An Empirical Study on GitHub Pull Requests' Reactions

MOHAMED AMINE BATOUN, École de Technologie Supérieure, Canada
KA LAI YUNG, Queen's University, Canada
YUAN TIAN, Queen's University, Canada
MOHAMMED SAYAGH, École de Technologie Supérieure, Canada

The pull request mechanism is commonly used to propose source code modifications and get feedback from the community before merging them into a software repository. On GitHub, practitioners can provide feedback on a pull request by either commenting on the pull request or simply reacting to it using a set of pre-defined GitHub reactions, i.e., "Thumbs-up", "Laugh", "Hooray", "Heart", "Rocket", "Thumbs-down", "Confused" and "Eyes". While a large number of prior studies investigated how to improve different software engineering activities (e.g., code review and integration) by investigating the feedback on pull requests, they focused only on pull requests' comments as a source of feedback. However, the GitHub reactions, according to our preliminary study, contain feedback that is not manifested within the comments of pull requests. In fact, our preliminary analysis of six popular projects shows that a median of 100% of the practitioners who reacted to a pull request did not leave any comment suggesting that reactions can be a unique source of feedback to further improve the code review and integration process.

To help future studies better leverage reactions as a feedback mechanism, we conduct an empirical study to understand the usage of GitHub reactions and understand their promises and limitations. We investigate in this paper how reactions are used, when and who use them on what types of pull requests and for what purposes. Our study considers a quantitative analysis on a set of 380k reactions on 63k pull requests of six popular open-source projects on GitHub and three qualitative analyses on a total number of 989 reactions from the same six projects. We find that the most common used GitHub reactions are the positive ones (i.e., "Thumbs-up", "Hooray", "Heart", "Rocket" and "Laugh"). We observe that reactors use positive reactions to express positive attitude (e.g., approval, appreciation, and excitement) on the proposed changes in pull requests. A median of just 1.95% of the used reactions are negative ones, which are used by reactors who disagree with the proposed changes for six reasons, such as feature modifications that might have more downsides than upsides or the use of the wrong approach to address certain problems. Most (a median of 78.40%) reactions on a pull request come before the closing of the corresponding pull requests. Interestingly, we observe that non-contributors (i.e., outsiders who potentially are the "end-users" of the software) are also active on reacting to pull requests. On top of that, we observe that core contributors, peripheral contributors, casual contributors and outsiders have different behaviors when reacting to pull requests. For instance, most core contributors react in the early stages of a pull request, while peripheral contributors, casual contributors and outsiders react around the closing time or, in some cases, after a pull request is merged. Contributors tend to react to the pull request's source code, while outsiders are more concerned about the impact of the pull request on the end-user experience. Our findings shed light on common patterns of GitHub reactions usage on pull requests and provide taxonomies about the intention of reactors, which can inspire future studies better leverage pull requests' reactions.

CCS Concepts: • **Software and its engineering → Collaboration in software development**.

Authors' addresses: Mohamed Amine Batoun, mohamed-amine.batoun.1@ens.etsmtl.ca, École de Technologie Supérieure, 1100 Notre-Dame St W, Montreal, Quebec, Canada, H3C 1K3; Ka Lai Yung, Queen's University, 99 University Ave, Kingston, Ontario, Canada, kyung@cs.queensu.ca; Yuan Tian, Queen's University, 99 University Ave, Kingston, Ontario, Canada, y.tian@queensu.ca; Mohammed Sayagh, mohammed.sayagh@etsmtl.ca, École de Technologie Supérieure, 1100 Notre-Dame St W, Montreal, Quebec, Canada, H3C 1K3.

Manuscript submitted to ACM

Additional Key Words and Phrases: GitHub Reactions, Pull Requests, Software Collaboration, Feedback

ACM Reference Format:
Mohamed Amine Batoun, Ka Lai Yung, Yuan Tian, and Mohammed Sayagh. 2018. An Empirical Study on GitHub Pull Requests' Reactions. 1, 1 (August 2018), 36 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Developers use the pull request mechanism to get feedback about their changes before integrating them into an existing software system. Such a mechanism is commonly adopted by open-source and commercial software projects and supported by project hosting sites such as GitHub. In particular, on GitHub, developers fork a project, make a contribution, then submit it through a pull request to receive feedback from other developers or users. Such a feedback can be in the form of a code review, a discussion around the possible changes or a comment to describe the general impression about the pull request.

However, in many cases, only a simple reaction is used as a sort of feedback. On GitHub specifically, developers can react to a pull request with one of the eight reactions shown in Figure 1. We refer to these reactions as *pull request reactions*. According to De Zoysa et al. [26], the GitHub reactions "Thumbs-up", "Laugh", "Heart", "Hooray" and "Rocket" are positive reactions, "Eyes" is a neutral reaction, while "Thumbs-down" and "Confused" are considered as negative reactions. Figure 2 shows an example of a pull request that received a total of 493 reactions, out of which 464 are positive reactions and 25 are neutral. On the other hand, four people negatively reacted by giving a "Confused" reaction.

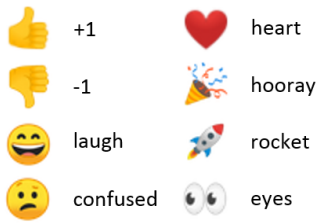


Fig. 1. Eight types of GitHub reactions.

The feedback provided to pull requests is leveraged by researchers to improve different software engineering activities, including the pull request mechanism itself. For instance, researchers have shown that feedback provided via pull request comments can be leveraged to improve software project management via supporting pull request acceptance prediction [37], pull request reviewer recommendation [41, 42] and contributor behaviour prediction [24]. However, the literature focuses mainly on the feedback provided via comments to pull requests and yet does not use the GitHub reactions (introduced in 2016).

In fact, reactions to pull requests can be a new opportunity for researchers to further investigate the feedback that is not manifested in the comments of a pull request to further improve the review and integration mechanism. According to our preliminary analysis of six popular projects, the GitHub pull requests reactions hold information that does not exist in a pull request's comments, which are studied by prior work [24, 37, 41, 42]. For instance, we observe from our preliminary analysis that a median of 100% of the reactors (i.e., practitioners who react to a pull request) over the six projects do not leave any comment in pull requests, which indicates that the pull request reactions represent a unique dataset about the feedback of practitioners on pull requests and that can be further explored to improve the feedback

mechanisms of a software system. Furthermore, a recent study [35] advocates that while GitHub reactions reduce the amount of noisy comments on a pull request, there is a need to investigate the intentions behind the usage of reactions.

Fig. 2. Example of reactions to a pull request on GitHub.

To help future studies better leverage pull request reactions, we conduct a first empirical study on six popular open source projects (Cataclysm, Julia, Laravel, Node, RPCS3, and Rust) to better understand the promises of using GitHub reactions and their limitations. To do so, our empirical study focuses on answering the 5Ws (Who, What, When, Where, and Why). In particular, we study *what* GitHub reactions are used and *why*, *who* use pull request Reactions, *When* pull request reactions are used according to the lifespan of pull requests, and on *what type of pull requests* (i.e., referring to the location in which reactions are added - the where of 5Ws) practitioners react to. Such an empirical understanding can help future researchers better understand the pull request reactions as well as their limitations. Answering the 5Ws is made according to the following research questions, which summarize our contribution.

RQ1: What are the most commonly used GitHub reactions?
In this research question, we investigate which type of reaction is the most popular, and why practitioners use each of them. We observe that our studied projects' reactors mostly use positive reactions as the median percentage of positive reactions over the six projects is 95.71%. Via two qualitative analyses of 616 reactions, we observe that reactors tend to use positive reactions to express positive attitude (e.g., approval, appreciation and excitement) about a pull request's change. New features can also receive negative reactions when they have more downsides than upsides. Similarly, wrong fixes or fixes causing regression issues also receive negative reactions.

RQ2: Who reacts to pull requests?
In this research question we classify reactors into five categories based on the number of commits they have in a project at the time of their reaction, and the degree of authorship of source files: *TF-authors*, *core contributors*, *peripheral contributors*, *casual contributors* and *outsiders*. Interestingly, we find that outsiders are actively engaged in pull request reactions. They have even more reactions in five out of our six studied projects than contributors.

In fact, a median percentage of 81.36% of the reactions are made by outsiders in Julia, Laravel, Node, RPCS3 and Rust, while in Cataclysm-DDA outsiders' reactions represent 36.38% of the total reactions. On the other hand, an outsider reacts to significantly less pull requests than contributors. Via a qualitative analysis of 373 reactions, we find that contributors are most likely to react to the code of the pull request, while outsiders' reactions are more related to the end user experience.

RQ3: When do practitioners react to pull requests?

We investigate at which stage of a pull request's lifespan do people react, then we study the differences between the reaction times of the five categories of reactors (TF-authors, core contributors, peripheral contributors, casual contributors and outsiders). A median of 53.57% of the reactions are added as soon as the first 25% of the lifespan of the pull requests, and a median of 78.40% of the reactions are added before the closing time of the pull requests. We observe that practitioners, specifically peripheral and casual contributors alongside outsiders, can still react to a pull request after closing it.

Our results suggest future researchers who want to leverage GitHub reactions to (1) be aware that negative reactions are less popular than positive ones since practitioners rarely react negatively to a pull request. (2) We recommend future studies to leverage reactions for different software engineering aspects, such as source code refactoring, as reactions are not limited to specific types of pull requests. We advocate future studies leveraging reactions on pull requests to (3) not weigh the reactions coming from outsiders and contributors the same way, as outsiders and contributors can have different intentions behind their reactions, and (4) leverage merged or closed pull requests to assess the reactions made by contributors as well as outsiders, since outsiders might still react to a pull request even after its closing time.

To facilitate the replication package of our work, we provide a replication package with all of our data and scripts [1]. Our replication package also contains a set of 8,004 projects and their number of reactions on their respective last 1,000 pull requests, and the percentage of pull requests with at least one reaction. These 8,004 are the projects that have been updated within the last year (before September 2022), have source code files, and are among the top 10k most popular projects.

We summarise our main contributions as follows:

- We shed light on the importance of leveraging GitHub reactions in future studies as they can be used as a unique source of data to capture practitioners' perceptions of pull requests.
- We conduct a first empirical study based on qualitative and quantitative analyses to understand the use of GitHub reactions, and provide insights into how different categories of reactors (e.g, *TF-authors*, *core contributors*, *peripheral contributors*, *casual contributors* and *outsiders*) leverage such reactions, which can be used to improve the pull request mechanism.
- We define a set of promises and limitations that researchers need to consider when leveraging GitHub reactions. These promises and limitations can provide a clear guideline for researchers, and help them avoid bias and ensure that the results they obtained using GitHub reactions are reliable.
- We provide a dataset of projects that use GitHub reactions in their pull requests, allowing future researchers to replicate our study and analyze the data to improve the pull request mechanism.

The rest of the paper is structured as follows: Section 2 explains our research methodology. Section 3 discusses the results of our preliminary study. Section 4 answers the three research questions. Section 5 lists various promises and

[1] https://doi.org/10.5281/zenodo.7665265

Table 1. Key statistics in the studied projects. Note that the number of commits and contributors are obtained from the Git log

Project Name	Project Description	PRs with Reactions	Reactions	Commits	Contributors
Cataclysm	Open-source game	17,706	94,157	100,828	1,833
Julia	Programming language for financial and scientific calculations	9,263	54,172	60,108	1,405
Laravel	Open-source PHP web framework	1,044	11,570	6,783	723
Node	Open-source cross-platform JavaScript runtime environment	16,596	80,075	65,950	3,508
RPCS3	Open-source game console emulator and debugger	2,700	34,260	14,257	197
Rust	High performance and safety programming language	16,368	106,071	176,154	4,451

limitations of the paper. Section 6 discusses related work. Section 7 presents the threats to validity of our findings. Finally, Section 7 concludes the paper.

2 DATA COLLECTION

To answer the three proposed research questions in Section 1, we conduct an empirical study on six open-source projects that satisfy the following criteria: (1) projects use a large number of reactions so we can have enough data to answer our research questions, (2) projects should be popular and actively maintained projects so we do not study toy projects or dead projects that are not maintained anymore. Our studied projects are shown in Table 1. To obtain our six projects that satisfy our two criteria, we select from the top 10,000 most starred GitHub repositories projects that have been updated for a year and are software repositories (i.e., have no source code files). We end up with 8,005 projects. Since from this last dataset some projects have just one pull request, we eliminate projects with less than 1,000 pull requests. From 1,852 remaining projects, we select the six case studies that are shown in Table 1. These six projects are among the top 200 projects according to the number of pull requests' reactions and number of pull requests with at least one reaction (i.e., in the last 1,000 pull requests of these projects). Overall, our projects have at least 1,044 pull requests with one reaction, a minimum of 11,570 reactions and they have a large number of commits (minimum of 6,783) and at least 197 contributors, which suggests that our studied projects use reactions on their pull requests and are not toy projects. That said, we do not generalize our results to other software systems as further discussed in our threats to validity.

For each of the selected six projects, we collect all the pull requests with each of their associated reactions using the GitHub v4 API [2]. Our pull requests data covers a period that spans over six years, starting from March 2016 (i.e., the time at which GitHub introduced the reactions) to July 2022. To answer our research questions, we collect the metrics that are shown in Table 2. We collect data about the pull requests, the reactions on each pull requests, the comments, as well as the source code commits that are made throughout the whole history to measure the expertise of reactors, which we further discuss in our approach of RQ2. Table 2 summarizes our metrics and on which research question we use them. We further discuss in details the approach to answer each of our research questions in their related sections.

[2] https://docs.github.com/en/graphql/overview/about-the-graphql-api

Table 2. Collected metrics

Dimension	Metric	Usage	Description
Pull Request	PRdescription	RQ1-2	the content of the title and description of the pull request
	PRstart	RQ3	the opening time of the pull request
	PRend	RQ3	the closing time of the pull request
	PRauthor	RQ2	the author of the pull request
	PRtype	RQ1-2-3	the type of the pull request (e.g., "bug", "feature"), which we identify as discussed in the approach of RQ1
Reaction	reactType	RQ1-2-3	the reaction on the pull request (e.g.,"Thumbs-Up", "Hooray"...)
	reactAuthor	PQ2, RQ2-3	the author of the reaction (GitHub username)
	reactTime	RQ3	the timestamp specifying when the reaction was made
	relativeReactTime	RQ3	the time of the reaction on a pull request according to its creation and closing time
Comment	comment	RQ1-2	the content of a comment on the pull request
	commentAuthor	PQ2, RQ1-2	the author of the comment
Commit	commitID	RQ1-2	the ID of the commit
	commitMessage	RQ1	the message of the commit
	commitAuthor	RQ2	the author of the commit (Git username)
	changedFiles	RQ1	the files that were changed by the commit

3 PRELIMINARY STUDY

The goal of our preliminary study is to investigate how prevalent are pull request reactions and whether they hold exclusive information, which are not available in the pull request comments. To do so, we address the following preliminary research question:

PQ1. Do reactors comment on a pull request?

