Exploring Developer Perceptions on the Potential of CI Recommendation Systems

Osamah H. Alaini
Trent University, Department of Computer Science, Peterborough, Ontario, Canada

Taher A. Ghaleb
Trent University, Department of Computer Science, Peterborough, Ontario, Canada

Abstract

Continuous Integration (CI) is central to modern software development, yet developers often struggle to choose the most suitable CI service. Prior work has identified barriers to CI adoption but offers little empirical evidence on how developers select CI services or whether adoption decisions are driven by genuine project needs versus social influence. This report presents an exploratory survey study addressing that gap. We aim to contact about 5,000 active GitHub developers, including both CI users and non-users. The study investigates: (1) what drives CI adoption and service selection, distinguishing need-driven from socially influenced motivations; (2) whether developers consider CI universally necessary or context-dependent and what barriers hinder adoption; and (3) developers' perceptions of automated CI recommendation systems. Our findings will inform researchers developing CI recommendation systems and practitioners aiming to streamline CI adoption in open-source projects.

2012 ACM Subject Classification Software and its engineering → Software configuration management and version control; Software and its engineering → Empirical software engineering

Keywords and phrases Continuous Integration (CI), CI adoption, CI services, Recommendation system, Developer survey, Empirical study

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Continuous Integration (CI) is central to modern software development, automating testing and deployment while reducing integration errors [12, 21]. CI adoption has increased from about 40% in open-source projects [22] to over 50% with GitHub Actions [19], yet developers still struggle to choose services amid complex trade-offs in features, pricing, and workflow [6, 29]. The impact is substantial: 18.8% of Java projects abandon CI entirely [6], 52% of developers want easier configuration [21], and socially driven adoption rather than project needs wastes time and resources. This challenge is expected to grow with the increasing adoption of AI agents for configuring CI/CD workflows [14].

Prior work has documented adoption barriers through systematic reviews [30], pain-point taxonomies [33], and analyses of technical and social factors [28, 18, 17]. Three critical gaps remain: (1) limited understanding of whether adoption decisions are driven by project needs or social influence, (2) no empirical evidence on whether developers view CI as universally necessary or context-dependent and what barriers prevent adoption among those who view it as inappropriate, and (3) no empirical evidence on developers' receptiveness to automated CI recommendation systems that can advise on suitability, suggest compatible services, and assist with configuration based on project context. These gaps are particularly relevant given the desire for easier configuration [21] and prior work on CI migration [23]. Together, these gaps motivate the first stage of a broader research agenda toward context-aware, AI-enabled CI support, where project suitability, service selection, and configuration are guided by empirical project characteristics rather than defaults or social influence [3].


© Osamah H. Alaini and Taher A. Ghaleb;
licensed under Creative Commons License CC-BY 4.0

This report addresses these gaps through a study that aims to survey about 5,000 active GitHub developers (expected responses: 250–500). Unlike prior surveys targeting only CI users [22, 28] or experienced adopters [29], we explicitly include non-adopters to enable comparative analysis across three RQs: (RQ1) What drives CI adoption and service selection, distinguishing need-driven from socially-influenced motivations? (RQ2) Do developers view CI as universally necessary or context-dependent, and what barriers prevent adoption? (RQ3) Do developers perceive value in CI recommendation systems, and what features do they desire? Specifically, we examine perceived value across three dimensions: assessing project suitability for CI, recommending services given project characteristics (e.g., language, team size, testing practices), and supporting configuration and setup.

The rest of this report is organized as follows. Section 2 reviews related work on adoption barriers, service selection, and CI recommendation systems. Section 3 outlines the study design, sampling strategy, survey instrument, analysis procedures, execution plan, ethical considerations, and threats to validity. Section 4 concludes the report.

2 Background and Related Work

2.1 Background

2.1.1 Continuous Integration: Overview

Continuous Integration (CI) is a software development practice in which developers frequently integrate code changes into a shared repository, triggering automated builds and tests to detect integration issues early [12]. A typical CI workflow involves: (1) committing code changes to version control, (2) automatically retrieving and building the latest code, (3) executing automated tests, and (4) reporting build status to the team [12].

2.1.2 CI Services and Ecosystem

Modern CI platforms fall into two categories: self-hosted solutions (e.g., Jenkins), which require organizations to maintain their own infrastructure, and cloud-based services (e.g., Travis CI, CircleCI, GitHub Actions, AppVeyor), which integrate directly with version control platforms. Before 2019, CI research focused mainly on Travis CI. Since then, the landscape has shifted with GitHub Actions [9, 19], which integrates tightly with GitHub repositories, provides a marketplace of reusable workflow components, and supports matrix builds across multiple environments [9]. This diverse ecosystem forces developers to make complex choices, balancing pricing models, feature sets (e.g., parallelization, caching, deployment integrations), platform support, and configuration complexity [6, 29].

