An Insight into the Pull Requests of GitHub

Mohammad Masudur Rahman Chanchal K. Roy
University of Saskatchewan, Canada
{mor543, ckr353}@mail.usask.ca

ABSTRACT

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, we 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, we analyze pull request discussion texts, project specific information (e.g., domain, maturity), and developer specific information (e.g., experience) in order to report useful insights, and use them to contrast between successful and unsuccessful pull requests. We believe our study will help developers overcome the issues with pull requests in GitHub, and project administrators with informed decision making.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Analysis—*maintenance, open source development*

General Terms

Theory

Keywords

Commit comments, pull request, topic model

1. INTRODUCTION

GitHub, a popular web-based source code hosting service, provides a convenient way for the software developers to collaborate on open source software development with one another. In order to contribute, a developer either creates her own repository or forks from a *base repository*, and continues her work. GitHub maintains the source code and associated content (e.g., committed code, commit comments) for both base and forked repositories separately. The idea is to allow the developer to continue her work without reporting every single commit instantly to the *base repository*. The approach helps her to avoid the *frequent merge conflicts* with other developers of the project, and also provides flexibility in the development. Once the developer completes a milestone (e.g., module) involving several commits to the forked repository, she makes a *pull request* to the owner (i.e., administrator) of the *base repository*, and attempts to get her commits merged. Then other members of the project analyze the posted commits, review the code, and the streams of discussion among them are captured in terms of *pull request commit comments*. The posted commits are generally accepted if both the merge operation succeeds without conflicts and the identified concerns by other developers are properly addressed. Unfortunately, not all the pull requests succeed and there are growing concerns of how to make successful pull requests in GitHub [2]. In this research, we perform a comparative study between successful (i.e., merged with base repository) and unsuccessful (i.e., failed to merge with base repository) pull requests by analyzing different related artifacts such as the pull request discussion texts (i.e., code review comments), pull request history, and project and developer specific statistics. The study can provide important insights into the success and the failure of a pull request at GitHub repositories.

A number of existing studies focus on the analysis of email messages, bug reports, MSR papers, and commit messages of source code repositories [3, 4] for various software maintenace activities. Our work is closely related with the study by Hindle et al. [4], where they extract the hidden topics from the commit comments of a code repository, and then map to different cross-project non-functional requirements in order to analyze the software maintenance activities. In this paper, we examine the pull request discussion texts along with project and developer specific information using a machine learning technique and then report the frequent technical issues and inefficiencies in the source code hosted at GitHub. We use MSR dataset [2], and collect information about 78,517 pull requests made to 78 base projects by 20,142 developers from 103,192 forked projects. We extract 100 underlying topics that the reported issues of 9,421 pull requests (containing pull request discussion) are based on. In order to extract the topics, Latent Dirichlet Allocation (LDA) with Gibbs sampling is used, and we manually label 64 topics. We identify eight frequently discussed technical topics, and manually analyze the pull request discussion texts for useful insights. From the analysis of project and developer specific information, our study reports that programming language and domain specific factors can influence the success and failure rates of the pull requests. More

MSR'14, May 31 – June 1, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2863-0/14/05...$15.00
http://dx.doi.org/10.1145/2597073.2597121

