What are the Most Common Types of Pull Requests Between Bug Fixes, New Features and Refactoring?

Extended Abstract

Austin Kelly
Northern Arizona University
Flagstaff, AZ, USA
ak678@nau.edu

Karsten Nguyen
Northern Arizona University
Flagstaff, Arizona, USA
khn6@nau.edu

Austin Collins
Northern Arizona University
Flagstaff, Arizona, USA
aac332@nau.edu

ABSTRACT

In this document, we discuss our methods and findings from researching common types of pull requests. We mine data from GitHub using its Application Program Interface, and search for commonalities based on key phrases. In our study, we decided on three main categories: bug fixes, new features and refactoring. This paper is designed to present our findings in an unbiased manner that does not favor our own hypothesis and interpretation. We hope to provide valuable data for future researchers who pursue analyzing attributes of pull requests.

CCS CONCEPTS

• **Software and its engineering** → *Collaboration in software development*;

KEYWORDS

GitHub, Pull Requests, Refactoring, Features, Bug Fixes

ACM Reference Format:
Austin Kelly, Karsten Nguyen, and Austin Collins. 2018. What are the Most Common Types of Pull Requests Between Bug Fixes, New Features and Refactoring?: Extended Abstract. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Writing code that is readable and easy to understand can speed up the code review turnaround time of ones pull requests. In the future, reviewers or future contributors can be more productive when they are able to understand what someone's code is doing just by reading and analyzing it. Readable code is helpful beyond just the code review process and allows future developers to quickly adapt to projects. In most code, the majority of the time someone reads code it is either up to the developer or their peers to debug, add features and refactor, long after it was added.

In software development, bugs are most often introduced at the boundaries between applications. Unit testing usually works for testing the interactions that an open-source admin knows about. Sometimes they are not able to grasp all the interesting, and often unexpected ways that a community might put a library to use. Poor mistakes in early-on pull requests often turn out to be code smells, which are described as code structures that may cause harm to the software later on. Code smells will force writers to go back to refactor the code. Redoing things is fundamental to every creative process. According to Jim Highsmith, "Refactoring may be the single most important technical factor in achieving agility." Findings show that programmers fault reduction effort can be directed to refactoring those code smells mostly likely to be related to bugs.

Before pull requests were introduced, bug tracking was something that existed for quite a while. There are already multiple studies that discuss the prioritization of bug reports and issues. GitHub's pull requests also allows itself to a variety of usage patterns. Along with basic patch submission, pull requests can also be used as a requirements or design discussion tool. In this case, the title lays out the overall purpose of the pull requests, and the comments serve as a discussion board for soliciting the opinions of other developers while a new feature is being implemented. Pull requests can also be used as a progress tracking tool. Pull requests are a crucial aspect of GitGub today, and we wanted to gather more information on how they are being used. We have listed our questions of research below.

Q1: What are the most common type of pull request between bug fixes, new features and refactoring?

We came up with this research question to discover any existing factors or similarities that help determine the most common types among the pull requests tested. This is mainly attributed to the lack of research, to the best of our knowledge regarding resolved bugs, new features and refactoring code respective to pull requests. We aim to be the first to research this specific topic and ideally, our findings will be of relative benefit wherever possible.

Q2: What research, outside of our own, has been conducted on code smells and refactoring to fix bugs?

We want to know of previous analyses of code smells and refactoring to have a better understanding of what kind of research was carried out and what it set out to achieve. This should inform us of parts to look out for while researching code smells and refactoring that are associated to pull requests themselves so we can refine our research methods as needed. In addition, this will shed light on why further and more extensive research was not pursued by those who previously took on the task.

Q3: What are the most common types of bugs that are successfully fixed by the most common type of pull requests?

This was an interesting question because it is meant to show any cause and effect relationships among a given bug fix and pull request. If there is a relationship, we can track it back to the source

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of the bug and find out why and how it occurred in the first place and possibly provide a suggestion to its resolution. We can then attribute our findings to the topics lack of research.

Our primary goal is to research common pull requests between bug fixes, new features and refactoring. Our reasoning for this is due to there being very little research of the topic in the past. This will hopefully demonstrate useful results that can serve beneficial purposes for existing bug fixes associated with pull requests. To do so, we will need to review previous research of our topic. We will then compare the research questions, problems, purposes, goals, and findings to on another. This will inform us of went when well and not so well for them so we know of what specifications should and should not be taken on. The comparison will also give us an educated approximation as to why no further research or action was brought about, given their results. We will then need to heavily utilize the GitHub API to find bug fixes, new features and code refactoring. Before moving on, we will narrow our search by finding pull requests linked to those specific bug fixes, new features and refactorization. This will make our research more finite and will narrow it down to exactly what we are looking for. Finally, we will cross reference pull request types to type of bug fixes which will highlight the most common of each. This will show similarities, if any, among aspects like code smells, refactoring techniques or even additional factors that slipped through the cracks.

