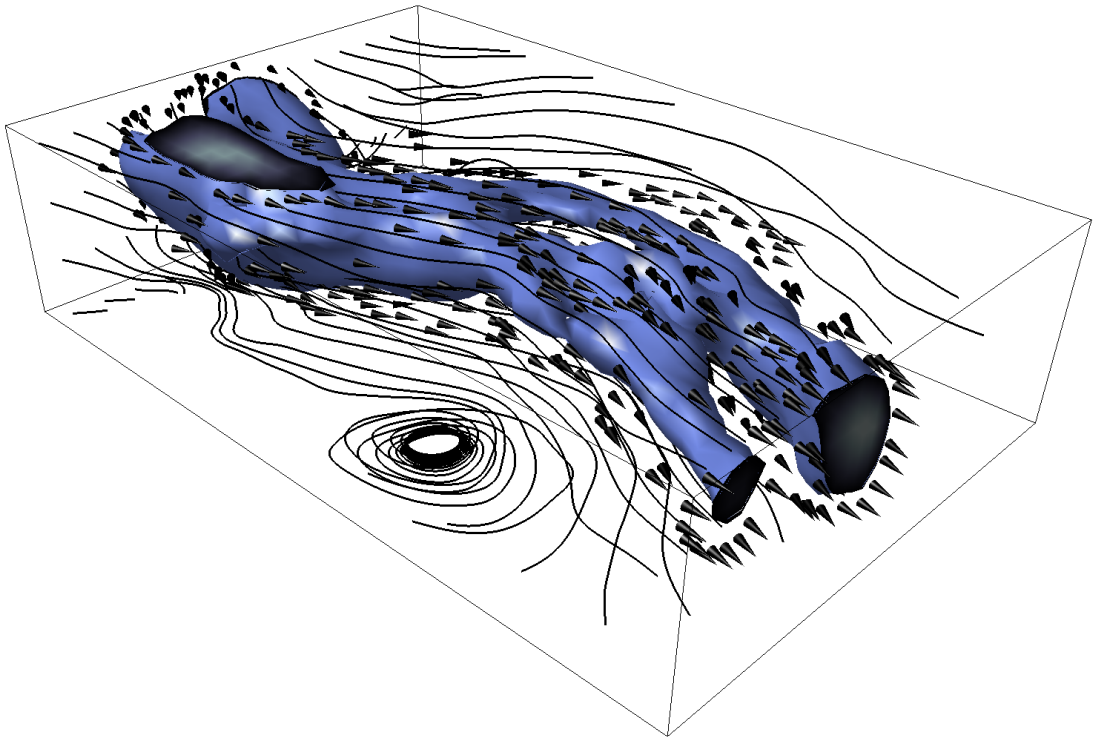
CHALMERS
UNIVERSITY OF TECHNOLOGY



A Chalmers University of Technology Master's thesis template for LaTeX

A Subtitle that can be Very Much Longer if Necessary

Master's thesis in Master Programme Name

DAVID FRISK

Department of Some Subject or Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Master's thesis 2015:NN

An Informative Headline describing the Content of the Report

A Subtitle that can be Very Much Longer if Necessary

NAME FAMILYNAME

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Some Subject or Technology
Division of Division name
Name of research group (if applicable)
Chalmers University of Technology
Gothenburg, Sweden 2015

An Informative Headline describing the Content of the Report
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME

Supervisor: Name, Company or Department
Examiner: Name, Department

Master's Thesis 2015:NN
Department of Some Subject or Technology
Division of Division name
Name of research group (if applicable)
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Wind visualization constructed in Matlab showing a surface of constant wind speed along with streamlines of the flow.

Typeset in LaTeX
Printed by [Name of printing company]
Gothenburg, Sweden 2015

An Informative Headline describing the Content of the Report
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME
Department of Some Subject or Technology
Chalmers University of Technology

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: lorem, ipsum, dolor, sit, amet, consectetur, adipisicing, elit, sed, do.

