Quality Assurance Awareness in Open Source Software Projects on GitHub

Ali Khatami
Delft University of Technology
Delft, The Netherlands
s.khatami@tudelft.nl

Andy Zaidman
Delft University of Technology
Delft, The Netherlands
a.e.zaidman@tudelft.nl

***Abstract*—Software engineers employ a variety of approaches to ensure the quality of software systems, including software testing, modern code review, automated static analysis, build automation, and continuous integration. To make effective decisions regarding quality assurance (QA), software engineers need to have an awareness of (1) the QA approaches that are in use in a project, and (2) how they are used. Through an exploratory, mixed-methods investigation we set out to better understand the awareness of software engineers in open-source software (OSS) development with regard to QA practices. This involved a large-scale survey of 471 maintainers and contributors on GitHub. Our findings indicate that a high-level awareness among the respondents is common, but also that the respondents are less certain about how the practices are adopted; we further consider the perspective of both the contributor and the maintainer.**

***Index Terms*—Software Quality Assurance, Open Source Software (OSS), Software Engineering, Software Testing, Code Review, Continuous Integration, Automation Workflows, GitHub**

I. Introduction

As we have grown accustomed to living in a software-filled world, the quality assurance of that software is indispensable [1], because of the potential catastrophic consequences [2]. To ensure the quality of these software systems, software engineers can use a range of quality assurance approaches, e.g., software testing [3]–[12], modern code review [5], [13]–[15], automated static analysis [16]–[21], and build automation [22]–[27].

In order to make effective decisions with regard to quality assurance, we postulate that software engineers need to have an awareness of the quality assurance (QA) approaches that are in use in a project. This corresponds with the *situational awareness* (SA) theory from psychology, that describes that an understanding of an environment, its composing elements, and its evolution over time, is important for effective decision making in many environments [28]. This study projects situational awareness onto the domain of QA in software engineering [29]. Our motivation to study the awareness that software engineers working in the Open Source Software (OSS) domain have, is driven by the fact that while many QA tools and principles are available, and many OSS projects adopt them, it is as yet unclear how *aware* software engineers are about the use of QA in the project. Building up our understanding of their awareness, can be a stepping stone towards understanding the decision making process surrounding QA, and potentially improving existing tools to stimulate awareness.

While there have been investigations into the information needs in such contexts as agile software development [30], software development analytics [31], collocated software development teams [32], contemporary code review [33], and CI & CD [34], there is a notable research gap concerning the *overall awareness* of quality assurance practices among software engineers in the context of OSS development. An additional dimension of interest is that GitHub as a social coding platform offers several features to enable or stimulate quality assurance, e.g., the pull-based development model [35], modern code review [13], code coverage reporters [36], and the utilization of GitHub Actions [37], Apps, and Bots [38],

We carry out an exploratory investigation to find out more about OSS developers' awareness about quality assurance in their projects on GitHub. For our investigation, we conduct an online survey with 471 maintainers [39] and contributors [40]. As a means of measuring awareness, we include questions regarding their adoption of various QA tools and practices, but we equally gauge for their knowledge regarding what actually happens during these QA practices, e.g., how many code review comments did you get, or what type and level of code coverage is reported for your project? Our investigation is steered by the following five research questions.

RQ1 How aware are developers of testing in their projects?

Our first research question investigates how aware OSS maintainers and contributors are when it comes to automated testing, a QA practice that was earlier identified to be important in the pull-based development model [40].

RQ2 How aware are developers of code reviews in their projects?

In our second RQ we gauge how aware the software engineers that we survey are with regard to code reviewing, a QA practice that has been shown to decrease functional and maintainability defects in code [13], [15], [41].

RQ3 How aware are developers of automation workflows in their project, and what are the purposes for which they employ these workflows?

Third, our investigation delves into the awareness of maintainers and contributors regarding *automation workflows* [42], [43]. These encompass various tools such as GitHub Actions

(GHA)[1], GitHub Checks[2], GitHub Apps[3], and bots, and can help ensure the quality of software during development.

RQ4 How aware are developers of continuous integration in their projects?

Next, we zoom in on the awareness of maintainers and contributors in OSS of CI, as an established QA practice that checks builds, tests, static analysis, etc. [17], [22], [44].

RQ5 How aware are developers of guidelines in their project?

Lastly, we ask OSS maintainers and contributors to find about whether they are aware of their projects contributing guidelines (e.g., a CONTRIBUTING.md file). We mainly aim to investigate whether projects have contribution guidelines for newcomers, and if projects have any guidelines to keep their software buildable after making changes [29], [45], [46].

Our research findings indicate that respondents are generally aware of the quality assurance practices in use in their OSS projects, however, when it comes to more detailed questions like when a specific QA should be used, or to what level a QA practice is (or should be) done, e.g., levels of code coverage, there is generally a lack of awareness. Similarly, we also noted a lack of awareness surrounding the use of automated workflows, and the benefits of code reviews. Overall, maintainers of projects are more aware of QA practices than contributors, with the notable exception of the presence and use of contribution guidelines.

II. Background and Related Work

Software quality assurance is composed of an ensemble of techniques to guard the quality of a piece of software under scrutiny. As such, various sources of information together provide a comprehensive understanding of the state-of-the-quality. These sources include testing reports, code reviews, CI & CD reports, documentation, and potentially other reports. The knowledge of, and ability to access and utilize this information enhances software engineer's awareness of quality problems, and forms an important prerequisite for further (preemptive) actions. In the next sections, we explore literature of quality assurance practices in software engineering.

A. Information Needs in Software Engineering

Several studies have explored the information needs of software engineers. Baysal *et al.* have investigated the information needs of software engineers when working with bugs, and have made suggestions for improving issue tracking systems [47]. Buse and Zimmermann have studied the information needs of software development managers and have distilled a set of guidelines for software analytics tools [31]. Pascarella *et al.* highlighted seven high-level information needs in modern code review [33], while Ahmad *et al.* have categorized information needs in the context of Continuous Integration [48].

[1]https://docs.github.com/en/actions, last visit on July 2023.

[2]https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/collaborating-on-repositories-with-code-quality-features/about-status-checks#checks, last visit on July 2023.

[3]https://github.com/marketplace?type=apps, last visit on July 2023.

B. Quality Assurance Practices

Key QA practices include testing, code reviews, continuous integration (CI), and automated static analysis. These practices have been extensively studied both individually and in combination [29].

Kochhar *et al.* found that open-source projects with more lines of code, and a higher number of developers tend to have a greater number of test cases [49]. Vassallo *et al.* investigated automated testing and source code quality inspection in OSS and found that only 11% of builds do quality control in CI [50]. Hilton *et al.* zoomed in on usage, costs, and benefits of CI in OSS. They found popular projects are more likely to use CI, and projects using CI, release more often, accept pull requests faster, and have developers who are less concerned about build breakage compared to projects not using CI [24]. Cassee *et al.* investigated the impact of CI on code reviews. They found adopting CI leads to an average reduction of one review comment per pull request, highlighting the time-saving benefits of adopting CI in code review [51]. McIntosh *et al.* studied the impact of modern code review on software quality. Their results indicate the negative impact of poorly reviewed code on software quality [52]. Beller *et al.* found that 60% of the most popular open-source projects seem to use automated static analysis tools (ASATs) [16]. Zampetti et al. found that open-source software (OSS) build breakages resulting from automated static analysis tools (ASATs) failures mostly result from failure to adhere to code standards and missing licenses; vulnerabilities contributed less frequently to build failures [53]. In the context of automated workflows, studies looked at expectations of code review bots [54], usage of GHA [37], [55], [56] and its influence on the adoption of other CI services on GitHub [57]. However, there remains a knowledge gap regarding the awareness of all these tools within the QA context. Khatami and Zaidman investigated the state of QA practices in OSS quantitatively, revealing that QA techniques are mostly *not* being used in conjunction [29]. Complementing their study, our study will provide insights into the awareness of software engineering professionals regarding QA practices.

