Empirical Software Engineering (2022) 27:126
https://doi.org/10.1007/s10664-022-10143-4

Pull request latency explained: an empirical overview

Xunhui Zhang[1] · Yue Yu[1] · Tao Wang[1] · Ayushi Rastogi[2] · Huaimin Wang[1]

Accepted: 9 March 2022

Abstract
Pull request latency evaluation is an essential application of effort evaluation in the pull-based development scenario. It can help the reviewers sort the pull request queue, remind developers about the review processing time, speed up the review process and accelerate software development. There is a lack of work that systematically organizes the factors that affect pull request latency. Also, there is no related work discussing the differences and variations in characteristics in different scenarios and contexts. In this paper, we collected relevant factors through a literature review approach. Then we assessed their relative importance in five scenarios and six different contexts using the mixed-effects linear regression model. The most important factors differ in different scenarios. The length of the description is most important when pull requests are submitted. The existence of comments is most important when closing pull requests, using CI tools, and when the contributor and the integrator are different. When there exist comments, the latency of the first comment is the most important. Meanwhile, the influence of factors may change in different contexts. For example, the number of commits in a pull request has a more significant impact on pull request latency when closing than submitting due to changes in contributions brought about by the review process. Both human and bot comments are positively correlated with pull request latency. In contrast, the bot's first comments are more strongly correlated with latency, but the number of comments is less correlated. Future research and tool implementation needs to consider the impact of different contexts. Researchers can conduct related studies based on our publicly available datasets and replication scripts.

Keywords Pull-based development · Pull request latency · Distributed software development · GitHub

1 Introduction

As an important paradigm of distributed software development, pull-based development is widely used in social coding communities including GitHub and GitLab. Because of the

Communicated by: Igor Steinmacher

✉ Yue Yu
yuyue@nudt.edu.cn

Extended author information available on the last page of the article.

Published online: 04 July 2022

temporal and spatial asynchrony of the project participants in this model, the latency of pull requests is an important issue. For reviewers, understanding the latency of pull requests leads to a predictable development process and helps managers make plans and decisions. [1] For contributors, the latency prediction can remind developers about the remaining time and accelerate the completion of pull requests (Maddila et al. 2019; Maddila et al. 2020), which can minimize the abandonment behavior from contributors (Li et al. 2021). For the pull-based paradigm, latency covers the entire pull request lifecycle and is an important research area to grasp the pull-based development model as a whole.

Considering the entire lifetime (see Fig. 1), the pull-based development model mainly consists of the following stages.

1. Contributors (also known as requesters or submitters) first submit the code changes of their cloned repository to reviewers for inspection in the form of pull requests.
2. With/Without automatic continuous integration (CI) builds, reviewers manually inspect the code changes and discuss them in comments.
3. The integrator (also called the closer or the merger) evaluates the code changes based on the review process information and then decides to merge or reject the pull request.
4. After rejecting the pull request, the contributor or project manager may also reopen the pull request and make further changes for the final merge.

A pull request has multiple states throughout the process, including submit, review, close, merge, and reopen. Due to CI tools, the review state consists of two sub-states, namely CI construction and manual review state.

It can be seen from the above states that the lifetime of a pull request is complex, and there is a lot of research on each stage of the entire lifetime. Research in the submitting state focuses on the automatic generation of pull request descriptions (Liu et al. 2019), reviewer recommendations (Jiang et al. 2017; Yu et al. 2014), duplication assessment (Wang et al. 2019; Yu et al. 2018), etc. For the review state, research focuses on the interplay between continuous integration builds and pull requests (Zampetti et al. 2019) and the influence of static analysis tools on code review effort (Singh et al. 2017). For the manual review state, research focuses on the prioritization method of review (v. d. Veen et al. 2015) and commenter recommendation (Jing et al. 2017). For the reject or merge stage, research focuses on the prediction of pull request decisions (Gousios et al. 2014). Reopen state focuses on the evaluation and analysis of the impact of reopening (Jiang et al. 2019). For the entire lifetime, there are studies of factors influencing pull request decisions (Tsay et al. 2014; Zhang et al. 2021) and latency (Yu et al. 2015), etc.

Therefore, we can see that exploring the factors across the whole lifetime of a pull request is helpful to understand the entire process and various states of the pull request. Also, the open time and close time are two important time points in the pull request lifecycle. The analysis of the association between factors and latency of the two snapshots can help us understand the impact of factors' changes in the review process, which can provide actionable suggestions to those involved in the pull request. We have previously studied the factors that influence the final merge decision of pull request throughout its lifetime (Zhang et al. 2021). As a critical research area in the entire lifetime of pull requests, the study of pull request latency also covers all the states of pull requests. Meanwhile, as part of the effort evaluation (Maddila et al. 2019), pull request latency analysis and prediction can also help reviewers save review time, improve review efficiency, and optimize pull request review

[1] https://en.wikipedia.org/wiki/Software_development_effort_estimation

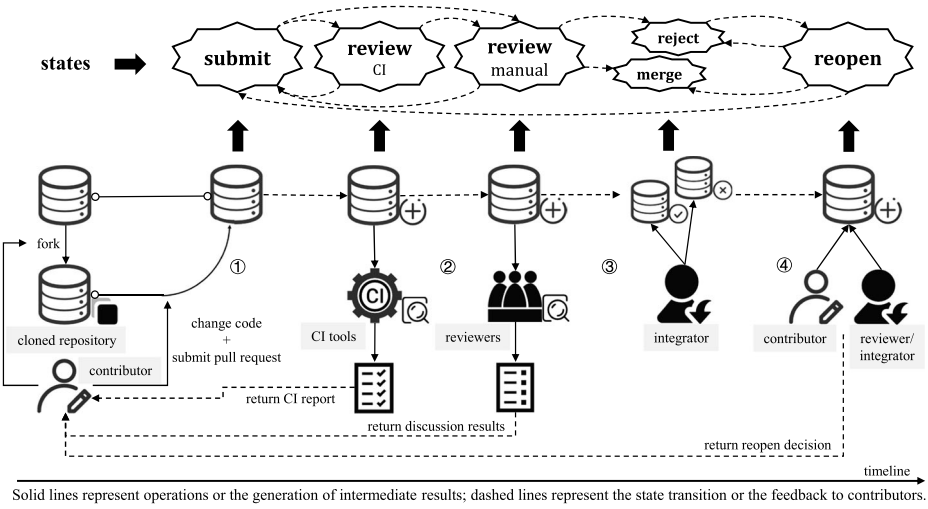

Fig. 1 Workflow of a pull request

priority. Therefore we would like to build on our previous work (Zhang et al. 2021) and explore the impact of various factors on pull request latency.

Although related work has explored the factors that affect the latency of pull requests, there is no systematic analysis of the influencing factors and the exploration of factors' influence in different situations and contexts. Therefore, building on a large-scale and diverse dataset, this paper conducts an empirical study on the impact of factors in different situations and contexts on the latency of pull requests. Notably, we explore the following two research questions:

RQ1 *How do factors influence pull request latency?*
RQ2 *Do the factors influencing pull request latency change with a change in context?*

To answer the above research questions, we first collect a comprehensive list of factors influencing pull request latency by conducting a systematic literature review. Then, we create a large-scale and diverse dataset of these factors. Finally, we build models (mixed-effect linear regression model) for different scenarios (e.g., pull request using CI) and contexts (e.g., pull request closing time).

This paper makes the following contributions:

1. We collect a dataset of 11,230 projects on GitHub with 63 factors and 3,347,937 pull requests relating to the latency of pull requests. We open-source the relevant dataset (Zhang et al. 2021) and model-building scripts.[2]
2. We present a synthesis of the factors identified in the literature, indicating their significance and direction in inferring pull request latency.
3. We show the importance of these factors in explaining the latency of pull requests in different scenarios and how they change within different contexts.

[2]https://github.com/zhangxunhui/ESE_pull_request_latency

The rest of the paper is organized as follows. We present related works in Section 2 and explain our research design in Section 3. We show the results in Section 4 and discuss bot-related factors in Section 5. We present the discussion and threats in Sections 6 and 7. We conclude the paper in Section 8.

2 Related Work

This paper interprets software effort estimation from the perspective of modern code review. Therefore, the related work in this chapter mainly includes traditional software effort estimation, modern code review, and related research work for pull request latency.

2.1 Software Effort Estimation

Effort estimation has been a critical activity in software engineering that helps software managers and developers to plan software project development time and monitor the development process (Kocaguneli et al. 2011; Trendowicz and Jeffery 2014). Many studies are focusing on related research in this area. Sehra et al. (2017) analyzed 1178 articles related to software effort estimation in terms of topic classification and research trends. The article modeled the research themes by the LDA method and classified 15 topics, including "factors affecting estimation," "estimation methods," etc. There are also many approaches to software effort estimation, including the neural network-based approach (Dasheng and Shenglan 2012), expert evaluation-based approach (Jørgensen 2011), hybrid approach (Minku and Yao 2011), use-case-based approach (Wang et al. 2009), etc. The influencing factors associated with effort assessment are complex and diverse, including language and cultural differences, team structure, work pressure, team size, and team familiarity (Lenarduzzi 2015; Altaleb et al. 2020).

Unlike traditional software effort estimation, open source software development differs in many aspects, including the types of participants involved (*e.g.,* contributors, reviewers), support mechanisms (*e.g.,* CI tools, bot), contribution and merging strategies, etc. Therefore the factors involved in effort estimation are different. The importance of these factors may also vary significantly. Therefore, the pull request effort estimation is very different from the traditional software effort estimation.

2.2 Modern Code Review

Code review has always been a critical aspect of software engineering practice and is widely used in open source and industrial software. Unlike traditional code review methods, today's form of code review relies more on automated tools and is more lightweight (Bacchelli and Bird 2013). Among them, CRITICS (Zhang et al. 2014), ReviewClipse (Bernhart et al. 2010), and Mylyn Reviews (Ecplise 2011) are IDE-integrated code review tools. Gerrit (Gerrit 2021) and CodeFlow (Bacchelli and Bird 2013) are platforms supporting many successful open source software and companies.

Acting as a new social coding paradigm, the pull-based development model (Gousios et al. 2014) involves multiple social media, *e.g.,* follow, watch, fork, and is becoming more and more popular recently (Yu et al. 2014). Unlike Gerrit, the pull-based model focuses not only on a single commit but also on a whole branch (Vogel 2020). In contrast, pull request is easy to participate in the contribution process without having to master many git

operations (Atkins 2012). Its well-designed user interface and support for social collaboration help improve the usability and code review process of GitHub (Gerrit 2013). These characteristics help GitHub get more than 79 million users and 238 million repositories.

Therefore, this paper will take the pull-based development model as an entry point to explore the impact of factors on effort estimation in modern code review, i.e., which factors affect the pull request latency.

2.3 Latency Analysis for Pull Requests

The research of pull requests has been a hot topic, including pull request reviewer recommendation (Yu et al. 2014), automatic description generation (Liu et al. 2019), @-mechanism empirical study (Zhang et al. 2014), etc. The work involving the whole pull request lifecycle mainly includes two aspects, i.e., pull request decision and latency (Zhang et al. 2021). Therefore, a large amount of related research focuses on the study of pull request latency.

Gousios et al. (2014) constructed a classification prediction model for pull request latency, where the time required to integrate pull requests is divided into three categories. Through a small number of factors created at pull request submission time, they calculated the relative importance of the factors. However, this work only considers the effect of factors at the time of pull request submission but does not consider the factors that occurred during the review process, e.g., comments, CI results. Yu et al. (2015) and Yu et al. (2016) also explored the impact of various factors on pull request latency. They collected potentially relevant factors through a heuristic approach, including the latency and build results of CI tools. Since many related studies have emerged, there is a lack of a scientific process to integrate all possible influencing factors. And the discussion of these factors is limited to the general context only, lacking a discussion of the influence of factors in different contexts (e.g., whether to include review comments, whether use CI tools, etc.), where differences in context may lead to differences in the influence of factors. Therefore, to improve the comprehensive understanding of pull request latency, this paper integrates the possible factors and considers the influence of factors from different contexts. Maddila et al. (2019) performed in-process prediction of latency and analyzed the time savings brought by this prediction tool to software development. However, the work is limited to Microsoft, and the factors involved are mainly proposed for the characteristics of Microsoft internal teams, lacking a generalization. Our work takes a global perspective and exploits the diversity of the data we collect to explore the impact of the factors. Other studies such as Jiang et al. (2019), Hu et al. (2018), and Zhao et al. (2017) focus on the impact of pull request reopen, bug fixes, and CI usage on pull request latency, respectively. However, they only focus on one factor instead of exploring the differences between factors and their relative importance in explaining pull request latency in a comprehensive perspective.

Overall, this paper integrates all the general quantifiable factors mentioned in related works that may affect pull request latency from a comprehensive perspective and explores the impact of factors on pull request latency from different contexts.

3 Study Design

Figure 2 shows the framework of our study. There are mainly four parts. First, we gather all the papers explaining pull request latency (systematic literature review (SLR)). Second, we collect data for factors extracted in the above step from diverse GitHub projects (data

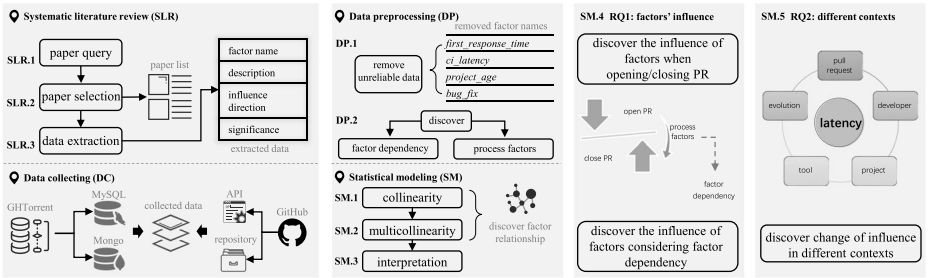

Fig. 2 Framework of this paper

collecting (DC)). Third, we preprocess the data for further analysis (data preprocessing (DP)). Finally, we do exploratory analysis and build models to answer the research questions (statistical modelling (SM)).