Motivation: This preliminary question explores whether developers would use reactions without providing written expressions in the form of comments and whether users provide a feedback via a comment alongside a reaction. The answer to this question would reveal how exclusive are both mechanisms (i.e., reactions and comments) in providing feedback on pull requests, hence there is a need to consider both reactions and comments as community feedback on pull requests to gain a complete picture. For instance, if the reactors do not always leave a comment, their opinions on the pull requests would not be captured by analyzing pull requests' comments and vice-versa.

Approach: To answer this preliminary research question, for each collected pull request with at least one reaction in the six target projects, we identify all reactors and investigate the percentage of reactors who do not leave a comment on the same pull request. We also identify the percentage of commenters to a pull request who did not react to the same pull request. Then we study the distributions of such percentages in each project. For instance, if a pull request contains five reactions given by five GitHub users and three of the five reactors leave at least one comment in the pull request while the other two do not, then the percentage of reactors that do not comment is 40% (two out of five).

Results: Most of the reactors do not leave any comments. The median percentage of such reactors in pull requests with reaction(s) is 100% in all six studied projects, as shown in Figure 3. Moreover, the percentages of pull requests which have at least one reactor who did not comment are 97.67%, 89.97%, 98.93%, 96.76%, 98.17% and 91.35% for Cataclysm, Julia, Laravel, Node, RPCS3 and Rust, respectively. We also observe 78.46%, 62.73%, 90.49%, 85.44%, 71.17%

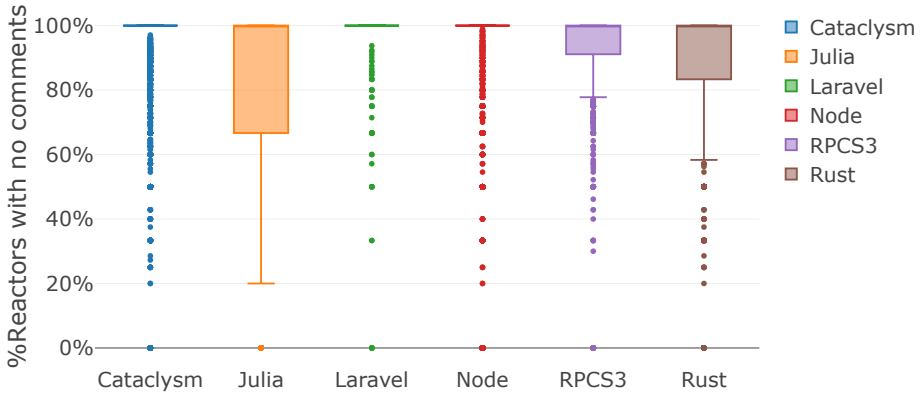


Fig. 3. Distribution of reactors who do no leave comments in pull requests of our six studied projects.

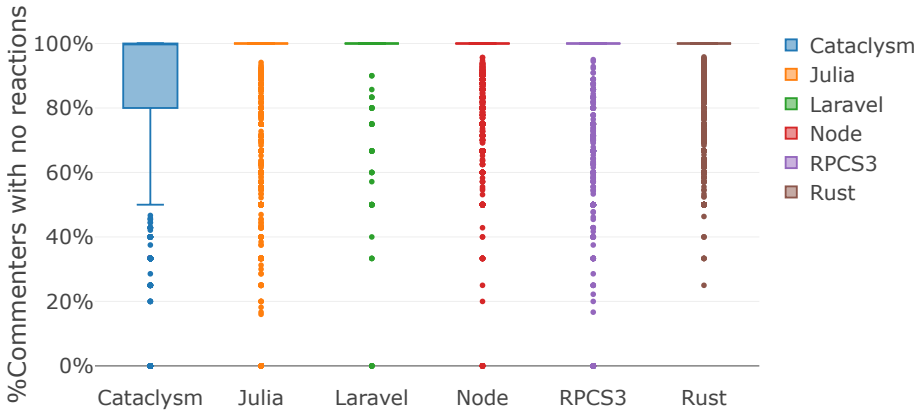


Fig. 4. Distribution of commenters who do no react to pull requests of our six studied projects.

and 68.49% of the pull requests where all of the reactors did not leave any comments at all. Just as designed [3], GitHub reactions enable people to express feedback without providing textual comments, and therefore reduce the noise in the pull request discussions.

On the other side, most of the commenters do not use reactions either. As shown in Figure 4, a median of 100% of the commenters do not react to pull requests. A median of 95.47%, 98.28%, 98.34%, 99.90%, 96.68% and 100.00% of the pull requests have at least one commenter who did not react for the Cataclysm, Julia, Laravel, Node, RPCS3 and Rust projects. Finally, a median of 72.21%, 82.29%, 87.16%, 93.92%, 78.40% and 93.01% of the above projects' pull requests contain no commenter reaction, i.e., all their commenters did not react to the pull request.

Similarly, most practitioners who comment on the follow-up discussion of pull requests do not use reactions, and half of the reactors do not leave comments. We find that a median of 100% of the commenters do not use reactions. Specifically, a median of 99.99%, 98.57%, 99.68%, 99.99%, 99.30% and 99.99% of the pull requests have at least one commenter who did not react for Cataclysm, Julia, Laravel, Node, RPCS3 and Rust, respectively. We also observe 74.60%, 68.45%, 79.55%, 67.82%, 78.84% and 74.10% of the pull requests where all of the commenters did not react at all. On the other hand, a median of 53.57% of the reactors do not comment. The percentages of pull requests which

have at least one reactor who did not comment are 69.23%, 66.59%, 96.92%, 52.15%, 81.78% and 58.87% for Cataclysm, Julia, Laravel, Node, RPCS3 and Rust, respectively. We also observe 33.47%, 31.39%, 80.76%, 21.80%, 41.14% and 23.91% of the pull requests where all of the reactors did not leave any comments at all.

Summary of the preliminary analysis

Pull request reactions represent a promising dataset to capture practitioners' perception about pull requests, as their reactions may capture exclusive information that are not in the comments of a pull request (i.e., practitioners often react without leaving any comment). Before leveraging pull request reactions, our finding motivates the need to first understand the usage of GitHub reactions and how to interpret them.

4 RESULTS

The goal of our empirical study is to understand who leverage GitHub reactions, when, how, and why they do so, as well as which reactions are used and on what type of pull requests. In particular, we address the following research questions:

- RQ1. What are the most commonly used GitHub reactions?
- RQ2. Who reacts to pull requests?
- RQ3. When do practitioners react to pull requests?

RQ1. What are the most commonly used GitHub reactions?

Motivation: The goal of this research question is to quantitatively identify what type of reactions (e.g., positive or negative) practitioners use on pull requests, qualitatively investigate why they use each type of reactions, and quantitatively and qualitatively investigate on what type of pull requests practitioners react to.

Approach: To investigate which reactions practitioners use and why, we conduct a quantitative and qualitative study. Our quantitative study consists of analyzing the distribution of the existing eight reactions (shown in Figure 1) that belong to three types of reactions (i.e., positive, negative, and neutral) in our six studied projects. We also investigate if the distributions vary among different types of pull requests (e.g., bug-fixing and refactoring pull requests). We also conduct two qualitative analyses to understand why practitioners use different types of reactions. Our manual analyses cover two statistically representative samples that together consists of 616 reactions. Our qualitative analyses propose a taxonomy of potential intents behind the usage of different types of reactions. Our quantitative and qualitative analyses are further discussed below.

Quantitative study on reactions. Following the classification proposed by De Zoysa [26], we categorize the eight GitHub reaction types into three types, i.e., "Positive", "Negative", and "Neutral":

- **Positive:** "Thumbs-up", "Laugh", "Hooray", "Heart" and "Rocket".
- **Negative:** "Thumbs-down" and "Confused".
- **Neutral:** "Eyes"

We then calculate the ratio of positive, negative, and neutral reactions that are used on the pull requests of our six studied projects. Note that for this research question we study reactions to merged, closed and open pull requests.

We also studied whether there are any type (i.e., fixing a bug, refactoring, etc.) of pull requests that receive more positive than negative reactions or vice-versa. We think that diverse types of pull requests may receive different ratios

of the three types of reactions. For instance, we intuitively think that practitioners would be happy when bugs are fixed and hence provide positive reactions, while they can have negative reactions when features are deprecated.

To categorize pull requests, we follow the approach that is proposed by Wang et al. [33]. While their approach is originally for classifying the commits, we adapt it for pull requests. Wang et al. defined eight types of commits, i.e., i.e., *test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug*, and *others*. We use their approach to automatically identify the type for each of our studied pull requests with reactions by checking the appearance of a set of predefined keywords in the descriptions of the merged commit(s) associated with the same pull request. In particular, we leveraged the following steps:

Step 1: The first author manually classifies a sample of 100 pull requests from the six projects and compares his labeling results with those generated by the automatic approach. In the first iteration, the automatic approach leverages the same keywords that are used by Wang et al [33].

Step 2: Then, the first author iteratively refines the set of keywords to improve the correctness of the algorithm defined by Wang et al. [33].

Step 3: In this step, the first and fourth authors manually and separately classified a sample of 100 pull requests and compared their classifications to the algorithm and keywords of the previous step, then the first and fourth authors cross-validate the results to measure the agreement/disagreement scores between the two raters from one side and the two raters with the algorithm from another side. The authors obtained an agreement score of 77.93% (Kripperndorff's $\boldsymbol{\alpha}$ [18]) for the classification. To solve the disagreement, they discussed the disagreement cases one by one to reach a final agreement score of 100%. Thus, the final agreement between the two raters and the algorithm is 85.87% , which indicates an almost perfect agreement.

The final set of keywords are shown in Table 3. Note that a pull request can belong to different types depending on its commits. For example, a pull request can add a new feature and refactor existing source code. However, since each pull request ends up with a merge commit, we do not consider a pull request under the merge type of pull requests, except if all of its composing commits are for merging. Finally, we evaluate the overall ratios of the three types of reactions on each pull request type.

Qualitative study on pull requests with negative, extremely positive and positive reactions. To investigate the intention behind the usage of different reaction types, we conducted two qualitative analyses on two different representative samples that sum up to 616 reactions. Our first and second qualitative analysis investigates the intention behind using positive and negative reactions by leveraging 372 (Confidence Level = 95%, Confidence interval = 5%, Population size = 11,585) and 244 (Confidence Level = 95%, Confidence interval = 5%, Population size = 670) positive and negative reactions respectively. Note that the whole data we sampled from consists of reactions for which the same reactor has a comment, as discussed in the following paragraph. Our selected random samples considers distinct pull requests so our conclusions are not biased by a few pull requests that receive a large number of reactions. On top of that, we select a similar number of pull requests from each project so our results are not biased by one project.

Our manual analysis consists of reading the comments that a reactor added to a pull request in order to get a better understanding of how that reaction's author feels about the pull request. We do not consider any comment on a pull request, but only comments of the same person who gave a studied reaction. In addition to that, each of the two authors read the description of a pull request to understand the context of the committed source code change. Finally, we

Table 3. Set of keywords used for pull request classification

PRtype	Description	Set of keywords
Bug	pull request to fix a bug	the commit message contains the keywords: “fix”, “bug”, “repair”, “correct”, “prevent”, “issue”, “problem”, “error”, “exception”, “typo” or “failure”
Feature	pull request to implement new or update existing features	The commit message contains the expressions : “add feature”, “new feature”, “create new”, “add new”, “add missing” OR the pull request is not classified under any category and the commit message contains the keywords : “enable”, “add”, “update”, “improve”, “support”, “new”, “upgrade”, “optimize” or “implement”
Test	pull request to add new or update existing test cases	the commit message contains the keyword “test” or the changed files contain the keyword “test”
Deprecate	pull request to remove deprecated code	the commit message contains the keywords: “deprecate”, “delete”, “remove”, “disable”, “obsolete” or “downgrade”
Refactoring	pull request to refactor existing code	the commit message contains the keywords: “refactor”, “refact” or “style”
Resource	pull request to update non-source code resources, configurations or documents	The commit message contains the keywords: “config”, “licence”, “legal”, “readme”, “gitignore” or “doc” OR the pull request is not classified under any category and none of the changed files are source code files
Merge	pull request to merge branches	1. the commit message contains the keywords: (“merge” or “integrate”) AND NOT (“fail” or “fix”) 2. all the commits of the pull request should be classified as Merge
Others	pull request that belongs to none of the previous types	N/A

leverage an open card sort analysis technique [3] to infer a taxonomy of possible intentions behind the positive and negative types of reactions. In particular, we leveraged the following steps:

Step 1: The first, the third, and the fourth authors manually and separately read the comments that a reactor added to a pull request in order to get a better understanding of how that reaction’s author feel about a pull request. The first author labeled all the 616 reactions, the third and fourth authors labeled 308 positive and negative reactions each. Each author separately mapped each reaction under a different category that she/he defined. For example, given a “Thumbs-down” reaction made by an author named “smolbird” on a pull request (https://github.com/CleverRaven/Cataclysm-DDA/pull/24005) in Cataclysm, each author separately read the description of the pull request to understand that it is an effort to fix some unbalanced weapons in the game. Then, they read the comments of the same reaction author to understand her/his intention behind the negative reaction. On his first comment, the reactor describes the pull request as “the most astoundingly lazy, half-baked, and narrow-minded change [she/he has] seen so far”. The same reactor adds other negative comments which makes it clear that she/he completely refuses this fix. Therefore, we assign this reaction to the third category, namely “Disagreement with a fix”. We allowed the raters to add new labels.

Step 2: The first and last authors met to discuss the labels they separately obtained. The authors then reviewed their labels before discussing the agreement and disagreements, as discussed in the following step.

Step 3: In this step, we first cross validated the results to measure the agreement/disagreement score between the two raters. The authors obtained an agreement score of 73.80% and 63.51% (Kripperndorff’s α [18]) for the qualitative

[3]Open card sort is a categorization approach that is widely used in information processing to extract taxonomies from input data

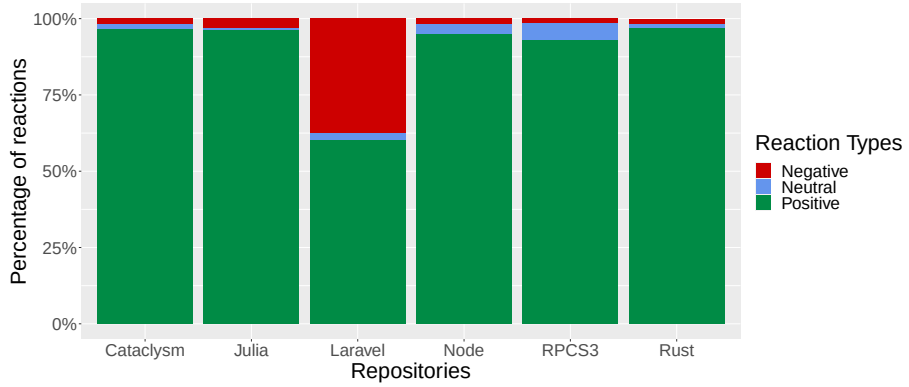


Fig. 5. Percentages of the three reaction types within the studied projects.