III. Study Methodology

With approval from our university's ethics committee, we conducted a mixed-methods exploratory study, utilizing multiple-choice and open-ended questions in our survey. We used both qualitative and quantitative approaches to analyze the collected data to assess the level of awareness among 471 contributors and maintainers of open-source software projects on GitHub regarding quality assurance in their projects.

A. Survey Design

Our survey follows a specific structure, encompassing the following components: informed consent, 7 demographic questions, 24 question regarding quality assurance practices within GitHub (see Table II).

Prior to the main survey, we conducted a pilot run with six colleagues to gather feedback on the survey. Their feedback

contributed to refining the phrasing of questions and answer options, and adding a back button to our survey.

We run our survey on the Qualtrics[4] platform.

B. Sampling

In our study, we focused on engaging with active and popular open-source software projects on GitHub. To ensure a representative sample, we utilized GitHub Search [58] to select projects. We selected projects with more than 100 stars, over 50 contributors, and more than 100 pull requests, while excluding forks. We also only considered recent projects with a last commit made after January 1, 2023. This yielded a total of 5,273 projects. We used this set of projects to invite participants for the subsequent phase of our study.

C. Attracting participants

In addition to promoting the online survey through our Twitter accounts to invite open-source developers to participate, we proactively reached out to the developers of our selected projects on GitHub (Section III-B) to encourage their involvement in our survey. We specifically targeted three groups of developers in the selected projects: assignable users[5] of projects, authors, and assignees of their most recent pull requests. In doing so, we aimed to ensure a balanced representation of both contributors and maintainers.

We collected 18,287 data records containing developers' contact information. We removed any redundant records, resulting in a final dataset of 12,337 unique records.

To invite these developers to voluntarily participate in our survey, we sent personalized emails to 12,337 email addresses. In each email, we included relevant information such as their GitHub user account, the repository they have contributed to, a clear explanation of our research objectives, and estimation of the time needed to complete the survey (15 minutes). Besides, we emphasized the anonymity of their participation by providing an anonymous link to the survey.

Because of the anonymous link to the survey, we are not able to provide an exact response rate for each source of recruitment (targeted recruitment versus Twitter).

We conducted our survey from February 19 to March 10, 2023, and we received 644 non-empty submissions. After filtering out incomplete submissions, we obtained a final total of 471 complete responses. However, this does not mean that all questions were answered since the survey questions were optional. Precise information regarding how many participants answered each question is presented in the results section.

D. Data Analysis

Out of the 24 non-demographic and non-consent questions in the survey, 11 were open-ended. To analyze these open-ended questions, we employed an open card sorting approach [59]. Initially, the first author independently performed coding and categorization, manually sorting the responses into meaningful groups [60]. To establish consensus, a discussion meeting was conducted between both authors to finalize the code names and categories for each question. We opted for open card sorting to allow codes and categories to emerge naturally, without imposing pre-defined groups.

For questions primarily focused on naming tools or indicating the presence or absence of specific practices, we categorized the responses into a limited number of emerging categories. Conversely, for questions allowing more open-ended responses, we employed a combination of open coding and card sorting techniques to identify themes.

Additionally, we included 10 text-entry fields for certain answer options in multi-option questions. We did this to clarify the "Other" option and to collect specific information such as comment counts based on selected options, or relevant links. We also used open card sorting.

[4]https://www.qualtrics.com/, visited on July 2023.

[5]https://docs.github.com/en/graphql/reference/objects

TABLE I
PARTICIPANT RESPONSES TO DEMOGRAPHIC QUESTIONS.

	<1	**1—3**	**3—6**	**7—10**	**>10**
Experience in Software Development (in years)	2 (0.4%)	38 (8.1%)	101 (21.9%)	76 (16.1%)	252 (53.5%)
OSS Experience (in years)	25 (5.3%)	82 (17.6%)	122 (26.1%)	83 (17.6%)	157 (33.3%)
Occupation(s) (multiple choice)	Academia 15.4%		Industry 65.2%	OSS 34%	Other 12.8%
Role in Project	**Contributor** 187 (39.8%)				**Maintainer** 283 (60.2%)
Contributing to an OSS project is the main professional activity?	**Yes** 174 (36.9%)				**No** 297 (63.1%)

IV. PARTICIPANTS

Our survey had seven demographic questions, gauging the experience of our 471 participants. In particular, we asked how long they have been developing software, how long ago they first contributed to an OSS project, the repository handle of the project they recently contributed to, how many pull requests they made, their role in the project, their occupation, and if contributing to OSS is their main professional activity.

Table I shows an overview of the participants to our survey. Regarding participant roles, the majority (60.2%) identified themselves as maintainers, integrators, project (co-)owners, or other similar positions. We refer to this group as *maintainers*. The remaining participants self-identified as *contributors*, and only one respondent left this question open.

Of all respondents, 451 provided the handle (`username/repository`) of the most recent GitHub project they had contributed to. There were 415 distinct OSS repositories mentioned in total.

V. RESULTS

In this section, we present an analysis of the survey data, structured along the quality assurance practices and our research questions, see Table II.

In our investigation, we aimed to gather insights into the current landscape of QA awareness among OSS developers by examining both their awareness and adoption of consolidated QA

resources on GitHub. We evaluated respondents' awareness level based on five types of answers to our survey questions: (1) "I don't know'' answers, these responses explicitly indicate a lack of awareness. (2) "No or Zero'' answers: these answers indicate that the practice is not followed in the project or no information is available in that area, which we use as a proxy for lack of awareness. (3) "No Answers'': when respondents leave a question blank, we also consider it as a proxy for lack of awareness, although it is a limitation of our work as some respondents might be aware but chose not to answer. (4) "Invalid Answers'': these are answers that are not related to the question or are too general or vague, implicitly indicating a lack of awareness about the topic. (5) "Other Answers": these are informative and valid responses that explicitly indicate the respondent's awareness of the topic. A summary of all answers is shown Figure 1; we will use this information to answer each of the research questions.

A. *RQ1) How aware are developers of testing in their projects?*

T-Q1) Does the project have automated (scripted) tests?: The majority of respondents (87.5%) reported having automated tests in their project, while 10% said they do not have tests, and 2.5% did not know the answer to this question. Respondents who did not select "Yes" as their answer to this question were excluded from getting any subsequent questions about testing in the survey.

T-Q2) What kind of (functional) tests do you have in the project?: Out of all respondents, 84.7% reported knowing the specific type(s) of testing used in their project (e.g., unit testing, integration testing, etc.). A large subset of the population indicated to have unit tests in their project (92%), while 72% mentioned having integration tests (72%). A smaller group of respondents reported using validation (29%) and system testing (24%). Another 30 respondents (7.5%) selected the "Other" option and specified additional testing methods not included in the answer options. Among those end-to-end testing (7), fuzzing, performance, and UI testing (3) were the most frequent ones.

T-Q3) What kind of test coverage information is available for the project?: Of the respondents who answered this question, 80.8% reported that statement/line coverage information was available in their project. Branch coverage was indicated to be available by 41.4% of respondents, while 8.8% reported using mutation score to assess their tests. Additionally, 43 respondents (17.3%) selected the "Other" option and specified additional types of code coverage; out of those responses, 32 were valid, with 25 indicating that no code coverage information was available to them, while 2 were unsure. Other responses included using *Coveralls.io*[6] (1), class coverage (1), line coverage (2), and branch coverage (1).