3.1 Systematic Literature Review (SLR)

To gather papers related to pull request latency and collect measurable factors mentioned in previous studies, we conduct SLR following the guideline of Kitchenham and Charters (2007). There are mainly three steps, i.e., paper query, paper selection and data extraction.

3.1.1 Paper Query (SLR.1)

Our previous study (Zhang et al. 2021) found that the keyword "pull based" can be related to the "peer to peer" related studies. Therefore, to limit the search scope, we use keyword "pull based development" and "pull based model" instead of "pull based". For the time related keyword, we choose "lifetime" (Maddila et al. 2019) and "latency" (Yu et al. 2016). Finally, we defined the boolean search string as follows:

("pull based model" OR "pull based development" OR "pull request") AND ("latency" OR "lifetime")

We conducted the query on May 15th, 2020. We got 1,273 papers, including 164 from ACM Digital Library, 1 from IEEExplore, 5 from Web of Science, 5 from Ei Compendex and 1,098 from Google Scholar.

3.1.2 Paper Selection (SLR.2)

We manually checked the keywords, title and abstract of each paper and excluded papers for the following reasons:

- not written in English (4 papers)
- have duplications (241 papers)
- have an extension (3 papers)
- not related to pull request latency (1,004 papers)
- explain factors not applicable to GitHub (1 paper), i.e., factors based on mailing list (Jiang et al. 2013).
- not generalizable to a wider range of projects on GitHub (2 papers), i.e., Microsoft and C# related factors (Maddila et al. 2019).

– not measurable quantitatively (1 paper), *i.e., the features relating to the pull request latency found in a qualitative study* (Gousios et al. 2015).

To avoid the missing of related papers, we conducted a one-round backward snowballing process (Jalali and Wohlin 2012). However, we found no new articles related to the pull request latency. All in all, we found 17 papers presenting the factors related to the pull request latency (before May 2020).

3.1.3 Data Extraction (SLR.3)

In total, we extract 45 features from the previous selected 17 studies together with the description of measurement, the influence direction and significance. To clearly show the extraction results, we list the symbolic representations of the factors in Table 1 and the factor's relationship with pull request latency in Table 9. Table 1 shows 12 developer-related (1 newly added - *same_user*), 10 project-related (1 newly added - *has_comments*), and 25 pull request-related factors. • marks 29 factors that make sense at the submission time of pull request (*e.g., open_pr_num*), while the other 18 factors occurred during the review process. ⋆ marks 8 code-related factors present below the dashed line that can change during the review process (*e.g., num_commits*). Table 9 presents the articles related to pull request latency and the factors included in studies, where the meaning of shape, color and filling can be seen in the title of the table. For example, Yu et al. (2016) proposed the factor *core_member*, which is negatively and significantly related to pull request latency in their conclusion.

With the foundation of the prequel work (Zhang et al. 2021), the first author and the fourth author jointly completed the screening and extraction of relevant factors after the article search. During the paper filtering phase, compared to our previous study on pull request decisions (Zhang et al. 2021), some papers are new, i.e., related to pull request latency only. Therefore, we only discuss these newly added papers (e.g., Pinto et al. 2018; Hu et al. 2018) as the papers related to pull request decisions had already reached a consensus between the two authors. In the data extraction stage, the first author extracted the factors of the papers focusing on pull request decisions and latency at the same time. For the newly added papers, the fourth author went through a double-check. For factors that were not sure whether to be included, such as factors related to Microsoft teams and C# program language only (Maddila et al. 2019). After discussion, we reached a consensus by removing factors that are not generalizable.

3.2 Data Collecting (DC)

According to the described measurement in related work, we collect the extracted factors from SLR. The data collection process is based on GHTorrent MySQL[3] and Mongo data dump on June 1st, 2019. Also, we directly extract factors based on GitHub API and the cloned source repository.

Some pull request-related factors change dynamically throughout the pull request lifetime (e.g., the number of commits contained in a pull request changes when modifying the pull request during the review process). We refer to these factors as dynamic factors, which are shown in Table 3. Based on our previous dataset (Zhang et al. 2021), we calculated the values of these dynamic factors at the time of pull request submission and count the changes of these factors at the time of pull request submission and closing. By adding these factors,

[3] http://ghtorrent-downloads.ewi.tudelft.nl/mysql/mysql-2019-06-01.tar.gz

Table 1 Comprehensive list of the factors known to influence pull request latency on GitHub

Factor	Description	Factor	Description
	Developer Characteristics		
first_pr•	first pull request? yes/no	prior_review_num	# of previous reviews in a project
core_member•	core member? yes/no	first_response_time	# of minutes from pull request creation to the reviewer's first response
contrib_gender•	gender? male or female	contrib_ affiliation•	contributor affiliation
same_affiliation	same affiliation contributor/integrator? yes/no	inte_affiliation	integrator affiliation
social_strength•	fraction of team members interacted with in the last three months	prev_pullreqs•	# of previous pull requests
followers•	# of followers at pull request creation time	same_user	same contributor and integrator? yes/no
	Project Characteristics		
sloc•	executable lines of code	team_size•	# of active core team members in the last three months
project_age•	# of months from project to pull request creation	open_pr_num•	# of open pull requests
integrator_ availability•	latest activity of the two most active integrators	test_lines_per_ kloc•	# of test lines per 1K lines of code
test_cases_per_ kloc•	# of test cases per 1K lines of code	asserts_per_kloc•	# of assertions per 1K lines of code
perc_external_ contribs•	% of external pull request contributions	requester_succ_ rate•	past pull request success rate
	Pull Request Characteristics		
bug_fix•	fixes a bug? yes/no	description_ length•	length of pull request description
hash_tag	"#" tag exists? yes/no	num_participants	# of participants in pull request comments

Table 1 (continued)

Factor	Description	Factor	Description
ci_exists●	uses CI? yes/no	part_num_code	# of participants in pull request and commit comments
ci_latency	# of minutes from pull request creation to the first CI build finish time	num_code_ comments	# of code comments
reopen_or_not	pull request is reopened? yes/no	friday_effect●	pull request submitted on a Friday? yes/no
has_comments	pull request has a comment? yes/no	num_comments	# of comments
num_comments_ con	# of contributor comments	at_tag	"@" tag exists? yes/no
num_code_ comments_con	# of contributor's code comments	ci_test_passed	all CI builds passed? yes/no
comment_conflict	keyword "conflict" exists in comments? yes/no		
num_commits_ open⋆●	# of commits at pull request open time	num_commits_ close/change⋆	# of commits at close time/changed compared to open time
src_churn_ open⋆●	# of lines changed (added + deleted) at pull request open time	src_churn_ close/change⋆	# of lines changed (added + deleted) at close time/changed compared to open time
files_changed_ open⋆●	# of files touched at pull request open time	files_changed_ close/change⋆	# of files touched at close time/changed compared to open time
commits_on_ files_touched_ open⋆●	# of commits on files touched at pull request open time	commits_on_ files_touched_ close/change⋆	# of commits on files touched at close time/changed compared to open time
churn_addition_ open⋆●	# of added lines of code at pull request open time	churn_addition_ close/change⋆	# of added lines of code at close time/changed compared to open time

Table 1 (continued)

Factor	Description	Factor	Description
churn_deletion_ open⋆●	# of deleted lines of code at pull request open time	churn_deletion_ close/change⋆	# of deleted lines of code at close time/changed compared to open time
test_churn_ open⋆●	# of lines of test code changed (added + deleted) at pull request open time	test_churn_ close/change⋆	# of lines of test code changed (added + deleted) at close time/changed compared to open time
test_inclusion_ open⋆●	test case existing at pull request open time? yes/no	test_inclusion_ close/change⋆	test case existing at close time/changed compared to open time? yes/no

Factors marked as ⋆ are dynamic factors that can change during the review process. Here we collect their values at two different snapshots (pull request submission and close). Factors marked as ● are those make sense when creating a pull request

All metrics are relative to a referenced pull request in a project

Factors that change over time (e.g., core team) are measured using the previous three months of development activities in a project

we can determine whether the importance of these dynamic factors changes under different snapshots (pull request submission and close) during the whole lifetime of the pull request.

The final dataset offers a total of 3,347,937 pull requests from 11,230 projects. The diversity of the dataset is reflected in 6 programming languages, different projects sizes and different project activities (Zhang et al. 2021).

3.3 Data Preprocessing (DP)

There are mainly two parts for data preprocessing, *i.e.* unreliable data removal (DP.1) and special factor discovery (DP.2).

3.3.1 Unreliable Data Removal (DP.1)

There are some unexpected values for factors, including *first_response_time*, *ci_latency* and *project_age*. We need to fix the problem for future reliable analysis. Detailed reasoning for the unexpected problems can be seen in our technical report (Zhang et al. 2020b).

- *first_response_time* has *negative* values since our metric considers the comments under the related code, and some comments exist before pull request creation (0.4% pull requests have negative values).
- *ci_latency* has *negative* values when commits exist before pull request creation, and the time of the first CI build is earlier than the creation time of a pull request (1.5%).
- *project_age* has *negative* values where the creation time of a user account on GHTorrent is different from that on Github API (0.1%).
- We also remove *bug_fix*, which has 99.3% empty values, to avoid its impact on further analysis. Empty values represent the pull requests or their related issues that do not have tags determining whether the pull request is a bug fix according to Fan et al.'s method (2017).

After removing the above-unexpected values, to further verify the validity of the remaining data, we calculated the accuracy of the 100 selected pull requests by random sampling of the time-related factors, respectively. The accuracy of factors *first_response_time*, *account_creation_days*, *project_age*, *ci_latency* in the subset of randomly selected samples are 98%, 97%, 96%, and 94%, respectively. Relative to the size of our dataset, unexpected values represent only a small fraction. Since it is difficult to avoid the problem caused by the inconsistency between GHTorrent and GitHub APIs, we have added the issue to Threats to Validity (Section 7) with a corresponding description.

3.3.2 Special Factor Discovery (DP.2)

Some factors would not make sense unless a precondition were met (see Table 2). For example, factor *ci_latency* only make sense when *ci_exists* was true. Therefore, *ci_exists* presents a precondition contingent on which *ci_latency* is meaningful, also referred to as a postconditional factor. Other factors which have no dependencies will not be affected by conditions.

- *first_response_time*, it only make sense when there exists comment (*has_comments=1*).
- *ci_test_passed* and *ci_latency* depend on the existence of CI tools (*ci_exists=1*).
- *same_affiliation*, it only makes sense when the contributor and integrator are different persons (*same_user=0*).

Table 2 Factors with dependency

postconditional factor	preconditional factor
first_response_time	has_comments
ci_test_passed	ci_exists
ci_latency	
same_affiliation	same_user

Some pull request related factors are related to the review process and can change from the creation time to the close time of pull request (see Table 3). To comprehensively understand factors' influence on pull request latency, we consider these factors when submitting (opening) and closing a pull request separately.

3.4 Statistical Modeling (SM)

To analyze the influence of factors on pull request latency, we first discover the relationship between factors and solve the collinearity and multicollinearity problems (see SM.1 and SM.2 in Fig. 2).[4] Then we transform the data for further interpretation (SM.3). Finally, we build models in different scenarios (SM.4) and models within different contexts (SM.5).

3.4.1 Discover Factor Relationship

To ensure the reliability for the building of further analysis models, we removed the collinearity and multicollinearity problems via two steps.

Collinearity removal (SM.1) We first calculated the correlations among all the factors. For continuous factors, we used the Spearman correlation coefficient (ρ) (Gousios et al. 2014) and marked $\rho > 0.7$ as a strong correlation (Gousios et al. 2014). For categorical factors, we used Cramér's V value (Φ_c)[5], where $\Phi_c > \frac{0.5}{\sqrt{df^*}}$[6] was considered as a strong correlation (Cohen 1969). For the correlation between continuous and categorical factors, we used the partial Eta-squared value (η^2) (Jones 2019) and marked $\eta^2 > 0.14$ as a strong correlation (Cohen 1969). The heatmap of correlation results can be seen in the technical report.[7] Then we extracted the strongly correlated factors and created strong correlation networks, which are also shown in the technical report.

After calculating factor correlation, we removed the strongly correlated factors by keeping popular factors while removing factors strongly correlated with many others. The reasons are as follows:

- Our objective was to keep as many factors as possible to enrich our model and make the study more meaningful. Therefore, for strongly correlated factors, we remove the one with the biggest number of strong correlations with other factors.
- Among the strongly correlated factors, we keep the factors that are more discussed in related work. Since the goal of our study is to uncover the relative importance of factors, it is more valuable and instructive to consider popular factors for research purposes.

[4] https://quantifyinghealth.com/correlation-collinearity-multicollinearity/
[5] https://www.statstest.com/cramers-v-2/
[6] http://www.real-statistics.com/chi-square-and-f-distributions/effect-size-chi-square/
[7] https://github.com/zhangxunhui/ESE_pull_request_latency

Table 3 List of dynamic factors

factor name	split by process
churn_addition	churn_addition_open churn_addition_close
churn_deletion	churn_deletion_open churn_deletion_open
commits_on_files_touched	commits_on_files_touched_open commits_on_files_touched_close
files_changed	files_changed_open files_changed_close
num_commits	num_commits_open num_commits_close
src_churn	src_churn_open src_churn_close
test_churn	test_churn_open test_churn_close
test_inclusion	test_inclusion_open test_inclusion_close

The description of the workflow[8] and the code[9] are open source.