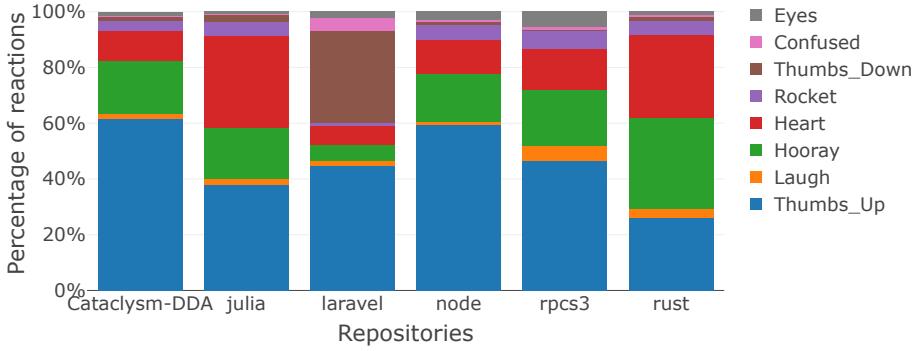


Fig. 6. Reaction usage within the studied projects.

analysis related to the positive and the negative reactions, respectively. To solve the disagreement, they discussed the disagreement cases one by one to reach a final agreement score of 100% for both qualitative analyses.

Results: We observe that our studied projects' reactors mainly use reactions to convey positive reactions for pull requests rather than negative and neutral reactions. Figure 5 shows that positive reactions are dominant in all of our six studied projects. Specifically, positive reactions account for 60.19% to 96.81% (a median of 95.71%) of the total pull request reactions in our six studied projects. In contrast, we observe a median of 1.95% and 1.81% of the reactions are negative and neutral, respectively. We observe one exceptional case in Laravel where negative reactions amount to 37.54% of the total reactions. For individual reaction types, we observe that "Thumbs-up" is the most used reaction in pull requests, followed by "Hooray" and "Heart", with median percentages of 45.70%, 18.80% and 13.42% over the six projects, respectively as shown in Figure 6. Other reactions are less used in pull requests - "Rocket", "Laugh", "Thumbs-down", "Confused" and "Eyes" represent median percentages of 4.94%, 1.92%, 1.81%, 1.26% and 0.68%, respectively.

Further, we find that the above findings still hold on each type of pull request (*test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug*, and *others*), as shown in Figure 7. Most of their reactions belong to the positive type, followed by negative and neutral types. We notice that projects may have different ratios of reactions over the eight types of pull

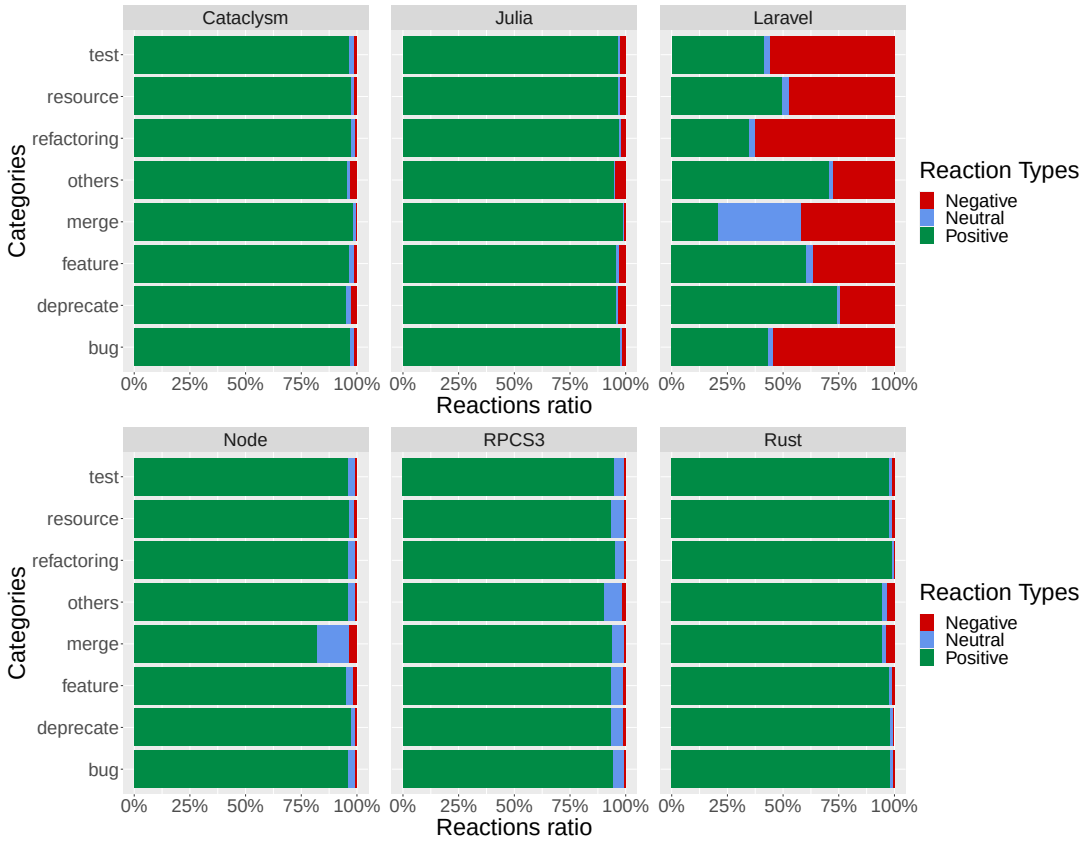


Fig. 7. Percentages of the three reaction types within the studied projects for each type of pull request.

requests. For instance, in Laravel, the percentage of positive reactions range from 21.05% to 74.31%. We also observe the same abnormality in Laravel (i.e., same as figure 5) where the percentages of negative reactions are more important compared to the other five projects. They even exceed the positive reactions in three types of pull requests (i.e., "bug", "refactoring" and "test"). Furthermore, we observe that negative and neutral reactions are dominant in the *merge* pull requests in Laravel with percentages of 42.10% and 36.84%. The reason behind this is the low number of *merge* pull requests with reactions in Laravel. In fact, only six *merge* pull requests have received reactions, most of which are negative or neutral.

The same pull request can combine different reactions that can be even contradictory. For instance, negative reactions are not representative of the impression of practitioners on a pull request, as negative reactions tend to be combined with positive ones. We observe that the median ratio of positive reactions in pull requests with a least one negative reaction is between 41.42% and 50.00% in five out of the six projects, while in Laravel the median ratio is 0.00%, as shown in Figure 8. Furthermore, a median of 94.91% (between 81.13% and 95.84%) of the pull requests, over the six projects, that have consistent reactions (either just positive, just negative reactions, or just neutral reactions). A median of 95.58% (between 76.83% and 96.67%) of the pull requests with at least one positive reaction have just positive reactions. Similarly, a median of 36.93% (between 25.31% and 65.98%) of the pull requests have just negative reactions,

whereas a median of 39.57% (between 6.67% and 44.90%) of the pull requests have just neutral reactions. Our results suggest future studies to leverage all the reactions on a pull requests to get a whole picture of practitioners feedback.

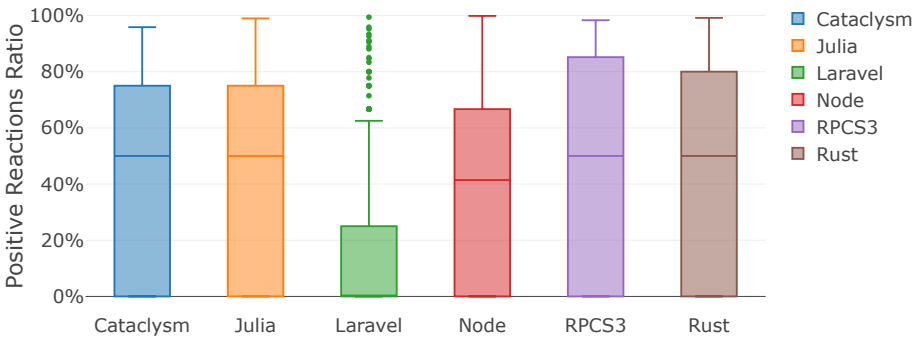


Fig. 8. Distribution of the ratios of positive reactions in pull requests with at least one negative reaction.

Our findings are consistent for reactions to the follow-up discussions of pull requests. We observe that practitioners use reactions mostly to convey positive reactions (median of 94.34% over the six projects). "Thumbs-up" is the most used reaction in pull request discussions with a median percentage of 72.32%, followed by "Hooray", "Laugh" and "Heart" with median percentages of 7.75%, 7.14% and 7.06%, respectively. Less used are the reactions "Rocket", "Thumbs-down", "Confused" and "Eyes" with median percentages of 1.17%, 2.46%, 1.63% and 1.11%, respectively. The same findings hold for each type of pull request (*test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug* and *others*). Finally, discussions to pull requests can combine different reactions, as the median percentage of positive reactions in discussions with at least one negative reaction is between 60.00% and 81.82% over the six projects. Moreover, a median of 90.33% (between 77.72% and 94.49%) of the pull requests with at least one positive reaction have just positive reactions. Similarly, a median of 15.99% (between 8.45% and 23.84%) of the pull requests' comments have just negative reactions, whereas a median of 17.66% (between 8.54% and 23.07%) of the pull requests have just neutral reactions.

We observe that our studied projects' reactors leverage negative reactions to express disagreement with pull requests, especially on pull requests proposing new features, feature modifications, or feature removals. Among the 244 qualitatively studied negative reactions, we observe 120 cases (49.18%) have comments from reactors expressing their disagreement with the proposed feature or the feature modification in the pull request. For instance, a pull request [4] in rust adds a new match expression syntax, and a reactor who gave a negative reaction commented that "the problem is that Rust now has two syntaxes to express the same thing" and also mentioned that the feature "had 37 downvotes compared to 19 upvotes". Bug fixes and refactorings that are considered unnecessary or inadequate also receive negative reactions (45 cases). For example, a reactor stated how a fix is "*rather pointless and a terrible approach to the problem*" [5]. We find 28 cases expressing reactors' disagreement with the proposed removal of existing features. This is unsurprising as removing or modifying existing functions can break client application. We also observe 16 cases where a negative reactor (i.e., a negative reaction author) is reporting failures caused by the pull request. Interestingly, we observe 37 cases when the reactor's comment is neutral (e.g., they can be asking questions, responding to other comments or making suggestions for the pull request in a neutral way) or not explicitly clear about the content of the

[4] https://github.com/rust-lang/rust/pull/47947
[5] "https://github.com/CleverRaven/Cataclysm-DDA/pull/23317"

pull request. For example, a negative reactor added a comment on a pull request [6] in Julia, where she/he simply asked whether the change "[can] break doc-tests?". Finally, we do not observe any case where people have negative reactions about the text of the pull request, e.g., negative reactor leaving a comment complaining about the unclear description of a pull request. Table 4 shows more sample usage cases of negative reactions and the details of our classification on the 126 negative reaction usage cases.

Finally, we observe that our studied projects' reactors leverage positive reactions to exhibit positive attitude (e.g., appreciation, approval and excitement) towards the pull request. Among the 372 qualitatively studied positive reactions, we observe that they can be used to express positive attitude for features' modifications and the introduction of new features that provide users with more functionalities (122 cases). They are also used by several reactors to approve or appreciate a fix or refactoring (101 cases). We observe 6 cases where reactors are excited about a release, and 2 cases where they agree with the removal of buggy features. Surprisingly, we observe 138 cases where reactors who make a positive reaction are neutral about the pull request. Specifically, their comments can be discussing technical concerns regarding the pull request, e.g., explaining a code block to another user, or arguing about the functional and implementation choices of a feature, without expressing any sentiment towards the feature itself. Table 5 shows more details about our classification of positive reactions.

Summary of RQ1

The majority of reactions to the description or discussions of pull requests, regardless of their types, are positive reactions. Negative and positive reactions are often used to express disagreement, and appreciation of the pull request, respectively. More than 37% of the positive reactors are neutral in their comments, i.e., they discuss technical details of the pull request rather than expressing how they feel about the change.

[6] https://github.com/JuliaLang/julia/pull/33864

Table 4. Manual qualitative analysis of pull requests with negative reactions

Category	Definition	# Cases	Sample case
Disagreement with a feature	Reactors can leverage a negative reaction when they disagree with the modification of a feature, either because it has more downsides than upsides, it uses the wrong approach to treat a problem, or it can be confusing to new contributors. Reactors can also partially disagree with a feature modification. This happens when they approve the change but disagree with some details that are either related to the code or the changes made to the feature. New features can also receive negative reactions when contributors question the motivation behind them or find them confusing or more annoying than useful.	120	The season length of Cataclysm was increased from 14 to 91 days [7]. While such a change makes the game more realistic, it amplifies its already quite-high difficulty. This makes it harder for beginners who might quickly abandon the game. Therefore, a user gave a negative reaction (i.e., "Thumbs-down") stating later on via a comment that "*Most players using the defaults will never live to see a winter*" and added that if such a change is not addressed then they are "*going to end up with a boring survival experience.*".
Disagreement with a fix/refactoring	People can use negative reactions when they disagree with a fix either because it is unnecessary or considered as a poor solution to the issue. Negative reactions can also be used in case of disagreement with choices made in a refactoring, that can cause confusion or break users' code.	45	In order to fix an issue, a pull request [8] prevents bicycle archery in Cataclysm. So, a reactor left a negative reaction because she/he disagrees with the fix and suggested that the better solution should be a decrease of shooting accuracy and vehicle control.
Disagreement with the removal	Negative reactions can be used to express complete or partial disagreement with the removal of a feature, because it is valuable or widely used by several members.	28	Half the mods (i.e. game alterations) in the source tree of Cataclysm were obsolete, so a pull request [9] was made to remove them. Consequently many reactors, who still use these mods, expressed their disagreement in comments and used negative reactions. "*This seems like a pretty drastic cut to make*" a contributor stated, and continued to say "*I am concerned that you're being a little overzealous here.*".
Pull request causing failures	Pull requests can receive negative reactions when they break other features or cause failures.	16	Even though a pull request [10] fixes an issue of black screen on Linux, it creates other problems too. That is why a reactor gave it the "Thumbs-down" emoji and commented "*MGS3, RDR, Killzone 2, no longer work [...], games stuck the emulator*".