T-Q4) Considering your answer to the previous question, do you know how much is the code coverage of the project?: Around half of the participants (53.7%) reported that code coverage information was not available on the project's GitHub page. In contrast, 32.6% of respondents indicated that they knew their project's code coverage, and 13.7% reported that the information could be found on the project's GitHub page.

We asked the participants who indicated that they knew the value of code coverage to provide this value through a text input field. Out of 124 entries received, 106 contained valid code coverage information. For participants that reported that the code coverage information was available on their project's GitHub page, we requested the specific location of the information through a separate text input field; of the 52 entries received, 11 provided a valid location for the code coverage information, while 21 provided the actual code coverage. The participants identified various locations to access code coverage information, including the README on Github, GitHub Actions, external tool reports (e.g., *Codecov*[7], pull requests, CI status/pipeline, and build artifacts.

Further analysis based on the classification scheme proposed by Heitlager *et al.* [61] showed that most (61.3%) respondents' projects have either high (39) or very high code coverage (26), while 24.5% had a moderate level of code coverage (26), and the remaining 14.2% of projects had either a low (12) or very low (3) code coverage levels.

Participants were more responsive when asked about the type of code coverage in their project (T-Q3 with 249 responses) compared to when asked about its value (T-Q4 with 176 responses saying either they know the value or they can find it on project's GitHub repository).

T-Q5) What kind of tooling do you use for testing the project?: Of all responses, 343 provided valid inputs. Upon analysis (explained in Section III-D – Data Analysis) of responses, we found that all respondents mentioned the name of at least one tool, framework, library, or approach for testing in their project, with some mentioning up to 15 such tools. The mean number of tools mentioned was 2.16, and the median was 2. From these responses, we selected the most commonly mentioned tools, which are shown in Table III.

RQ1 summary. Figure 1 shows that ∼87% of the participants to our survey indicate to know whether the project they most recently contributed to has test scripts (T-Q1). Similarly, ∼85% of the respondents know the type(s) of tests a project employs (T-Q2), and ∼74% know the testing tools in use (T-Q5). However, the awareness surrounding the type of test coverage (T-Q3; ∼53%) and the actual level of test coverage (T-Q4; ∼37%) is considerably lower.

B. *RQ2) How aware are developers of code reviews in their projects?*

For the following questions, we asked the survey participants to reflect on their most recent involvement in a pull

[6]https://coveralls.io/, last visit on July 2023.

[7]https://about.codecov.io/), visited on July 2023.

TABLE II
SURVEY QUESTIONS DETAILS

Code	QA-related Category	Question	Type
T-Q1	Testing	Does the project have automated (scripted) tests?	multiple-choice
T-Q2	Testing	What kind of (functional) tests do you have in the project? (leave open if you don't know)	multiple-choice
T-Q3	Testing	What kind of test coverage information is available for the project? (leave open if you don't know)	multiple-choice
T-Q4	Testing	Considering your answer to the previous question, do you know how much is the code coverage of the project?	multiple-choice
T-Q5	Testing	What kind of tooling do you use for testing the project? (frameworks, libraries, etc.)	text entry
CR-Q1	Code Review	How many comments did you receive (as the author of the pull request), or give (as the maintainer)?	text entry
CR-Q2	Code Review	The comments that you received/gave were about:	multiple-choice
CR-Q3	Code Review	How many code reviewers were involved in the pull request? (Not considering code review bots)	multiple-choice
CR-Q4	Code Review	Does the project use GitHub's Checks in pull requests?	multiple-choice
CR-Q5	Code Review	What kind of reports do you get from GitHub's Checks' comments on your pull requests?	multiple-choice
CR-Q6	Code Review	How does code reviewing help you?	text entry
AW-Q1	Automation Workflows	Do you use GitHub Actions and/or GitHub Apps in the project?	multiple-choice
AW-Q2	Automation Workflows	Which type of GitHub Actions do you use in the project? (Please name them)	text entry
AW-Q3	Automation Workflows	Which GitHub Apps do you use in the project? (Please name them)	text entry
AW-Q4	Automation Workflows	Do you use bots in the project?	multiple-choice
AW-Q5	Automation Workflows	Which bots do you use in the project? (Please name them)	text-entry
AW-Q6	Automation Workflows	Do you use ASATs in the project?	multiple-choice
CI-Q1	Continuous Integration	Does the project have CI workflows configured on GitHub?	multiple-choice
CI-Q2	Continuous Integration	How do you (or other people in the project) configure CI workflows on GitHub?	text entry
CI-Q3	Continuous Integration	When and how do you check the project's CI results on GitHub?	text entry
CI-Q4	Continuous Integration	Describe a situation in which CI workflows helped you in your contribution to the project.	text entry
G-Q1	Guidelines	Does the project have a checklist/guideline to check before contributors submit or maintainers review a pull request?	multiple-choice
G-Q2	Guidelines	Does a newcomer get any guidelines about how to contribute to the project? If yes, can you provide a link to it?	text entry
G-Q3	Guidelines	Does the project have specific guidelines (e.g., CONTRIBUTING.md) to make the latest version of the project buildable/compilable?	text entry

request (PR) when responding.

CR-Q1) How many comments did you receive (as the author of the pull request), or give (as the maintainer)?: Figure 2 displays a box plot based on the number of comments provided by 373 respondents. Also, 57 responses were excluded due to imprecise or vague numerical ranges (e.g., thousands, hundreds, or less/more than 10, etc.). Only exact values or specific ranges were included, and in cases of ranges, the average was calculated. Accordingly, the median number of comments per PR is 3. Furthermore, 50% of the respondents' PRs had between 1 and 3 comments, while the remaining 50% ranged from 3 to 11 comments.

TABLE III
TOP 50% TOOLING, FRAMEWORKS, LIBRARIES, OR APPROACHES USED FOR TESTING (T-Q5). ALL WEBSITES VISITED JULY 2023.

Categories	URL	# of mentions	%
PyTest	https://pytest.org/	50	6.8%
PHPUnit	https://phpunit.de/	35	4.8%
GitHub Actions	https://docs.github.com/en/actions	35	4.8%
Go test	https://pkg.go.dev/testing	30	4.1%
JUnit	https://junit.org/	26	3.5%
Cargo (Rust)	https://doc.rust-lang.org/cargo/	24	3.3%
Custom		22	3%
Rust test	https://doc.rust-lang.org/cargo/	20	2.7%
Jest	https://jestjs.io/	19	2.6%
CodeCov	https://about.codecov.io	15	2%
Rspec (Ruby)	https://rspec.info/	12	1.6%
Scripts (bash, etc.)		12	1.6%
Unittest (Python)	https://docs.python.org/3/library/unittest.html	11	1.5%
Google test	https://github.com/google/googletest	10	1.4%
Coveralls	https://coveralls.io	9	1.2%
Xunit (.Net)	https://xunit.net/	8	1.1%
Coverage.py	https://coverage.readthedocs.io/en/7.2.7/	8	1.1%
Playwright	https://playwright.dev/	8	1.1%
Jenkins	https://www.jenkins.io/	7	1%
Cypress	https://www.cypress.io/	7	1%
Others (246 different categories)		366	50%
Total		734	100%

CR-Q2) The comments that you received/gave were about?: We received 387 responses for this question. The majority of respondents (63%) reported that their comments were related to *Code Quality or Maintainability Defects* [15], while 35.7% of respondents indicated that their comments were about "Functional Defects". A slightly lower percentage indicated the "Tests" option (32%). Moreover, 39.5% selected the "Other" option to specify additional topics for their comments in a PR.