Multicollinearity removal (SM.2) For the multicollinearity problem, we excluded factors with variance inflation factor (VIF) values ≥ 5, as such values could inflate variance, measured using the *vif* function of the *car* package in R (Cohen et al. 2013). In this way, we removed *num_comments*.

The factors removed in different cases are shown in Table 4. For RQ2, we consider meaningful factors in all the corresponding contexts, so we need to take the intersection of response cases. For example, we consider factors in both the submission and close states for the process context, so factors such as *has_comments* and *num_code_comments* are not considered.

3.4.2 Ease of Interpretation (SM.3)

We stabilize the variance in features log-transforming continuous variables after adding "0.5" (continuous factors) (Hall 2009). Then we transformed the features into a comparable scale with a mean value of "0" and a standard deviation of "1".

3.4.3 Factors' Influence on Pull Request Latency

To explain the influence on pull request latency of all the factors, we need to consider that different factors only make sense in different situations (e.g., factor *ci_latency* only makes sense when a pull request uses CI tools). Figure 3 presents all the situations of model construction in this paper, where situation I-V are used to explain factors' influence. In

[8] https://github.com/zhangxunhui/ESE_pull_request_latency/blob/main/report.pdf
[9] https://github.com/zhangxunhui/ESE_pull_request_latency/blob/main/ese_latency_factor_selection.py

Table 4 The removal of strongly correlated factors

	Submission	Close	has_comments=1	ci_exists=1	same_user=0
first_pr	†	†	†	†	†
perc_external_contribs	†	†	†	†	†
test_cases_per_kloc	†	†	†	†	†
test_lines_per_kloc	†				†
contrib_affiliation	†	†	†	†	†
social_strength	†	†	†	†	†
requester_succ_rate	†	†	†	†	†
churn_deletion	†	†	†	†	†
churn_addition	†	†	†	†	†
asserts_per_loc		†	†	†	
comment_conflict		†	†	†	
part_num_code		†	†	†	†
num_code_comments_con		†	†	†	†
num_comments_con		†	†	†	†
inte_affiliation		†	†	†	†
num_participants		†	†	†	†
at_tag		†	†	†	†
num_comments		‡		‡	‡
has_comments			⸸		
ci_exists				⸸	
same_user					⸸
first_response_time	⸶	⸶		⸶	⸶
ci_latency	⸶	⸶	⸶		⸶
ci_test_passed	⸶	⸶	⸶		⸶

†marks the factors removed for collinearity problem

‡marks the factors removed for multicollinearity problem

⸶marks the postconditions for special cases

⸸marks the preconditions for special cases

these situations, we include as many factors and relevant pull requests as possible to explore which of all potential factors are more important in different cases.

- I Pull requests at submission time. We can learn how factors that can be measured at the pull request submission time influence latency in this situation. Here, factors marked by • (see Table 1) are included when building up the model.
- Ii Pull requests at close time. Comparing to situation I, factors merged during review process (*e.g., has_comments*) are included. Both close time and submission time are just snapshots of the entire lifetime of pull requests, which can help us understand the impact of different factors on pull request latency throughout the lifetime.
- III Pull requests with comments (*has_comments=1*). The first response latency of a pull request (*first_response_time*) only makes sense when there exists comment.
- IV Pull requests using CI tools (*ci_exists=1*). The latency of CI builds (*ci_latency*) and the build results (*ci_test_passed*) only make sense when pull requests use CI tools.

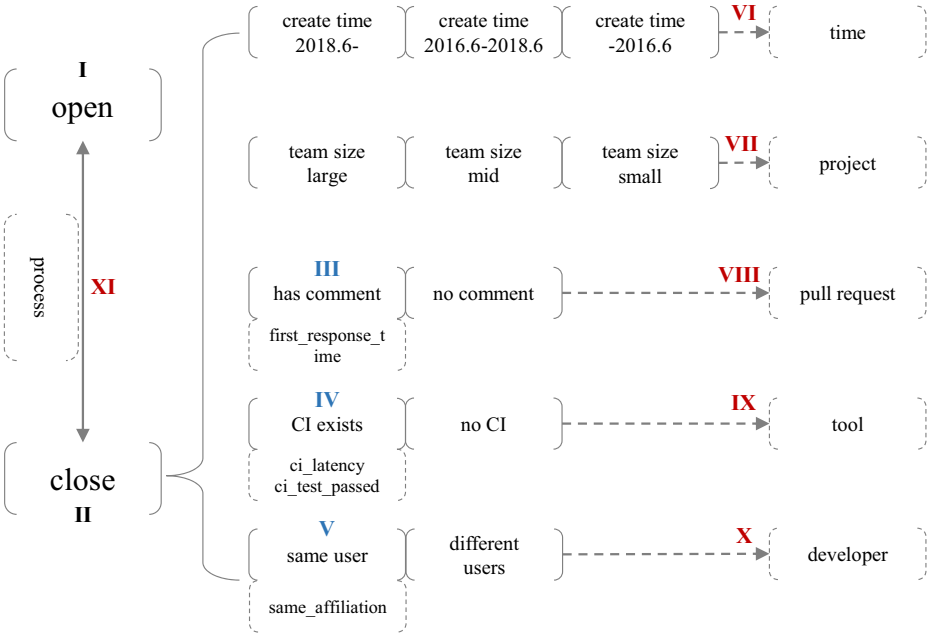

Fig. 3 Various situations of model construction

IV Pull requests submitted and integrated by different users (*same_user=0*). Whether contributor and integrator of a pull request are affiliated to the same organization (*same_affiliation*) only makes sense when this pull request is submitted and integrated by different people.

3.4.4 Influence Change Within Different Contexts

To explore the relevance of context in explaining pull request latency, we studied six scenarios (see context VI-XI in Fig. 3). Context VI-X inherit from our previous study (Zhang et al. 2021). We use the same set of factors in each context but different pull requests to build our model for studying change of factor's importance.

- VI Time context: the period before June 1st, 2016, June 1st, 2016 and June 1st, 2018, and the period after June 1st, 2018.
- VII Project context: team size small ($team\_size \leq 4$), medium ($4 < team\_size \leq 10$), and large ($team\_size > 10$).
- VIII Pull request context: *has_comments=1* and *has_comments=0*.
- IX Tool context: *ci_exists=1* and *ci_exists=0*.
- X Developer characteristic: *same_user=1* and *same_user=0*.
- XI Process context. We select the dynamic factors that can change during the review process (refer to factors marked by ⋆ in Table 1). We study whether the importance of these factors changes after the review process. We can understand whether the impact of factors changes under different snapshots in the pull request lifetime through this context. It is also possible to understand the effect of pull request modifications on pull request latency during the review process.

3.5 Model Interpretation

The resulting mixed-effect linear regression models explain the influence of factors in models and their relative relevance. Section 4 presents the findings from these mixed-effect linear regression models. We report the regression coefficients together with their *p-value*s. The term *p-value* indicates the statistical significance of a factor, which was indicated by asterisks: $^{***}p < 0.001; ^{**}p < 0.01; ^{*}p < 0.05$. Meanwhile, we present the explained variance of each factor derived from ANOVA type-II analysis (Langsrud 2003). Finally, the *percentage of explained variance* (calculated by $explained\ variance/total\ amount\ of\ variance$) was used as a proxy for the relative importance of a factor (Overney et al. 2020). For the *goodness of fit* of each model, we report both marginal and conditional R^2 (Nakagawa and Schielzeth 2013), which considers the variance of the fixed effects and both fixed and random effects, respectively.

4 Results

In this section, we first present how factors influence pull request latency (answering **RQ1**). It includes models for pull requests at open time, pull requests at close time, pull requests with comments, pull requests using CI tools, and pull requests submitted and integrated by different users (see Section 3.4.3). After that, we present how factors influence pull request latency change with context change (answering **RQ2**), consisting of models for six contexts, namely time, project, pull request, tool, developer, and process contexts (see Section 3.4.4).

4.1 RQ1: How do Factors Influence Pull Request Latency?

4.1.1 Open Time

Columns 2 and 3 in Table 10 show the influence of factors at pull request open time, where *description_length* (row 2), *src_churn_open* (row 3), *prev_pullreqs* (row 4), *integrator_availability* (row 5) and *open_pr_num* (row 6) are the top 5 important factors. They together explain 88.7% of the total variance and are all significantly important regarding the latency of pull requests.

Among all the factors, the length of pull request description (*description_length*) contributes the most to the latency of pull request (46.3% variance), which positively correlates with the latency. The pull request text description's length may indicate the pull request's complexity, and it is likely that more complex or challenging to understand pull requests take longer to review.

The amount of source code change (*src_churn_open*) ranks the second (20.4% variance), which is also positively correlated with pull request latency. One possible explanation is that the amount of source code changed affects the amount of time the maintainer has to check the correctness of the software.

In addition to these technical factors, the contributor's experience (*prev_pullreqs*) also has a sizable effect, which explains 8.9% variance and negatively correlates with the latency of pull requests. Comparing to other developer characteristics, the contributor's experience in the target project is more decisive regerding the latency at pull request open time.

For project characteristics, the *integrator_availability* and *open_pr_num* have sizable effects, explaining 6.7% and 6.5% variance, respectively. These results indicate that the top active reviewers and workload of projects influence the latency of pull requests, increasing the latency with the increase of description length.

> When submitting a pull request, both social and technical factors (pull request, developer, and project characteristics) influence the latency of pull request with sizable effects, among which the length of pull request description and the size of the source code change are the most influential.

4.1.2 Close Time

Columns 4 and 5 in Table 10 present the influence of factors at pull request close time, among which *has_comments* (row 21), *same_user* (row 22), *description_length* (row 2), *num_code_comments* (row 23) and *num_commits_close* (row 14) are the top 5 important factors, explaining 80.8% of the total variance and are all significantly important.

Factor *has_comments* ranks the first in explaining pull request latency (37.5% variance) with a positive correlation. Communication between reviewers and the contributor may indicate that reviewers ask for changes or ask the contributor relevant questions to understand the contribution content.

For factor *same_user*, it explains 16.9% of the total variance with a positive correlation. It indicates that when the contributor and integrator are not the same (*same_user=0*), the latency of a pull request is more likely to increase compared to the situation when the contributor and integrator are the same. A possible explanation is that when reviewing pull requests submitted by other people, reviewers need to spend more time making the final decision.

Among all the factors, *num_code_comments* has sizable positive effect, which explains 8.4% of the total variance. This indicates that the presence or absence of comments on the code during the review process has a moderate correlation with pull request latency. Moreover, comments on the code increase the decision time.

For the length of pull request description, its importance ranks the 3rd to explain pull request at close time (12.3% variance). It still has a sizable impact in increasing pull request latency even comparing with factors that occurred during the review process.

We see that factor *num_commits_close* has a sizable positive effect by explaining 5.7% of the total variance. This indicates that the number of commits in the pull request has a moderate impact on the pull request latency, as seen in the snapshot of the pull request closing. In Session 4.2.6, we will examine how the same factors differ in the interpretation of pull request latency in the two contexts of pull request submission and closing.

From an overall perspective, the close time model fits better than the submission model according to the R-square values. The result is likely to be related to adding factors that occurred during the review process (*e.g., has_comments*) and the change of dynamic factors (*e.g., num_commits*).

When closing a pull request, factors that occurred during the review process (*e.g., has_comments*) have a significant impact on the latency of pull request, which weakens the effect of factors measured at pull request submission time. Meanwhile, the overall fitness of the model becomes better by adding the factors that occurred during the review process.

4.1.3 Pull Requests With Comments

Columns 6 and 7 in Table 10 present the influence of factors when pull requests have comments (*has_comments=1*), where *first_response_time* (row 28), *num_comments* (row 29), *num_code_comments* (row 23), *hash_tag* (row 24) and *prev_pullreqs* (row 4) rank the top 5, explaining 95.4% of the total variance.

Among all the factors, we can see that the latency is highly dependent on the comments during the review process, where the importance of *first_response_time*, *num_comments* and *num_code_comments* ranks the 1st, 2nd and 3rd, explaining 58.7%, 31.4% and 2.6% variance respectively. However, for factor *hash_tag* and *prev_pullreqs*, which ranks the 4th and 5th, explaining only 1.8% and 0.9% variance respectively.

When there exist comments in pull requests, comment-related factors during the review process play a decisive role in the latency of pull requests. The latency of the first reviewer's response is the most important.

4.1.4 Pull Requests Using CI Tools

Columns 8 and 9 in Table 10 show the results of factors' influence on pull request latency when pull requests use CI tools. The top 5 important factors are *has_comments* (row 21), *ci_latency* (row 30), *ci_test_passed* (row 31), *num_code_comments* (row 23) and *same_user* (row 22), with 64.7% variance explained in total. All of the five factors are factors that occurred during the review process. For factors that can be measured at the opening time of pull request, they also have a sizable effect although not ranking in the top 5. For example, *description_length* (row 2) and *integrator_availability* (row 5), they explain 6.1% and 4.6% of the total variance respectively.

Factors *ci_latency* and *ci_test_passed* explain 18.2% and 10.4% of the total variance, respectively, which indicate that for pull requests using CI tools, the time used for making decisions is moderately relevant to the build time and build outcome of CI tools. For the influence direction, pull requests that need longer CI builds are more likely to take more time for review. Further, pull requests which passed the CI builds are more likely to be handled in a shorter time.

For other situations except for *ci_exists=1* in Table 10, we can see that pull requests using CI tools are likely to take more time to be handled (row 8).

When pull requests use CI tools, both factors that occurred during the review process (*e.g., has_comments*) and factors that can be measured at pull request submission time (*e.g., description_length*) have a sizable effect on pull request latency. Meanwhile, we find that the usage of CI tools slows down the processing of pull requests.