2.1.3 CI Principles and Practices

CI emphasizes automated testing, fast feedback loops, and continuous deployment [12, 24], with workflows typically covering dependency installation, compilation, unit and integration tests, and static analysis. Despite these standardized principles, open-source projects exhibit considerable inconsistencies in CI configurations [15, 1, 2]. Modern CI ecosystems also integrate security tooling; for example, `codeql` is among the most common workflow names, appearing in 3.5% of GitHub Actions repositories [9].

2.2 Related Work

2.2.1 CI Adoption Barriers and Service Selection

Prior research has documented persistent adoption challenges. Hilton et al. [22] surveyed 442 developers and identified setup difficulties, configuration complexity, and lack of experience as key barriers. Pinto et al. [28] surveyed 158 Travis CI users and found that 33.1% were uncertain whether job failures constituted build failures, revealing conceptual gaps. Widder et al. [33] replicated these results, confirming that these barriers persist. Overall, these studies show that adoption barriers are not only technical but also knowledge-based and contextual. Beyond adoption, developers may also struggle with complex service selection decisions. Chopra and Ghaleb [6] analyzed 19K Java projects and found that 64% rely on a single CI maintainer (creating knowledge bottlenecks) and 18% use multiple CI services (indicating uncertainty about which service fits their project). Travis CI abandonment (29.3%) exceeded the overall rate (18.8%), but they did not study CI recommendation system adoption. Gallaba and McIntosh [13] reported that 48.16% of CI configuration code configures job processing nodes and 9.60% of projects contain anti-patterns. Prior work has also shown trade-offs between build speed and reliability [18, 17, 16]. These findings directly motivate our research: understanding what drives adoption and service selection, distinguishing need-driven from socially-influenced motivations (RQ1), whether developers view CI as universally necessary or context-dependent and what barriers prevent adoption (RQ2), and assessing developers' perceptions on developing CI recommendation systems (RQ3).

2.2.2 Qualitative Studies of CI Migration

Complementing quantitative surveys, Mazrae et al. [29] conducted interviews with 22 practitioners working with 31 CI tools, revealing that migration decisions stem from evolving project needs, service discontinuation, or improved platform integration. Despite documenting migration patterns, developer perceptions on developing automated CI recommendation systems remain unexplored.

2.2.3 Tool-Development Efforts and Research Gap

Researchers have developed automated solutions to address CI challenges, including tools for CI service migration [23] approaches addressing CI/CD workflow resource usage and maintenance [31, 4]. While implementing CI recommendation systems involves technical challenges, such as extracting project characteristics and maintaining service knowledge, these efforts suggest that the underlying components are feasible. Despite this progress, little is known about whether developers would adopt automated CI recommendation systems or which features they would value. Understanding these factors is critical, as technology acceptance models (TAM [8] and TAM2 [32]) suggest that perceived usefulness strongly influence whether practitioners adopt new tools, while trust is also widely recognized as an important factor shaping technology adoption decisions. Without empirical evidence on these aspects, future tool development risks becoming misaligned with developer needs. Hence, we ground RQ3 in the TAM constructs of perceived usefulness and behavioral intention, together with trust to evaluate developers' anticipated acceptance of CI recommendation systems.

2.2.4 Prior Developer Surveys

Several large-scale CI surveys have been conducted. Table 1 compares our study with prior CI surveys across key research dimensions. Hilton et al. [22] surveyed 442 developers; Pinto et al. [28] surveyed 158 Travis CI users; Widder et al. [33] replicated Hilton's methodology; Elazhary et al. [11] conducted organizational interviews. While these studies documented barriers and practices, they share three methodological limitations that our study addresses: (1) minimal representation of non-adopters (Hilton et al. surveyed only 7.9% non-users, while Pinto et al., Widder et al., and Elazhary et al. focused exclusively on CI users), which we address via inclusive sampling of both CI users and non-users; (2) no assessment of developers' perceptions on CI recommendation systems despite well-documented configuration challenges (e.g., 52% desire easier configuration [21]), which we address via direct Likert-scale measurement of perceived value and desired features; and (3) the absence of a comprehensive survey of the post-GitHub Actions (2019) CI ecosystem, which we address by surveying the current landscape across RQ1–RQ3.