Our contributions will be a further investigation of what was left out of previous research on the same topics. We will essentially pick up where others left off. The novelty of our findings will be that ours specific to pull requests and related bug fixes. This will produce what is expected to be the first research of its kind regarding the most common type of pull request between bug fixes, new features and refactoring. It can be used for future developments as either a starting point or a reference for a more robust study. While possible limitations are not fully known at this time, we hope our findings will tap into a new market of where others did not previously look into, and discover beneficial information regarding pull requests and bug fixes.

2 RELATED WORK

We reviewed similar researches of most common pull request among bug patches, new features, and refactoring in this section. Studies that consider a common theme upon pull request are bare. Most studies that incorporate pull request, are usually criticizing the structure and how they should have a guideline of what a pull request should look like. The most related work we can find was "An insight into pull request on GitHub" [4]. Given the increasing number of unsuccessful pull requests in GitHub projects, insights into the success and failure of these requests are essential for the developers. In this paper, Mohammad Masudur Rahman and Chanchal K. Roy provide a comparative study between successful and unsuccessful pull requests made to 78 GitHub base projects by 20,142 developers from 103,192 forked projects. In the study, they analyze pull request discussion texts, project specific information and developer specific information in order to report useful insights, and use them to contrast between successful and unsuccessful pull requests. They believe the study will help developers overcome the issues with pull requests in GitHub, and project administrators with informed decision making. [4] Roy and Rahman research is about the relationship between successful and failed pull requests, while are our research is related to common pull request. Among developers are ignored.

Their findings can be narrowed down and categorized into eight topics being Recursion and Refactoring, Database Query Execution, Arrays and functions, Actor Model, OOP Paradigm, and Space Indents. Even though these technical issues proved useful for them, we are researching common pull requests which only relates to one of their topics, refactoring. They concluded recursion and refactoring were brought up in 18.35% documents, as a pair and not individually. This was the highest result of the eight topics most discussed which caught our attention as 92 other topics collectively consisted of 4% of the discussion. Upon further research, it was made clear to us that this was not saying recursion and refactoring were the most effective but the most talked about. While the two had a high success rate they also had a high fail rate. This can be attributed to Recursion and refactoring being the most used among pull requests associated with code smells, bug patches, and new features as refactoring is a popular "go-to" strategy. These underlying problems are often the cause of code smells which can be incredibly hard to fix later on and are important to catch early. In the end, the previous research we found has proven to be a valuable reference to our own studies.

3 METHODS

3.1 Introduction

Git is a popular version control system for open source development created by Linus Torvalds. Users create repositories which use a branching system that allows for easy and transparent collaboration. Pull requests let developers tell others about changes they have pushed to a branch in a repository on GitHub. Once a pull request is opened, developers can discuss and review the potential changes with collaborators and add follow-up commits before their changes are merged into the base branch. A detailed commit history allows reviewers to view the differences between one commit to another so they can easily verify whether their concerns have been addressed. The commits will be squashed when the pull request is merged. There is an issue tracking system set up for repositories which often have comment threads, but these are only limited to a few categories. For our research, we decided on searching keywords in pull request titles to determine what type of requests were being made. This is a different approach to get a broader understanding of what requests are typically comprised of. We initially randomly picked 15 repositories to derive our pull request from. Once we had our set projects, we wrote a python script to find all the titles of the pull requests in each of those 30 categories. We used the GitHub API to find keywords that mentioned bug fixes, new features and refactoring. We also added some variations of our keywords to a list to account for different ways of speaking. Our program appends the data to a CSV file after each repository is entered. Each line of the CSV is formatted in the order: REPO-TITLE, TOTAL-TITLES-WITH-KEYWORDS, BUG-FIXES, NEW-FEATURES, REFACTORING. The CSV was then put into R studio, where variables were set and compared to find the mean and standard deviation. We did an

analysis on the first 15 repositories, the 15 we handpicked, and then all 30 as a whole.

Below are the results from each CSV.