4.1.5 Pull Requests Submitted and Integrated by Different People

Columns 10 and 11 in Table 10 present the influence of factors on pull request latency when pull requests are submitted and integrated by different people. Among all the factors, *has_comments* (row 21), *num_code_comments* (row 23), *description_length* (row 2), *integrator_availability* (row 5) and *num_commits_close* (row 14) rank the top 5, explaining 70.1% of the total variance.

Factor *has_comments* stands out, which explains 33.9% variance and is much higher than all the other factors. A possible explanation is that when the contributor and integrator are not the same, the review process highly depends on communication, which helps the reviewers understand the contribution and make the final decision.

When pull requests are submitted and integrated by different people, comments under a pull request play the most important role in influencing pull request latency. This reflects that pull request reviewers rely on communicating with contributors through comments for the standardised pull-based development, which is likely to bring time costs to the review process.

4.2 RQ2: Do the Factors Influencing Pull Request Latency Change with a Change in Context?

4.2.1 Time Context

Columns 2-7 of Table 11 present the results of comparison among pull requests in different time contexts. Among all the factors, the variance explained by factor *has_comments* (row 2) and *project_age* (row 20) change more than 5%.

For factor *project_age*, its explained variance increases directly from 0.1% to 26.2% as projects evolve, with negative influence on pull request latency. Since the project age can reflect the project maturity to some extent, this result indicates that project maturity is increasingly important to distinguish the pull request latency as projects evolve. Meanwhile, the more mature the project, the faster the review of the pull request will be.

For factor *has_comments*, the explained variance decreases from 43.1% to 27.9% as projects evolve. It indicates that comment under a pull request cannot be a decisive factor in measuring its latency as the project evolves. We calculated the percentage of pull requests with comments for each period and found that the ratio decreases as projects evolve (62.7% for the period before 1st June, 2016; 61.3% for the period between 1st June, 2016-1st June, 2018; 57.7% for the period after 1st June, 2018). However, the percentage of pull requests using CI tools increases as projects evolve (65.7% for the period before 1st June, 2016;

83.9% for the period between 1st June, 2016-1st June, 2018; 86.0% for the period after 1st June, 2018). CI tools are likely replacing the manual review process as projects evolve, leading to decreased importance of comments.

> As the project evolves, the maturity of the project becomes more and more important for determining pull request latency, while the comment existence no longer plays a decisive role.

4.2.2 Project Context

Columns 2-7 of Table 12 present the results of comparison among projects with different team sizes when submitting pull requests. Among all the factors, the variance explained by factor *num_code_comments* (row 7) and *same_user* (row 3) change more than 5%.

For factor *num_code_comments*, the explained variance increases from 4.2% to 15.8% with the increase of team size. We calculate the median value of project workload (*open_pr_num*) with different team sizes and find that the workload increases a lot with the team size (small: 6, mid: 13, large: 77). Therefore, it is likely that the increased workload distracts the reviewers, so if code comments exist, it may take longer to wait for the changes to be reviewed again.

For factor *same_user*, its explained variance decreases from 21.7% to 10.5% with the increase of team size. It is likely that due to the standardization of the pull-based development process, pull requests need to be reviewed by others. We calculated the ratio of pull requests submitted and integrated by the same user for different team sizes (small: 45.3%, mid: 44.1%, large: 40.5%). It is likely that the decrease of pull requests submitted and integrated by the same user lead to the decline in importance in explaining the pull request latency.

> As the team size of projects become larger, the discussion under pull request code snippets becomes more and more important for explaining pull request latency, while the relationship between contributor and integrator becomes less important.

4.2.3 Pull Request Context

Columns 8-11 of Table 11 present the comparison results between pull requests with and without comments. Among all the factors, *same_user* (row 2), *num_commits* (row 6), *src_churn* (row 8) and *prev_pullreqs* (row 14) are outstanding, with explained variance change more than 5%.

For factor *same_user*, the explained variance increases from 8.8% (*has_comments=true*) to 50.8% (*has_comments=false*), which indicates that when there does not exist comment, whether submitted and integrated by the same user, plays a decisive role in the latency of pull request. Statistically, we calculated the latency (in minutes) of pull requests in different situations (see Table 5). It is clear to see that although the median latency of *has_comments=false* situation is much smaller than *has_comments=true* situation, the discrimination for *same_user* is much stronger when no comment exists (with more than 10

Table 5 Pull request median latency (in minutes) for *has_comments* and *same_user* cross situations

	has_comments=true	has_comments=false
same_user=true	1,348	30
same_user=false	2,881	311

times difference in pull request median latency). This indicates that contributors can easily decide according to the project standard by themselves. In contrast, for others, it takes some time to complete the review and make the final decision, even though no need to communicate with the contributor.

For factor *num_commits*, the explained variance decreases from 28.1% (*has_comments=true*) to 5.9% (*has_comments=false*). A possible explanation is that when there exist comments, it is likely that the pull request needs to be modified to meet project standards. Thus its latency is highly dependent on the code changes during the review process.

Likewise, for factor *src_churn*, it is directly related to the code change during review process. Therefore, its explained variance decreases from 14.6% (*has_comments=true*) to 3.2% (*has_comments=false*).

For factor *prev_pullreqs*, the explained variance decreases from 8.2% (*has_comments=true*) to 0.1% (*has_comments=false*). It is likely that when there exist comments, experienced contributors can modify their contributions easier and faster than the non-experienced.

> Comparing to pull requests without comments, when there exists communication between the contributor and reviewers, the size of the contribution and the contributor's experience become more important in determining the latency of pull requests. Without communication, the relation between contributor and integrator plays a decisive role in pull request latency.

4.2.4 Tool Context

Columns 8-11 of Table 12 present the results of comparison between pull requests using CI tools and not using CI tools. Among all the factors, only *same_user* (row 3) has more than 5% change of explained variance, which increases from 14.1% (*ci_exists=true*) to 25.3% (*ci_exists=false*). Likewise, we calculate the latency (in minutes) in different situations (see Table 6). We can see from the result that using CI tools increases the latency of pull requests in an overall perspective. Contributors likely need to spend more time on modifying contributions to meet project standards. When not using CI tools, the latency of pull request is easier to distinguish using factor *same_user* (with more than 18 times difference in pull request median latency). A possible explanation is that pull request decisions can be made quickly for developers with merge access due to the lack of automatic feedback from CI tools. However, it takes relatively long for other people's contributions to be reviewed.

> The use of CI tools increases the overall latency of pull requests. The latency differs significantly for pull requests that do not use CI tools, considering the relationship between the submitter and the integrator.

Table 6 Pull request median latency (in minutes) for *ci_exists* and *same_user* cross situations

	ci_exists=true	ci_exists=false
same_user=true	310	51
same_user=false	1,461	926

4.2.5 Developer Context

Columns 8-11 of Table 13 present the results of pull requests within different developer contexts (whether contributor and integrator are the same). Among all the factors, *has_comments* (row 21), *description_length* (row 2), *integrator_availability* (row 5) and *open_pr_num* (row 6) are outstanding, with explained variance change of more than 5% across different contexts.

For factor *has_comments*, the explained variance decreases from 50.3% (*same_user=true*) to 33.8% (*same_user=false*). According to Table 5, factor *has_comments* is more distinguishable for pull request latency when *same_user=true*. For factor *description_length*, the explained variance decreases from 17.6% (*same_user=true*) to 9.1% (*same_user=false*). Through the data statistics, we found that the description information was generally long for cases with different integrators and contributors (*same_user=true* median: 19, *same_user=false* median: 26). This illustrates that while self-integrated pull request description messages are generally shorter and in this case, it is easier to explain the latency of the pull request by the length of description.

For factor *integrator_availability*, the explained variance increases from 2.1% (*same_user=true*) to 8.8% (*same_user=false*). The result may be explained by the fact that only pull request reviewed by others highly depends on the availability of other integrators. Likewise, the workload (*open_pr_num*) of projects only have a sizable effect on the latency of pull request submitted and integrated by different people.

> If the contributor and integrator are the same, pull request latency depends on the communication between the contributor and reviewers and the length of the pull request description. When the contributor and integrator are different, pull request latency depends on the project workload and availability of active integrators.

4.2.6 Process Context

Columns 2-5 of Table 13 present the results of comparison for pull requests at submission time and close time. We study whether the dynamic factors perform differently regarding pull request latency (column 2-5) and whether the change of these factors influences the latency of pull requests (column 6-7).

Among all the factors, the explained variance of *description_length* decreases from 46.3% (open time) to 36.5% (close time). It indicates that the description length becomes less important in determining the latency of pull requests as the review process evolves. During the review process, pull requests can be modified, which may be the reason for the importance increase of factor *num_commits* (0.3% at submission time, 22.8% at close time). To verify the explanation, we built another model (process change) by replacing factor *num_commits* with its change during the review process (*num_commit_change*), and found

that the change of commits is significantly important for pull request latency with a sizable effect (row 14, columns 6 and 7; 34.5% variance).

Surprisingly, we find that the explained variance of factor *src_churn* decreases (20.4% at submission time, 11.1% at close time). A possible explanation is that the change in the number of commits indicates the change in time compared to the change in lines of source code, which in turn can better explain the time consumption of the pull request review process.

The model fits better at the close time than the submission time. Meanwhile, we find that if we replace the dynamic factors with the amount of variation during the review process, the model fits better. It indicates that the factors explain the pull request latency better for snapshots that are closer to the merge/reject state. Also, the change of factors during the review process has a significant impact on pull request latency.

> For dynamic factors that can change during the review process, the number of commits significantly influences the latency of pull requests. As pull requests approach the merge/reject state, the factors become more explanatory of the pull request delay.

5 Case Study

We found that bots are widely used in the pull-based development model (Wessel et al. 2018). As an essential support mechanism, we explore the impact of bot participation on pull request latency.

5.1 Bot Recognition

There exists a lot of bot identification methods, including identifying based on user name (Wessel et al. 2018), commit information (Dey et al. 2020; Golzadeh et al. 2021), comment information (Cassee et al. 2021; Golzadeh et al. 2021). Golzadeh et al. (2021) created the tool "BoDeGHa" to identify bots using the comments of pull requests and issues in a repository, and the F1 value reached 0.98. In this paper, we use this method for bot identification.

We have made some modifications to the tool. We removed the extraction process of the pull request and issue comments from BoDeGHa, and instead got all the historical comment information (not limited to a particular repository) of each pull request-related commenter from the GHTorrent dataset. By increasing the number of comments in the prediction, we can break the limitation of "minimal number of non-empty comments" in BoDeGHa and identify more users. At the same time, the increase of users' comment information can theoretically improve prediction accuracy. The comments are then fed into BoDeGHa's prediction model to classify the reviewers. For the pull requests containing comment information in our dataset, 154,278 reviewers were successfully classified using BoDeGHa. 2,551 of them were bots (1.7%). After removing the pull requests containing unsuccessfully classified reviewers (reviewers identified as "unknown" by BoDeGHa), 33.8% have bot comments among all the commented pull requests.

Based on the recognition of the reviewer, we add four new factors, i.e., whether there exists human comment (*has_human_comments*), whether there exists bot comment

(*has_bot_comments*), the number of human comments (*num_human_comments*), the number of bot comments (*num_bot_comments*).

5.2 Influence of Bot Comments on Pull Request Latency

We constructed two mixed-effect linear regression models for the effects of bot reviews. The results are shown in Table 7.

Table 7 Results for bot-related factors in influencing pull request latency

	(1)	(2) has_bot/human_comments		(3)		(4) num_bot/human_comments		(5)	
		Coeffs (Err.)		Sum sq		Coeffs (Err.)		Sum sq	
(1)	(Intercept)	-0.66(0.01)	***	-		-0.02(0.01)	***	-	
(2)	first_response_time	0.49(0.00)	***	158504.90	***	0.47(0.00)	***	157003.22	***
(3)	followers	0.00(0.00)	***	7.80	***	-0.01(0.00)	***	141.70	***
(4)	prev_pullreqs	-0.11(0.00)	***	4059.50	***	-0.08(0.00)	***	2082.64	***
(5)	integrator_availability	0.04(0.00)	***	1272.83	***	0.04(0.00)	***	1274.51	***
(6)	open_pr_num	0.14(0.00)	***	1981.48	***	0.14(0.00)	***	2045.94	***
(7)	project_age	0.02(0.00)	***	85.93	***	0.03(0.00)	***	182.45	***
(8)	team_size	-0.04(0.00)	***	140.69	***	-0.03(0.00)	***	138.27	***
(9)	description_length	0.08(0.00)	***	4817.24	***	0.05(0.00)	***	2155.81	***
(10)	num_code_comments	0.16(0.00)	***	17519.44	***	-0.06(0.00)	***	1502.88	***
(11)	commits_on_files_touched	-0.03(0.00)	***	631.82	***	-0.04(0.00)	***	813.84	***
(12)	files_changed	0.00(0.00)		0.44		-0.01(0.00)	***	23.24	***
(13)	num_commits	0.11(0.00)	***	6107.63	***	0.07(0.00)	***	2238.16	***
(14)	src_churn	0.08(0.00)	***	2887.80	***	0.06(0.00)	***	1920.20	***
(15)	test_churn	0.01(0.00)	***	33.51	***	0.00(0.00)		0.34	
(16)	core_member	-0.06(0.00)	***	325.34	***	-0.05(0.00)	***	287.70	***
(17)	ci_exists	0.03(0.00)	***	58.26	***	0.03(0.00)	***	74.76	***
(18)	friday_effect	0.06(0.00)	***	612.78	***	0.06(0.00)	***	645.31	***
(19)	hash_tag	0.20(0.00)	***	7520.40	***	0.16(0.00)	***	4981.70	***
(20)	test_inclusion	0.00(0.00)		1.24		0.01(0.00)	***	18.55	***
(21)	same_user	0.08(0.00)	***	939.90	***	0.06(0.00)	***	582.59	***
(22)	contrib_gender	0.01(0.00)	*	2.81	*	0.00(0.00)		0.48	
(23)	prior_review_num	0.00(0.00)	**	5.49	**	0.00(0.00)	***	7.33	***
(24)	sloc	0.01(0.00)	***	23.72	***	0.01(0.00)	***	17.63	***
(25)	test_lines_per_kloc	-0.01(0.00)	***	38.66	***	-0.01(0.00)	***	17.29	***
(26)	reopen_or_not	0.25(0.00)	***	1400.45	***	0.12(0.00)	***	340.66	***
(27)	has_human_comments	0.42(0.00)	***	11421.56	***		-		
(28)	has_bot_comments	0.48(0.00)	***	21091.72	***		-		
(29)	num_human_comments		-			0.36(0.00)	***	49984.60	***
(30)	num_bot_comments		-			0.25(0.00)	***	26603.94	***
	nobs	1,170,192				1,170,192			
	R_m^2	0.40				0.46			
	R_c^2	0.47				0.52			