For this question, we asked participants to indicate the number of comments they had per specific topic they selected from the answer options. The comment count distribution for each topic is depicted in Figure 3. The median number of comments for the "Tests" and "Functional Defects" categories was 1, which was lower compared to the other two categories with a median of 2. The number of responses provided for each topic were as follows: "Tests" (52), "Code Quality or Maintainability Defects" (100), "Functional Defects" (62), and "Others" (49).

We present the qualitative analysis (see Section III-D) of the "Other" responses in Table IV. The "functional aspects" of change encompass comments related to business logic, code behavior, design decisions, architecture, features, functionality, implementation, solution/approach, compatibility, dependencies, CI, build, testing, bugs, defects, and API. The "maintainability aspects" involve comments related to code quality of the contribution, textual content, formatting, code style, documentation, change log, and typos. The "non-functional aspects" of change include mentions of performance, accessibility, and appearance. In the category of "social and communication", respondents highlighted comments

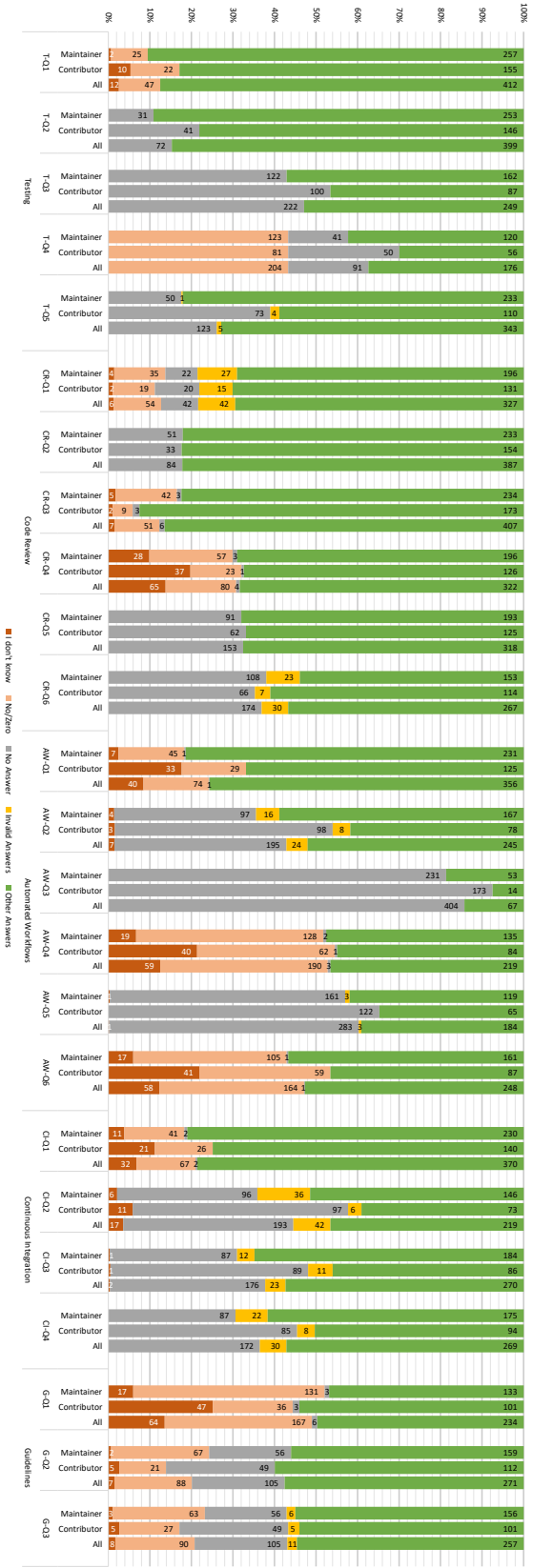

Fig. 1. Summary of answers to all questions.

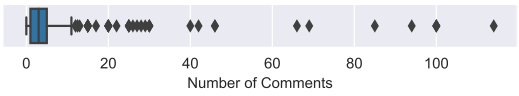

Fig. 2. Comments received/given in a pull request.

that involve pinging someone to review the PR, expressing gratitude towards contributors, and providing encouragement. Lastly, the "suggestions" category includes comments that offer improvements, suggestions, clarification, verification of changes, and discussions about unnecessary changes.

Of interest to note here is that respondents used the "Other" category to list defects that we would classify as either *functional* or *maintainability* aspects (also see the work of Beller *et al.* [15]). However, the respondents were perhaps not aware of the exact terminology. We explicitly chose to not reclassify these answers.

CR-Q3) How many code reviewers were involved in the pull request? (Not considering code review bots): Among the respondents that provided an answer, 11% reported that no reviewers were involved in their PR, 46.6% reported one reviewer, while 28.4% had two reviewers, and 12.5% selected the "more than 2 reviewers" option. With 1.5% of the respondents indicating to be unsure of how many reviewers were involved, we observe a high level of awareness.

CR-Q4) Does the project use GitHub Checks in pull requests?: GitHub Checks provide information on build outputs, static code analysis results for the changes made in a PR. This information is provided on GitHub's PR page, and contributes to the general QA awareness. We asked respondents whether they use this feature in their projects.

Half of the respondents (50.3%) indicated to use *Checks*, while 35.9% said they do not, and 13.8% are not aware of it; four did not provide an answer.

CR-Q5) What kind of reports do you get from GitHub's Checks' comments on your pull requests?: Participants who

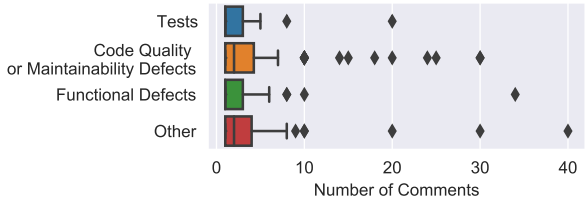

Fig. 3. Count of comments per topic in a pull request.

TABLE IV
CATEGORIES OF OTHER TOPICS OF COMMENTS IN A PR (CR-Q2)

Categories	**# of mentions**	**%**
Functional, logical, behavioral aspects of change	51	38%
Maintainability aspects of change	29	22%
Social and communication	21	16%
Suggestions, improvements, and verification of change	18	13%
Non-functional aspects of change	3	2%
Other	12	9%
Total	134	100%

TABLE V
CATEGORIES OF HOW DOES CODE REVIEWING HELP YOU (CR-Q6)

Categories	# of mentions
Non-Functional (173)	
Code Quality	161
Code Performance	8
Security	4
Functional (124)	
Error/Bug Prevention	120
Verification	14
Knowledge Sharing (65)	
Learning & Knowledge Sharing	65
Other (29)	
Design (code, architecture, and software)	12
Testing	6
Speed-up Workflow	3
Community	3
Collective Decision-making	1
Static Analysis	1
Team Confidence	1
Think More	1
Project Policy	1

answered *Yes* to CR-Q4 were asked about the types of reports they receive from the *Checks* feature. The most commonly selected ones were "Test results" (57.8%) and "Build checks" (55%); these two options were frequently selected together because tests are run during the build. Other options selected were "Code quality warning" (39.4%) and "Code coverage reports" (23.3%). Only a small percentage (4%) selected the "Other" option. Of those, 22 respondents provided additional information about the types of checks they use, including DCO (Developer Certificate of Origin)/commit format checks, licensing checks, compatibility checks, and security checks.