[7] "https://github.com/CleverRaven/Cataclysm-DDA/pull/25429"
[8] "https://github.com/CleverRaven/Cataclysm-DDA/pull/35201"
[9] "https://github.com/CleverRaven/Cataclysm-DDA/pull/37272"
[10] "https://github.com/RPCS3/rpcs3/pull/4580"

Table 5. Manual analysis of pull requests with positive reactions

Category	Definition	Cases	Sample case
Positive attitude towards a feature	Reactors can leverage positive reactions when they appreciate modifications that enhance a feature, when they simply agree to the idea of the change or when they appreciate the functionalities introduced by a new feature.	123	A member of Laravel appreciates the feature modifications in a a pull request [11] so she/he reacts with "Thumbs-up" and explains in comments that she/he likes it because the output is "*much more informative and easier to understand*".
Positive attitude towards the fix/refactoring	Users can react positively to express appreciation of the results of a fix and confirm that there are no longer any test failures or regressions, or to simply appreciate a refactoring.	102	After performing tests, a user leverages "Thumbs-up" reaction in a pull request [12] in RPCS3 to confirm that "most of the regressions are [now] fixed".
Positive attitude towards the release	Positive reactions can be used to express appreciation or excitement for a release	7	In this example, a reactor to a pull request [13] in Julia, exhibits his excitement for the release by commenting "Do it! Just do it!".
Agreement with the removal	Positive reactions can also be used to approve the removal of a feature that can be buggy.	2	An unnecessary package in Laravel was removed, so upon the removal a user reacted positively to the pull request [14] and commented "This is the only thing I remove on a new Laravel app. I am in favor of this removal".
Others	This is when the reactors' comments do not show their intention behind the positive reaction. Such reactors can be discussing functional choices, asking or answering questions about the pull requests in the comment section, making suggestions to help fix an issue or improve a feature, but they do it using neutral expressions.	138	In this example, a contributor reacts positively to a pull request [15] in Rust, and in the comment section she/he argues about the functional choices of the pull request in a neutral way.

RQ2: Who reacts to pull requests?

Motivation: We aim to investigate how different stakeholders react to pull requests. These stakeholders are defined according to their contribution to a project, i.e., TF-authors [16], core contributors, peripheral contributors, casual contributors or non contributors (aka., outsiders). In particular, we quantify the amount of reactions of each type of reactors and qualitatively investigate on what each type of reactors reacts on.

Approach: We first categorize reactors into the following five categories, based on their commits in the studied projects at the time when they react to a pull request and the degree of authorship of source files:

[12] https://github.com/laravel/laravel/pull/4523
[13] https://github.com/RPCS3/rpcs3/pull/6388
[14] https://github.com/JuliaLang/julia/pull/28521
[15] https://github.com/laravel/laravel/pull/4593
[16] Truck Factor (TF) authors are contributors without whom a project can be incapacitated

- **TF-authors:** contributors who have the highest degree of authorship of the source files in the project (determined following the approach of Avelino et al. [13]). To identify TF-authors, we leverage the algorithm of Avelino et al. [13] on each month and each project to estimate the truck factor of a project.
- **Core contributors:** contributors who have significantly more commits than others (determined by an automatically learned threshold as discussed below).
- **Peripheral contributors:** contributors who have less commits than core contributors, i.e., their commit number is smaller than the threshold learned for identifying the core.
- **Casual contributors:** contributors who have exactly one commit at the time of their reaction, following the work of Crowston et al. [16].
- **Outsiders:** reactors who have not yet contributed any commit to the project at the time of their reactions.

We leverage the weighted uni-variate clustering algorithm [32] to identify **core and peripheral contributors**. Crowston et al. [16] found that Bradford's law analysis is the appropriate approach to classify developers based on the number of their "postings" (i.e., commits) and differentiate between core and peripheral contributors. They define the core group as the members who contribute 1/3 of the total number of postings. However, they also suggest that "rather than using Bradford's level of 1/3 as the cutoff for the core group, it may be that a different level will provide a more useful definition of the core group". Therefore, we followed Bradford's law approach, but we chose the weighted uni-variate clustering algorithm [32] instead of the 1/3 level in order to better classify core and peripheral contributors based on their number of commits at the time of their reaction to a pull request and according to the other developers.

We leverage the elbow method [17] and weighted uni-variate clustering algorithm to identify a dynamic threshold of commits to classify core from peripheral contributors. To differentiate core from peripheral contributors, one could consider a fixed threshold of commits. For instance, assuming the developers with more than ten commits are the experts in the project. However, such design ignores the evolution of a software project, e.g., in the early development phase, even main contributors may have few commits. Once the software became mature, non-experts would also have accumulated many commits. Moreover, it can not capture the intuition that core contributors should relatively contribute more than peripheral contributors. Thus we choose to learn a dynamic threshold that is monthly updated automatically by considering all contributors' commit numbers.

To collect the commits of our studied reactors, we first collect the Git commit history ("git shortlog –all –no-merges") with each commit's author and date. Then, we map the commits' authors to the reactions' authors. Since our reactions' authors are represented as a set of GitHub usernames, we map each GitHub username to its associated Git commits' author names. Such a mapping is nontrivial as developers might not use the same GitHub username for Git commits. To resolve this issue, we design a heuristic that first merges Git authors with the same associated email. Next, we use the user's latest commit id for each unique commit author to retrieve the GitHub user name who made the commit using the GitHub v4 API.

First, we collect the number of commits made by each contributor from the repository creation date up to a specific month. Then, we cluster all contributors (for one particular month and project) based on their cumulative number of commits by applying the weighted uni-variate clustering algorithm, with four clusters (i.e., k=4) that we justify in the following paragraph. The first cluster (i.e., with the lowest number of commits) represents peripheral contributors, while the third and fourth clusters (i.e., with the highest numbers of commits) represent core contributors. We do not consider developers of cluster 2 to better discretize the differences between core and peripheral contributors.

[17] A method used to determine the optimal number of clusters by looking at the percentage of the comparison between the number of clusters that will form an elbow at a point [29]

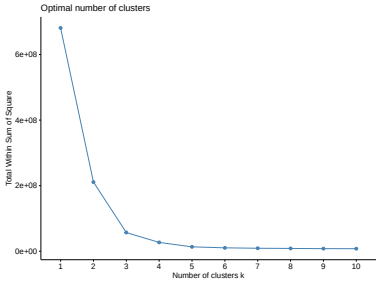
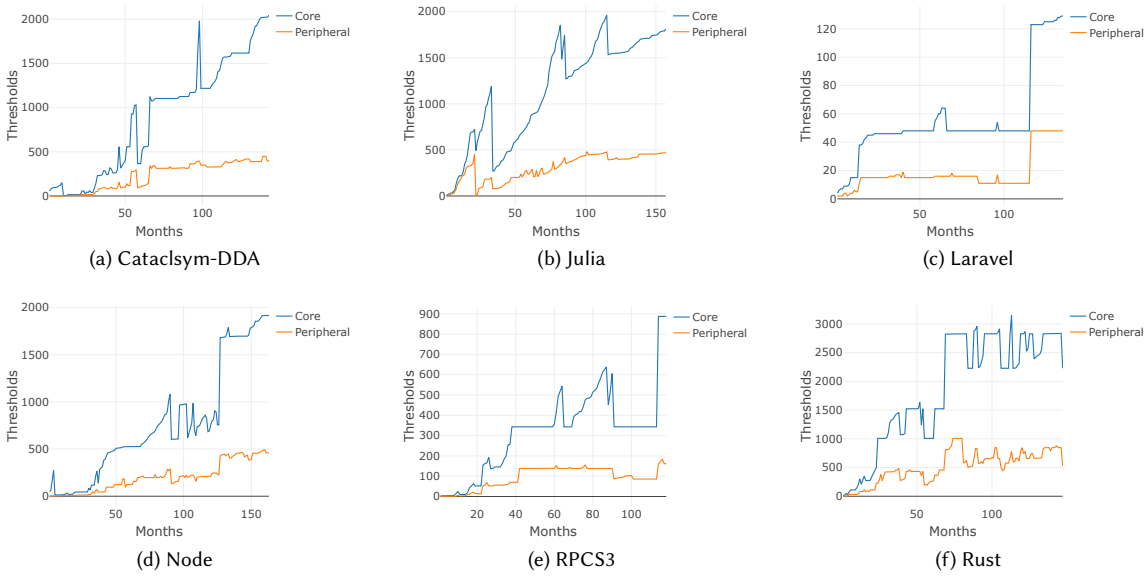


Fig. 9. Elbow method applied to a random month in the project Rust. The elbow point (*k=4*) represents the optimal number of clusters for the studied data set. It is identified automatically using the KElbowVisualizer [18] in Python.



(a) Cataclsym-DDA (b) Julia (c) Laravel

(d) Node (e) RPCS3 (f) Rust

Fig. 10. The distribution of core and peripheral thresholds per month, for the six studied projects. Core contributors are developers with more commits than the core threshold, while peripheral contributors are developers with less commits than the peripheral threshold.

We leverage the elbow method to identify the optimal number of clusters. We apply it on each month for all the six projects and we find a median optimal number of clusters *k=4* and an average *k=4.2*. So, we choose *k=4* as our optimal number of clusters over the six projects to have consistent results. Figure 9 shows an example of how the optimal number of clusters is automatically identified for a random month (May 2021) for the Rust project. The same figure shows that four, which is the elbow of the curve, is the optimal number of clusters. When we apply the weighted univariate clustering algorithm with an input of *k=4* on the same month (May 2021), we obtain a core threshold of 2558 (i.e., developers with more than 2558 commits in the same month are core) and a peripheral threshold of 672 (i.e., developers with less than 672 commits in the same month are peripheral). As a result, we find 9 core developers and

3,826 peripheral developers. Figure 10 shows the distribution of the core and peripheral thresholds per month for each project.

Following the prior work of Pinto et al. [14], we identified **casual developers** as contributors "that performed at most one commit to a software project" at the time of their reaction. In particular, we further divided our "Peripheral Contributors" category into two sub-categories: "Casual Contributors" (i.e., contributors who have at most one commit) and "Peripheral (non-casual) Contributors" (i.e., contributors who have at least two commits). If we take the previous example, 1,321 out of the 3,826 peripheral developers have contributed with exactly one commit. Therefore, we classify them as Casual and remove them from the list of peripheral developers, resulting in 2,505 peripheral (non-casual) developers instead of 3,857.

We note that since we use different algorithms to classify TF-authors and core developers, we proceed to remove the identified TF-authors from the list of core developers as they would be duplicated. For intance, "Kevin Granade" [19] is identified as a TF-author of the Cataclysm project in July 2022 based on the TF algorithm. At the same time he appears in the list of core developers in July 2022 since he has the largest number of commits. Therefore we remove him from the list of core developers and keep him only as a TF-author.

According to our clustering, we determine the category of a reactor at the time of her/his reaction. Note that the same reactor can have different status according to the time at which she/he reacted to a pull request. For example, a reactor can be an outsider if she/he had no commits when she/he reacted to a pull request. Later on, the same reactor can be a casual or peripheral contributor if she/he contributed to the project. Similarly, the same reactor can be later on a core contributor or a TF-author.

Quantitative analysis on who reacts to pull requests: Based on the results of the truck factor algorithm and the identified thresholds, we determine the category of each reactor on the month she/he reacted in to end up with a set of reactions and the category of their associated reactors. Based on that set of data, we perform a quantitative analysis to identify which category of reactors have more pull request reactions. In particular, we quantify whether outsiders react to pull requests, how different is the number of reactions provided by the five categories of reactors, and to how many pull requests the same person and category of reactors reacts to. We also quantify whether any of the five categories of reactors focus on a specific type of pull requests (*test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug*, and *others*) compared to the other categories of reactors. Since we compare between the number of reactions and the number of pull requests each category of reactors reacts on, we only consider reactions to merged and closed pull requests, without including the open pull requests. That is because an open pull request can still receive reactions from different reactors.

Qualitative analysis on what do the three categories of reactors react about: We conduct a qualitative analysis to better understand what each category (i.e., novices, experts, outsiders) of reactors react to.

Since we have a different distribution of the number of reactors per category, we select a weighted statistically representative sample (confidence level = 95%, confidence interval = 5%) of 373 reactions. Our sample is weighted according to the number of reactors on each category (i.e., TF-authors, core contributors, peripheral contributors, casual contributors and outsiders). We end up with a dataset of 11 TF-authors' reactions, 33 core contributors' reactions, 221 peripheral contributors' reactions, 14 casual contributors' reactions and 94 outsiders' reactions. For each of the previous categories, the reactions are distributed equally over the six projects. Similarly to RQ1, our qualitative analysis consists of reading the pull request to understand its context and the comment provided by the same reactor whose reaction is in our selected dataset. We also follow an open-card sorting technique and the three steps mentioned in RQ1 to identify

[19] "https://github.com/kevingranade"

a taxonomy of elements each category of reactors focuses on. Our original agreement score is 66.48% (Kripperndorff's α). After discussing the disagreement cases, we reach an agreement score of 100% (Kripperndorff's α).

Results: We surprisingly observe that outsiders are actively engaged in pull request reactions often (in five of our six studied projects) more than contributors. We observe that in five out of the six projects, outsiders have more reactions than contributors. For instance, the percentages of reactions made by outsiders are 54.33%, 88.65%, 81.36%, 88.18% and 51.16% of the total reactions in Julia, Laravel, Node, RPCS3 and Rust, respectively. In Cataclysm, reactions are mostly made by contributors with a percentage of 63.62% of the total reactions, as shown in Figure 11. Further, most of the contributors' reactions over five of the six projects are made by peripheral contributors, as they represent between 56.41% and 89.94% of the total contributors' reactions. Exceptionally in Laravel, reactions from peripheral contributors account to 42.85% of the total contributors' reactions. On the other hand, some pull requests have received reactions by only one category of reactors. For instance, a median percentage of 35.71% pull requests (over the six studied projects) only receive reactions from outsiders, while for TF-authors, core contributors, peripheral contributors and casual contributors, the numbers are 0.04%, 1.66%, 8.39% and 0.86%.

Our findings are consistent over the eight types of pull requests (i.e., *test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug*, and *others*). For instance, in Cataclysm, the median percentages of reactions made by TF-authors, core contributors, peripheral contributors, casual contributors and outsiders are 0.01%, 4.18%, 57.21%, 2.01% and 36.38%, respectively. We observe a similar distribution on all the types of pull requests of the same project. For example, 0.43%, 7.07%, 54.11%, 4.09% and 34.28% of the reactions on *"bug"* pull requests are made by TF-authors, core contributors, peripheral contributors, casual contributors and outsiders, respectively. The median percentages of such reactions over the eight types of pull requests in Cataclysm are 0.47%, 7.33%, 51.99%, 4.38% and 34.93%.