From the table, we find that all four factors (*has_human_comments*, *has_bot_comments*, *num_human_comments*, *num_bot_comments*) positively affect the pull request latency. This indicates that no matter from humans or bots, as long as a pull request contains comments, it is likely that the lifetime will be lengthened. We found by manual inspection that some bots do not trigger when a pull request is submitted but automatically generate comments during the review process.[10][11] Some other bots may refer to many reviewers to review the pull request, and subsequent manual review will lead to the long latency of the pull request.[12] Therefore, many reasons may cause the increase of pull request latency when bot comments exist. For the goodness of fit, the model containing the number of comments fits better (R_c^2=0.52), which indicates that the number of reviews is more correlated with pull request latency than whether it contains comments or not. Multiple reviews imply multiple revisions and therefore have a stronger correlation with pull request latency. However, we also find from the results that the effect size of *has_bot_comments* is larger than *has_human_comments* when we consider whether or not to include the corresponding comments. However, when we consider the number of comments, the effect size of *num_bot_comments* is smaller than *num_human_comments*. One possible reason is that the bot's comment directly points out the problems in the pull request. The contributor makes code changes based on the comments. The first comment is more critical since the bot's comments are fixed.[13] As for humans, the focuses of comments may be different. Contributors may need to focus on different modifications, thus leading to the number of human comments being more relevant to the latency of pull requests.

6 Discussion

By studying the factors affecting pull request latency, we now have a deeper understanding of the entire lifetime of pull requests. To better understand the impact of the review process on pull request latency, we add dynamic factors to the data collection in the pull request submission and closure states compared to previous work. We analyze the impact of factors on pull request latency in different situations and contexts with a more comprehensive dataset.

6.1 Interpretation

RQ1 How do factors influence pull request latency? The impact of factors on pull request latency cannot be simply summarized in a holistic manner alone. For example, when submitting a pull request, the length of the pull request description, the number of source code changes, developer experience, project reviewer activity, and project workload are the five most important factors. However, when we consider the snapshot of when the pull request is closed, we find that all the factors are less important in explaining the pull request's latency than those generated during the review process. Similarly, for pull requests with comment information, the latency of the first comment and the number of comments are most important. Pull requests that use CI tools depend on the CI builds latency and the build results. If the person who finally integrates the pull request is not the same as the submitter, it depends

[10] https://github.com/ajaxorg/ace/pull/865
[11] https://github.com/palantir/atlasdb/pull/900
[12] https://github.com/ansible/ansible/pull/38773
[13] https://github.com/openshift/openshift-tools/pull/2937

on comment information. This phenomenon indicates that in the future, when conducting research related to pull request latency or building prediction tools, we should consider the characteristics of pull requests in-depth and build different models.

RQ2 Do the factors influencing pull request latency change with a change in context? Yes. The effect of factors on pull request latency is subject to change with context, and we consider the impact of the same set of factors in different contexts. The project's maturity is found to have a better explanatory role with pull request latency as it evolves. When the project team size increases, the code comments during the process become more important. Whether the pull request contains discussion information is correlated with whether its integrator and contributor are the same. If one ends up making decisions about own pull requests, this decision depends on the comments during the review process, which in turn affects the review latency. For the whole pull request review process, the closer a factor is to the final state, the better the effect of factors on the interpretation of the pull request, and the change of factors in the process has a significant impact on the latency of the pull request. These results illustrate that the importance of factors on the interpretation of pull request is changed with the change of pull request state in the lifetime of the pull request.

6.2 Implication

Compared to previous work, this paper discusses the impact of factors on pull request latency at both the submission and close time. While considering all factors at different states, we find that the factors generated during the process have a more significant impact. For example, when submitting a pull request, the relatively important factors, such as the number of lines of modified source code, the contributor's experience, the integrator's idle status, and the project's workload, become less important after considering in-process factors, such as the existence of comments and the number of code comments.

When considering the relatively important factors for all pull requests, Kononenko et al. (2018) did not consider the relative importance of code comments after combining various factors. Our results find that this factor is relatively essential, ranking the 4th. Yu et al. (2016) discussed the effect of the number of comments on results and found that it is less important than the number of added code lines and the latency of CI builds. However, we introduced whether there exist comments (no related work considered this factor). We found that its relative importance is even more significant than the code change size and the latency of CI builds. In the future, when building predictive tools, the existence of comments can be an excellent alternative to the number of comments as a predictor of pull request latency. Similarly, for factor *same_user* (not discussed in related work), we find that other-integrated pull requests take longer than the self-integrated. McIntosh et al. (2014) found that self-approved contributions are bug-prone, while contributors prefer self-rejection over self-acceptance (Zhang et al. 2021). In combination with the short review time of self-integrated pull requests, we argue that future pull requests that are self-approved and take a short time need special attention from other developers in the community.

When considering the impact of factors among different contexts, we find that the number of changed commits during the review process significantly impacts latency. In contrast, changes in the number of lines of code during the review process do not have a strong correlation. Kononenko et al. (2018) only analyzed the correlation between the changed lines of code at one snapshot of pull request lifetime and its latency. However, they did not consider the impact of in-process changes. Our conclusions suggest that the amount of code changed during the process is not a key factor affecting the latency of pull requests. Instead,

the number of commits submitted during the review process is essential. So how to reduce the number of revisions is critical in accelerating the pull request review process.

Since bot plays a vital role in the pull-based development model (Wessel et al. 2018), we add a case study of bot comments, which were not considered in previous studies. It is found that both human and bot comments increase pull request latency. However, the number of human comments is more important for explaining pull request latency than the number of bot comments. Since bot checks are relatively homogeneous, we think that the content of the first review has a significant impact on the review process. Future research or construction of prediction tools need to specifically examine the first review comment of bots and increase the weight of prediction.

This paper can generate the following actionable suggestions.

1. For research
   - When conducting research related to pull request latency, our results can help researchers select control factors under different contexts in the pull request lifecycle and then conduct empirical studies for new factors. The recommendations of specific factors are shown in Table 8.
   - Since bot and human comments have different effects on the latency of pull requests, subsequent researchers can discuss the differences in the effects of the type of bots on the latency of pull requests.
2. For practice
   - Future developers need to consider different factors in different contexts when building predictive tools for pull request latency. For example, when a pull request is just submitted, the focus should be on the length of the pull request description. While during the review process, the focus should be on whether there exist comments, the delay of first response, etc. When pull requests using CI tools, the latency of CI tools needs to be considered.
   - For pull requests reviewed by the same person, if the review time is much lower than the average time and no one else is involved in the review process of a self-approved pull request, project managers can be alerted to the quality of the merged code by adding a self-approved tag (Tables 9, 10 and 11).

7 Threats to Validity

This work is based on our previous dataset (Zhang et al. 2020a). The construction of the dataset relies on the measurement methods of related work and the GitHub API and GHTorrent dataset. Therefore, this paper inherits all the threats of the previous work.

At the same time, the paper also has the following limitations:

1. Different algorithms have different ways of evaluating the importance of factors. Here we use the mixed-effect linear regression model. The reason is that while explaining the relative importance of factors, we can determine their linear influence.
2. When preprocessing the dataset, we deleted factor *bug_fix*, which had many missing values. Thus this paper lacks a discussion about this factor.
3. Although we have discussed the six kinds of contexts, there are many more contexts, and we have not divided the contexts according to each factor. And because the dataset

Table 8 The recommended control factors for different contexts

	overall-submission	overall-close	other-integrated	self-integrated	has comment	no comment	use CI	no CI
description_length	✓	✓	✓	✓		✓	✓	✓
src_churn	✓	✓	✓	✓		✓	✓	✓
prev_pullreqs	✓		✓				✓	
integrator_availability	✓	✓	✓	✓		✓	✓	✓
open_pr_num	✓	✓	✓	✓		✓	✓	✓
core_member	✓					✓		✓
ci_exists	✓			✓		✓		
test_churn	✓							
commits_on_files_touched	✓		✓				✓	
num_commits		✓	✓	✓		✓	✓	✓
has_comments		✓	✓	✓			✓	✓
same_user		✓				✓	✓	✓
num_code_comments		✓	✓	✓	✓		✓	✓
hash_tag		✓	✓	✓	✓		✓	✓
reopen_or_not		✓	✓					
comment_conflict			✓	✓				
first_response_time					✓			
num_comments					✓			
ci_latency							✓	
ci_test_passed							✓	

✓marks the recommended control factors when building linear regression models for pull request latency

Table 9 Factors related to pull request decisions in related articles

	Gousios et al. 2014	Yu et al. 2016	Baysal et al. 2016	Kononenko et al. 2018	Jiang et al. 2019	Baysal et al. 2012	Pinto et al. 2018	Bosu and Carver 2014	Lee and Carver 2017	Hechtl 2020	Yu et al. 2015	Bernardo et al. 2018	Hilton et al. 2016	Zhao et al. 2017	Imtiaz et al. 2019	Sadowski et al. 2018	Hu et al. 2018
Developer Characteristics																	
contrib_affiliation			▲	▲													
contrib_gender															△		
core_member		▲				△	△	▲	▲	△	▲						
first_pr									▲								
first_response_time		▲									▲						
followers		▲									▲						
inte_affiliation			▲													▲	
prev_pullreqs	△		▲	▲													
prior_review_num			▲														
same_affiliation			▲														
social_strength		▲									▲						
Project Characteristics																	
asserts_per_kloc	◬																
integrator_availability		▲									▲						
open_pr_num		▲	△								▲						
perc_external_contribs	△																
project_age		▲									▲						
requester_succ_rate	△										▲						
sloc	△																
team_size	△	▲									▲						
test_cases_per_kloc	◬																
test_lines_per_kloc	△																

Table 9 (continued)

	Gousios et al. 2014	Yu et al. 2016	Baysal et al. 2016	Kononenko et al. 2018	Jiang et al. 2019	Baysal et al. 2012	Pinto et al. 2018	Bosu and Carver 2014	Lee and Carver 2017	Hechtl 2020	Yu et al. 2015	Bernardo et al. 2018	Hilton et al. 2016	Zhao et al. 2017	Imtiaz et al. 2019	Sadowski et al. 2018	Hu et al. 2018
Pull Request Characteristics																	
at_tag		▲									▲						
bug_fix																	▲
churn_addition		▲									▲						
churn_deletion		▲									▲						
ci_exists												△	▲	▲			
ci_latency		▲									▲						
ci_test_passed		▲									▲						
comment_conflict	△																
commits_on_files_touched	△	▲									▲						
description_length		▲									▲						
files_changed	△			△													
friday_effect		▲									▲						
hash_tag	△	▲									▲						
num_code_comments				△													
num_code_comments_con				△													
num_comments	△	▲		△							▲						
num_comments_con				△													
num_commits	△	▲		▲							▲						

Table 9 (continued)

	Gousios et al. 2014	Yu et al. 2016	Baysal et al. 2016	Kononenko et al. 2018	Jiang et al. 2019	Baysal et al. 2012	Pinto et al. 2018	Bosu and Carver 2014	Lee and Carver 2017	Hechtl 2020	Yu et al. 2015	Bernardo et al. 2018	Hilton et al. 2016	Zhao et al. 2017	Imtiaz et al. 2019	Sadowski et al. 2018	Hu et al. 2018
num_participants	△			▲													
part_num_code				▲													
reopen_or_not					△												
src_churn	△		▲	▲													
test_churn	△																
test_inclusion		▲									▲						

First column lists factors in alphabet ascending order in each class, the rest columns list related articles and the result of each factor. Horizontal Line in the middle of shape (△) means the factor is removed when building models because of collinearity or multicollinearity. Filling: Filled (▲) means *significance is reported* and unfilled (△) means *significance is not reported because of not using statistical model or inconsistent conclusions*. Size of filled shape: Big shape (▲) shows *statistically significant* relation and small shape (▲) *statistically insignificant* with 95% confidence threshold. Color: ▲ means a *negative relation* (meaning decrease pull request latency), red ▲ means a *positive relation*, gray ▲ means *uncertain relation* because of not using statistical model or nonlinear conclusion

Table 10 Results for explaining factors' influence on pull request latency. - means the factor is not included in the model