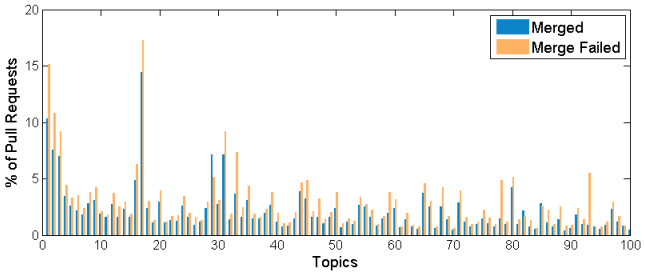

Figure 1: Pull Requests vs. Topics

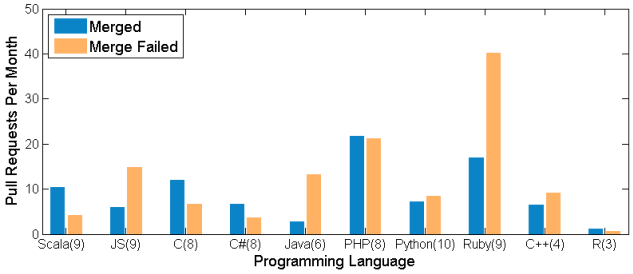

Figure 2: Pull Requests vs. Languages

importantly, it finds out that success rate of pull requests for a project degrades comparatively with a large number developers (e.g., more than 4,000) or a large number of forked projects (e.g., more than 3,000). While the extracted frequent technical topics and language or domain specific insights can help the developers with successful pull requests, project and developer specific insights can aid the GitHub project administrators with informed decision making in the management of pull requests, projects and developers involved.

2. DATASET

MSR challenge dataset [2] contains 78,955 pull requests (33,910 merged and 45,045 merge failed) made to 88 *base projects* by 20,142 developers. Among them, 9,601 pull requests (4,091 merged and 5,510 merge failed) made to 78 base projects contain *pull request commit comments*. Pull request commit comments are generally short comments containing code reviews by other developers of the project. Each pull request contains a series of conversations (e.g., comments), and discusses a few topics concerning the committed code. We use the challenge datatset, and consider the whole sequence of conversations associated with a successful or a failed pull request as a document, and collect 9,601 conversation documents for the experiment. It should be noted that we limit our study to the 78 *base projects* and their forked projects, and also use other 68,916 pull requests (i.e., not containing discussion) for the experiment. We collect the details of each project (e.g., domain, programming language, age and maturity) and the corresponding developers (e.g., number, experience) for the comparative analysis. The hosted projects belong to different application domains such as *reusable frameworks* and *libraries, networking, database management, IDE, statistics* and so on. The projects are written in 13 different programming languages such as *Scala, Python, Java, C#* and so on.

3. TOOLS AND METHODOLOGY

We apply Latent Dirichlet Allocation (LDA), a popular topic modeling technique, on the document collection to find out the underlying topics discussed in each document (e.g., pull request conversation). In order to apply topic modeling on the corpus, we normalize the content of each document given that they contain natural language texts. We remove stop words from them using a word list[1] provided by Google, and perform stemming to extract the root form of each word. Then we represent the content of each document as a collection of stemmed tokens, and we use 9,421 such documents. It should be noted that after stemming and removal of stop words, we find the content of 180 documents insignificant (e.g., empty, contains single word), and they are discarded from the corpus.

Topic Modeling: We use *JGibbLDA*[2], a LDA implementation that uses Gibbs sampling, in order to determine the probabilistic topic model. We use 3,000 iterations of sampling, and collect 100 topics discussed in the document collection, where each topic is described using ten relevant words. We also collect the probabilistic measures of the extent to which a corpus document discusses each of the 100 topics, which we use for comparative analysis in the later phase.

Topic Labeling: The tool returns a list of ten relevant words along with their probabilistic expressiveness for each topic. However, the extracted topics should be more comprehensive for effective analysis which demands topic labeling. In order to label a topic, we analyze the corresponding word list and choose the top four words representing the topic. These words are not necessarily the words with the topmost probabilities. The labeling approach is partially motivated by the approach of Lau et al. [5], where they use the article titles extracted from *Wikipedia* search for automatic topic labeling. In our research, we use the most representative words from the list and a few programming or technical jargons extracted from various online sources[3], and manually label each topic. We successfully label 64 out of 100 topics, and the labeled topics are hosted elsewhere [1] due to space limitation. We show the top four representative words for each of the topics in the table, and the complete word list for each of the topics can also be found online [1].

4. COMPARATIVE ANALYSIS

In this study, we analyze different artifacts and aspects associated with the pull requests in order to extract useful information that can provide insights into the success and failure of the pull requests.