Table 1 Coverage of CI/CD surveys across key dimensions. Earlier work examined barriers but not service selection, need vs. social influence, or recommendation systems

Question/Objective	Hilton 2016	Hilton 2017	Pinto 2018	Widder 2019	Elazhary 2021	Rostami 2023	This Study
CI adoption barriers	✓(pre-GHA)	✓(pre-GHA)	✓(pre-GHA)	✓(pre-GHA)	✓(pre-GHA)	Partial	✓(post-GHA)
Service selection criteria	✗	✗	✗	✗	✗	✓(n=22)	✓(n=250–500)
Need vs. social influence	✗	✗	✗	✗	✗	Partial	✓(explicit)
Multi-CI usage reasons	✗	✗	✗	✗	✗	✓	✓
Service migration/switching	✗	✗	✗	✓(leavers)	✗	✓(n=22)	✓(n=250–500)
Non-user perspectives	Limited (7.9%)	✗	✗	✗	✗	✗	✓(40–60%)
Developer support for CI recommendation systems	✗	✗	✗	✗	✗	✗	✓
Post-GitHub Actions era	✗	✗	✗	✗	✗	✓	✓

3 Study Design and Execution Plan

3.1 Research Questions and Methodology

We will conduct a survey-based empirical study following established practices from prior CI surveys [22, 28] and GitHub mining guidelines [25] and general empirical software engineering methodology [34]. Respondents will be segmented by CI involvement status (CI users versus non-CI users) based on their self-declaration. In this study, CI users are respondents who report having configured, maintained, or actively used CI in a software project, whereas non-CI users report no such involvement. This segmentation enables comparative analyses, which are essential for understanding developer needs in the development of CI recommendation systems. Based on prior evidence that approximately 40% of GitHub projects use CI [22], and more recent studies indicating adoption above 50% in the post-GitHub Actions era [19], we anticipate our sample will naturally reflect a distribution within this range. Figure 1 provides an overview of our methodology. In particular, we address three research questions:

RQ1: What factors drive CI adoption and service selection, and how do project needs versus social influence shape these decisions?

We investigate three dimensions:

- **RQ1.1:** What motivates initial CI adoption, and to what extent are they driven by project-specific needs versus social influences (e.g., following popular projects)?

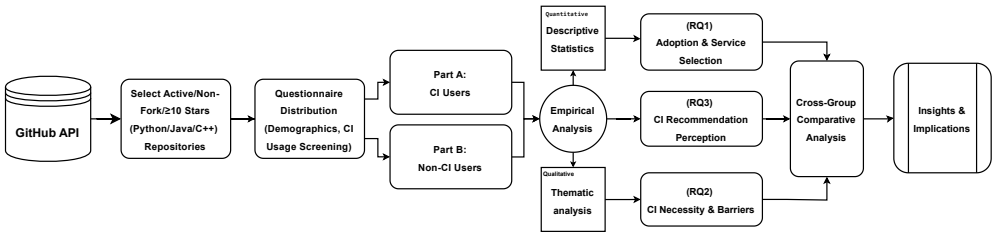

Figure 1 Overview of our survey study design

- **RQ1.2:** Which criteria do developers prioritize when choosing specific CI services (e.g., ease of use, cost, features, integration, reliability)?
- **RQ1.3:** How do developers switch between CI services or adopt multiple services concurrently, and what factors influence these decisions?

Understanding these factors helps identify which adoption motivations and service features a CI recommendation system should prioritize.

RQ2: How do developers perceive the necessity of CI, and what contextual factors limit its adoption?

We investigate three dimensions:

- **RQ2.1:** Do developers consider CI essential for all projects, or beneficial only in specific contexts?
- **RQ2.2:** What reasoning guides developers in choosing to adopt CI selectively rather than universally?
- **RQ2.3:** What types of barriers prevent CI adoption among non-adopters, and how prevalent is each barrier type?

Understanding whether CI is seen as universally necessary or context-dependent informs whether CI recommendation systems should suggest CI for all projects or first assess each project's suitability.

RQ3: How do developers perceive the usefulness and desired features of a CI recommendation system?

We aim to measure the perceived value using Likert scales in terms of usefulness, trust, and likelihood of using recommendations. Desired needs are captured via multi-select checkboxes (e.g., service comparison, configuration templates, troubleshooting guides, integration examples) and open-ended responses, enabling identification of actionable requirements for developing such systems. Based on prior findings that 52% of developers desire easier configuration [21] and documented service selection challenges [6], we anticipate exploring patterns in perceived value across different developer groups.