CR-Q6) How does code reviewing help you?: A total of 297 responses were collected for the question. However, 30 of these responses were excluded as they were considered too general (e.g., "yes," "no," "it doesn't") or unrelated to the question. We conducted a qualitative analysis (see Section III-D) of the remaining responses and identified several high-level categories. The frequency of these categories is summarized in Table V. It should be noted that some respondents mentioned multiple reasons why code reviewing was helpful to them. The most frequently mentioned category was "improving code quality" (173 mentions), followed by "error/bug prevention" (124 mentions), and "knowledge sharing" (65 mentions).

RQ2 summary. Figure 1 shows that ∼69% (CR-Q1) and ∼68% (CR-Q4) of respondents are aware of the number of code review comments and the usage of *Checks* during PRs. Similarly, ∼68% of survey respondents indicate to be familiar with GitHub Check reports (CR-Q5). Most respondents are aware of the number of reviewers involved (∼86%; CR-Q3) and the type(s) of comments they give (∼82%; CR-Q2). We observe a lack of awareness regarding the benefits of code reviewing (∼57%; CR-Q6).

C. *RQ3) How aware are developers automation workflows in their project, and what are the purposes for which they employ these workflows?*

AW-Q1) Do you use GitHub Actions and/or GitHub Apps in the project?: GitHub Marketplace offers two distinct types of tools for automating development workflows: GitHub Apps (Apps) and GitHub Actions (GHA). Both offer similar functionality, including the ability to automate tasks, customize and extend GitHub's features, and integrate with third-party services. However, there are also differences between these two tools. GHA is built into GitHub's platform, whereas Apps are integrated through the GitHub API. As a result, GHA is primarily focused on automating workflows within a repository, while Apps are more versatile and can be used to interact with multiple repositories and third-party services.

We included this survey question to determine participants' awareness and usage of these two tools. Out of 470 responses, 56.6% reported using only GHA, while 17.2% used both GHA and Apps, and 15.7% used only Apps. Also, 15% reported not using either tool, and 8.5% were unaware of the answer.

AW-Q2) Which type of GitHub Actions do you use in the project? (Please name them): From the 347 respondents that indicate to use GHAs in AW-Q1, 276 provided input for this question. After excluding 24 invalid inputs, we qualitatively analyzed the responses (see Section III-D), resulting in the categories presented in Table VI. CI/CD, environment setup, and code quality analysis are among the top Actions types used by the respondents of our survey.

These findings offer insights into the awareness of our survey participants about types of GHAs used in their projects.

AW-Q3) Which GitHub Apps do you use in the project? (Please name them): Similar to AW-Q2 we surveyed the 90 respondents who reported to use Apps in AW-Q1, and 67 of them (74.4%) provided input to this question. The respondents mentioned a wide variety of Apps, with *Codecov* (19) , *Renovate* (9), *CodeQL* (5), *Slack* (4), *Travis CI* (4), *Dependabot* (3), *DCO*[8] (3), *Azure Pipelines* (3), *AppVeyor* (3), *Vercel* (3), and *SonarCloud* (3) being the most commonly mentioned ones.

AW-Q4) Do you use bots in the project?: Before gauging the awareness surrounding bots, we made it clear to respondents that we were referring to bots such as *Dependabot*. We received 46.8% positive answers, while 40.6% responded to not use a bot in their projects; 12.6% selected "I don't know".

AW-Q5) Which bots do you use in the project?: From the 219 respondents who indicated to use bots in their projects, 188 participants provided at least one bot. In total, we identified 54 distinct bots. Some notable bots are: Dependabot, which was mentioned 111 times, something that could potentially be attributed to the fact that we cited Dependabot as an example in the question, *Stale* (14 mentions), *Renovate*

[8]This App enforces the Developer Certificate of Origin (DCO) on pull requests. From https://github.com/apps/dco, visited on July 2023.

TABLE VI
CATEGORIES OF GHA TYPES ACCORDING TO RESPONSES (AW-Q2)

Categories	# of mentions
Continuous Integration and Deployment	
Testing	45
CI	37
Build	36
Build/Test	31
Release/Publish	26
Deployment	18
Versioning	3
Dependency Management/Review	3
Compile	2
Close Milestones	1
Code Quality and Analysis	
Code Quality/Code Style	32
CodeQL	22
Codecove/Code Coverage	15
Static Code Analysis	5
Security	2
Documentation and Management	
Upload/Generate Artifacts	20
Issue/PR Management	12
Wiki/Documentation	5
Notifications	3
Update Changelog	2
UI Translation	1
License Analysis	1
Other	
Environment Setup	77
Link to GHA Configurations	12
Custom	7
Check File Structure	1

(9 mentions), *Bors*[9] (9 mentions), and *Codecov* (8 mentions). Moreover, 8 respondents specified to use custom bots.

AW-Q6) Do you use ASATs in the project?: After providing an explanation of automated static analysis tools (ASATs) to the participants of our survey, we inquired whether they utilize them in their projects. Out of the 470 respondents who answered this question, the results were as follows: 52.8% responded with "Yes", 34.9% replied with "No", and 12.3% selected the "I don't know" option.

RQ3 summary. Figure 1 shows that ∼76% of the survey respondents indicate that they use GHA or GitHub Apps in their project (AW-Q1). However, there is lack of awareness about the specific type of Apps they use (∼14%; AW-Q3) compared to Actions (∼52%; AW-Q2). Additionally, we observe relatively low awareness of ASATs among respondents, with ∼53% indicating to know about ASAT usage in their projects (AW-Q6). Also, less than half of survey participants are aware of bot usage (∼46%; AW-Q4) and which ones they use (∼40%; AW-Q5) in their projects.

D. *RQ4) How aware are developers of continuous integration in their projects?*

CI-Q1) Does the project have CI workflows configured on GitHub?: In response to AW-Q2, the respondents mentioned Testing, CI, and Build as the most commonly used categories for GHA types in their projects. This finding is consistent with the purpose of GitHub's automated workflow tools, which aim to integrate continuous integration (CI) into the software development process. These tools automate tasks such as building and testing code changes[10]. In this question, we asked participants whether they had configured CI workflows for their projects using GitHub's automated workflows.

Of the 469 responses received, 78.9% stated they had CI workflows, while 9.2% reported configuring CI workflows on platforms other than GitHub, 6.8% said they did not know about it, and 5.1% indicated they did not have CI workflows for their projects.

CI-Q2) How do you (or other people in the project) configure CI workflows on GitHub?: We posed an open-ended question to the 370 participants who reported using CI workflows on GitHub, asking them to share how they or other people in their project configure these workflows.

We received 278 responses, with 93 respondents reported using GHA (GitHub Actions) for configuration, while 108 mentioned editing YAML configuration files. It is important to note that some responses were either invalid or vague (42 in total), and 17 participants indicated that they were unsure about the answer. Additionally, 20 respondents mentioned services or tools outside of GitHub.

We received further insights from some of the respondents regarding their approach to configuring CI workflows: 1) 14 respondents said they manually edit the (.yml) configuration files, 2) 6 respondents mentioned "copying" configurations from other projects or Stack Overflow, 3) 3 respondents mentioned using GHA examples and shared workflows (of GitHub) when configuring their CI.

CI-Q3) When and how do you check the project's CI results on GitHub?: We posed an open-ended question to participants, asking them about *when* and *how* they check CI results on GitHub. We received a total of 295 responses, out of which 23 were deemed invalid or vague, and 2 indicated not knowing the answer to this question. After qualitatively analyzing the remaining 270 responses (described in Section III-D), categories presented in Table VII were emerged.