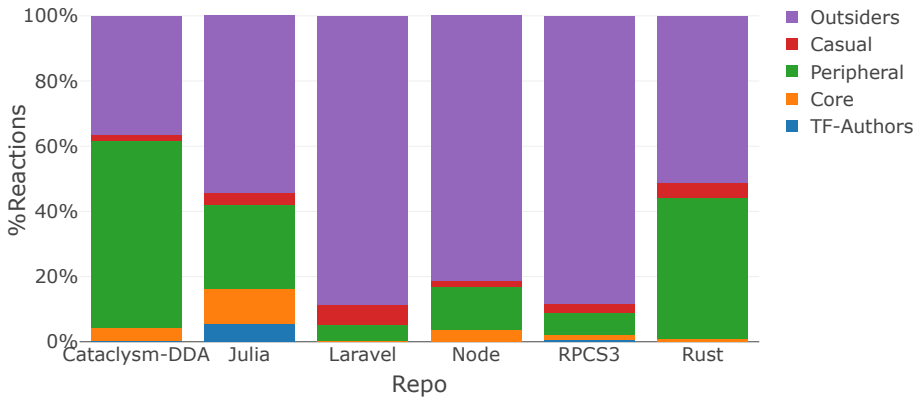


Fig. 11. Percentage of reactions given by each category of reactors. For example, 6.14%, 11.40%, 56.10%, 5.88% and 20.47% of the total reactions on Julia are provided by TF-authors, core contributors, peripheral contributors, casual contributors and outsiders, respectively.

In each of the six projects, we find that the same outsider can react to significantly less pull requests compared to a given contributor. We observe statistically significant differences (Wilcoxon test; p-values < 0.05) between each pair of the five categories of reactors (i.e., core vs. peripheral, core vs. casual, core vs. outsider, peripheral vs. casual, peripheral vs. outsider and casual vs. outsider) in terms of the number of pull requests on which a person reacts. We also observe small to large effect sizes between each pair of reactors over the six projects. For instance, we

find median Cohen's d values of 0.81 (between 0.28 and 1.22), 0.78 (between 0.02 and 1.29), 1.08 (between 0.75 and 2.12), 0.13 (between 0.05 and 0.45), 0.37 (between 0.11 and 0.56) and 0.28 (between 0.08 and 0.64) for the pairs core vs. peripheral, core vs. casual, core vs. outsider, peripheral vs. casual, peripheral vs. outsider and casual vs. outsider, respectively. Note that we did not have enough observations of TF-authors to perform statistical analyses. As shown in Figure 12, core contributors react to the largest number of pull requests with a median between eight and 870 pull requests over the six projects. Peripheral and casual contributors are next with medians between two and 14 and between two and 23 pull requests, respectively. Outsiders react to a median of one to two pull requests only. Our observation suggests that studying how the reactions of the same person evolves over time (e.g., do stakeholders get more positive or more engaged over time?) might be more feasible for contributors (e.g., core contributors who also were peripheral or casual contributors at a certain time of the project lifecycle) than it is the case for outsiders who do not react to multiple pull requests.

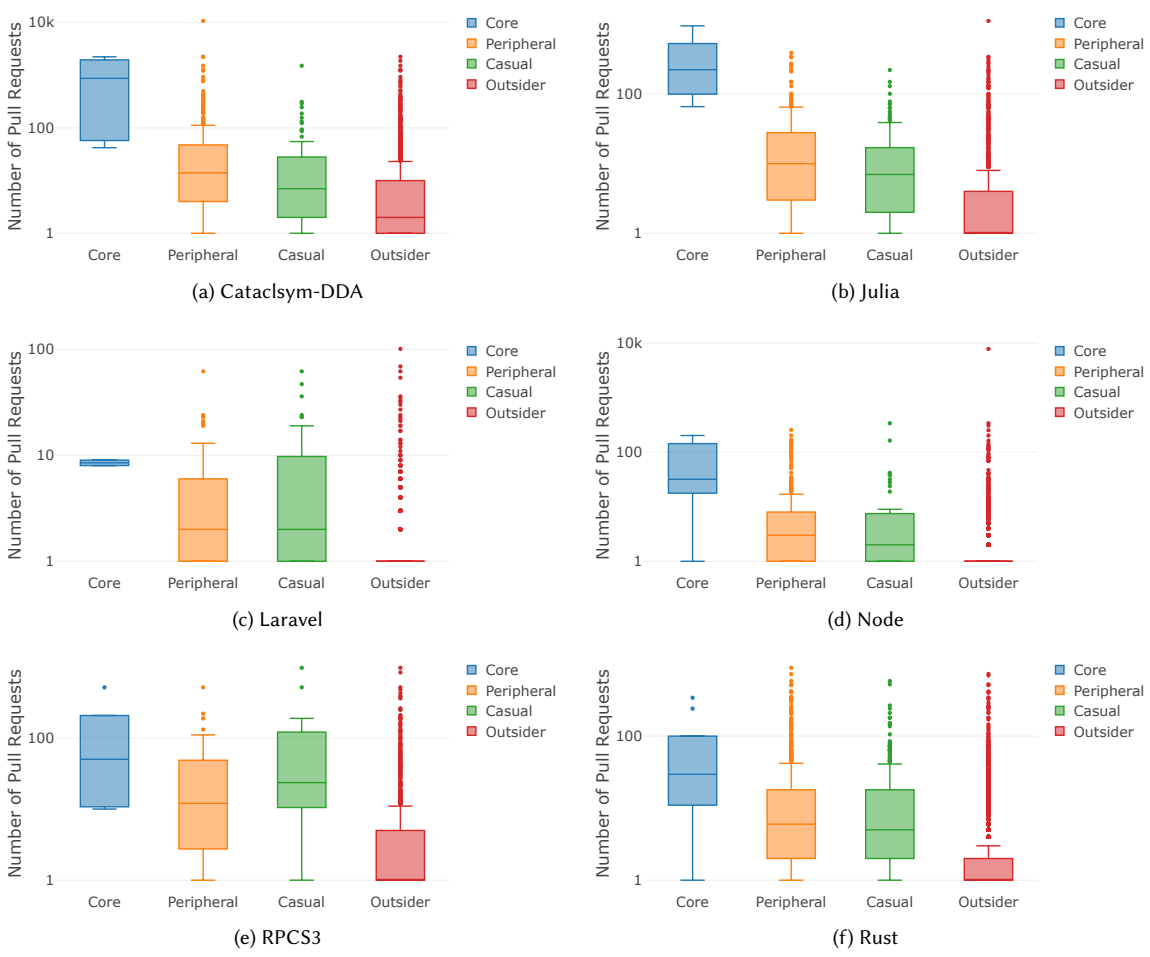


Fig. 12. The number of pull request the same reactor reacts on. Note that we did not have enough observations for the TF-authors category.

Our findings are consistent for the reactions on the follow-up discussions of pull requests. Outsiders are active in pull request reactions with a median of 49.50% of the reactions. They also have more reactions than contributors in three out of the six projects, with percentages of 54.09% in Julia, 89.76% in Laravel and 86.24% in RPCS3. In Cataclysm, 65.18% of the reactions are made by contributors. These results are also consistent over the eight types of pull requests (i.e., *test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug*, and *others*). Finally, the same outsider reacts to significantly less pull requests than a given contributor. We observe statistically significant differences (Wilcoxon test; p-values < 0.05) with small to large effect sizes between each pair of the five categories of reactors.

Contributors' reactions are mostly related to the functional and implementation choices of the pull request while outsiders' reactions are more related to the end user experience and the results of the feature testing. Contributors can react to either the features in a pull request (115 cases), the code (103 cases), the results of testing a feature (44 cases), or even the user experience (11 cases). We observe that contributors who reacted to features tend to suggest new ideas to improve them. Similarly, the comments of the contributors who reacted to the code were accompanied with alternatives to improve or fix issues related to the code. On the other hand, outsiders can react positively or negatively to a pull request according to the results of testing a feature (23 cases), or according to its impact on the end user experience (20 cases). Outsiders can also react to the functional choices of a pull request (37 cases), where they express their opinions (e.g., agreement or disagreement with the pull request) and even discuss other options for the proposed approaches. Surprisingly, we observe 9 cases where outsiders react to the code of the pull request and can even make suggestions to improve it. Note that we were not able to classify 23 cases (6.16%) which did not belong to any of the aforementioned categories. Our results suggest future studies distinguish between the reactions coming from outsiders and contributors as they target different aspects of pull requests.

Summary of RQ2

We observe that outsiders are as active as contributors on reacting to pull requests. However, the same outsider react to statistically significantly less pull requests compared to contributors, core in particular. Contributors' reactions are more related to the code of the feature, while outsiders' reactions are more related to user experience.

Table 6. Results of the qualitative analysis of reactions and comments from different GitHub users. Columns #TF, #C, #P, #c and #O stand for the number of cases for TF-authors, core contributors, peripheral contributors, casual contributors and outsiders, respectively. The taxonomy here is different than RQ1 since we are not analyzing the purpose of reactions (covered in RQ1), instead, we analyze the concerns of different categories of reactors (i.e., what do they react to)

Category	Definition	#TF	#C	#P	#c	#O	Sample case
Functional choices	Reactors can react to the functional choices of the pull request. At the same time they can suggest new ideas to improve a feature or solutions to fix issues related to the pull request, or simply express their opinion (e.g., agreement or disagreement with the pull request) without adding anything new to the discussion.	3	10	78	6	30	In Cataclysm, an expert reacts to the functional choices of a pull request [20] and suggests a less extreme alternative solution to the pull request that will make a feature more realistic and reasonable.
Implementation choices	One can react to the implementation choices of the pull request while reviewing the code, proposing alternatives to improve the code or or simply expressing their opinion without bringing anything new to the discussion of the pull request.	6	20	55	2	6	A contributor appreciates the code of a pull request [21] in Node and reacts accordingly, precising that she/he "likes [its] polymorphic approach".
Results of the manual tests	Reactions can also concern the results of the manual tests performed by either the reactors themselves or end users who test the pull request and report back.	0	0	33	2	21	An outsider reacts negatively after testing a pull request [22] and receiving a "fatal error message" caused by the refactoring.
Impact on the end user experience	Reactors can react to a pull request accordingly to its impact on the end user experience, and explain how it is either improving their experience or affecting it negatively.	0	0	5	0	9	An outsider reacts positively to a pull request [23] in Julia that is allowing him access to the console/command buffer "which is useful for grabbing printed output from previous commands".
Excited about the pull request	This is when reactors explicitly express their excitement towards the pull request without discussing the functional or implementation choices of the pull request, reporting the results of the manual tests nor describing the impact on their user experience	0	0	1	0	7	An outsider reacts positively to a release pull request [24] in Node and follows it with a comment saying that this release is "the greatest news in this morning!".

RQ3. When do practitioners react to pull requests?

Motivation: We aim to investigate when pull request reactions are given and if different categories of reactors (*TF-authors*, *core contributors*, *peripheral contributors*, *casual contributors*, and *outsiders*) tend to react at different stages

[21] "https://github.com/CleverRaven/Cataclysm-DDA/pull/31611"
[22] "https://github.com/nodejs/node/pull/13013"
[23] "https://github.com/RPCS3/rpcs3/pull/5456"
[24] "https://github.com/JuliaLang/julia/pull/23319"

of a pull request. For instance, one could react at the early stage of a pull request right after its opening or around the time when the pull request is closed. By answering this question, we would also understand at which stage of a pull request the received reactions can approximate the final distribution of reactions given by different community members.

Approach: Each reaction we collected contains a timestamp specifying when the reaction was given on a specific pull request, i.e., *reactTime* Since the lifespans of pull requests can largely vary from a few seconds (e.g., a pull request [25] in Cataclysm was closed after less than one minute) to several months (e.g., a pull request [26] in Julia stayed open for two years and seven months), we normalise the time at which a reaction is added according to the opening and closing time of a pull request by measuring the relative reaction time, which we define as follows:

$$relativeReactTime = \frac{reactTime - PRopen}{PRclose - PRopen} \quad (1)$$

Where *PRopen* and *PRclose* represent the opening and closing times of a pull request, respectively. $PRclose - PRopen$ refers to the lifespan of a pull request. Note that the value of *relativeReactTime* is positive and can be larger than one, indicating a reaction given after the closing of a pull request, namely post-closing reaction. *relativeReactTime* values of zero and one refer to the opening and closing time of a pull request, respectively.

We also investigate if the three different categories of reactors (i.e., novices, experts, and outsiders) react at different stages of a pull request. We further verify if our observed findings still hold for the eight types of pull requests (i.e., *test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug*, and *others*). Note that in this research question, we only study merged and closed pull requests, since it is not possible to normalize the time of reactions in open pull requests.

Results: Most reactions are given before the closing times of pull requests. Specifically, 62.53% to 82.99% (median of 78.40%) of the total pull request reactions in our studied projects were given before the closing time of the pull requests. Figure 13 shows the percentage of reactions over *relativeReactTime* in our six studied projects. The first reaction of a studied pull request can arrive at a median relative time that ranges between 0.02 and 0.47, which corresponds to a median of 0.98 to 5.66 hours after the creation time. Moreover, we observe medians of 57.14%, 66.66%, 0.00%, 100.00%, 33.33% and 50.00% of the reactions are made in the first 25% relative time of a pull request, for Cataclysm, Julia, Laravel, Node, RPCS3 and Rust, respectively as shown in Figure 14. On the other hand, in each of the six projects, we observe medians of 0.00% of the reactions are made in the second, third and fourth 25% relative times of pull requests. The above findings can be observed over the eight types of pull requests (*test*, *resources*, *refactoring*, *merge*, *feature*, *deprecate*, *bug*, and *others*). For example, in Cataclysm, the median percentage of reactions made in the first 25% relative time of the pull requests is 57.14%. We observe a median of 66.67% over the eight types (between 50.00% and 66.67%). As for the second, third and fourth 25% relative times of the pull requests, we also observe medians of 0.00% over the eight types of pull requests.