Technical Topics Discussed: The discussion texts associated with a pull request often contain useful information about the technical concerns identified in the code. In our research, we identify and use them to contrast between successful and failed pull requests. From *JGibbLDA* tool, we collect the probabilistic alignments of a document (e.g., pull request discussion) to each of the 100 extracted topics. We sort the measures in descending order and collect the top five dominant topics (i.e., due to short volume of each discussion) discussed in the document. Then for each topic, we collect the frequency of the pull requests (i.e., discussion documents) and the information regarding their successes and failures. Fig. 1 shows the percentage of the 9,421 pull

[1]https://code.google.com/p/stop-words/

[2]http://jgibblda.sourceforge.net/

[3]http://www.webopedia.com/Programming

requests that involve each topic in the discussion. We note that eight topics are widely discussed (i.e., each topic on average in 17.54% documents), and six of them can be labeled– *Recursion and Refactoring* (in 18.35% documents), *Database Query Execution* (in 16.16% documents), *Arrays and functions* (in 11.12% documents), *Actor Model* (in 12.22% documents), *OOP Paradigm* (in 16.29% documents) and *Space and Indents* (in 10.98% documents). The two unlabeled dominant topics are discussed in 28.60% documents on average. The remaining 92 topics are less discussed (i.e., each topic in less than 4% documents on average). We also note that each topic is *more prevalent* in the discussion of the unsuccessful pull requests than that of the successful pull requests except a dominant topic– *Actor Model*. Thus, the study reports that a few technical problems (i.e., topics) are frequently associated with the pull requests; however, most of the time, they are not properly solved, and therefore, the pull requests do not succeed.

Programming Language: Programming language of a project is an important aspect to take into account when we are interested in comparative analysis of pull requests. We find 13 programming languages used in the 78 GitHub projects, and we find nine of them having 8-10 projects each. However, we also select *R* language containing three projects, and discard *CSS*, *Go* and *TypeScript* from the experiment due to their insignificant number of projects. We consider the average number of successful and failed pull requests for a project from each of the programming languages. We also note that age of the project can be an influencing factor in this case, and therefore, we determine the average number of successful and unsuccessful pull requests made per month. Fig. 2 shows the average number of pull requests made per month to any single base project of each programming language by its forked projects. We note that projects using *Scala, C, C#, R* and *PHP* programming languages made more successful pull requests on average than failed ones, whereas projects of *JavaScript(JS), Java, Python, Ruby* and *C++* did the opposite. We investigate the forks and the developer pool associated with those projects, and find out that the first group of projects except the *PHP*-based ones have less forks but more developers involved than those of the latter group. We also note that on average, *PHP* and *Ruby*-based projects made the highest number of pull requests per month; however, Ruby-based projects are often found with increasing number of failed pull requests per month. Although we speculate, this is due to the maximum number of forks in *Ruby* projects, the finding can encourage the research on the language specific factors on pull requests.

Application Domain: Domain specific concerns can be introduced in the pull request discussions, and they can affect the chance of merging for a pull request. In our research, we consider the domain of the project, and identify seven major domains– *Networking, Database, IDE, Statistics, Framework, Library* and *Client Apps*. We manually categorize each of the 78 base projects into different domains consulting their documentations provided online. We also determine the number of pull requests received each month by a *base project* from a particular domain. Fig. 3 shows the pull request statistics for each domain. We note that projects from *Framework* and *IDE* domains made the maximum number of pull requests each month, and their success rates are relatively higher than that of the projects

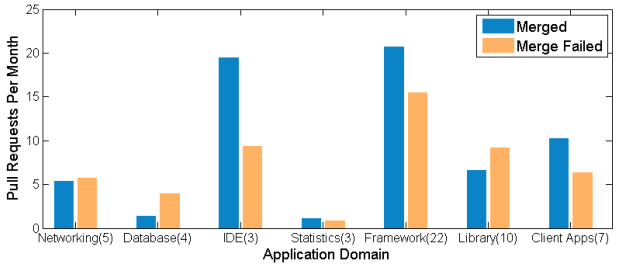

Figure 3: Pull Requests vs. Domains

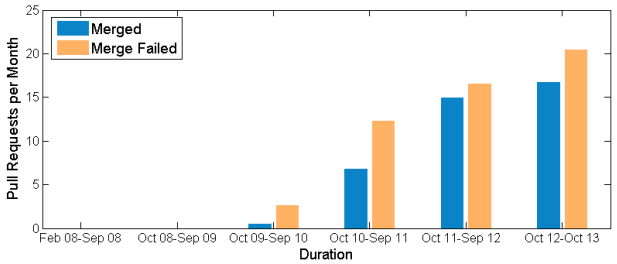

Figure 4: Pull Requests vs. Durations

from other domains. Projects from *Networking, Library* and *Client apps* domains showed comparatively similar success and failure rates in the merging of pull requests.