Chosen

	V1	V2	V3	V4	V5
1	diveintodeeplearning/d2l-en	12	8	9	2
2	GoogleChromeLabs/squoosh	18	12	5	1
3	Antergos/Cnchi	21	15	5	1
4	minimaxir/big-list-of-naughty-strings	12	5	7	0
5	sindresorhus/awesome	113	7	106	0
6	torvalds/linux	30	23	7	0
7	firecracker-microvm/firecracker	67	18	40	9
8	facebook/facebook-android-sdk	14	9	5	0
9	PyGithub/PyGithub	54	15	39	0
10	mozilla/thimble.mozilla.org	231	137	87	7
11	brigade/scss-lint	46	23	23	0
12	airbnb/lottie-android	32	16	12	4
13	apache/incubator-superset	812	537	241	34
14	ruby/rdoc	34	21	13	0
15	linkedin/css-blocks	13	8	2	3

Figure 1: Chosen Repositories

Random picks

	V1	V2	V3	V4	V5
1	d3/d3	132	75	57	0
2	electron/electron	1148	664	390	94
3	vim/vim	208	131	77	0
4	ethereum/aleth	35	17	14	4
5	randombit/botan	131	79	48	4
6	nasa/openmct	103	53	45	5
7	ianstormtaylor/slate	225	149	60	16
8	kanaka/mal	100	26	71	3
9	infernojs/inferno	28	22	5	1
10	palantir/plottable	46	21	24	1
11	Semantic-Org/Semantic-UI	87	60	27	0
12	BoltsFramework/Bolts-ObjC	20	13	7	0
13	JohnCoates/Aerial	16	12	4	0
14	apple/swift-evolution	141	54	85	2
15	dvyukov/go-fuzz	27	14	13	0

Figure 2: Random Repositories

3.2 Qualitative Analysis

We performed a qualitative analysis on the titles of pull requests in GitHub. The first search was to simply get results of repositories of any kind. Fifteen random and fifteen systematically chosen. We only used repositories with over 50 pull requests so we would have a enough data within each one. It was apparent some of the data would not be free from bias due to our keywords only being found in a portion of the total pull requests. Concurrently, each member would choose five random and five and systematic repositories. For the systematic repositories, we were looking for higher amounts of pull requests but nothing drastic as it would undermine the randomly chosen pull requests. We decided on using only closed pull requests, and there needed to be at least 50 total in a repository to be valid for use. We then began the mining process that searched for words and phrases, exactly like or similar to what would constitute a tally for that category. The four keywords chosen for the bug category were: "bug", "fix", "patch" and "smell." For new features we chose: "new", "feature","add" and "implement." Finally, we decided on "refactor", "rewr", "revise" and "restructure" for refactoring. Note that "rew" would cover both the present and past tense of "rewrite."

The analysis of pull request titles matched our expectation of which topic would have the least and most. Bug patches resulted in the majority and new features was not far behind. However, refactored came up short with only a small portion. All though this was the order we anticipated, we did not expect "refactor", or anything along the lines of it, to yield as few results as it did. For refactoring alone, we even used an extra keyword, since "rewr" covers both tenses. In the end, it still came out way below our expectations.

3.3 Quantitative Analysis

Our quantitative analysis consisted of 30 total repositories. 15 of them were chosen, and 15 were pulled randomly. We then mined all the closed pull requests and compared the types between the the chosen and random sets of data. During our analysis, we broke the pull request in three categories: bug patches, new features and refactoring. We chose 4 keywords for each type. We measured the ratio of the pull request type over the total amount of titles that contained any keyword. Below is a graph of our findings.

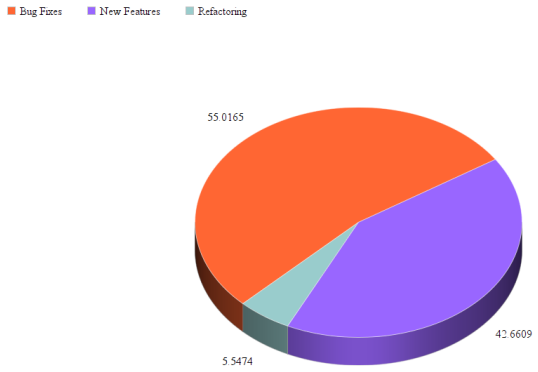

Figure 3: Type Distribution

4 RESULTS

Between bug fixes, new features, and refactoring related titles, the most common pull request was bug fixes, taking 55.0165% of all titles that contained a keyword. Closely behind, are new features with 42.6609%. Refactoring had very few results comparatively, and took up 5.5474% of the total. Below, is a table for the respective mean and standard deviation in each category.