We observe that 12.78% to 72.89% of the pull requests in the six target projects have at least one reaction after the closing time. Such percentages are 39.66%, 27.92%, 64.42%, 12.78, 72.89% and 41.99% for Cataclysm, Julia, Laravel, Node, RPCS3 and Rust, respectively. The post-closing reactions are given at a median time that ranges from 1.55 to 32.35 days after the closing time. In an extreme example, 11 out of 25 reactions on a pull request [27] in Rust were given after the closing time. More specifically, among the 11 reactions, six were given one week after the closing time, two given after one year, two given after three years and the last reaction was given four years after the pull request

[25] "https://github.com/cleverraven/cataclysm-dda/pull/19664"
[26] "https://github.com/JuliaLang/julia/pull/18594"
[27] "https://github.com/rust-lang/rust/pull/30652"

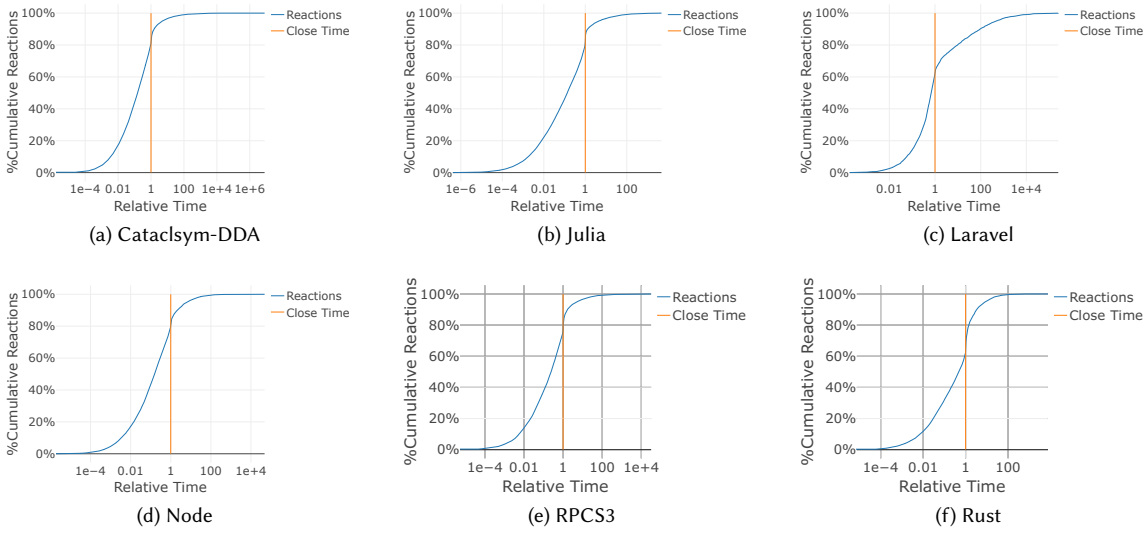


Fig. 13. Cumulative percentage of reaction occurrences over relative time. The x-axis represents the *relativeReactTime* metric where zero is the moment of creation of a pull request and one is the closing time. For instance, if a pull request's lifespan is two weeks, a relative time of 0.5 would be one week. The y-axis represents the cumulative percentage of reactions over all the pull requests. For example, 81.51% of all the reactions in Cataclysm were given before the closing time of their respective pull requests, while 17.19% were given before a relative time of 0.01 (i.e., first 10% of the lifespan of the pull requests).

was merged. In all of the six projects, we observe statistically significantly more positive reactions before the closing time compared to positive reactions after the closing time (Wilcoxon test; p-values < 0.05). We also observe small to large effect sizes over the six projects. Specifically, we find Cohen's d values of 0.65, 0.35, 0.17, 0.10, 0.63 and 0.07 for Cataclysm, Julia, Laravel, RPCS3 and Rust, respectively. Similarly, we observe similar statistically significant differences between negative reactions before and after the closing time (Wilcoxon test; p-values < 0.05) with Cohen's d values of 0.28, 0.89, 0.13, 0.20, 0.06 and 0.21. Figure 15 shows the distribution of these two types of reactions before and after the closing time.

Core contributors tend to react to pull requests before peripheral contributors, casual contributors and outsiders, as shown in Figure 16. The distribution of the relative react times of the four categories show that core contributors react at a median relative time that ranges between 0.07 and 0.39, while peripheral contributors react slightly after that, at a median relative time of 0.06 up to 0.49. Casual contributors and outsiders tend to be the last ones to react to pull requests, after median relative times that range between 0.06 and 0.97 and between 0.15 and 0.90, respectively. The relative times at which the four categories of developers react are statistically significantly different (Wilcoxon test; p-value $< = 0.05$) according to our pair comparison of core vs. peripheral, core vs. casual, core vs. outsider, peripheral vs. casual, peripheral vs. outsider and casual vs. outsider. We also observe that outsiders and novice contributors are more likely to react after the closing time. For instance, in Laravel, where more than a third of the reactions were added after the closing time, we find that outsiders and casual contributors are responsible for 91.55% and 4.54% of those reactions, respectively, which adds up to 96.09% of the total post closing reactions. As for the other five projects, a median of 63.38% (between 54.60% and 93.46%) of such reactions are made by outsiders, and a median of 4.00% (between 1.40% and 5.94%) is made by casual contributors. Therefore, according to our findings, researchers

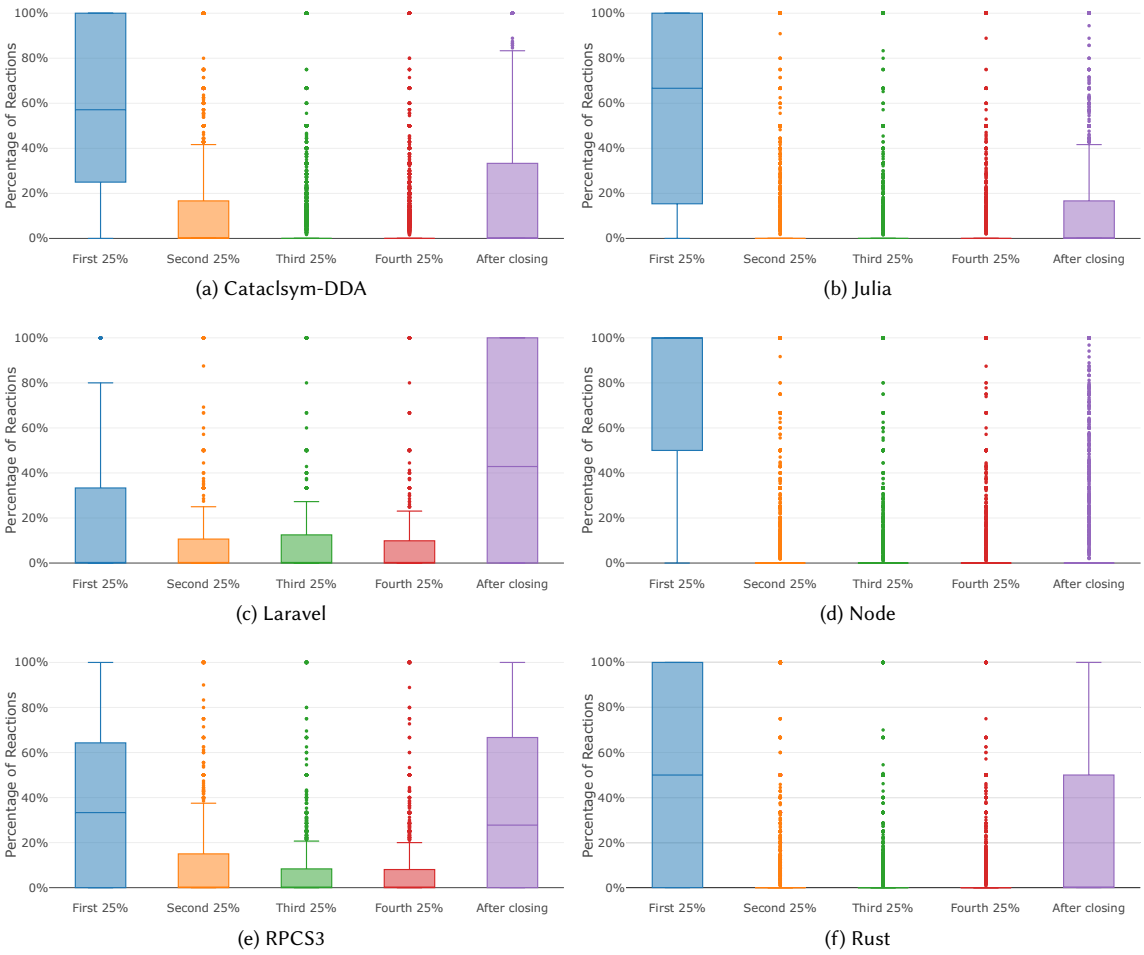


Fig. 14. Distribution of reactions in different parts of pull requests lifespans.

aiming at studying how core contributors react to the changes in source code can leverage the reactions of recently merged or closed pull requests, which is not the case for researchers who wish to investigate the reactions of outsiders on pull requests.

Most of our findings are consistent with the results for reactions on the follow-up discussions of pull requests. We observe that a median of 76.45% (between 51.24% and 80.49%) of the discussion reactions were made before the closing time of the pull requests. Unlike reactions to the description of pull requests, we observe that less reactions are made in the first 25% relative time with medians of 33.33%, 20.00%, 0.00%, 33.33%, 0.00% and 12.50% for Cataclysm, Julia, Laravel, Node, RPCS3 and Rust, respectively. On the contrary, we observe medians of 100% and 14.55% of the reactions are made after the closing time in Laravel and RPCS3, respectively. As for the other four projects, such medians are all equal to 0%. We consistently observe these findings over the eight types of pull requests (*test*, *resources*,

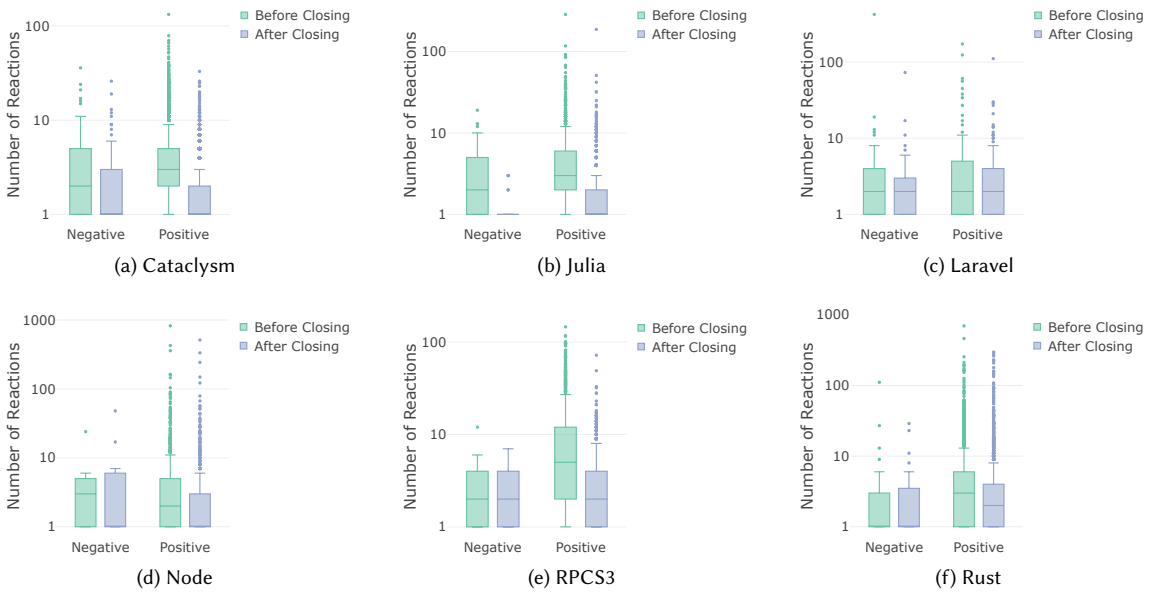


Fig. 15. Distribution of the number of reactions before and after the closing time.

refactoring, *merge*, *feature*, *deprecate*, *bug*, and *others*). On the other hand, we find that 32.59% to 86.04% of the pull request discussion have at least one reaction after the closing time.

We also observe statistically significantly more positive reactions before the closing time compared to positive reactions after the closing time (Wilcoxon test; p-value < = 0.05). Note that we observe small to large effect sizes over the six projects. However, we did not have enough data points to perform statistical analysis on negative reactions.

Finally, we observe that core contributors tend to react to pull requests' comments before peripheral contributors, casual contributors and outsiders. Core contributors react at a median relative time between 0.29 and 0.61, while peripheral contributors react at median relative time between 0.28 and 0.60. Casual contributors and outsiders tend to be the last reactors with median relative times that range between 0.24 and 1.00 and 0.49 and 0.95, respectively. We also observe statistically significant differences between the relative times at which the four categories of developers react (Wilcoxon test; p-value < = 0.05).

Summary of RQ3

While the first 25% of the pull requests' lifespan is the most active period on reactions, some of them are made even after a pull request is merged. Core contributors tend to react in the early stages of pull requests, while peripheral contributors, casual contributors and outsiders are more likely to react around or after the closing time. Most of our findings for reactions on the description of pull requests are consistent with the results for reactions on the follow-up discussions of pull requests. However, reactions on the follow-up discussions are more likely to be made after reactions on the pull requests' descriptions.

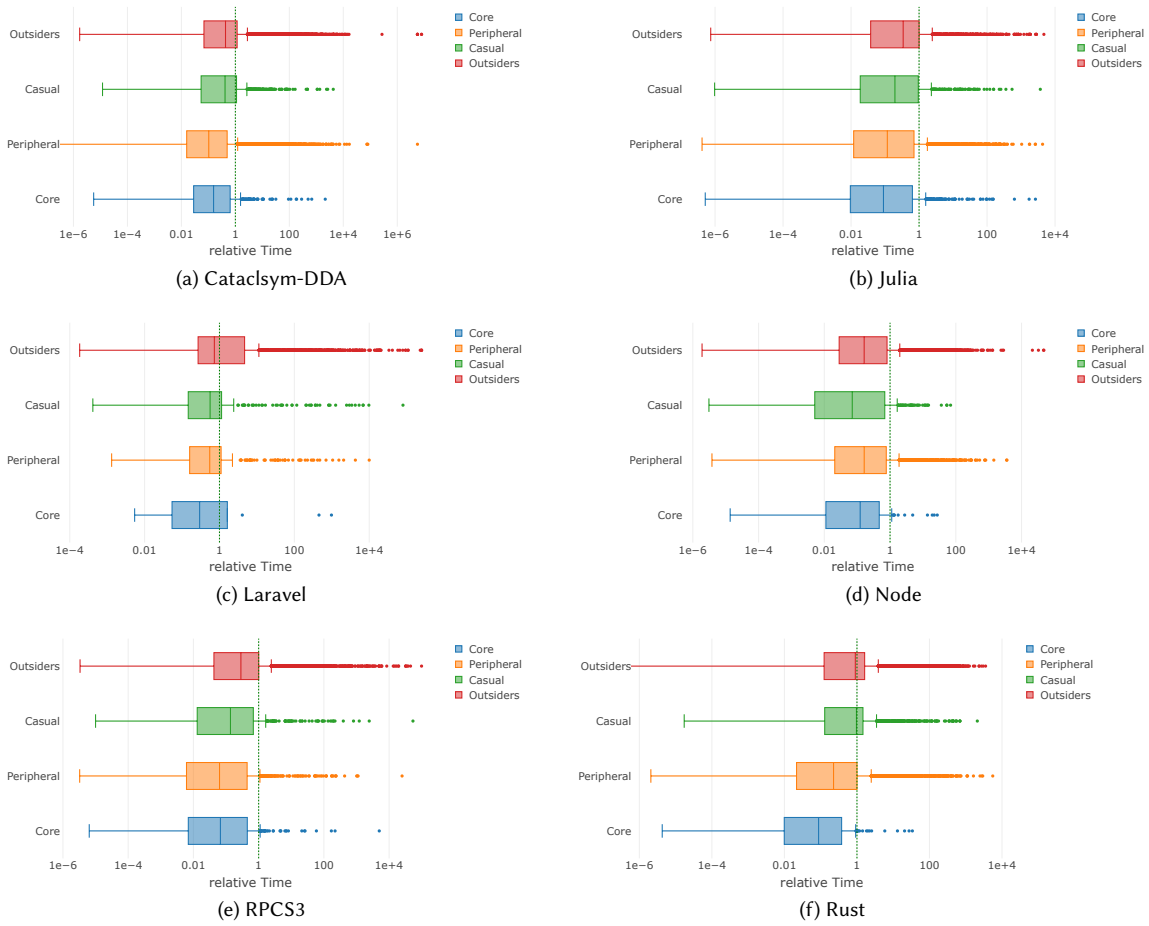


Fig. 16. Distribution of relativeReactTimes at which experts, novice contributors and outsiders react.