Project Age & Maturity: Over time, a project either may get lost or get matured through the collaboration of a number of open source developers online. We believe that *age* (i.e., the time interval between project creation date and latest recorded date in the dataset, October 5, 2013) and *number of forked projects* are two important factors that may influence the success rate of the pull requests of a project. We consider a timeline of five plus years from February, 2008 to October, 2013 with one year interval, and determine the average number of pull requests made to any single base project each month during each interval. Fig. 4 shows the results of the experiment. We note that up to September, 2009, no pull request are made (i.e., not recorded in the challenge dataset), and from October, 2009 to onward, the pull requests (i.e., both successful and failed) increase almost exponentially. It should be noted that throughout the intervals there is an increasing trend on age and number of projects, and developer participation, which actually help the higher rate of pull requests for the projects.

We consider the number of forked projects as an estimate of the maturity of a base project. We find at most 103,192 forks for 78 base projects, and we choose certain ranges. Then, for each fork count range, we determine the average number of pull requests made to a corresponding base project each month. Fig. 5 shows the results of the experiment. We note that with increase in forked projects, the average number of pull requests per month increases; however, it does not show a regular pattern. Moreover, with the increase in forked projects, the failure rate of pull requests increases especially for the projects with more than 2,000 forks.

Project Developers & Experience: Number of developers involved into a project along with their working experience with the project are also two contributing factors that can influence the success and failure rate of the pull requests. We collect the information of 20,142 developers involved into 78 base projects, whose working experience varies from five months to 68 months. We sort the projects according to their total developer range, and Fig. 6 shows

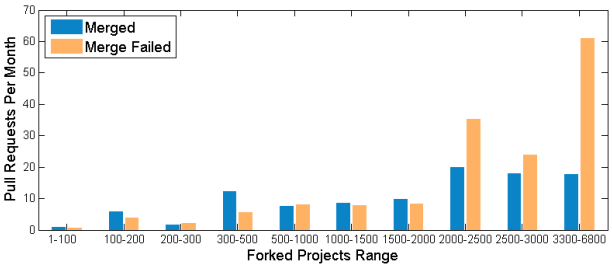

Figure 5: Pull Requests vs. Project Maturity

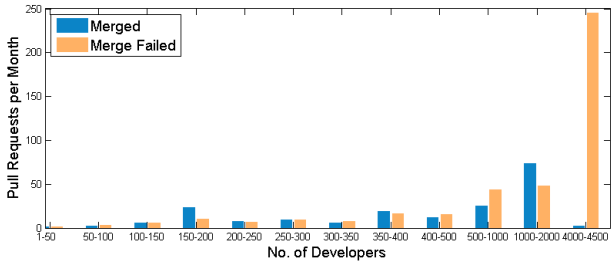

Figure 6: Pull Requests vs. No. of Developers

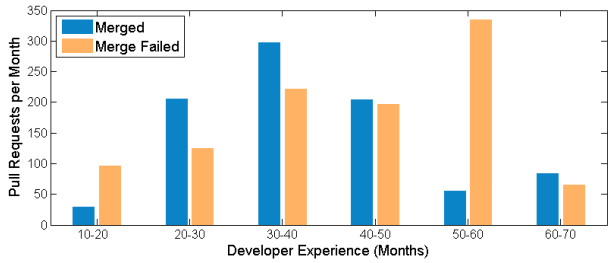

Figure 7: Pull Requests vs. Experience

the average number of pull requests made to a base project each month with a certain range of developers. We note that the average number of pull requests does not increase comparatively with higher participation; however, the projects with a large developer crowd may make an excessive number of unsuccessful pull requests.