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
		Submission time		*Close time*		*has_comments=1*		*ci_exists=1*		*same_user=0*	
		Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq
(1)	(Intercept)	-0.04(0.01)***		-0.49(0.00)***		-0.03(0.01)***		-0.18(0.01)***		-0.29(0.01)***	
(2)	description_length	0.20(0.00)***	66429.42 ***	0.13(0.00)***	26396.35 ***	0.05(0.00)***	2063.70 ***	0.09(0.00)***	5817.93 ***	0.10(0.00)***	8934.70 ***
(3)	src_churn_open/close	0.16(0.00)***	29349.67 ***	0.08(0.00)***	7214.10 ***	0.06(0.00)***	1913.66 ***	0.09(0.00)***	3556.99 ***	0.09(0.00)***	5112.53 ***
(4)	prev_pullreqs	-0.13(0.00)***	12717.35 ***	-0.05(0.00)***	1770.14 ***	-0.08(0.00)***	2477.51 ***	-0.08(0.00)***	1966.67 ***	-0.09(0.00)***	3438.87 ***
(5)	integrator_availability	0.07(0.00)***	9570.64 ***	0.07(0.00)***	8606.00 ***	0.04(0.00)***	1397.17 ***	0.07(0.00)***	4372.21 ***	0.09(0.00)***	8694.52 ***
(6)	open_pr_num	0.20(0.00)***	9335.41 ***	0.17(0.00)***	6623.60 ***	0.14(0.00)***	2072.11 ***	0.16(0.00)***	2934.40 ***	0.22(0.00)***	6618.30 ***
(7)	core_member1	-0.17(0.00)***	5336.89 ***	-0.10(0.00)***	2057.12 ***	-0.06(0.00)***	397.56 ***	-0.09(0.00)***	613.24 ***	-0.06(0.00)***	389.44 ***
(8)	ci_exists1	0.16(0.00)***	4106.24 ***	0.11(0.00)***	1860.33 ***	0.03(0.00)***	88.90 ***	-		0.05(0.00)***	232.92 ***
(9)	test_churn_open/close	0.05(0.00)***	2309.25 ***	0.02(0.00)***	437.97 ***	0.00(0.00)*	2.95 *	0.01(0.00)***	67.21 ***	0.02(0.00)***	264.38 ***
(10)	commits_on_files_touched_open/close	-0.04(0.00)***	1734.60 ***	-0.03(0.00)***	1273.88 ***	-0.03(0.00)***	855.42 ***	-0.04(0.00)***	1175.53 ***	-0.04(0.00)***	1490.81 ***
(11)	followers	-0.03(0.00)***	905.21 ***	-0.01(0.00)***	68.37 ***	-0.02(0.00)***	255.71 ***	-0.02(0.00)***	153.68 ***	-0.02(0.00)***	183.81 ***
(12)	friday_effect1	0.05(0.00)***	853.15 ***	0.05(0.00)***	873.60 ***	0.06(0.00)***	643.79 ***	0.07(0.00)***	704.96 ***	0.06(0.00)***	684.75 ***
(13)	files_changed_open/close	-0.02(0.00)***	374.92 ***	-0.01(0.00)***	96.69 ***	-0.01(0.00)***	40.90 ***	0.00(0.00)	1.50	0.00(0.00)	0.24
(14)	num_commits_open/close	0.02(0.00)***	362.10 ***	0.11(0.00)***	12127.66 ***	0.05(0.00)***	1324.34 ***	0.11(0.00)***	5788.89 ***	0.12(0.00)***	8229.60 ***
(15)	sloc	-0.02(0.00)***	118.65 ***	-0.01(0.00)***	49.20 ***	0.01(0.00)***	18.60 ***	-0.01(0.00)***	11.85 ***	-0.01(0.00)**	4.74 **
(16)	project_age	0.01(0.00)***	44.17 ***	0.02(0.00)***	163.59 ***	0.03(0.00)***	229.88 ***	-0.01(0.00)***	13.17 ***	0.00(0.00)*	2.71 *
(17)	test_inclusion_open/close1	0.01(0.00)***	17.73 ***	0.00(0.00)*	3.80 *	0.01(0.00)***	13.85 ***	0.00(0.00)	1.21	0.02(0.00)***	30.84 ***
(18)	asserts_per_kloc	0.01(0.00)***	16.26 ***	-	-	-	-	-	-	-0.01(0.00)***	29.46 ***
(19)	contrib_genderMale	0.00(0.00)	0.20	0.01(0.00)***	18.86 ***	0.00(0.00)*	1.99 *	0.04(0.00)***	94.10 ***	0.03(0.00)***	100.81 ***
(20)	team_size	0.00(0.00)	0.00	-0.05(0.00)***	588.16 ***	-0.04(0.00)***	151.61 ***	-0.04(0.00)***	200.88 ***	-0.08(0.00)***	936.72 ***
(21)	has_comments1	-		0.47(0.00)***	80378.48 ***	-		0.36(0.00)***	21174.32 ***	0.41(0.00)***	33233.39 ***
(22)	same_user0	-		0.33(0.00)***	36133.44 ***	0.08(0.00)***	980.43 ***	0.21(0.00)***	6403.17 ***	-	
(23)	num_code_comments	-		0.11(0.00)***	17972.88 ***	-0.13(0.00)***	7267.37 ***	0.10(0.00)***	7218.64 ***	0.11(0.00)***	9580.65 ***
(24)	hash_tag1	-		0.15(0.00)***	7064.17 ***	0.16(0.00)***	5128.81 ***	0.17(0.00)***	5217.46 ***	0.16(0.00)***	4637.90 ***
(25)	reopen_or_not1	-		0.26(0.00)***	2239.82 ***	0.10(0.00)***	242.24 ***	0.22(0.01)***	870.18 ***	0.28(0.01)***	1678.87 ***
(26)	prior_review_num	-		0.01(0.00)***	87.49 ***	0.01(0.00)***	107.50 ***	0.00(0.00)***	10.49 ***	0.02(0.00)***	222.46 ***
(27)	test_lines_per_kloc	-		0.00(0.00)*	2.60 *	-0.01(0.00)***	9.66 ***	-0.02(0.00)***	48.92 ***	-	
(28)	first_response_time	-		-		0.44(0.00)***	164130.55 ***	-		-	
(29)	num_comments	-		-		0.47(0.00)***	87815.23 ***	-		-	
(30)	ci_latency	-		-		-		0.16(0.00)***	17307.41 ***	-	
(31)	ci_test_passed1	-		-		-		-0.23(0.00)***	9586.20 ***	-	
(32)	comment_conflict1	-		-		-		-		0.40(0.01)***	3226.00 ***
	nobs	2,413,390		2,413,439		1,251,905		1,112,605		1,368,236	
	R^2_{fn}	0.17		0.32		0.47		0.32		0.26	
	R^2_c	0.33		0.44		0.53		0.42		0.40	

Red color marks the top 5 factors in each situation

is just diverse to a certain extent, it can only guarantee a certain degree of diversity in each context.

Table 11 Results for comparing factors influencing pull request latency in different contexts. - means the factor is not included in the model

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
		Time context						*Pull request context*			
		before 1st June, 2016		*1st June, 2016-1st June, 2018*		*after 1st June, 2018*		*has_comments=1*		*has_comments=0*	
		Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq
(1)	(Intercept)	-0.52(0.01)***		-0.49(0.01)***		-0.41(0.02)***		-0.15(0.01)***		-0.22(0.01)***	
(2)	has_comments1	0.54(0.00)***	27024.46 ***	0.47(0.00)***	22226.89 ***	0.45(0.00)***	9537.36 ***	-		-	
(3)	same_user0	0.34(0.00)***	9807.64 ***	0.34(0.00)***	9884.86 ***	0.33(0.00)***	4579.98 ***	0.20(0.00)***	7477.58 ***	0.54(0.00)***	39546.20 ***
(4)	description_length	0.13(0.00)***	7233.40 ***	0.13(0.00)***	7253.72 ***	0.12(0.00)***	2823.24 ***	0.13(0.00)***	16223.52 ***	0.16(0.00)***	17414.14 ***
(5)	num_code_comments	0.11(0.00)***	5526.18 ***	0.11(0.00)***	5712.80 ***	0.11(0.00)***	2241.39 ***	-		-	
(6)	num_commits_close	0.09(0.00)***	2622.18 ***	0.10(0.00)***	2955.44 ***	0.10(0.00)***	1417.88 ***	0.19(0.00)***	23782.36 ***	0.10(0.00)***	4592.70 ***
(7)	integrator_availability	0.07(0.00)***	2516.23 ***	0.06(0.00)***	2028.36 ***	0.06(0.00)***	821.51 ***	0.08(0.00)***	6132.84 ***	0.08(0.00)***	4520.66 ***
(8)	src_churn_close	0.08(0.00)***	1821.71 ***	0.09(0.00)***	2332.21 ***	0.08(0.00)***	996.32 ***	0.14(0.00)***	12406.83 ***	0.07(0.00)***	2451.66 ***
(9)	open_pr_num	0.13(0.00)***	1570.13 ***	0.20(0.00)***	1388.25 ***	0.26(0.01)***	650.90 ***	0.19(0.00)***	4508.61 ***	0.17(0.00)***	3547.34 ***
(10)	hash_tag1	0.14(0.00)***	1442.20 ***	0.15(0.00)***	2140.90 ***	0.14(0.00)***	866.26 ***	-		-	
(11)	core_member1	-0.13(0.00)***	810.98 ***	-0.11(0.00)***	530.48 ***	-0.09(0.00)***	179.52 ***	-0.03(0.00)***	128.43 ***	-0.12(0.00)***	1022.80 ***
(12)	reopen_or_not1	0.24(0.01)***	469.40 ***	0.25(0.01)***	635.63 ***	0.25(0.01)***	319.84 ***	0.30(0.00)***	2623.68 ***	0.30(0.01)***	431.33 ***
(13)	ci_exists1	0.10(0.00)***	461.30 ***	0.16(0.00)***	717.84 ***	0.15(0.01)***	245.24 ***	0.06(0.00)***	308.57 ***	0.17(0.00)***	1882.87 ***
(14)	prev_pullreqs	-0.05(0.00)***	399.17 ***	-0.04(0.00)***	228.80 ***	-0.04(0.00)***	140.82 ***	-0.13(0.00)***	6972.60 ***	-0.02(0.00)***	104.49 ***
(15)	commits_on_files_touched_close	-0.02(0.00)***	217.50 ***	-0.03(0.00)***	249.85 ***	-0.03(0.00)***	132.54 ***	-0.03(0.00)***	716.28 ***	-0.03(0.00)***	565.75 ***
(16)	friday_effect1	0.05(0.00)***	175.50 ***	0.06(0.00)***	305.50 ***	0.06(0.00)***	138.76 ***	0.07(0.00)***	1034.33 ***	0.04(0.00)***	186.30 ***
(17)	followers	-0.02(0.00)***	167.76 ***	-0.01(0.00)***	22.10 ***	0.01(0.00)**	4.72 **	-0.02(0.00)***	224.10 ***	0.00(0.00)	0.21
(18)	prior_review_num	0.02(0.00)***	137.72 ***	0.01(0.00)***	10.29 ***	0.00(0.00)*	2.76 *	0.03(0.00)***	486.89 ***	0.04(0.00)***	587.32 ***
(19)	test_churn_close	0.02(0.00)***	115.71 ***	0.02(0.00)***	130.11 ***	0.02(0.00)***	49.67 ***	0.04(0.00)***	818.95 ***	0.03(0.00)***	412.94 ***
(20)	project_age	-0.03(0.00)***	80.65 ***	-0.16(0.00)***	2274.93 ***	-0.60(0.00)***	8963.48 ***	0.01(0.00)***	61.38 ***	0.03(0.00)***	200.11 ***
(21)	team_size	-0.03(0.00)***	74.44 ***	-0.09(0.00)***	409.34 ***	-0.06(0.01)***	50.45 ***	-0.06(0.00)***	501.34 ***	-0.02(0.00)***	78.14 ***
(22)	files_changed_close	-0.01(0.00)***	18.24 ***	-0.02(0.00)***	73.28 ***	-0.01(0.00)***	17.81 ***	-0.02(0.00)***	118.29 ***	-0.02(0.00)***	154.74 ***
(23)	sloc	-0.02(0.00)***	16.70 ***	-0.01(0.00)**	4.17 **	-0.02(0.01)**	4.93 **	-0.01(0.00)***	11.18 ***	-0.03(0.00)***	122.12 ***
(24)	contrib_genderMale	-0.02(0.00)***	9.25 ***	0.00(0.00)	0.31	0.02(0.00)***	9.40 ***	0.02(0.00)***	33.90 ***	-0.01(0.00).	2.40 .
(25)	test_inclusion_close1	0.01(0.00)**	5.20 **	0.00(0.00)	0.76	0.00(0.01)	0.10	0.02(0.00)***	49.70 ***	0.00(0.00)	0.00
(26)	test_lines_per_kloc	-0.01(0.00)*	3.48 *	-0.02(0.00)***	27.64 ***	0.01(0.01)	0.58	-0.02(0.00)***	71.70 ***	0.01(0.00)**	5.00 **
	nobs	628,827		683,034		332,046		1,407,427		1,006,012	
	R^2_{fn}	0.32		0.31		0.37		0.18		0.2	
	R^2_c	0.43		0.46		0.67		0.3		0.36	

Red color marks the factors with at least 5% of variance change across different contexts

Table 12 Results for comparing factors influencing pull request latency in different contexts. - means the factor is not included in the model