5 IMPLICATIONS

We observe through our study that reactions to the pull requests is a promising dataset to capture different stakeholders' perceptions on pull requests. However, such a dataset has certain limitations. Below, we list a set of promises that reactions on pull requests have as well as their limitations.

Promise 1: Reactions on pull requests is a unique source of data to capture practitioners' perception on a pull request. As we observe from our preliminary study (PQ1) that reactors do not leave any comments, we suggest future studies leverage reactions to the pull requests to better understand stakeholders' perceptions of pull requests. For example, the existing study on developer sentiment by Huq et al. [11] looked at the comments to investigate how developers' sentiments are related to bugs, and found that too much positive emotions in comments may lead to buggy code, as they can turn developers overconfident and careless, reducing their ability to scrutinize their own code. We suggest extending their work to consider the GitHub reactions too as they can provide exclusive data about developers' sentiments.

Promise 2: Reactions exist on different types of pull requests. Different topics on software engineering can benefit from GitHub reactions to the pull requests. For example, researchers can investigate how practitioners perceive different types of refactoring. For example, future studies can investigate whether large refactored code is positively perceived by developers. Future studies can also investigate the impact of fixing bugs on a software system's stakeholders. For example, future studies can leverage outsiders' reactions to bugs to see what types of bugs are more expected to be fixed and design machine learning models that identify and prioritize these types of bugs (e.g., bugs related to the UI of a software system).

Promise 3: Positive reactions are related to the content of the pull requests. Reactions can be used as means to investigate the impact of a pull request on different stakeholders. We also observe that reactors who leave a comment, their comments are about the content (e.g., the implemented feature, the fixed bug, the source code modification, etc.) of the pull requests. Through our manual analyses, we did not observe any reaction that is related to the description or title of the pull request. For example, a pull request [28] received 56 positive reactions, while that pull request does not have any description. Similarly, another pull request [29] received 87 positive reactions, while the description of that pull request is a simple link to Wikipedia. Our results suggest future work to leverage GitHub reactions to investigate source code changes that practitioners prefer. For example, future studies can investigate whether readable code, small changes, and changes that are touching certain types of files (e.g., code files in the core of a system, build files, configuration files, etc.) are more likely to be appreciated by other developers. We also suggest future studies to design models that early predict whether a change will be appreciated by different stakeholders such that developers improve their code before submitting it for review.

Promise 4: GitHub pull request reactions can capture the reactions of not just the contributors to a project, but the reactions of external stakeholders. For instance, we observed that a median of 63.06% (between 20.47% and 86.09%) of the reactions over the six studied projects are coming from external users, which indicate that reactions can be used as a mean to investigate the impact of different software engineering practices on a software system's stakeholders apart from the developers. For example, researchers can investigate the impact of the amount of changes on users. For instance, we observe cases in which users react to a pull request because of their appreciation of a new feature or negatively react to a pull request because it introduces bugs in the system. We also observe 9 out of 94 cases where outsiders react positively or negatively according to the impact of the pull request on the end-user experience and explain how it is either improving or downgrading their experience. Similarly, we observe cases where users are excited about a feature and waiting for its release. Thus, we suggest future studies to leverage external reactions to identify the characteristics of features that are more expected by users and build machine learning models to prioritize features that can be more relevant for outsiders.

Promise 5: Researchers can use the reactions of pull requests as soon as they are closed, especially when studying the perception of the core developers of a project. We observe that most of the reactions are added before the closing time of the pull requests. That said, practitioners, especially outsiders, keep reacting on a pull request after its closing time. So, when comparing pull requests, researchers should compare the pull requests that were closed at the same time-frame.

Limitation 1: While the reactions are promising data sources to mine, they do not capture the negative reactions of practitioners. For instance, the majority of reactions that are used are positive. So researchers aiming at investigating how to improve the pull requests mechanisms cannot investigate how practitioners are negative about

[28] https://github.com/RPCS3/rpcs3/pull/4816
[29] https://github.com/RPCS3/rpcs3/pull/5188

pull requests. Even when a practitioner negatively reacts to a pull request, that does not reflect the overall impression of a software system's stakeholders as negative reactions co-exist with positive ones. Therefore, the reactions dataset may not be crucial for studies like Pletea et al. [4] who leveraged sentiment analysis on pull requests' comments to find that the majority of security-related comments have a negative sentiment, nor studies like Freira et al. [23] who studied the impact of negative comments on developers' mood variation.

Limitation 2: Researchers leveraging reactions on pull requests need to distinguish between different types of users and should not combine reactions of different types of reactors. For instance, different stakeholders react differently to pull requests. While contributors react to the functional and implementation choices of pull requests, outsiders' reactions are more related to the end-user experience and the results of the feature testing. Thus, combining the reactions coming from different stakeholders can lead to misinterpretations. Prior work, for instance, differentiated comments based on the role of the commenters, Skriptsova et al. [8] used sentiment analysis to compare between the comments of experts and new developers. They find that comments are mostly neutral, yet negativity is slightly higher for newcomers. Eleni et al. [9] studied pull request comments as a social factor amongst other factors leading developers to abandon the RubyGems and npm software ecosystems, and distinguished between core and peripheral developers. As we suggest future work to replicate these studies using GitHub reactions (following Promise 1), we suggest separating reactions based on the role of reactors.

Limitation 3: A type of reactor can be a minority class. For instance, we observe for certain projects that the majority of reactions are coming just from outsiders, so leveraging reactions to investigate how the developers react to pull requests might not be as promising. For example, 88.65% of the Laravel's reactions are from outsiders, and the remaining reactions are mostly from peripheral and casual developers, leaving only 0.37% of the reactions that were given by the core contributors, and 0% by TF-authors.

Limitation 4: The evolution of reactions can be studied just for the core contributors and not outsiders. For instance, the core contributors react to multiple pull requests allowing researchers to investigate how the reactions of the core contributors change over time. For example, studying the relation between certain practices (e.g., refactoring code) and the impact of these practices on the reactions of a core contributor. However, that is not the case for outsiders as they react to a median of just one to two pull requests.

6 RELATED WORK

The closest studies to our paper focused on reactions and emojis in the context of Software Engineering (SE) activities (Section 6.1) and leveraging reactions for other non-software engineering related activities such as social media (Section 6.2). We discuss studies that leveraged the comments on pull requests (Section 6.3). Finally, we discuss prior work on sentiment analysis, which can be improved by leveraging users reactions (Section 6.4).

6.1 Studies on Reactions and Emojis in the SE Context

Prior studies [15, 19–21, 35, 45] on reactions and emojis focused on their popularity, impact on software artifacts and how can they support sentiments analysis in SE. Borges et al. [15] showed that GitHub users are increasingly using issue reactions, and the issues with reactions take more time to be addressed compared to the issues without any reactions. Lu et al. [19] observed that emojis are widely used in GitHub, mostly to convey positive and supportive sentiments in issues, pull requests and comments. According to the empirical study, emojis are also used to emphasize contents in README files. Note that GitHub emojis are different than GitHub reactions. There are more than 1,400 emojis that are available for free usage in the description and comment threads of issues and pull requests descriptions

as well as README files. Claes et al. [20] showed that emojis in Apache's and Mozilla's issue trackers are widely used by developers, mainly express joy, and can be leveraged for sentiment analysis. Chen et al. [45] proposed SEntiMoji, a learning approach leveraging emojis in software artifacts (e.g., issue comments) for sentiment analysis in SE, which outperforms the existing sentiment analysis methods on four representative benchmark datasets. Venigalla et al. [21] presented StackEmo, a Google Chrome plugin that appends emojis to comments based on their sentiment analysis.

Though existing studies have analyzed the usage of emojis and reactions in GitHub, no prior work has analyzed reactions on pull requests. Our work is aligned with the work of Teyon et al. [35]. They suggested through a study protocol to investigate the intentions behind the usage of reactions beyond reducing the commenting noise in the pull request discussions. In our paper, we also raise the problem of interpreting pull request reactions by considering multiple dimensions, i.e., the types of reactions (*Positive*, *Extremely Positive* or *Negative*), the roles of reactors (*Expert*, *Novice Contributor* or *Outsider*) as well as the relative react time in the lifespan of a pull request. We also examine our findings on eight types of pull requests separately.

6.2 Studies on Reactions and Emojis in Other Domains

In other domains, reactions and emojis are mainly studied in social media and communication platforms such as Twitter and Facebook, as ways to support sentiment analysis of user content [2, 6, 28, 34, 40, 43, 44]. Giuntini et al. [34] found that reactions in Facebook are correlated to emotional expressions and thus can be used to improve sentiment analysis algorithms. Davidov et al. [6] presented a sentiment classification framework based on data from Twitter (i.e., emojis and hashtags). Felbo et al. [2] showed that emojis in Twitter can also be leveraged to pre-train models for a better performance in learning representations of emotional content in texts. Xu et al. [40] studied specific emojis and words related to mourning on Twitter. They found that emojis alone are not enough to contextualize mourning in tweets, instead they should be combined with words. Chen et al. [43] investigated emoji usage in Android apps through a keyboard input app named Kika keyboard. They found that differences between genders regarding emoji usage are statistically significant. In another study, Chen et al. [44] presented ELSA, a framework for representation learning based on emojis, that can better identify sentiments in the source and target languages for cross-lingual sentiment classification task. Bai et al. [28] conducted a systematic review on the use and application of emojis and proposed ideas for future research.

Similar to the above studies, we argue that reactions should be considered together with textual feedback expressed in comments as they carry extra information. Different from them, we target a different context, i.e., pull requests, and consider more SE-specific factors like the reactor roles in a software project, lifespan of a pull request, and pull request type.

6.3 Studies on Comments in GitHub Pull Requests

Prior work [24, 37, 41, 42] studied comments as feedback to GitHub pull requests. Tsay et al. [37] analyze the influence of social and technical factors such as comments on the evaluation of contributions in GitHub. They found that pull requests with many comments are less likely to be accepted. Yue et al. [42] and Ying et al. [41] considered pull request comments as one feature for pull request reviewer recommendation. Middleton et al. [24] leverage pull request comments, amongst other forms of contribution, to predict whether a new developer become a long-term contributor of a software project.

Similar to the above studies, we analyze reactions as another type of community feedback on GitHub pull requests. Unlike the above studies, which leverage pull request comments to characterize the contribution and expertise of developers, we focus on interpreting reactions.

6.4 Sentiment Analysis in Software Engineering

A large number of prior work [1, 5, 7, 8, 10, 17, 22, 25, 27, 30, 31, 36, 38, 39, 46] studied the relation between sentiments and software developers' performance or productivity, and highlighted the importance of emotional awareness in SE communities. For example, Ortu et al. [22] showed that happier developers fix issues faster, and slower issues are associated with negative emotions. They also found that politeness increases productivity. Souza et al. [31] found that negative sentiment affects the result of the build process, and vice versa. Wrobel et al. [38] presented results of a survey on emotions experienced by developers in their work, where they find that positive and negative emotions have a direct impact on productivity. Later on Wrobel et al. [39] proposed a new approach using the method of participant observation to investigate the impact of emotions on productivity in IT projects. Huang et al. [46] built a model to predict community smells (e.g., Organizational Silo, Lone Wolf and Bottleneck) and their effect on developers. Mäntylä et al. [25] studied symptoms of productivity loss in software engineering using a novel metric, namely VAD, to measure Valence, Arousal and Dominance. Gachechiladze et al. [5] built a sentiment classifier which detects anger towards self, others and objects, based on 723 comments in Apache issue reports. Skriptsova et al. [8] used sentiment analysis to compare between the comments to contributions of old and new developers. They find that comments are mostly neutral, yet negativity is slightly higher for newcomers. Guzman et al. [10] presented an approach combining topic modeling with sentiment analysis to improve emotional awareness by extracting emotions expressed in collaboration artifacts (e.g., html tags, email headers, file attachments).

Another line of research focused on evaluating and improving sentiment analysis tools in SE context. First, Jongeling et al. [30] evaluated the consistency of the results of different sentiment analysis tools and compared them with human evaluations. They also investigated the impact of the choice of such tools on software engineering studies. Novielli et al. [27] benchmarked three SE customized sentiment analysis tools in terms of performance. They found that sentiment analysis tools in SE can be reliable and that supervised learning algorithms show the best performance. Lin et al. [1] presented limitations of applying the existing sentiment analysis tools that are developed for general text data in SE data. Zhang et al. [36] refined pre-trained Transformer-based models for sentiments analysis in SE to outperform the state-of-the-art tools by 6,5 to 35,6% in terms of macro/micro-averaged F1 scores. Biswas et al. [7] also made a significant improvement over the state-of-the-art sentiment analysis tools by leveraging BERT (a language representation model) with larger data sets. Sun et al. [17] presented a novel approach by considering sentence structures. Their approach was more generalizable than machine learning baselines and outperformed two dictionary-based baselines (e.g., SentiStrength).

We believe our analysis of pull request reactions can complement the above line of work to capture the sentiment of different community members in the software development process.

7 THREATS TO VALIDITY

7.1 External validity

Threats to external validity concern the generalizability of our results. While our work is based on six open-source projects in GitHub, of varying size with different domains and programming languages, we do not generalize our results

to other software systems. Despite the relatively small number of selected projects, we investigated a total of 380k reactions on 63k pull requests. That said, we suggest future studies further explore our findings in other projects.

When interpreting the reaction usage in RQ1 and RQ2 via our qualitative analyses, we leverage the comments that are given by reactors as the optimal threshold to understand the intention behind practitioners' reactions. Even as we observed in the preliminary study that most reactors do not leave a comment making the reactions an exclusive dataset, we explored the 14k existing reactions for which reactors left a comment and from which we selected three different representative samples to better understand the pull requests reactions. That said, our findings might not represent all reactors.

