.CSCW (2019), pp. 1–29.

[45] Stephan Rabanser, Tim Januschowski, Valentin Flunkert, David Salinas, and Jan Gasthaus. “The effectiveness of discretization in forecasting: An empirical study on neural time series models”. In: *arXiv preprint arXiv:2005.10111* (2020).

[46] Nornadiah Mohd Razali, Yap Bee Wah, et al. “Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests”. In: *Journal of statistical modeling and analytics* 2.1 (2011), pp. 21–33.

[47] Patrick Royston. “Remark AS R94: A remark on algorithm AS 181: The W-test for normality”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 44.4 (1995), pp. 547–551.

[48] Tiberiu Sabau. *Investigating Agentic AI Contributions on “Good First Issues” in Open-Source Projects - Replication Package*. Apr. 2026. DOI: 10.6084/m9.figshare.32044074. URL: https://figshare.com/articles/thesis/Investigating_Agentic_AI_Contributions_on_Good_First_Issues_in_Open-Source_Projects_-_Replication_Package/32044074/1.

[49] Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. “Vibe coding vs. agentic coding: Fundamentals and practical implications of agentic ai”. In: *arXiv preprint arXiv:2505.19443* (2025).

[50] Soumyajit Sarkar. “Comparative Analysis of AI-Powered Coding Assistants for Software Engineering Practices: An Evaluation of Claude Sonnet, ChatGPT, Google Gemini, and Competing Platforms”. In: *ChatGPT, Google Gemini, and Competing Platforms (September 09, 2025)* (2025).

[51] Carolyn B. Seaman. “Qualitative methods in empirical studies of software engineering”. In: *IEEE Transactions on software engineering* 25.4 (1999), pp. 557–572.

[52] Samuel Sanford Shapiro and Martin B Wilk. “An analysis of variance test for normality (complete samples)”. In: *Biometrika* 52.3-4 (1965), pp. 591–611.

[53] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. “A systematic literature review on the barriers faced by newcomers to open source software projects”. In: *Information and Software Technology* 59 (2015), pp. 67–85.

[54] Shuyan Sun, Wei Pan, and Lihshing Leigh Wang. “A comprehensive review of effect size reporting and interpreting practices in academic journals in education and psychology.” In: *Journal of Educational Psychology* 102.4 (2010), p. 989.

[55] Wannita Takerngsaksiri et al. “Human-in-the-loop software development agents”. In: *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. 2025, pp. 342–352.

[56] Ronald J Tallarida and Rodney B Murray. “Chi-square test”. In: *Manual of pharmacologic calculations: with computer programs*. Springer, 1987, pp. 140–142.

[57] Xin Tan, Yiran Chen, Haohua Wu, Minghui Zhou, and Li Zhang. “Is it enough to recommend tasks to newcomers? understanding mentoring on good first issues”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 653–664.

[58] Xin Tan, Minghui Zhou, and Zeyu Sun. “A first look at good first issues on GitHub”. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 398–409.

[59] David Thissen, Lynne Steinberg, and Daniel Kuang. “Quick and easy implementation of the Benjamini-Hochberg procedure for controlling the false positive rate in multiple comparisons”. In: *Journal of educational and behavioral statistics* 27.1 (2002), pp. 77–83.

[60] Martin A Volker. “Reporting effect size estimates in school psychology research”. In: *Psychology in the Schools* 43.6 (2006), pp. 653–672.

[61] 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”. In: *arXiv preprint arXiv:2509.14745* (2025).

[62] Carla Willig and Wendy Stainton Rogers. *The SAGE handbook of qualitative research in psychology*. Sage, 2017.

[63] Clark Wissler. “The Spearman correlation formula”. In: *Science* 22.558 (1905), pp. 309–311.

[64] Wenxin Xiao, Hao He, Weiwei Xu, Xin Tan, Jinhao Dong, and Minghui Zhou. “Recommending good first issues in github oss projects”. In: *Proceedings of the 44th international conference on software engineering*. 2022, pp. 1830–1842.

[65] Wenxin Xiao, Jingyue Li, Hao He, Ruiqiao Qiu, and Minghui Zhou. “Personalized first issue recommender for newcomers in open source projects”. In: *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2023, pp. 800–812.

[66] Haruhiko Yoshioka, Takahiro Monno, Haruka Tokumasu, Taiki Wakamatsu, Yuki Ota, Nimmi Weeraddana, and Kenichi Matsumoto. “Let’s Make Every Pull Request Meaningful: An Empirical Analysis of Developer and Agentic Pull Requests”. In: *arXiv preprint arXiv:2601.18749* (2026).

[67] Mengxi Zhang, Huaxiao Liu, Chunyang Chen, Yuzhou Liu, and Shuotong Bai. “Consistent or not? An investigation of using pull request template in GitHub”. In: *Information and Software Technology* 144 (2022), p. 106797.