We observe that Pull Requests (PRs) play a crucial role in the evaluation of CI workflows. They emphasized the importance of reviewing CI results for each commit and push, particularly in cases of failures. Moreover, respondents mentioned receiving notifications on GitHub or via emails to stay informed about these failures.

CI-Q4) Describe a situation in which CI workflows helped you in your contribution to the project.: In total, we received 269 responses (excluding 30 invalid and vague ones) and qualitatively analyzed them (as described in Section III-D).

We summarized the responses in Table VIII. The majority emphasized the benefits of CI in terms of testing and bug prevention. Additionally, participants highlighted the importance of building the project in different environments, to mitigate

[9]https://github.com/apps/bors, visited on July 2023.

[10]From https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration, visited on July 2023.

TABLE VII
WHEN/HOW DEVELOPERS CHECK CI RESULTS ON GITHUB (CI-Q3)

	Categories	# of mentions
When	Per/during/after/before each **PR/merge/change**	154
	Per each **commit**	42
	In case of **failure/error**	28
	Getting **email/notification**	26
	Per each **push**	25
	Per/during/before **(code) reviewing**	19
	On a **regular basis**, daily or monthly	8
	Others: per/during deploy, after manual test, part of branch workflow, after workflows run	5
How	On **PR page** and **PR Checks tab**	73
	By checking **emails/notification**	17
	On **Actions tab**	11
	On **commit Checks**	9
	On **(badges in) README file**	7

the "works-on-my-machine" mentality. They also mentioned the impact of CI on code quality, which was mentioned almost as frequently as building in different environments.

RQ4 summary. Based on the summary of answers presented in Figure 1, the majority of respondents indicate to have knowledge about CI workflows configured on GitHub (∼79%; CI-Q1). Also, ∼57% of them show awareness of how CI helps in contributing to a project (CI-Q4). Similarly, ∼57% know how and when to check GitHub's CI workflows results (CI-Q3), however, there is less awareness on how to configure them (46.5%; CI-Q2).

E. *RQ5) How aware are developers of guidelines in their project?*

G-Q1) Does the project have a checklist/guideline to check before contributors submit or maintainers review a pull request?: Of the total number of respondents, 234 (50.3%) answered *yes* to this question, while 167 (35.9%) answered *no*. Another 64 (13.8%) answered that they do not know, and 6 did not answer.

Out of 234 respondents who answered positively, 173 (74%) provided input on the location of the contribution checklist/guideline. The CONTRIBUTING.md file was the most popular location (81 respondents), or the similar DEVELOPER.md file (3 respondents). Another 42 respondents pointed to an issue or pull request template, while 29 respondents pointed to a separate document, and 5 persons indicated the README.md file. Lastly, two respondents said the information is provided privately to contributors, and five did not provide valid responses.

TABLE VIII
CATEGORIES OF HOW CI HELPS DEVELOPERS (CI-Q4)

	Categories	# of mentions
How & in what areas?	Testing	106
	Bug/failure/mistake/breakage prevention	98
	Build with different versions/configurations/environments & avoid works-on-my-machine mentality	47
	Static code analysis/code style/formatting/code quality	46
	Build/compile	29
	PR/contribution verification	18
	Automation of manual tasks: reduce workload, local environment limitation, commits' sign-off message, and maintaining repo files	12
	Prevention of regressions	11
	Deployment/release/publishing	11
	(Code) Reviews	11
	Improve contribution confidence	6

G-Q2) Does a newcomer get any guidelines about how to contribute to the project? If yes, can you provide a link to it?: We manually reviewed a total of 366 responses to this question. Out of the respondents, 270 (74.8%) confirmed the presence of a contribution guideline in their project. While 216 participants provided a link to the guideline, 87 respondents indicated that their project did not possess such a guideline, and 8 participants were uncertain about its existence.

Interestingly, respondents shared various additional practices they employ to support newcomers in their projects. Six participants mentioned leveraging communication platforms like Discord or Matrix to keep in touch with and guide newcomers. They also highlighted the significance of directing newcomers to *good first issues* [62], [63] and encouraging them to initiate discussions regarding their motivations for contributing.

G-Q3) Does the project have specific guidelines (e.g., CONTRIBUTING.md) to make the latest version of the project buildable/compilable?: Similar to the previous question, we inquired about the existence of guidelines regarding making the latest version of participants' projects buildable or compilable. We manually reviewed a total of 366 text-entry responses to gather insights on this topic.

Around 70% of them indicated that guidelines are in place to ensure buildability or compilability, pointing to the CONTRIBUTING.md, or the README.md file for these guidelines. Another 90 respondents (24.5%) indicate to not have these guidelines for their projects, while 8 respondents were unsure whether these guidelines exist for their project. Moreover, 5 respondents claim that building is not applicable to their project, and 6 answers were considered invalid.

Additionally, (4) respondents pointed out the importance of their CI build and test workflows in ensuring that the latest version of their software builds successfully.

RQ5 summary. Figure 1 shows a summary of survey respondents awareness about three types of guidelines in projects: for newcomer contributions to project (G-Q2; ∼58%), keeping the latest version of software buildable/compilable (G-Q3; ∼55%), and as a checklist when reviewing or submitting pull requests (G-Q1; ∼50%. In all, we observe an average level of awareness about project guidelines in OSS.

F. Maintainers & Contributors Level of Awareness

As we compare the awareness of maintainers and contributors, we turn our attention to Figure 1.

In general, we observe that maintainers are more aware of QA activities in their projects compared to contributors.

This is in line with earlier findings from Gousios *et al.* who established that maintainers are more involved in ensuring a project's quality [39]. This is especially true for questions related to *testing*, *the automation workflows in use*, and *CI*.

However, we see a mixed picture for *code reviewing*, where we see indications of higher awareness among contributors compared to maintainers, in particular when it comes to the number of reviewers active in a pull request (CR-Q3) and the benefits of code review (CR-Q6).

Also, contributors are (slightly) more aware of newcomer guidelines and pull requests' checklist/guidelines than maintainers (∼60% vs. ∼56%; G-Q2, and ∼57% vs. ∼47%; G-Q1).

VI. Limitations

We conducted a survey to investigate the level of awareness among OSS developers regarding the quality assurance (QA) practices in their GitHub projects. To ensure unbiased responses, we carefully formulated the survey questions, avoiding leading or ambiguous language. However, it is important to acknowledge that our study has certain limitations that could potentially impact the validity of our findings.

Generalizability. Our findings may not apply to other populations of OSS developers, or OSS projects. While we tried to avoid constraints in our sampling, e.g., limiting the programming language, we acknowledge the need for replication.

Researcher bias may introduce categorization bias when using qualitative research methods to categorize answers of open-ended questions in our survey. Additionally, this bias could potentially influence the wording of the questions. To address this limitation, we took measures to mitigate it by conducting a pilot round of the survey (see Section III-A).

Survey responses validity. The order of questions, the presence of open-ended questions, and respondents' inclination to present themselves in a positive light (e.g., by claiming awareness of all QA activities in their project) may have influenced the accuracy of the provided answers. Similarly, some answers might be estimations, e.g., T-Q4 about test coverage, instead of very precise numbers.

Measuring awareness. While our approach described in Section V, serves as an approximation to measure respondents' awareness, it has limitations. Specifically, using "No Answers","No/Zero", and "Invalid Answers" as proxies of lack of awareness may not always be accurate, e.g., respondents may leave a question unanswered despite having the knowledge.

VII. Conclusion, Implications, and Future Work

We set out to better understand the situational awareness of software engineers when it comes to quality assurance practices in their open source software projects on GitHub.