3.2 Study Design

Target population and sampling. Following GitHub mining best practices [25], we will use the GitHub REST API to sample about 5,000 active open-source developers with publicly available email addresses from repositories that have at least 10 stars and 10 commits in the past 12 months. For each candidate repository, we will identify recent committing developers from the past 12 months and randomly select one developer. We will exclude bots and accounts with non-contactable emails. To prevent repeated invitations to the same person, we will remove duplicated email addresses across all repositories, if any. This 5,000 represents our *sampling population*, i.e., the number of invitation emails to be sent, not the expected number of completed responses. We aim to receive 250–500, as detailed below.

Prospective response rate. We aim for 250-500 complete responses (5-10% response rate), which is motivated by prior CI developer surveys: Hilton et al. [22] achieved 9.8% (442 of 4,508), Pinto et al. [28] achieved 14.4% (158 of 1,100), and Ghaleb et al. [17] achieved 3.5% (139 of 4,366). Hence, we aim for the middle of this range (5-10%) following established strategies: personalized outreach, optimized survey length (≈15 minutes), and mobile-responsive design. Research shows these strategies can collectively increase response rate in developer surveys [10]. We acknowledge that non-CI developers may respond at lower rates, so we will monitor response rate throughout the process and may prioritize non-CI developers, if needed, in the subsequent rounds of invitations.

Expected sample composition Prior work estimates CI adoption in GitHub projects at 40% [22] to over 50% in the post-GitHub Actions era [19]. We therefore conservatively expect 40–60% CI users in our sample. For a minimum sample of 250 developers, this yields about 100 CI and 150 non-CI users; for our 500-developer target, about 200 CI and 300 non-CI users. This distribution provides adequate power for subgroup analyses: Mann-Whitney U tests with a medium effect size ($r = 0.3$) require $n \geq 87$ per group for 80% power at $\alpha = 0.05$ [7], which both sample sizes satisfy.

Survey instrument: We used a two-part questionnaire with conditional routing implemented in Qualtrics.[1] Table 2 gives an overview of the types of questions and answers included at a high-level (full survey instrument is available online[2]).

Participants indicate whether they have been involved with CI (including configuring, maintaining, or actively using CI), routing them to the appropriate part. To ensure consistent interpretation of "CI recommendation systems," the survey presents respondents with the explicit definition provided in Section 1.

- ***Part A (CI Users, 18 questions)*** focuses on adoption motivations, service selection, and switching (RQ1.1, RQ1.2, RQ1.3), opinions about CI universality and selective adoption (RQ2.1, RQ2.2), and perceived value of CI recommendation systems (RQ3).
- ***Part B (Non-CI Users, 14 questions)*** explores views on CI universality (RQ2.1), adoption barriers (RQ2.3), and perceived value of CI recommendation systems (RQ3).

Perceived value. Both *Part A* and *Part B* include 3-4 Likert questions measuring perceived value: (1) usefulness ("Would help me"), (2) trust ("I would trust recommendations"), (3) adoption likelihood ("I would use a CI recommendation system"). Desired needs are captured via a multi-select list (service comparison, configuration templates, troubleshooting guides, integration examples, performance benchmarks, community feedback) and an open-ended text field for additional features.

3.3 Quantitative Analysis

RQ1 (adoption motivations): We will compute descriptive statistics (frequencies, medians, interquartiles) and compare need-driven versus socially-influenced motivation distributions via Chi-square tests. We will also Mann-Whitney U tests [26] (effect size: r) [20] to compare ordinal Likert outcomes. For RQ1.2 (service selection criteria), we will analyze frequencies and crosstabs of prioritized criteria. For RQ1.3 (switching/multi-service adoption), we will compute frequencies and conduct thematic analysis of open-ended explanations.

[1] `https://trentu.qualtrics.com/jfe/form/SV_0fbtOMEDAyNQpam`
[2] `https://docs.google.com/document/d/1xcDl2n1QO9fgpUvvWLDnVPnimkqcMmYi_UQ71FlTpZs`

Table 2 Survey instrument mapping questions to research questions (RQ1: adoption factors; RQ2: barriers; RQ3: perceived value)