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
		Project context						*Tool context*			
		$team\_size \leq 4$		$4 < team\_size \leq 10$		$team\_size > 10$		ci exists		no ci	
		Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq
(1)	(Intercept)	-0.45(0.01)***		-0.47(0.01)***		-0.61(0.01)***		-0.40(0.01)***		-0.43(0.01)***	
(2)	has_comments1	0.46(0.00)***	37112.20 ***	0.48(0.00)***	27619.74 ***	0.49(0.00)***	27203.79 ***	0.46(0.00)***	60777.88 ***	0.51(0.00)***	19134.56 ***
(3)	same_user0	0.39(0.00)***	21264.21 ***	0.32(0.00)***	11840.65 ***	0.27(0.00)***	8130.93 ***	0.31(0.00)***	23586.66 ***	0.42(0.00)***	11974.70 ***
(4)	description_length	0.12(0.00)***	11504.22 ***	0.13(0.00)***	9027.68 ***	0.14(0.00)***	10182.40 ***	0.13(0.00)***	19837.71 ***	0.14(0.00)***	6071.56 ***
(5)	num_commits_close	0.11(0.00)***	6270.21 ***	0.12(0.00)***	4899.68 ***	0.10(0.00)***	3407.39 ***	0.12(0.00)***	12085.98 ***	0.08(0.00)***	1342.72 ***
(6)	integrator_availability	0.07(0.00)***	4163.59 ***	0.07(0.00)***	3678.56 ***	0.06(0.00)***	2361.51 ***	0.07(0.00)***	6846.15 ***	0.07(0.00)***	1782.75 ***
(7)	num_code_comments	0.07(0.00)***	4104.44 ***	0.1(0.00)***	5209.26 ***	0.15(0.00)***	12261.78 ***	0.12(0.00)***	16251.90 ***	0.09(0.00)***	2552.44 ***
(8)	open_pr_num	0.12(0.00)***	3375.73 ***	0.1(0.00)***	1482.83 ***	0.17(0.00)***	1833.10 ***	0.19(0.00)***	5479.78 ***	0.12(0.00)***	855.65 ***
(9)	hash_tag1	0.14(0.00)***	2947.39 ***	0.17(0.00)***	2933.73 ***	0.15(0.00)***	2767.72 ***	0.16(0.00)***	7148.17 ***	0.12(0.00)***	680.55 ***
(10)	src_churn_close	0.08(0.00)***	2622.45 ***	0.09(0.00)***	2897.87 ***	0.09(0.00)***	3227.73 ***	0.09(0.00)***	6598.17 ***	0.07(0.00)***	1035.85 ***
(11)	core_member1	-0.13(0.00)***	1377.26 ***	-0.1(0.00)***	659.70 ***	-0.04(0.00)***	97.23 ***	-0.08(0.00)***	907.11 ***	-0.14(0.00)***	889.73 ***
(12)	ci_exists1	0.10(0.00)***	715.16 ***	0.12(0.00)***	663.40 ***	0.16(0.00)***	1210.63 ***			-	
(13)	commits_on_files_touched_close	-0.03(0.00)***	567.43 ***	-0.04(0.00)***	559.78 ***	-0.03(0.00)***	388.77 ***	-0.03(0.00)***	1160.42 ***	-0.03(0.00)***	213.24 ***
(14)	reopen_or_not1	0.21(0.01)***	555.12 ***	0.25(0.01)***	636.78 ***	0.32(0.01)***	1559.68 ***	0.28(0.00)***	2188.86 ***	0.21(0.01)***	248.20 ***
(15)	prev_pullreqs	-0.04(0.00)***	348.33 ***	-0.04(0.00)***	332.61 ***	-0.08(0.00)***	1862.50 ***	-0.06(0.00)***	1981.65 ***	-0.05(0.00)***	404.70 ***
(16)	friday_effect1	0.04(0.00)***	295.62 ***	0.06(0.00)***	384.84 ***	0.06(0.00)***	400.28 ***	0.06(0.00)***	859.80 ***	0.04(0.00)***	90.20 ***
(17)	test_churn_close	0.03(0.00)***	267.15 ***	0.02(0.00)***	124.42 ***	0.02(0.00)***	146.69 ***	0.02(0.00)***	375.95 ***	0.01(0.00)***	39.84 ***
(18)	project_age	0.03(0.00)***	187.53 ***	0.02(0.00)***	35.97 ***	0.00(0.00)*	2.79 *	0.02(0.00)***	187.82 ***	0.02(0.00)***	29.78 ***
(19)	team_size	-0.01(0.00)***	67.30 ***	0.00(0.00)*	3.61 *	-0.04(0.00)***	156.36 ***	-0.04(0.00)***	288.42 ***	-0.03(0.00)***	52.82 ***
(20)	prior_review_num	0.01(0.00)***	57.51 ***	0.02(0.00)***	131.74 ***	0.01(0.00)***	26.89 ***	0.01(0.00)***	180.30 ***	0.00(0.00)**	4.52 **
(21)	sloc	-0.02(0.00)***	40.34 ***	0.00(0.00)	0.00.35	-0.04(0.00)***	84.98 ***	-0.02(0.00)***	44.57 ***	0.00(0.00)	0.81
(22)	followers	-0.01(0.00)***	28.41 ***	-0.01(0.00)***	23.68 ***	0.00(0.00)***	7.40 ***	-0.01(0.00)***	76.38 ***	-0.01(0.00)***	8.77 ***
(23)	files_changed_close	-0.01(0.00)***	12.35 ***	0.00(0.00)	0.00.13	-0.02(0.00)***	152.28 ***	-0.01(0.00)***	98.66 ***	0.00(0.00).	1.78 .
(24)	contrib_genderMale	0.01(0.00)***	7.67 ***	0.00(0.00)	0.00.21	0.02(0.00)***	23.69 ***	0.01(0.00)***	19.63 ***	0.00(0.00)	0.60
(25)	test_lines_per_kloc	-0.01(0.00)***	7.90 ***	0.00(0.00).	1.62 .	0.01(0.00)***	18.61 ***	0.00(0.00)	0.00.39	-0.01(0.00)*	2.83 *
(26)	test_inclusion_close1	-0.01(0.00)*	2.10 *	0.00(0.00)	1.70	0.01(0.00)***	11.72 ***	0.00(0.00)*	2.53 *	0.01(0.00).	1.82 .
	nobs	1,148,976		816,327		820,450		1,885,836		527,603	
	R^2_m	0.32		0.31		0.32		0.32		0.34	
	R^2_c	0.44		0.43		0.43		0.42		0.48	

Red color marks the factors with at least 5% of variance change across different contexts

4. This paper only considers the factors that can be measured on GitHub and only considers the factors that can be quantitatively measured. We did not verify the generality of the results on other platforms.

Table 13 Results for comparing factors influencing pull request latency in different contexts. - means the factor is not included in the model

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
		Process context						*Developer context*			
		Open time		*Close Time*		*Process change*		*same_user=1*		*same_user=0*	
		Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq	Coeffs (Err.)	Sum sq
(1)	(Intercept)	-0.04(0.01)***		-0.04(0.01)***		-0.05(0.01)***		-0.37(0.01)***		-0.30(0.01)***	
(2)	description_length	0.20(0.00)***	66429.42 ***	0.19(0.00)***	57108.13 ***	0.19(0.00)***	55769.31 ***	0.15(0.00)***	15112.30 ***	0.10(0.00)***	8960.29 ***
(3)	src_churn_open/close/change	0.16(0.00)***	29349.67 ***	0.13(0.00)***	17393.69 ***	0.01(0.00)***	158.96 ***	0.08(0.00)***	2776.68 ***	0.09(0.00)***	5117.35 ***
(4)	prev_pullreqs	-0.13(0.00)***	12717.35 ***	-0.12(0.00)***	10971.99 ***	-0.11(0.00)***	10006.45 ***	-0.02(0.00)***	191.66 ***	-0.09(0.00)***	3338.93 ***
(5)	integrator_availability	0.07(0.00)***	9570.64 ***	0.07(0.00)***	9546.17 ***	0.07(0.00)***	9921.81 ***	0.05(0.00)***	1788.96 ***	0.09(0.00)***	8657.59 ***
(6)	open_pr_num	0.20(0.00)***	9335.41 ***	0.20(0.00)***	9211.72 ***	0.20(0.00)***	9478.98 ***	0.11(0.00)***	1187.87 ***	0.22(0.00)***	7017.60 ***
(7)	core_member1	-0.17(0.00)***	5336.89 ***	-0.18(0.00)***	6071.37 ***	-0.15(0.00)***	4312.15 ***	-0.07(0.00)***	284.71 ***	-0.06(0.00)***	407.69 ***
(8)	ci_exists1	0.16(0.00)***	4106.24 ***	0.17(0.00)***	4321.54 ***	0.17(0.00)***	4319.72 ***	0.21(0.00)***	2548.77 ***	0.05(0.00)***	255.69 ***
(9)	test_churn_open/close/change	0.05(0.00)***	2309.25 ***	0.05(0.00)***	1731.12 ***	0.01(0.00)***	64.38 ***	0.02(0.00)***	113.22 ***	0.02(0.00)***	267.34 ***
(10)	commits_on_files_touched_open/close/change	-0.04(0.00)***	1734.60 ***	-0.03(0.00)***	878.62 ***	-0.01(0.00)***	278.68 ***	-0.02(0.00)***	205.44 ***	-0.04(0.00)***	1462.44 ***
(11)	followers	-0.03(0.00)***	905.21 ***	-0.02(0.00)***	709.43 ***	-0.02(0.00)***	585.84 ***	0.01(0.00)***	17.26 ***	-0.02(0.00)***	198.53 ***
(12)	friday_effect1	0.05(0.00)***	853.15 ***	0.05(0.00)***	848.75 ***	0.05(0.00)***	900.57 ***	0.05(0.00)***	302.56 ***	0.06(0.00)***	685.56 ***
(13)	files_changed_open/close/change	-0.02(0.00)***	374.92 ***	-0.05(0.00)***	1786.72 ***	0.02(0.00)***	433.30 ***	-0.02(0.00)***	179.91 ***	0.00(0.00)	0.80
(14)	num_commits_open/close/change	0.02(0.00)***	362.10 ***	0.17(0.00)***	35663.50 ***	0.28(0.00)***	65113.56 ***	0.11(0.00)***	5428.18 ***	0.12(0.00)***	8220.84 ***
(15)	sloc	-0.02(0.00)***	118.65 ***	-0.02(0.00)***	107.54 ***	-0.01(0.00)***	49.50 ***	-0.02(0.00)***	39.52 ***	-0.01(0.00)*	3.44 *
(16)	project_age	0.01(0.00)***	44.17 ***	0.01(0.00)***	63.33 ***	0.02(0.00)***	107.20 ***	0.05(0.00)***	440.30 ***	0.01(0.00)***	27.66 ***
(17)	test_inclusion_open/close/change1	0.01(0.00)***	17.73 ***	0.02(0.00)***	41.47 ***	0.00(0.00)	0.56	0.00(0.00)	0.50	0.02(0.00)***	30.50 ***
(18)	asserts_per_kloc	0.01(0.00)***	16.26 ***	0.00(0.00)**	6.89 **	0.00(0.00)*	3.42 *	0.03(0.00)***	51.57 ***	-0.01(0.00)**	6.33 **
(19)	contrib_genderMale	0.00(0.00)	0.20	0.00(0.00)*	3.59 *	0.01(0.00)***	21.51 ***	-0.01(0.00)**	4.95 **	0.03(0.00)***	102.39 ***
(20)	team_size	0.00(0.00)	0.00	0.00(0.00).	2.48 .	-0.01(0.00)***	48.21 ***	-0.01(0.00)***	16.20 ***	-0.08(0.00)***	980.45 ***
(21)	has_comments_open/close/change	-		-		-		0.54(0.00)***	43302.83 ***	0.41(0.00)***	33184.19 ***
(22)	num_code_comments	-		-		-		0.11(0.00)***	7329.60 ***	0.11(0.00)***	9633.27 ***
(23)	hash_tag1	-		-		-		0.14(0.00)***	2544.51 ***	0.16(0.00)***	4675.21 ***
(24)	comment_conflict1	-		-		-		0.42(0.01)***	1420.16 ***	0.40(0.01)***	3225.13 ***
(25)	reopen_or_not1	-		-		-		0.26(0.01)***	783.50 ***	0.28(0.01)***	1679.58 ***
(26)	test_lines_per_kloc	-		-		-		-0.01(0.00)***	10.14 ***	0.00(0.00)	1.34
	nobs	2,413,390		2,413,390		2,413,390		1,045,154		1,368,236	
	R^2_m	0.17		0.20		0.23		0.31		0.26	
	R^2_c	0.33		0.36		0.38		0.45		0.40	

Red color marks the factors with at least 5% of variance change across different contexts

5. When dividing pull requests for project and time contexts, we split the pull requests into three subsets of similar size. Previous studies (Soares et al. 2015; Soares et al. 2015) split the data into subsets using similar approaches for the trend analysis. We tried to ensure the differences in subsets and use as much data as possible to portray trends in different contexts. However, the choice of other thresholds can always give different results.
6. The GHTorrent dataset differs from the GitHub API regarding time-related data, affecting the factors collected in our dataset. We calculated the accuracy by randomly selecting 100 samples. The accuracy of factor *first_response_time*, *account_creation_days*, *project_age*, and *ci_latency* are 98%, 97%, 96%, and 94%, respectively. Our dataset inherits the errors from the previous dataset, but the percentage of errors is smaller (Tables 12 and 13).

8 Conclusion

This paper sorts out the influence of factors on pull request latency. Our study shows that the relatively importance of factors in different scenarios are different. For example, when the pull request is closed, the process-related factors are more important than the pull request description. At the same time, the importance of factors varies with context. For example, when considering the same set of factors, due to changes brought about by the review process, the number of commits in a pull request becomes more important when the pull request is closed. This paper can be helpful to follow-up research and practice. We created a dataset and open-sourced the script for model replication, which facilitates future research. Meanwhile, follow-up research can reuse our results and build up models according to different contexts and scenarios. Finally, we put forward actionable suggestions for pull request contributors and reviewers. For example, contributors should refine the text description when submitting a pull request, and reviewers should respond to contributions as quickly as possible.

Acknowledgements This work is supported by National Grand R&D Plan (Grant No.2020AAA0103504).