We compared the 15 chosen repositories with random 15 repositories, and calculated the P value with the formula below. None

Bug Fixes	All 30	Picked 15	Random 15
Mean	55.02%	53.13%	56.90%
Standard Deviation	0.1683	0.1389	0.1388
New Features	All 30	Picked 15	Random 15
Mean	42.66%	45.21%	40.11%
Standard Deviation	0.1845	0.1967	0.1379
Refactoring	All 30	Picked 15	Random 15
Mean	5.55%	5.55%	2.99%
Standard Deviation	0.0584	0.0741	0.0354

Figure 4: Distribution Between Sets

of the categories were considered significant using P<0.05 as our metric for significance.

$$t = \frac{\bar{x_1} - \bar{x_2}}{se(\bar{x_1} - \bar{x_2})}$$

Figure 5: T Test Formula

5 DISCUSSION

Overall, we were not surprised that there was no significance between the chosen and random sets of repositories. While differences are to be expected, anything extreme would likely indicate an error with our calculations. We accurately predicted that bug fixes would be the most common type of pull request, as this was our general experience using GitHub ourselves. We also predicted that refactoring would be the least, because each refactoring pull request is typically larger, and usually compiled into a single branch. We did not expect to see results as low as 5.5% for this topic. Overall, we feel this was a successful analysis in terms of our expectations.

6 THREATS TO VALIDITY

Bug patches were the most common type of pull request according to our findings, but it is important to understand the process and potential flaws with our research. We found two primary reasons for flaws in our findings.

Firstly, while we collaboratively decided on the what we believed to be the best keywords for indicating each type, different results would occur with alternate choices. There are countless factors that could influence this, such as a user's regional linguistics or title formatting conventions set by managers of a repository. For this reason, our results should be viewed in terms of the overall category makeup, as the percentages would vary based on different wording.

Secondly, we are only taking in pull requests that contain a keyword from any of our categories. This data set is a smaller portion of the results, and the untested pull request titles could most likely be categorized if more robust keywords were used. We decided on 4 keywords per category, as we wanted to keep each one focused, minimizing the overlap that would result from broader words. Because of this flaw, we decided to measure our results out of the total number of titles which contained any keyword, rather than all of the total pull requests in the repository. While many pull requests are left out, this still creates transparent results for the categories found based on our metrics.

7 CONCLUSION

In this study we used a mixed-method approach to determine the most common type of pull request between bug fixes, new features, and refactoring. The plan called for selecting repositories in two different manners with respect to how much data we would need to produce valid findings. We did this by randomly selecting 15 repositories and systematically selecting another 15. The process included a qualitative analysis that required a biased and unbiased approach in multiple respects to the titles. This was conducted by mining GitHub repositories, and scanning all the closed pull request titles for relevant words and phrases that referenced our three main categories.

In conclusion, we believe we found sufficient evidence to determine the most common types of pull requests between bug fixes, new features, and refactoring. As mentioned, we exclusively used repositories with more than 50 closed pull requests to specify our search area. This allowed for more practical results that helped to mitigate potential underlying factors and outliers of any kind. We believe our findings do not demonstrate a lack of data as we made appropriate use of random and systematic selection to even the odds. Thus, we have determined the most common types of pull requests between bug fixes, new features, and refactoring to be bug fixes. Our findings should be taken for further interpretation on the category, and while there are flaws we have outlined, we took several approaches to mitigate them, and have explained our process thoroughly.

Source code available here: LINK

REFERENCES

[1] Georgios Gousios and Andy Zaidman. 2018. A Dataset for Pull Request Research. Retrieved. *Commun. ACM* (November 2018). https://www.testroots.org/assets/papers/pullreqs-dataset.pdf

[2] Eirini Kalliamvakou. 2018. The Promises and Perils of Mining GitHub. *Commun. ACM* (November 2018). https://kblincoe.github.io/publications/2014_MSR_Promises_Perils.pdf

[3] Francesco Pontillo. 2018. Github Data Mining: 101 (November 2017). *Commun. ACM* (December 2018). https://blog.novoda.com/github-data-mining-101/

[4] Mohammad Masudur Rahman and Chanchal K. Roy Plant. 2018. An Insight into the Pull Requests of GitHub. *Commun. ACM* (November 2018). https://www.cs.usask.ca/~croy/papers/2014/RahmanMiningGithubMSR2014.pdf

[4] [1] [2] [3]