#	Question	Answer Options	RQ
Section 1: Demographics (5 items)			
$Q_{1.1}$	Which country do you work from?	Open-ended (country)	–
$Q_{1.2}$	What is your age group?	[Under 24, 25–34, 35–44, 45–54, 55+]	–
$Q_{1.3}$	What is your gender?	[Man, Woman, Non-binary, Prefer not to say]	–
$Q_{1.4}$	How many years of experience do you have in software development?	[< 1, 1–3, 4–6, 7–10, 10+ years]	–
$Q_{1.5}$	Which best describes your current role?	[Engineer, Student, Researcher, DevOps, OSS contributor, Other (open-ended)]	–
Section 2: CI Usage Patterns (routing item)			
$Q_{2.1}$	Have you ever been involved with Continuous Integration (CI) in any of your software projects? This includes: configuring or setting up CI services (e.g., writing .yml files, creating workflows); maintaining or modifying existing CI configurations; and actively using CI (triggering builds, reviewing results, fixing failures).	[Yes → Part A; No → Part B]	–
Part A: CI adopters (18 items, A1–A18)			
$A_{1–3}$	CI experience, scope of adoption, and services adopted	Multiple-choice (experience bands; all vs. some projects; CI services adopted, select all + "Other")	–
A_4	Primary reason for CI adoption	Single-choice (improve quality, catch bugs, team requirement, best practice, automate tasks, recommended, deployment, "Other")	RQ1
A_5	CI service consistency across projects	Single-choice (same/different) + open-ended justification	RQ1
$A_{6–7}$	Service selection criteria consideration	Yes/No + conditional select all that apply	RQ1
A_8	Perceived usefulness of CI recommendation tool	5-point Likert (1: not useful at all . . . 5: very useful)	RQ3
$A_{9–10}$	CI adoption challenges and specific pain points	5-point Likert (overall difficulty) + select up to 3 challenges (build times, flaky tests, config complexity, debugging, timeouts, cost, documentation, integration, team resistance, learning curve, maintenance, none, "Other")	RQ1
$A_{11–12}$	CI service switching and multi-service usage patterns	Yes/No + open-ended justification (for each)	RQ1
$A_{13–14}$	Opinions about CI necessity and adoption suggestions	Yes/No + open-ended justification; open-ended feedback	RQ2
A_{15}	Satisfaction with current CI service selection	5-point Likert (1: very dissatisfied . . . 5: very satisfied) + conditional open-ended (if 1–3: "What would make you more satisfied?")	RQ1
A_{16}	Trust in automated CI recommendation system	5-point Likert (1: no trust at all . . . 5: complete trust)	RQ3
A_{17}	Likelihood to use CI recommendation system	5-point Likert (1: very unlikely . . . 5: very likely)	RQ3
A_{18}	Desired capabilities in CI recommendation system	Select all that apply	RQ3
Part B: Non-CI adopters (14 items, B1–B14)			
B_1	Familiarity with the concept of CI	5-point Likert (1: not familiar at all . . . 5: very familiar)	–
B_2	Reasons for not adopting CI so far	Select all that apply	RQ2
B_3	Single biggest barrier preventing CI adoption	Single-choice (lack of knowledge, time constraints, project complexity, cost, setup complexity, no perceived need, no team requirement, service uncertainty, "Other")	RQ2
B_4	Beliefs about CI necessity for all projects	Yes/No + open-ended justification	RQ2
B_5	Future adoption likelihood if easier setup	5-point Likert (1: very unlikely . . . 5: very likely)	RQ2
B_6	Factors that would motivate CI adoption	Select all that apply	RQ2
B_7	Current code checking process before deployment	Select all that apply	–
B_8	Receptiveness to CI recommendation system	5-point Likert (1: not interested at all . . . 5: very interested)	RQ3
B_9	Concerns, doubts, and feedback about CI adoption	Open-ended response	RQ2
B_{10}	Additional thoughts about CI	Open-ended response	–
B_{11}	Perceived effort to set up and maintain CI	5-point Likert (1: very little effort . . . 5: very significant effort)	RQ2
B_{12}	Trust in automated CI recommendation system	5-point Likert (1: no trust at all . . . 5: complete trust)	RQ3
B_{13}	Likelihood to use CI recommendation system	5-point Likert (1: very unlikely . . . 5: very likely)	RQ3
B_{14}	Desired capabilities in CI recommendation system	Select all that apply	RQ3

RQ2 (CI universality patterns): We will compare response frequencies on whether developers believe all projects should adopt CI between CI users and non-users using Chi-square tests. For RQ2.2 (reasons for selective adoption), we will thematically analyze open-ended responses from developers who adopt CI selectively. For RQ2.3 (barriers among context-dependent believers), we will compute descriptive statistics for informational, technical, and organizational barriers and compare subgroups using Mann-Whitney U tests.

RQ3 (perceived value): Descriptive statistics for perceived value Likert questions (mean, SD, frequencies). Mann-Whitney U tests comparing ratings between CI users (Part A) and non-CI users (Part B; effect size: r). Frequency analysis of multi-select developer needs.