References

Altaleb A, Altherwi M, Gravell A (2020) An industrial investigation into effort estimation predictors for mobile app development in agile processes. In: 2020 9th international conference on industrial technology and management (ICITM). IEEE, pp 291–296

Atkins M (2012) Gerrit code review, or github's fork and pull model?. https://softwareengineering.stackexchange.com/questions/173262/gerrit-code-review-or-githubs-fork-and-pull-model [Online; accessed 4-November-2021]

Bacchelli A, Bird C (2013) Expectations, outcomes, and challenges of modern code review. In: 2013 35th international conference on software engineering (ICSE). IEEE, pp 712–721

Baysal O, Kononenko O, Holmes R, Godfrey MW (2012) The secret life of patches: A firefox case study. In: 2012 19th working conference on reverse engineering, pp 447–455

Baysal O, Kononenko O, Holmes R, Godfrey MW (2016) Investigating technical and non-technical factors influencing modern code review. Empir Softw Eng 21(3):932–959. https://doi.org/10.1007/s10664-015-9366-8

Bernardo JH, Alencar da Costa D, Kulesza U (2018) Studying the impact of adopting continuous integration on the delivery time of pull requests. In: 2018 IEEE/ACM 15th international conference on mining software repositories (MSR), pp 131–141

Bernhart M, Mauczka A, Grechenig T (2010) Adopting code reviews for agile software development. In: 2010 Agile Conference. IEEE, pp 44–47

Bosu A, Carver JC (2014) Impact of developer reputation on code review outcomes in oss projects: An empirical investigation. In: Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement, Association for Computing Machinery, ESEM '14, New York, NY, USA. https://doi.org/10.1145/2652524.2652544

Cassee N, Kitsanelis C, Constantinou E, Serebrenik A (2021) Human, bot or both? a study on the capabilities of classification models on mixed accounts. In: 2021 IEEE international conference on software maintenance and evolution (ICSME). IEEE, pp 654–658

Cohen J (1969) Statistical power analysis for the behavioral sciences. Academic Press, Cambridge

Cohen J, Cohen P, West SG, Aiken LS (2013) Applied multiple regression/correlation analysis for the behavioral sciences. Routledge, London

Dasheng X, Shenglan H (2012) Estimation of project costs based on fuzzy neural network. In: 2012 World congress on information and communication technologies. IEEE, pp 1177–1181

Dey T, Mousavi S, Ponce E, Fry T, Vasilescu B, Filippova A, Mockus A (2020) Detecting and characterizing bots that commit code. In: Proceedings of the 17th international conference on mining software repositories, pp 209–219

Ecplise (2011) Mylyn reviews. https://projects.eclipse.org/projects/mylyn.reviews [Online; accessed 4-November-2021]

Fan Q, Yu Y, Yin G, Wang T, Wang H (2017) Where is the road for issue reports classification based on text mining? In: 2017 ACM/IEEE international symposium on empirical software engineering and measurement (ESEM). IEEE, pp 121–130

Gerrit (2013) Gerritforge blog - git and gerrit code review supported and delivered to your enterprise. https://gitenterprise.me/2013/10/17/gerrit-code-review-or-githubs-fork-and-pull-take-both/ [Online; accessed 4-November-2021]

Gerrit (2021) Gerrit code review. https://www.gerritcodereview.com/ [Online; accessed 4-November-2021]

Golzadeh M, Decan A, Legay D, Mens T (2021) A ground-truth dataset and classification model for detecting bots in github issue and pr comments. J Syst Softw 175:110911

Golzadeh M, Decan A, Mens T (2021) Evaluating a bot detection model on git commit messages. arXiv:2103.11779

Gousios G, Zaidman A, Storey M, V Deursen A (2015) Work practices and challenges in pull-based development: The integrator's perspective. In: 2015 IEEE/ACM 37th IEEE international conference on software engineering, vol 1, pp 358–368

Gousios G, Pinzger M, Deursen A (2014) An exploratory study of the pull-based software development model. In: Proceedings of the 36th international conference on software engineering, Association for Computing Machinery, New York, NY, USA, ICSE 2014, pp 345–355. https://doi.org/10.1145/2568225.2568260

Hall DB (2009) Data analysis using regression and multilevel/hierarchical models. J Am Stat Assoc

Hechtl C (2020) On the influence of developer coreness on patch acceptance: A survival analysis

Hilton M, Tunnell T, Huang K, Marinov D, Dig D (2016) Usage, costs, and benefits of continuous integration in open-source projects. In: 2016 31st IEEE/ACM international conference on automated software engineering (ASE), pp 426–437

Hu D, Wang T, Chang J, Zhang Y, Yin G (2018) Bugs and features, do developers treat them differently? 250–255

Imtiaz N, Middleton J, Chakraborty J, Robson N, Bai G, Murphy-Hill E (2019) Investigating the effects of gender bias on github. In: 2019 IEEE/ACM 41st international conference on software engineering (ICSE), pp 700–711

Jalali S, Wohlin C (2012) Systematic literature studies: Database searches vs. backward snowballing. In: Proceedings of the 2012 ACM-IEEE international symposium on empirical software engineering and measurement, pp 29–38

Jiang J, Mohamed A, Zhang L (2019) What are the characteristics of reopened pull requests? a case study on open source projects in github. IEEE Access 7:102751–102761

Jiang J, Yang Y, He J, Blanc X, Zhang L (2017) Who should comment on this pull request? analyzing attributes for more accurate commenter recommendation in pull-based development. Inf Softw Technol 84:48–62

Jiang Y, Adams B, German DM (2013) Will my patch make it? and how fast? case study on the linux kernel. In: 2013 10th Working conference on mining software repositories (MSR), pp 101–110

Jing J, Yun Y, He J, Blanc X, Li Z (2017) Who should comment on this pull request? analyzing attributes for more accurate commenter recommendation in pull-based development - sciencedirect. Inform Softw Technol 84(C):48–62

Jones JS (2019) Learn to use the eta coefficient test in spss with data from the niosh quality of worklife survey (2014)
Jørgensen M (2011) Contrasting ideal and realistic conditions as a means to improve judgment-based software development effort estimation. Inf Softw Technol 53(12):1382–1390
Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering
Kocaguneli E, Misirli AT, Caglayan B, Bener A (2011) Experiences on developer participation and effort estimation. In: 2011 37th EUROMICRO conference on software engineering and advanced applications. IEEE, pp 419–422
Kononenko O, Rose T, Baysal O, Godfrey M, Theisen D, de Water B (2018) Studying pull request merges: A case study of shopify's active merchant. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, Association for Computing Machinery, New York, NY, USA, ICSE-SEIP '18, pp 124–133. https://doi.org/10.1145/3183519.3183542
Langsrud O (2003) Anova for unbalanced data: Use type ii instead of type iii sums of squares. Kluwer Academic Publishers, Amsterdam
Lee A, Carver JC (2017) Are one-time contributors different? a comparison to core and periphery developers in floss repositories. In: 2017 ACM/IEEE international symposium on empirical software engineering and measurement (ESEM), pp 1–10
Lenarduzzi V (2015) Could social factors influence the effort software estimation? In: Proceedings of the 7th international workshop on social software engineering, pp 21–24
Li Z, Yu Y, Wang T, Yin G, Wang H (2021) Are you still working on this an empirical study on pull request abandonment. IEEE Trans Softw Eng PP(99):1–1
Liu Z, Xia X, Treude C, Lo D, Li S (2019) Automatic generation of pull request descriptions. In: 2019 34th IEEE/ACM international conference on automated software engineering (ASE). IEEE, pp 176–188
Maddila C, Bansal C, Nagappan N (2019) Predicting pull request completion time: a case study on large scale cloud services. In: Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering, pp 874–882
Maddila C, Upadrasta SS, Bansal C, Nagappan N, Gousios G, van Deursen A (2020) Nudge: Accelerating overdue pull requests towards completion. arXiv:2011.12468
McIntosh S, Kamei Y, Adams B, Hassan AE (2014) The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects. In: Proceedings of the 11th working conference on mining software repositories, pp 192–201
Minku LL, Yao X (2011) A principled evaluation of ensembles of learning machines for software effort estimation. In: Proceedings of the 7th international conference on predictive models in software engineering, pp 1–10
Nakagawa S, Schielzeth H (2013) A general and simple method for obtaining r2 from generalized linear mixed-effects models. Methods in Ecology and Evolution 4(2):133–142
Overney C, Meinicke J, Kstner C, Vasilescu B (2020) How to not get rich: an empirical study of donations in open source. In: ICSE '20: 42nd international conference on software engineering
Pinto G, Dias LF, Steinmacher I (2018) Who gets a patch accepted first? comparing the contributions of employees and volunteers. In: Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering. Association for Computing Machinery, New York, NY, USA, CHASE '18, pp 110–113. https://doi.org/10.1145/3195836.3195858
Sadowski C, Söderberg E, Church L, Sipko M, Bacchelli A (2018) Modern code review: A case study at google. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, Association for Computing Machinery, New York, NY, USA, ICSE-SEIP '18, pp 181–190. https://doi.org/10.1145/3183519.3183525
Sehra SK, Brar YS, Kaur N, Sehra SS (2017) Research patterns and trends in software effort estimation. Inf Softw Technol 91:1–21
Singh D, Sekar VR, Stolee KT, Johnson B (2017) Evaluating how static analysis tools can reduce code review effort. In: 2017 IEEE symposium on visual languages and human-centric computing (VL/HCC)
Soares DM, DLJúnior ML, Murta L, Plastino A (2015) Rejection factors of pull requests filed by core team developers in software projects with high acceptance rates. In: 2015 IEEE 14th international conference on machine learning and applications (ICMLA), pp 960–965
Soares DM, de Lima Júnior ML, Murta L, Plastino A (2015) Acceptance factors of pull requests in open-source projects. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, Association for Computing Machinery, New York, NY, USA, SAC '15, pp 1541–1546. https://doi.org/10.1145/2695664.2695856
Trendowicz A, Jeffery R (2014) Software project effort estimation. Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags 12:277–293

Tsay J, Dabbish L, Herbsleb J (2014) Influence of social and technical factors for evaluating contribution in github. In: Proceedings of the 36th International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, ICSE 2014, pp 356–366. https://doi.org/10.1145/2568225.2568315

v. d. Veen E, Gousios G, Zaidman A (2015) Automatically prioritizing pull requests. In: 2015 IEEE/ACM 12th working conference on mining software repositories, pp 357–361

Vogel L (2020) Gerrit code review - tutorial. https://www.vogella.com/tutorials/Gerrit/article.html [Online; accessed 4-November-2021]

Wang F, Yang X, Zhu X, Chen L (2009) Extended use case points method for software cost estimation. In: 2009 International conference on computational intelligence and software engineering. IEEE, pp 1–5

Wang Q, Xu B, Xia X, Wang T, Li S (2019) Duplicate pull request detection: When time matters. In: Proceedings of the 11th Asia-Pacific Symposium on Internetware, pp 1–10

Wessel M, De Souza BM, Steinmacher I, Wiese IS, Polato I, Chaves AP, Gerosa MA (2018) The power of bots: Characterizing and understanding bots in oss projects. Proceedings of the ACM on Human-Computer Interaction 2(CSCW):1–19

Yu Y, Wang H, Filkov V, Devanbu P, Vasilescu B (2015) Wait for it: Determinants of pull request evaluation latency on github. In: 2015 IEEE/ACM 12th working conference on mining software repositories, pp 367–371

Yu Y, Wang H, Yin G, Ling CX (2014) Who should review this pull-request: Reviewer recommendation to expedite crowd collaboration. In: 2014 21st Asia-Pacific software engineering conference, vol 1, pp 335–342

Yu Y, Li Z, Yin G, Wang T, Wang H (2018) A dataset of duplicate pull-requests in github. In: Proceedings of the 15th International Conference on Mining Software Repositories, pp 22–25

Yu Y, Yin G, Wang T, Yang C, Wang H (2016) Determinants of pull-based development in the context of continuous integration. Sci China Inform Sci 59(8):080104. https://doi.org/10.1007/s11432-016-5595-8

Zampetti F, Bavota G, Canfora G, Penta MD (2019) A study on the interplay between pull request review and continuous integration builds. In: 2019 IEEE 26th international conference on software analysis, evolution and reengineering (SANER), pp 38–48

Zhang T, Song M, Kim M (2014) Critics: An interactive code review tool for searching and inspecting systematic changes. In: Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering, pp 755–758

Zhang X, Rastogi A, Yu Y (2020) On the shoulders of giants: A new dataset for pull-based development research. In: Proceedings of the 17th international conference on mining software repositories, pp 543–547

Zhang X, Rastogi A, Yu Y (2020) Technical Report. https://github.com/zhangxunhui/new_pullreq_msr2020/blob/master/technical_report.pdf [Online; accessed 3-March-2021]

Zhang X, Yu Y, Gousios G, Rastogi A (2021) Pull request decision explained: An empirical overview

Zhang X, Yu Y, Wang T, Rastogi A, Wang H (2021) Dataset for ESE submission "Pull Request Latency Explained: An Empirical Overview". https://doi.org/10.5281/zenodo.5105117

Zhang Y, Yin G, Yu Y, Wang H (2014) A exploratory study of@-mention in github's pull-requests. In: 2014 21st Asia-Pacific software engineering conference, vol 1. IEEE, pp 343–350

Zhao Y, Serebrenik A, Zhou Y, Filkov V, Vasilescu B (2017) The impact of continuous integration on other software development practices: A large-scale empirical study. In: 2017 32nd IEEE/ACM international conference on automated software engineering (ASE), pp 60–71

Affiliations

Xunhui Zhang[1] · Yue Yu[1] · Tao Wang[1] · Ayushi Rastogi[2] · Huaimin Wang[1]

Xunhui Zhang
zhangxunhui@nudt.edu.cn

Tao Wang
taowang2005@nudt.edu.cn

Ayushi Rastogi
a.rastogi@rug.nl

Huaimin Wang
whm_w@163.com

[1] National University of Defense Technology, Changsha, China

[2] University of Groningen, Groningen, The Netherlands