

<Analyses Pull-request Characteristics on Pull Request acceptance in Distributed Software Development>

<Lingyi Jia>

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfillment of the requirements of the Degree of Master of Science at the University of Glasgow

<Date of submission placed here>

**Abstract**

<Abstract goes here…>

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.

<**Please note that you are under no obligation to sign this declaration, but doing so would help future students.>**

Name: Signature:

Acknowledgements

<Acknowledgements go here>

Contents

<Update the table of contents by right-clicking on it and selecting Update Field… and then select page numbers only.>

Chapter 1 Introduction <This is Heading 1> 1

1.1 A section <This is style Heading 2> 1

1.1.1 A subsection <This is style Heading 3> 1

Chapter 2 Survey 2

Chapter 3 Further Chapters 3

Chapter 4 Conclusion 4

Appendix A <Name of appendix> 1

Appendix B <Another appendix> 2

Bibliography 3

# Introduction <This is Heading 1>

Introduce the project.

<This is style Normal. We recommend you make use of styles to simplify creating a well-formatted document. We have used “space before” and “space after” in defining these styles, in order to space the headings and paragraphs appropriately. You should never need to enter a blank line.>

## A section <This is style Heading 2>

Please note your dissertation need not follow the included section headings – this is only a suggested structure. Also add subsections etc. as required.

### A subsection <This is style Heading 3>

Try to avoid this too much, but it’s here if you need it.

# Background

In software engineering development, large-scale projects often require collaboration among multiple people. In actual development, new functions and patches are usually modified or added on the basis of released program, which do not affect the normal use of projects. The version control system helps development team to work together regardless time and space. It also allows archive development histories, which makes iterative development more convenient for projects. At present, the main version control system has been transformed from centralized version control system to distributed version control system(DVCS). In the DVCS, each developer can establish an independent code warehouse locally, which is known as repositories. At the same time, various version management operations can be completed locally relying on DVCS. In the development of DVCS, the following concepts are created and widely used:

**Git.** Git is a widely used DVCS and many source code repository hosts have been developed relying on Git, such as Github, Gitlab, Gitee, etc. In these repository hosts, independent development can be carried out on individual branches. And the modified or new-added code can be merged into the main branch by establishing a merge request. When requesting for a merge, the code manager will review the submission, in order to ensureing the consistency and quality of the code. The personal modification will be merged into the main branch once the code manager approves the merge request.

**Pull-based model.** In recent years, DVCS based on the Pull-based model has become more and more popular, since more and more developers prefer to independent development and branch creation. In Pull-based development model, contributors can not access main repository directly, while they can clone any public repository locally. As a result, developers can modify code independently and managers can merge modification selectively by pull requests.

## Pull Request in Github

Github is the largest repositories host in the open source system, hosting more than 60 million repositories and 20 millions users. The open source software development allows different types of developers to participant and brings their contributions together. It has profound impact on software development, containing a lot of valuable data, such as software code, process data, software communication and feedback.

In Github, pull request(PR) has become an effective way of social programming and continuous code integration. At the same time, the number of collaborative projects using PR accounts for more than half, which will increase in the future.

PR in Github includes following 6 steps.

First things first, the users will be either create a new directory or pull from an existing GitHub repository. Then, the following steps are shown as below in Figure 1.

1. Create a branch:

Branches are by default “master” in git whenever you’ve created a new repository. With branch we can easily isolate our work to maintain stability from one another to ensure no disruption to the master code as testing can be done within the new branch to see if it fit for the purposes. In addition, branches are not available to the others only unless you have pushed it to the remote repository.

Creating a branch outside of master and switching to it within git is rather straightforward by using the following CLI “git checkout -b new\_function\_A”. Once the newly function has been tested within the branch, the branch for development can then be merged back to master upon its completion.

1. Commits

Commit is a process which requires the user to provide a message of their work which then uses it to create a history of work that you have done. Thus, the others are able to keep track of what you have done recently along with the reason of why it has been committed. However, git commit does not just automatically include everything that has been recently changed. Instead, it requires the user to specify by using “git add” to which the files to be included. Moreover, each commit is a separate unit of change. Hence, it allows the user to roll back to certain version in the case of a bug in recent commit.

1. Push your changes & Open a Pull Request

Push your changes is where you push the changes made within your local repository to the remote repository. Without pushing your changes, the others will not be able to view your branch.

A pull request commences a discussion within the development team that are involved with the project. Everyone within will be able review the exact changes that will be merged and decide if it they would accept the request.

1. Discuss and Review your code

During the pull request as mentioned earlier, the development team that are reviewing your changes may provide advises, or questions regarding to the changes. For instance, the code might not follow a software design pattern, improvement of code performance, or alternatively informing that the code is in good condition to be merged. All in all, PR are used to encourage discussion and interaction within the team to ensure everything is in order before merging.

1. Rebase and tests

Rebase command is used to rebasing a branch update with another. For instance, a current branch is out of date with the master branch. Thus, by rebasing the current branch it allows all the new commits in the master branch to be also included in the current branch. Hence, ensuring the latest commit in master to be included in the current branch. Once that is done, testing is carried out to ensure that it does not cause issues before merging it with the master branch.

1. Merge

Last but not least, once the changes has been checked and approved, it can now be merged into the master branch.

Diagram

Description automatically generated

Figure 1: Workflow of PR in Github

# Further Chapters

The content of these chapters depends on the project and should be agreed with your supervisor (e.g. description of the solution, evaluation results, etc).

<Figure below is in style “figure” which continues to style “figure caption” when you press Enter and then back to “Normal” when you press Enter again.>

Figure 1: Some important shapes.

<If you wanted to show any code fragments, you could use the following style called code, which could then be followed by figure caption..>

*# This is a little bit of Python*

**for** i in range( 10 ):

**for** j in range( 10 ):

**print** i\*j,

**print**

Figure 2: A crucial algorithm for the project.

# Conclusion

Main conclusions of your project. Here you should also include suggestions for future work.

###### <Name of appendix>

<Use Heading 6 for the Appendix heading>

###### <Another appendix>

# Bibliography

Ex. [1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[1] Chatziasimidis F, Stamelos I. Data collection and analysis of GitHub repositories

and users. International conference on information intelligence systems and

applications, 2015.

[2] Gousios G, Storey M A, Bacchelli A. Work Practices and Challenges in Pull-Based Development: The Contributor’s Perspective. ICSE.