3.4 Qualitative Analysis of Open-Ended Responses

We will conduct an in-depth thematic analysis following Braun and Clarke [5] using a hybrid coding approach. Initial codes will be informed by prior CI research [22, 21, 33, 28], while remaining open to emergent themes. The two co-authors (software engineering researchers experienced in CI) will collaboratively code all responses to ensure consistency. We will monitor thematic saturation in sequential batches, declaring it when no new codes appear across consecutive batches. The coders will jointly refine a coding scheme organized by research question (RQ1: adoption motivations, service selection trade-offs, switching triggers; RQ2: universality patterns, selective adoption reasoning, context-dependent barriers; RQ3: trust concerns, desired features). We will assess inter-rater reliability using Cohen's kappa and resolve discrepancies through discussion. The final scheme will be applied to the full dataset, with 100% of coded responses reviewed for quality assurance. We will extract code frequencies and illustrative quotes, segmented by CI user status.

3.5 Execution Plan

Phase 1: Research Ethics Approval and Setup. We have prepared the full survey protocol with all methodological and ethical details and submitted it for ethics approval. After approval, we will update the Qualtrics survey as required and run a pilot with 10–15 developers to estimate completion time and test technical functionality. We will refine the survey based on pilot feedback, so the final wording and number of questions may differ slightly from those proposed here, while preserving the same constructs and RQ mappings.

Phase 2: Developer Sampling. We will use Python scripts to query the GitHub API with star and activity filters; validate and de-duplicate email addresses; remove bots and invalid emails; and organize the final $\approx 5,000$ invitations by language, geography, and experience level for balanced distribution.

Phase 3: Survey Distribution. We will send personalized email invitations to developers in several rounds, track responses, send one generic reminder (to be ignored by those who already responded), and target 250–500 total responses. If response rate is lower than expected, we may add a small random-draw incentive, consistent with prior CI research [22, 17].

Phase 4: Data Cleaning. We will export anonymized responses; identify incomplete responses (threshold: <80% questions; document exclusion); validate final dataset; generate demographic profile; assess representativeness.

Phase 5: Quantitative Analysis. We will compute descriptive statistics (frequencies, medians, interquartiles) for all Likert questions by CI user status and subgroups (experience, role, project size); perform Mann–Whitney U tests on key questions (e.g., RQ3 usefulness), reporting p-values and effect sizes; create contingency tables for categorical questions and run Chi-square tests; produce visualizations (e.g., box plots and stacked bar charts).

Phase 6: Qualitative Analysis. We will develop coding scheme via initial subset coding (10%) with inter-rater agreement validation (target kappa ≥ 0.70); apply the scheme to the full dataset with spot-check (second coder reviews 20% of remaining responses). Compute code frequencies; extract representative quotes (2-5 per theme); cross-tabulate themes by CI user status and demographic subgroups.

Phase 7: Integration and Synthesis. We will triangulate quantitative and qualitative findings for each RQ: RQ1 (adoption motivations, service selection, switching patterns), RQ2 (universality patterns, selective adoption reasoning, barriers among context-dependent believers), and RQ3 (perceived usefulness ratings with qualitative statements on trust and needs). RQ3 Likert items will follow the Technology Acceptance Model [8], measuring perceived usefulness and behavioral intention as core TAM constructs, plus trust as an additional adoption factor. These items capture anticipated perceptions of a hypothetical CI recommendation system, not actual post-adoption acceptance, recognizing that these may differ. We will derive design implications for CI recommendation systems, document threats to validity (non-response bias, construct validity, limited generalizability), and assess whether results support distinct recommendation paths for CI versus non-CI users.

Phase 8: Manuscript Preparation. We will draft Results section synthesizing quantitative and qualitative findings with figures and tables. Draft Discussion section connecting findings to prior literature (Related Work), interpreting implications, and addressing validity threats. Finalize Methods, Abstract, and Introduction.

3.6 Ethical Considerations

This study involves human participants and will adhere to the Tri-Council Policy Statement: Ethical Conduct for Research Involving Humans [27]. All ethical safeguards detailed below will be followed to protect participant welfare, privacy, and autonomy.

Research Ethics Board (REB) Approval. An ethics application for this study has been approved by Trent University.

Informed Consent. Participants will provide informed consent via a Qualtrics consent page before starting the survey, covering study purpose, procedures, voluntary participation, withdrawal rights, risks, benefits, confidentiality, and contact information. Partial responses under 80% completion will be excluded.