We choose a set of ranges for the developer experience, and Fig. 7 shows the average number of pull requests made each month to a base project having developers in the forked projects with certain range of experience. We interestingly note that developers with 20 months to 50 months of experience are found the most productive, and they made the maximum number of pull requests each month. However, developers with further experiences are found either less productive or making a number of unsuccessful pull requests each month. We can speculate that the experienced developers might be involved in management activities rather than development; however, more insights could be extracted if the role information is provided, which the dataset does not provide.

5. DISCUSSION AND CONCLUSION

In this study, we conduct a comparative study between successful and unsuccessful pull requests made to 78 GitHub base projects by 20,142 developers from 103,192 forked projects. We analyze the pull request discussion texts that contain useful information about the frequent technical issues encountered. We also analyze the pull request history along with project and developer specific information in order to extract useful insights into the success and the failure of pull requests. This section reports our findings in brief as follows:

Eight *topics* (i.e., technical issues), six of which can be labeled–*Recursion and Refactoring*, *Database Query Execution*, *Arrays and functions*, *Actor Model*, *OOP Paradigm* and *Space and Indents*, are widely discussed in the texts of pull request discussion. Each of those eight technical issues is faced by 17.54% of the 9,421 pull requests on average, and 7.76% of the requests succeed to merge. The remaining 92 topics are less discussed, and each of them is covered only in 3.90% of the discussion.

In case of 24 GitHub *base projects* using three *programming languages–Ruby, Java* and *JavaScript*, average number of unsuccessful pull requests per month is exceptionally higher than that of successful pull requests. While the study points out *excessive forking* as a possible reason, future research can investigate into the language specific factors affecting the success and failure of the pull requests.

Most of the projects under study belong to seven major *application domains*, and three of them–*Framework, Library* and *Client Apps*, contain 39 projects. Projects from *IDE, Framework* and *Client Apps* domains demonstrate a comparatively higher success rate of pull requests, whereas projects from *Database* and *Statistics* domains show very limited activity in terms of pull requests. The study speculates about *less popularity* of the target domains (in terms of *developer participation* and *created forks* for a project on average) as a possible explanation of their low activity; however, it also encourages the future research on the domain specific concerns affecting pull requests.

The *age* and *maturity* (i.e., number of forks) of a GitHub project clearly affects the success and failure rates of pull requests. As time goes by and more forks are created from the base project, the number of pull requests increases and so do their success and failure rates. More importantly, our study finds that the failure rate of pull requests increases rapidly (for seven projects) when more than 3000 forks are created, which can be an important piece of information for the administrators of the *base projects*.

The *number of developers* involved into a project and their *experience* can affect the success and failure rates of pull request for a project. Our study finds that the average number of pull requests per month for a project increases almost regularly against increased participation of the developers; however, the rate of unsuccessful pull requests increases exponentially (for one project) with more than 4000 developers involved. The study also suggests that developers with 20 to 50 months experience are found the most productive in terms of submitting and getting pull requests accepted. While we cannot provide a suitable explanation for this due to lack of information in the dataset, the findings can help the project administrator to attract the appropriate audience, and manage the existing developer pool involved into the project.

REFERENCES

[1] Experiment Data. URL `http://www.usask.ca/~mor543/msr2014`.
[2] G. Gousios. The GHTorrent Dataset and Tool Suite. In *Proc. MSR*, pages 233–236, 2013.
[3] A. Hindle, M.W. Godfrey, and R.C. Holt. What's Hot and What's Not: Windowed Developer Topic Analysis. In *Proc. ICSM*, pages 339–348, 2009.
[4] A. Hindle, N.A. Ernst, M.W. Godfrey, and J. Mylopoulos. Automated Topic Naming to Support Cross-Project Analysis of Software Maintenance Activities. In *Proc. MSR*, pages 163–172, 2011.
[5] J.H. Lau, K. Grieser, D. Newman, and T. Baldwin. Automatic Labelling of Topic Models. In *Proc.HLT*, pages 1536–1545, 2011.