7.2 Internal validity

Threats to internal validity concern confounding factors that could affect our reported findings. In RQ1, we define reaction types "Hooray", "Hear" and "Rocket" as extremely positive reactions. Such categorization might not be ideal for all projects, and our reported results based on this categorization would vary if one changes the sentiment classification of reactions. However, we believe that a change in the classification would not alter the take-home message that our studied projects' reactors mostly use positive reactions.

In RQ2 and RQ3, we leverage a weighted univariate clustering algorithm to identify a dynamic threshold of commit number for identifying if a reactor is a core, peripheral or casual contributor in a project. Choosing a different threshold setup method would lead to a different result. To mitigate such a risk, we automatically identify the optimal number of clusters for each month and project separately to end up with the median and average optimal number of clusters over the six projects (which were estimated at *K=4* and *K=4.2*, respectively) using the elbow method.

Another internal threat to validity is related to our classification of contributors based on the number of their commits as classifying contributors based on other activities, such as code review, architectural designs, management, operation, etc. can lead to different results. In our study, we focus on classifying contributors based on just their number of commits similarly to prior work [12, 16]. We encourage future work to replicate our study using other contributions.

8 CONCLUSION

In this paper, we aim to better understand how to interpret reactions based on their type (Positive, Extremely Positive, Negative or Neutral), the status of their author (TF-author, core contributor, peripheral contributor, casual contributor or outsider) as well as their relative time (i.e., around the opening time or the closing time of the pull request). To do so, we performed quantitative and qualitative analyses on six open-source projects on GitHub and their reactions.

We observe that our studied projects' reactors leverage pull requests reactions to convey more positive reactions than negative ones. A median of only 1.95% of pull request reactions are negative. Yet that does not reflect the general impression of practitioners about a pull request, as a median of 41.42% to 50% of the pull requests with a negative reaction do have positive reactions too. Moreover, we observe that positive and extremely positive reactions are used to show appreciation, agreement and excitement for a pull request. We also observe that 37.90% of the practitioners leveraging positive reactions are neutral in their comments. On the other hand, negative reactions are generally used to express disagreement with feature modifications, refactorings and removals. Furthermore, new features and bug fixes can also receive negative reactions when they have more downsides than upsides or when they are considered as a poor solution to an issue. These given reactions are provided by both outsiders and contributors. In fact, outsiders are as active as the contributors of a project on reacting to pull requests. We observe that outsiders have more reactions (a median of 81.36%) than contributors in five out of six projects (Julia, Laravel, Node, RPCS3 and Rust). We also find

that contributors' reactions are related to the functional and implementation choices in a pull request, while outsiders' reactions are more related to the impact of the pull request on the end user experience and the results of the feature testing. Finally, we observe that most reactions (between 62.53% and 82.99%) are made before the closing time of pull requests, especially by a project's core contributors, while peripheral contributors, casual contributors and outsiders can still react to a pull request after closing it.

Our take-home message is that researchers should consider the possible differences between the pull requests reactions that can be manifested due to the popularity of certain types of reactions over other ones, the status of reactors according to the studied project, and the time at which practitioners react to pull requests. That is because our results found different reaction patterns over the three dimensions. Our findings also shed light on the usage of reactions and provide a taxonomy of the possible intentions behind the usage of pull requests reactions, which can inspire future studies on better leveraging pull request reactions (e.g., what are the characteristics of new features that outsiders appreciate).

REFERENCES

[1] Lin Bin, Zampetti Fiorella, Bavota Gabriele, Di Penta Massimiliano, Lanza Michele, and Oliveto Rocco. 2018. Sentiment Analysis for Software Engineering: How Far Can We Go?. In Proceedings of the 2018 International Conference on Software Engineering. 94–104.

[2] Felbo Bjarke, Mislove Alan, Søgaard Anders, Rahwan Iyad, and Lehmann Sune. 2017. Using Millions of Emoji Occurrences to Learn Any-Domain Representations for Detecting Sentiment, Emotion and Sarcasm. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.

[3] Jake Boxer. 2016. Add Reactions to Pull Requests, Issues, and Comments. Retrieved 30 May 2023 from https://github.blog/2016-03-10-add-reactions-to-pull-requests-issues-and-comments/

[4] Pletea Daniel, Vasilescu Bogdan, and Serebrenik Alexander. 2014. Security and Emotion: Sentiment Analysis of Security Discussions on GitHub. In Proceedings of the 2014 Working Conference on Mining Software Repositories. 348–351.

[5] Gachechiladze Daviti, Lanubile Filippo, Novielli Nicole, and Serebrenik Alexander. 2017. Anger and its Direction in Collaborative Software Development. In Proceedings of the 2017 IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track. 11–14.

[6] Davidov Dmitry, Tsur Oren, and Rappoport Ari. 2010. Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. In Coling 2010: Posters. 241–249.

[7] Biswas Eeshita, Karabulut M. Efruz, Pollock Lori, and Vijay-Shanker K. 2020. Achieving Reliable Sentiment Analysis in the Software Engineering Domain Using BERT. In Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution. 162–173.

[8] Skriptsova Ekaterina, Voronova Elizaveta, Danilova Elizaveta, and Bakhitova Alina. 2019. Analysis of Newcomers Activity in Communicative Posts on GitHub. In Proceedings of the 2019 International Conference on Digital Transformation and Global Society. 452–460.

[9] Constantinou Eleni and Mens Tom. 2017. An Empirical Comparison of Developer Retention in the RubyGems and Npm Software Ecosystems. Innovations in Systems and Software Engineering 13 (2017), 101–115.

[10] Guzman Emitza and Bruegge Bernd. 2013. Towards Emotional Awareness in Software Development Teams. In Proceedings of the 2013 Joint Meeting on Foundations of Software Engineering. 671–674.

[11] Huq S. Fatiul, Sadiq A. Zafar, and Sakib Kazi. 2019. Understanding the Effect of Developer Sentiment on Fix-Inducing Changes: An Exploratory Study on GitHub Pull Requests. In Proceedings of the 2019 Asia-Pacific Software Engineering Conference. 514–521.

[12] Avelino Guilherme, Constantinou Eleni, Valente M. Tulio, and Serebrenik Alexander. 2019. On The Abandonment And Survival Of Open Source Projects: An Empirical Investigation. In Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. 1–12.

[13] Avelino Guilherme, Passos Leonardo, Hora Andre, and Valente M. Tulio. 2016. A Novel Approach For Estimating Truck Factors. In Proceedings of the 2016 IEEE 24th International Conference on Program Comprehension. 1–10.

[14] Pinto Gustavo, Steinmacher Igor, and Gerosa M. Aurélio. 2016. More Common Than You Think: An In-Depth Study Of Casual Contributors. In Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering. 112–123.

[15] Borges Hudson, Brito Rodrigo, and Valente M. Tulio. 2019. Beyond Textual Issues: Understanding the Usage and Impact of GitHub Reactions. In Proceedings of the 2019 Brazilian Symposium on Software Engineering. 397–406.

[16] Crowston Kevin, Wei Kangning, Li Qing, and Howison James. 2006. Core And Periphery In Free/Libre And Open Source Software Team Communications. In Proceedings of the 2006 Annual Hawaii International Conference on System Sciences. 118a–118a.

[17] Sun Kexin, Gao Hui, Kuang Hongyu, Ma Xiaoxing, Rong Guoping, Shao Dong, and Zhang He. 2021. Exploiting the Unique Expression for Improved Sentiment Analysis in Software Engineering Text. In Proceedings of the 2021 IEEE/ACM International Conference on Program Comprehension.

149–159.

[18] Krippendorff Klaus. 2004. Reliability in content analysis: Some common misconceptions and recommendations. Human Communication Research 30, 3 (2004), 411–433.

[19] Xuan Lu, Yanbin Cao, Zhenpeng Chen, and Xuanzhe Liu. 2018. A First Look at Emoji Usage on GitHub: An Empirical Study. ArXiv Preprint arXiv:1812.04863 (2018).

[20] Claes Maëlick, Mäntylä Mika, and Farooq Umar. 2018. On the Use of Emoticons in Open Source Software Development. In Proceedings of the 2018 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. 1–4.

[21] Venigalla A.S. Manasa and Chimalakonda Sridhar. 2021. StackEmo: Towards Enhancing User Experience by Augmenting Stack Overflow With Emojis. In Proceedings of the 2021 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 1550–1554.

[22] Ortu Marco, Adams Bram, Destefanis Giuseppe, Tourani Parastou, Marchesi Michele, and Tonelli Roberto. 2015. Are Bullies More Productive? Empirical Study of Affectiveness vs. Issue Fixing Time. In Proceedings of the 2015 IEEE/ACM Working Conference on Mining Software Repositories. 303–313.

[23] Freira Mateus, Caetano Josemar, Oliveira Johnatan, and Marques-Neto Humbertor. 2018. Analyzing The Impact Of Feedback In GitHub On The Software Developer's Mood. In Proceedings of the 2018 International Conference on Software Engineering & Knowledge Engineering. 445–450.

[24] Justin Middleton, Emerson Murphy-Hill, Demetrius Green, Adam Meade, Roger Mayer, David White, and Steve McDonald. 2018. Which Contributions Predict Whether Developers Are Accepted Into GitHub Teams. In Proceedings of the 2018 IEEE/ACM International Conference on Mining Software Repositories. 403–413.

[25] Mäntylä Mika, Adams Bram, Destefanis Giuseppe, Graziotin Daniel, and Ortu Marco. 2016. Mining Valence, Arousal, and Dominance: Possibilities for Detecting Burnout and Productivity?. In Proceedings of the 2016 International Conference on Mining Software Repositories. 247–258.

[26] De Zoysa Nalin. 2020. Analysis of Textual and Non-Textual Sources of Sentiment in Github. Master's thesis. University of Waterloo, Waterloo, Ontario, Canada.

[27] Novielli Nicole, Girardi Daniela, and Lanubile Filippo. 2018. A Benchmark Study on Sentiment Analysis for Software Engineering Research. In Proceedings of the 2018 IEEE/ACM International Conference on Mining Software Repositories. 364–375.

[28] Bai Qiyu, Dan Qi, Mu Zhe, and Yang Maokun. 2019. A Systematic Review of Emoji: Current Research and Future Perspectives. Frontiers in Psychology 10 (2019), 2221.

[29] Nainggolan Rena, Resianta Perangin-angin, Emma Simarmata, and Astuti F. Tarigan. 2019. Improved the Performance of the K-means Cluster Using the Sum of Squared Error (SSE) Optimized by Using The Elbow Method. In Journal of Physics: Conference Series. 012–015.

[30] Jongeling Robbert, Datta Subhajit, and Serebrenik Alexander. 2015. Choosing Your Weapons: On Sentiment Analysis Tools for Software Engineering Research. In Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution. 531–535.

[31] Souza Rodrigo and Silva Bruno. 2017. Sentiment Analysis of Travis CI Builds. In Proceedings of the 2017 IEEE/ACM International Conference on Mining Software Repositories. 459–462.

[32] Joe Song and Haizhou Wang. 2020. Optimal (Weighted) Univariate Clustering. Retrieved 30 May 2023 from https://search.r-project.org/CRAN/refmans/Ckmeans.1d.dp/html/Ckmeans.1d.dp.html

[33] Wang Song, Chetan Bansal, Nachiappan Nagappan, and Adithya A. Philip. 2019. Leveraging Change Intents for Characterizing and Identifying Large-Review-Effort Changes. In Proceedings of the 2019 International Conference on Predictive Models and Data Analytics in Software Engineering. 46–55.

[34] Giuntini F. Taliar, Larissa Pires, Luziane D. F. Kirchner, Denise A. Passarelli, Andrew T. Campbell Maria D. J. D. Dos Reis, and Jo Ueyama. 2019. How Do I Feel? Identifying Emotional Expressions on Facebook Reactions Using Clustering Mechanism. IEEE Access 7 (2019), 53909–53921.

[35] Son Teyon, Xiao Tao, Wang Dong, Kula R. Gaikovina, Ishio Takashi, and Matsumoto Kenichi. 2021. More Than React: Investigating The Role of Emoji Reaction in GitHub Pull Requests. ArXiv Preprint arXiv:2108.08094 (2021).

[36] Zhang Ting, Xu Bowen, Thung Ferdian, Haryono S. Agus, Lo David, and Jiang Lingxiao. 2020. Sentiment Analysis for Software Engineering: How Far Can Pre-Trained Transformer Models Go?. In Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution. 70–80.

[37] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence Of Social And Technical Factors For Evaluating Contribution In GitHub. In Proceedings of the 2014 International Conference on Software Engineering. 356–366.

[38] Michal R. Wrobel. 2013. Emotions in the Software Development Process. In Proceedings of the 2013 International Conference on Human System Interactions. 518–523.

[39] Michal R. Wrobel. 2016. Towards the Participant Observation of Emotions in Software Development Teams. In Proceedings of the 2016 Federated Conference on Computer Science and Information Systems. 1545–1548.

[40] Xu Xinyuan, Manrique Ruben, and Pereira N. Bernardo. 2021. RIP Emojis and Words to Contextualize Mourning on Twitter. In Proceedings of the 2021 ACM Conference on Hypertext and Social Media. 257–263.

[41] Haochao Ying, Liang Chen, Tingting Liang, and Jian Wu. 2016. Earec: Leveraging Expertise And Authority For Pull-request Reviewer Recommendation In GitHub. In Proceedings of the 2016 IEEE/ACM International Workshop on CrowdSourcing in Software Engineering. 29–35.

[42] Yue Yu, Huaimin Wang, Gang Yin, and Charles X Ling. 2014. Who Should Review This Pull-request: Reviewer Recommendation To Expedite Crowd Collaboration. In Proceedings of the 2014 Asia-Pacific Software Engineering Conference. 335–342.

[43] Chen Zhenpeng, Lu Xuan, Ai Wei, Li Huoran, Mei Qiaozhu, and Liu Xuanzhe. 2018. Through a Gender Lens: Learning Usage Patterns of Emojis From Large-Scale Android Users. In Proceedings of the 2018 World Wide Web Conference. 763–772.

[44] Chen Zhenpeng, Cao Yanbin, Yao Huihan, Lu Xuan, Peng Xin, Mei Hong, and Liu Xuanzhe. 2021. Emoji-Powered Sentiment and Emotion Detection from Software Developers' Communication Data. ACM Transactions on Software Engineering and Methodology 30, 2 (2021), 1–48.

[45] Chen Zhenpeng, Cao Yanbin, Lu Xuan, Mei Qiaozhu, and Liu Xuanzhe. 2019. SEntiMoji: An Emoji-Powered Learning Approach for Sentiment Analysis in Software Engineering. In Proceedings of the 2019 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 841––852.

[46] Huang Zijie, Shao Zhiqing, Fan Guisheng, Gao Jianhua, Zhou Ziyi, Yang Kang, and Yang Xingguang. 2021. Predicting Community Smells' Occurrence on Individual Developers by Sentiments. In Proceedings of the 2021 IEEE/ACM International Conference on Program Comprehension. 230–241.