Privacy and Confidentiality. We will store recruitment emails separately in encrypted, access-restricted files. Data will be transmitted via HTTPS and stored on secure institutional servers. Survey responses will be anonymous. Only aggregated, de-identified insights may be shared publicly. Data retention will follow institutional guidelines, and identifiable contact information will be deleted after study completion.

Risk and Benefit Assessment. The survey poses minimal risk, asking only about professional experience and CI usage, with no sensitive, intrusive questions. Potential benefits include advancing understanding of CI adoption and improving CI tools. Foreseeable risks will be negligible compared to everyday professional activities.

Recruitment of Participants. We will recruit participants via publicly available GitHub emails collected using the GitHub API and filtered for active developers. Emails will clearly state the study purpose and emphasize voluntary participation and withdrawal rights.

Transparency and Open Science. We will share anonymized datasets, survey instrument, and analysis code after study completion, pending ethics approval, to ensure that no participants can be identified, balancing transparency with confidentiality.

3.7 Threats to Validity

External validity. Our sample will include only open source GitHub projects and developers with public email addresses, which may limit generalization to enterprise settings or other platforms. As in prior adoption studies, our respondents' experiences may not represent all developers [33]. We will mitigate this threat by sampling diverse GitHub projects and reporting participant and project characteristics. We should note that GitHub will serve only as a sampling frame, not as a proxy for CI usage. CI adoption or non-adoption will be considered through participants' self-reports, since developers sampled from non-CI GitHub repositories may still report CI usage based on experience in other projects or platforms.

Construct validity. Our findings are constrained by the survey questions we asked, which may not fully capture all aspects of CI adoption and developer perceptions. While the instrument will be piloted with 10-15 developers to identify ambiguity and survey timing, following best practices from prior surveys, some nuances may be overlooked or misunderstood, as indicated by Pinto et al. [28].

Internal validity. Self-selection bias is possible: respondents may differ from non-respondents, skewing results toward more interested or experienced developers [22]. As in Hilton et al. [22] and Widder et al. [33], recall and social desirability biases may also affect self-reported answers. We will mitigate these biases by sending reminders, offering small incentives, and monitoring demographics to adjust outreach if some groups are underrepresented. Also, selecting developers with recent commits may favor active over occasional contributors, thus biasing our sampling. Under-response from non-CI users may also skew the sample toward CI users, which will be monitored and addressed through targeted outreach.

Conclusion validity. Non-response and skipped questions limit the scope of inference, so statistical associations should be interpreted cautiously; like most CI survey research, our study is best suited to highlight patterns and generate hypotheses rather than establish causal effects. We will mitigate these limitations by encouraging complete responses and interpreting results cautiously.

4 Conclusion

This report addresses a key gap in CI research: little is known about whether developers perceive value in CI recommendation systems or view CI as universally necessary versus context-dependent. By aiming to survey a target of 250–500 GitHub developers (CI and non-CI users alike), we will provide the first empirical evidence on developers' perceptions of such systems, including perceived value, trust, and desired features. We make three anticipated contributions. First, we identify which adoption factors developers prioritize, distinguishing need-driven from socially-influenced motivations, service selection criteria, and switching patterns. Second, we determine whether developers view CI as universally necessary or context-dependent, revealing opinions on suitability across project types, reasoning for selective adoption, and barriers among those who view CI as context-dependent, thus informing whether recommendation systems should advise CI universally or first assess project suitability. Third, we provide empirical evidence on developers' perceptions of CI recommendation systems, testing whether the majority perceive value and whether non-CI users report higher perceived value than current users. Our findings will benefit both researchers developing CI recommendation systems and practitioners streamlining adoption, ensuring future efforts address actual developer needs rather than assumed ones.

References

1 Edward Abrokwah and Taher A Ghaleb. An empirical study of complexity, heterogeneity, and compliance of GitHub Actions workflows. *arXiv preprint arXiv:2507.18062*, 2025.

2 Edward Abrokwah and Taher A Ghaleb. How compliant are GitHub Actions workflows? a checklist-based study with LLM-assisted auditing. In *International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2026.

3 Osamah H. Alaini and Taher A. Ghaleb. A vision for context-aware CI adoption decisions. In *Companion Proceedings of the 34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2026.

4 Islem Bouzenia and Michael Pradel. Resource usage and optimization opportunities in workflows of GitHub Actions. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–12, 2024.

5 Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

6 Nitika Chopra and Taher A Ghaleb. From first use to final commit: Studying the evolution of multi-CI service adoption. In *2025 IEEE International Conference on Software Maintenance and Evolution*, pages 773–778. IEEE, 2025.