We found that while there is awareness surrounding test scripts (∼87%; T-Q1), types of tests (∼85%; T-Q2), and testing tooling (∼74%; T-Q4), awareness is lacking when it comes to test coverage (∼53%; T-Q3), either because coverage reports are unavailable or contributors are not familiar with them [**RQ1**].

In the context of code review, our findings indicate that respondents demonstrated a higher level of awareness compared to other practices (∼69%; CR-Q1, ∼68%; CR-Q4 & CR-Q5, ∼82%; CR-Q2, and ∼86%; CR-Q3). However, there is lack of awareness surrounding the benefits of code review (∼57%; CR-Q6) [**RQ2**].

We have observed that while 76% of the respondents indicate that a workflow automation is in use in their project, they generally lack awareness on the precise type of GitHub Action (∼52%), or App (∼14%). When it comes to bots, ∼46% indicates to know that a bot is in use, with less respondents indicating to know which bots are in use (∼40%) [**RQ3**].

We see that ∼79% of the respondents are aware that there is a CI configured for their project, yet only around 57% knows how and when to check the CI results [**RQ4**].

When we turn our attention to the awareness of guidelines in GitHub projects, we observe that only ∼50% to ∼58% of the respondents are aware of such a set of guidelines existing. Either creating such guidelines, or increasing the awareness surrounding them, can likely improve the onboarding and developer experience [**RQ5**].

When we compare participants' roles, we observe that in general maintainers are more aware of QA practices in their projects. However, when it comes to code reviewing or guidelines, like the number of reviewers in a pull request (CR-Q3), the benefits of code review (CR-Q6), newcomer guidelines (G-Q2), and PRs' checklist/guidelines (G-Q1) contributors show a higher level of awareness.

Implications. The identification of awareness gaps through this analysis can pave the way for **educators** to emphasize the importance of situational awareness among the next generation of software engineers. **Tool makers** can improve their tooling to stimulate awareness among software engineers, similar to what TestAxis does for testing [64]. **Researchers** should study more deeply how and why software engineers either gain or lack knowledge of specific QA practices.

Future work. A publicly available data set [65] with 456 anonymized survey answers of our study makes it possible to replicate our study (15 respondents did not give consent to archive their answers). Also, the dataset can support other potential future research directions like: (1) using respondents' repository handles in our data set to see how reported answer compare to the reality of QA in OSS projects, (2) expand the work on private repositories or industrial projects, (3) using GitHub projects with different characteristics to see if and how QA awareness changes among those, (4) building tools to help increase OSS developers knowledge of QA in their projects, particularly in areas that we observed lack of awareness.

Acknowledgment

This research was partially funded by the Dutch science foundation NWO through the Vici "TestShift" grant (No. VI.C.182.032).

REFERENCES

[1] M. Jazayeri, "The education of a software engineer," in *Proc. International Conference on Automated Software Engineering (ASE)*. USA: IEEE, 2004.

[2] A. J. Ko, B. Dosono, and N. Duriseti, "Thirty years of software problems in the news," in *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. ACM, 2014, pp. 32–39.

[3] M. Aniche, C. Treude, and A. Zaidman, "How developers engineer test cases: An observational study," *IEEE Trans. Software Eng.*, vol. 48, no. 12, pp. 4925–4946, 2022.

[4] M. Aniche, *Effective Software Testing: A Developer's Guide*. Manning Publications, 2022.

[5] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Trans. Software Eng.*, vol. 39, no. 6, pp. 757–773, 2013.

[6] G. Balogh, T. Gergely, Á. Beszédes, and T. Gyimóthy, "Are my unit tests in the right package?" in *2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2016, pp. 137–146.

[7] M. Beller, G. Gousios, A. Panichella, S. Proksch, S. Amann, and A. Zaidman, "Developer testing in the IDE: patterns, beliefs, and behavior," *IEEE Trans. Software Eng.*, vol. 45, no. 3, pp. 261–284, 2019.

[8] M. Beller, G. Gousios, A. Panichella, and A. Zaidman, "When, how, and why developers (do not) test in their IDEs," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 2015, pp. 179–190.

[9] C. Marsavina, D. Romano, and A. Zaidman, "Studying fine-grained co-evolution patterns of production and test code," in *14th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2014, pp. 195–204.

[10] A. Zaidman, B. Van Rompaey, A. van Deursen, and S. Demeyer, "Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining," *Empir. Softw. Eng.*, vol. 16, no. 3, pp. 325–364, 2011.

[11] A. Zaidman, B. Van Rompaey, S. Demeyer, and A. van Deursen, "Mining software repositories to study co-evolution of production & test code," in *First International Conference on Software Testing, Verification, and Validation (ICST)*. IEEE Computer Society, 2008, pp. 220–229.

[12] Z. Lubsen, A. Zaidman, and M. Pinzger, "Using association rules to study the co-evolution of production & test code," in *Proceedings of the 6th International Working Conference on Mining Software Repositories (MSR)*. IEEE, 2009, pp. 151–154.

[13] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *35th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2013, pp. 712–721.

[14] M. di Biase, M. Bruntink, and A. Bacchelli, "A security perspective on code review: The case of chromium," in *2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2016, pp. 21–30.

[15] M. Beller, A. Bacchelli, A. Zaidman, and E. Jürgens, "Modern code reviews in open-source projects: which problems do they fix?" in *11th Working Conference on Mining Software Repositories (MSR)*. ACM, 2014, pp. 202–211.

[16] M. Beller, R. Bholanath, S. McIntosh, and A. Zaidman, "Analyzing the state of static analysis: A large-scale evaluation in open source software," in *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 2016, pp. 470–481.

[17] C. Vassallo, S. Panichella, F. Palomba, S. Proksch, H. C. Gall, and A. Zaidman, "How developers engage with static analysis tools in different contexts," *Empir. Softw. Eng.*, vol. 25, no. 2, pp. 1419–1457, 2020.

[18] D. Han, C. Ragkhitwetsagul, J. Krinke, M. Paixao, and G. Rosa, "Does code review really remove coding convention violations?" in *2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2020, pp. 43–53.

[19] T. Buckers, C. Cao, M. Doesburg, B. Gong, S. Wang, M. Beller, and A. Zaidman, "UAV: warnings from multiple automated static analysis tools at a glance," in *24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2017, pp. 472–476.

[20] F. Zampetti, S. Mudbhari, V. Arnaoudova, M. D. Penta, S. Panichella, and G. Antoniol, "Using code reviews to automatically configure static analysis tools," *Empir. Softw. Eng.*, vol. 27, no. 1, p. 28, 2022.

[21] C. Vassallo, S. Panichella, F. Palomba, S. Proksch, A. Zaidman, and H. C. Gall, "Context is king: The developer perspective on the usage of static analysis tools," in *25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2018, pp. 38–49.

[22] M. Beller, G. Gousios, and A. Zaidman, "Oops, my tests broke the build: an explorative analysis of Travis CI with GitHub," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 356–367.

[23] A. Rahman, A. Partho, D. Meder, and L. Williams, "Which factors influence practitioners' usage of build automation tools?" in *International Workshop on Rapid Continuous Software Engineering (RCoSE)*, 2017, pp. 20–26.

[24] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. 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 (ASE)*. ACM, 2016, pp. 426–437.

[25] O. Elazhary, C. M. Werner, Z. S. Li, D. Lowlind, N. A. Ernst, and M. D. Storey, "Uncovering the benefits and challenges of continuous integration practices," *IEEE Trans. Software Eng.*, vol. 48, no. 7, pp. 2570–2583, 2022.