7 Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 1988.

8 Fred D Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 13(3):319–340, 1989.

9 Alexandre Decan, Tom Mens, Pooya Rostami Mazrae, and Mehdi Golzadeh. On the use of GitHub Actions in software development repositories. In *2022 IEEE International Conference on Software Maintenance and Evolution*, pages 235–245. IEEE, 2022.

10 Don A Dillman, Jolene D Smyth, and Leah Melani Christian. Internet, phone, mail, and mixed-mode surveys: The tailored design method. *Indianapolis, Indiana*, 17, 2014.

11 Omar Elazhary, Colin Werner, Ze Shi Li, Derek Lowlind, Neil A Ernst, and Margaret-Anne Storey. Uncovering the benefits and challenges of continuous integration practices. *IEEE Transactions on Software Engineering*, 48(7):2570–2583, 2021.

12 Martin Fowler. Continuous integration, 2006. Accessed: Oct. 27, 2025. URL: `https://martinfowler.com/articles/continuousIntegration.html`.

13 Keheliya Gallaba and Shane McIntosh. Use and misuse of continuous integration features: An empirical study of projects that (mis) use Travis CI. *IEEE Transactions on Software Engineering*, 46(1):33–50, 2018.

14 Taher A Ghaleb. When AI agents touch CI/CD configurations: Frequency and success. In *Proceedings of the 23rd International Mining Software Repositories Conference*, 2026.

15 Taher A Ghaleb, Osamah Abduljalil, and Safwat Hassan. CI/CD configuration practices in open source Android apps: An empirical study. *ACM Transactions on Software Engineering and Methodology*, 35(2):1–40, 2026.

16 Taher A Ghaleb, Daniel Alencar da Costa, and Ying Zou. The promise and reality of continuous integration caching: An empirical study of Travis CI builds. In *International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2026.

17 Taher A Ghaleb, Safwat Hassan, and Ying Zou. Studying the interplay between the durations and breakages of continuous integration builds. *IEEE Transactions on Software Engineering*, 49(4):2476–2497, 2022.

18 Taher Ahmed Ghaleb, Daniel Alencar Da Costa, and Ying Zou. An empirical study of the long duration of continuous integration builds. *Empirical Software Engineering*, 24(4):2102–2139, 2019.

19 Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the rise and fall of CI services in GitHub. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering*, pages 662–672. IEEE, 2022.

20 Robert J Grissom and John J Kim. *Effect sizes for research: A broad practical approach.* Lawrence Erlbaum Associates Publishers, 2005.

21 Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. Trade-offs in continuous integration: assurance, security, and flexibility. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 197–207, 2017.

22 Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. Usage, costs, and benefits of continuous integration in open-source projects. In *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*, pages 426–437, 2016.

23 Md Nazmul Hossain and Taher A Ghaleb. CIgrate: Automating CI service migration with large language models. *arXiv preprint arXiv:2507.20402*, 2025.

24 Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

25 Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining GitHub. In *Proceedings of the 11th working conference on mining software repositories*, pages 92–101, 2014.

26 Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

27 Panel on Research Ethics. Tri-council policy statement: Ethical conduct for research involving humans – tcps 2 (2018). `https://ethics.gc.ca/eng/policy-politique_tcps2-eptc2_2018.html`, 2018. Accessed: 2025-11-19.

28 Gustavo Pinto, Fernando Castor, Rodrigo Bonifacio, and Marcel Rebouças. Work practices and challenges in continuous integration: A survey with Travis CI users. *Software: Practice and Experience*, 48(12):2223–2236, 2018.

29 Pooya Rostami Mazrae, Tom Mens, Mehdi Golzadeh, and Alexandre Decan. On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. *Empirical Software Engineering*, 28(2):52, 2023.

30 Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943, 2017.

31 Pablo Valenzuela-Toledo, Alexandre Bergel, Timo Kehrer, and Oscar Nierstrasz. The hidden costs of automation: An empirical study on GitHub Actions workflow maintenance. In *2024 IEEE International Conference on Source Code Analysis and Manipulation*, pages 213–223. IEEE, 2024.

32 Viswanath Venkatesh and Fred D Davis. A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management science*, 46(2):186–204, 2000.

33 David Gray Widder, Michael Hilton, Christian Kästner, and Bogdan Vasilescu. A conceptual replication of continuous integration pain points in the context of Travis CI. In *Proceedings of the 2019 27th acm joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pages 647–658, 2019.

34 Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. *Experimentation in software engineering*, volume 236. Springer, 2012.