[26] C. Vassallo, G. Schermann, F. Zampetti, D. Romano, P. Leitner, A. Zaidman, M. D. Penta, and S. Panichella, "A tale of CI build failures: An open source and a financial organization perspective," in *2017 IEEE International Conference on Software Maintenance and Evolution, (ICSME)*. IEEE, 2017, pp. 183–193.

[27] C. Vassallo, S. Proksch, T. Zemp, and H. C. Gall, "Un-break my build: assisting developers with build repair hints," in *Proceedings of the 26th Conference on Program Comprehension (ICPC)*. ACM, 2018, pp. 41–51.

[28] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 32–64, 1995.

[29] A. Khatami and A. Zaidman, "State-of-the-practice in quality assurance in java-based open source software development," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2306.09665

[30] H. Bomström, M. Kelanti, E. Annanperä, K. Liukkunen, T. Kilamo, O. Sievi-Korte, and K. Systä, "Information needs and presentation in agile software development," *Information and Software Technology*, p. 107265, 2023.

[31] R. P. L. Buse and T. Zimmermann, "Information needs for software development analytics," in *34th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2012, pp. 987–996.

[32] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *29th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2007, pp. 344–353.

[33] L. Pascarella, D. Spadini, F. Palomba, M. Bruntink, and A. Bacchelli, "Information needs in contemporary code review," *Proc. ACM Hum. Comput. Interact.*, vol. 2, no. CSCW, pp. 135:1–135:27, 2018.

[34] A. Ahmad, O. Leifler, and K. Sandahl, "Software professionals' information needs in continuous integration and delivery," in *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing*. ACM, 2021, pp. 1513–1520.

[35] G. Gousios and A. Zaidman, "A dataset for pull-based development research," in *11th Working Conference on Mining Software Repositories (MSR)*. ACM, 2014, pp. 368–371.

[36] D. Kavaler, A. Trockman, B. Vasilescu, and V. Filkov, "Tool choice matters: Javascript quality assurance tools and usage outcomes in github projects," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 476–487.

[37] T. Kinsman, M. S. Wessel, M. A. Gerosa, and C. Treude, "How do software developers use github actions to automate their workflows?" in *18th IEEE/ACM International Conference on Mining Software Repositories (MSR)*. IEEE, 2021, pp. 420–431.

[38] M. Wessel, A. Zaidman, M. A. Gerosa, and I. Steinmacher, "Guidelines for developing bots for github," *IEEE Softw.*, vol. 40, no. 3, pp. 72–79, 2023.

[39] G. Gousios, A. Zaidman, M. D. Storey, and A. van Deursen, "Work practices and challenges in pull-based development: The integrator's perspective," in *37th IEEE/ACM International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2015, pp. 358–368.

[40] G. Gousios, M. D. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: the contributor's perspective," in *Proceedings of the 38th International Conference on Software Engineering (ICSE)*. ACM, 2016, pp. 285–296.

[41] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Review participation in modern code review: An empirical study of the android, qt, and openstack projects (journal-first abstract)," in *25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE Computer Society, 2018, p. 475.

[42] M. Wessel, T. Mens, A. Decan, and P. Rostami Mazrae, "The github development workflow automation ecosystems," in *Software Ecosystems: Tooling and Analytics*, T. Mens, C. De Roover, and A. Cleve, Eds. Springer Nature, 2023.

[43] O. Elazhary, "Investigating the interplay between developers and automation," in *43rd IEEE/ACM International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 2021, pp. 153–155.

[44] D. G. Widder, M. Hilton, C. Kästner, and B. Vasilescu, "A conceptual replication of continuous integration pain points in the context of travis CI," in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 2019, pp. 647–658.

[45] F. Hassan, S. Mostafa, E. S. L. Lam, and X. Wang, "Automatic building of java projects in software repositories: A study on feasibility and challenges," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE Computer Society, 2017, pp. 38–47.

[46] O. Elazhary, M. D. Storey, N. A. Ernst, and A. Zaidman, "Do as I do, not as I say: Do contribution guidelines match the github contribution process?" in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2019, pp. 286–290.

[47] O. Baysal, R. Holmes, and M. W. Godfrey, "Situational awareness: personalizing issue tracking systems," in *35th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2013, pp. 1185–1188.

[48] A. Ahmad, O. Leifler, and K. Sandahl, "Data visualisation in continuous integration and delivery: Information needs, challenges, and recommendations," *IET Softw.*, vol. 16, no. 3, pp. 331–349, 2022.

[49] P. S. Kochhar, T. F. Bissyandé, D. Lo, and L. Jiang, "An empirical study of adoption of software testing in open source projects," in *2013 13th International Conference on Quality Software (QSIC)*. IEEE, 2013, pp. 103–112.

[50] C. Vassallo, F. Palomba, A. Bacchelli, and H. C. Gall, "Continuous code quality: are we (really) doing that?" in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE)*. ACM, 2018, pp. 790–795.

[51] N. Cassee, B. Vasilescu, and A. Serebrenik, "The silent helper: The impact of continuous integration on code reviews," in *27th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 423–434.

[52] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "The impact of code review coverage and code review participation on software quality: a case study of the qt, VTK, and ITK projects," in *11th Working Conference on Mining Software Repositories (MSR)*. ACM, 2014, pp. 192–201.

[53] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, and M. D. Penta, "How open source projects use static code analysis tools in continuous integration pipelines," in *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*. IEEE Computer Society, 2017, pp. 334–344.

[54] M. S. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, "What to expect from code review bots on github?: A survey with OSS maintainers," in *34th Brazilian Symposium on Software Engineering (SBES)*. ACM, 2020, pp. 457–462.

[55] T. Chen, Y. Zhang, S. Chen, T. Wang, and Y. Wu, "Let's supercharge the workflows: An empirical study of github actions," in *21st IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2021, pp. 1–10.

[56] A. Decan, T. Mens, P. R. Mazrae, and M. Golzadeh, "On the use of github actions in software development repositories," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2022, pp. 235–245.

[57] M. Golzadeh, A. Decan, and T. Mens, "On the rise and fall of CI services in github," in *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 662–672.

[58] O. Dabic, E. Aghajani, and G. Bavota, "Sampling projects in github for MSR studies," in *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021*. IEEE, 2021, pp. 560–564.

[59] T. Zimmermann, "Card-sorting: From text to themes," in *Perspectives on Data Science for Software Engineering*, T. Menzies, L. Williams, and T. Zimmermann, Eds. Morgan Kaufmann, 2016, pp. 137–141.

[60] J. M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative sociology*, vol. 13, no. 1, pp. 3–21, 1990.

[61] I. Heitlager, T. Kuipers, and J. Visser, "A practical model for measuring maintainability," in *Quality of Information and Communications Technology, 6th International Conference on the Quality of Information and Communications Technology (QUATIC)*. IEEE Computer Society, 2007, pp. 30–39.

[62] W. Xiao, H. He, W. Xu, X. Tan, J. Dong, and M. Zhou, "Recommending good first issues in github oss projects," in *Proceedings of the 44th International Conference on Software Engineering (ICSE)*. ACM, 2022, pp. 1830–1842.

[63] J. W. D. Alderliesten and A. Zaidman, "An initial exploration of the "good first issue" label for newcomer developers," in *14th IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2021, pp. 117–118.

[64] C. Boone, C. E. Brandt, and A. Zaidman, "Fixing continuous integration tests from within the IDE with contextual information," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension (ICPC)*. ACM, 2022, pp. 287–297.

[65] A. Khatami and A. Zaidman, "Quality Assurance Awareness in Open Source Software Projects on GitHub Analysis Dataset," Jul. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8139381