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Name Familyname, Gothenburg, Month Year

Contents

List of Figures

List of Tables

1
Introduction

1.1 *

When developing software products using a version-control system, branching is often used, both to support variability [1] and when developing new features, which we, in this document, call feature branching. Feature branching will be the main focus in this study. It is of vital importance that the merging of these branches works smoothly, even if a conflict occurs. As of today, there do not exist many tools for automatically solving conflicts during merging. It is up to the programmers themselves to manually resolving the conflicts in the code. Understanding which patterns are used for resolving merge conflicts allows for future development of an autonomous merge-conflict tool.

The goal of this work is to conduct a feasibility study that aims at investigating to what extent and how it is feasible to analyze merge-conflict resolution patterns in large codebases comprising many open-source projects, such as those being hosted on GitHub or BitBucket.

Our long-term goal is to create an automated merge-resolution tool. Towards this end, this study aims at answering the following research questions:

- RQ1. Is it feasible to statically analyze conflict-resolution patterns in real-world, large version histories of open-source projects, and how?
- RQ2. What kind of mining and analysis infrastructure is needed for such a study?
- RQ3. Which patterns exists for resolving merge-conflicts? ...

Our main working hypothesis is that it is in fact feasible to automatically recognize and analyze patterns from all the metadata available about projects and from statically analyzing code. By studying examples of popular projects with many branches or forks, by developing a mining infrastructure, and by investigating to what extent static code-analysis tools can be utilized for recognizing any patterns, we will work towards testing this hypothesis.

1.2 Section

1.2.1 Subsection

1.2.1.1 Subsubsection

1.2.1.1.1 Paragraph

1.2.1.1.1.1 Subparagraph

2
Theory

How Conflicts Arise in a GIT-based Branching and Merging Scenario. Many software projects follow a branching model when using Git, such as the one explained by Giessen [2]. In these models, users create feature branches that provide an environment where new features can be implemented and tested without affecting the end-user version of the software [3]. There are various ways to use branches. A branch can be created for each new feature, for each new release, and for each product [1]. When a new feature has been implemented in a new branch, the branch needs to be merged into another branch, such as the main branch. Merging is the process of joining two branches together, both in case of two local branches or a local and a remote branch [4]. At GitHub, this is usually done using a pull request. A pull request is made to let the collaborators in the repository know that the commits in a branch are ready to be merged. The collaborators can review the new code and input their feedback until it is finally approved for merging into the end-user branch [5]. Merging can also be performed without using pull requests. When branches are to be merged, conflicts might arise. Conflicts are the problems that prevent git from automatically merging two branches together.

Problems of Resolving Merge Conflicts. Resolving merge conflicts, such as those arising from changes to different variants of features, is difficult. Merging might require refactoring the class hierarchy, introducing design patterns, or adding parameters to the feature. If it would be possible to develop an autonomous tool that can provide automated conflict resolution in this case, it would be of great value, since resolving such conflicts is a recurring problem that is solved manually today. The problem is that the development of such a tool requires more understanding of how the merge-conflict resolution is performed. Hence, in the scenario above, we're interested in studying merge-conflict resolutions.

Problems of an Empirical Study of Conflict Resolutions. However, even that is difficult. It is unclear whether and how we could study such resolutions as they are performed by developers in real-world software. For having representative results, one would also need to study the resolutions in large codebases, such as from GitHub/BitBucket. This requires some automated analysis. It is not trivial how this automated analysis could be designed. It is also unclear how many projects can be studied; it's even not clear which projects are well-suited for such a study.

Problem Reports and Empirical Studies. Cloning happens during all stages of a software-development process, and it is the responsibility of the developers

themselves to make sure that changes between copies of the clones are propagated correctly [1]. Because of this, there are risks that conflicts arise during all stages of the development process.

With the use of a version control system, cloning can be managed in a more smooth way by using branching and merging capabilities [6]. GitHub uses the version control system Git, which maintains a development history for each project. In this history lies the information about when merges have occurred. When cloning features, multiple versions of the same feature exists and their consistency needs to be managed [6].

Textual Merging. The most commonly used merging technique is textual merging [7]. Textual merging is based on the history and on textual differences. It does not make use of any knowledge of the syntax or semantic. One must also distinguish between two-way merging and three-way merging. In two-way merging, only the two conflicting clones are analyzed to resolve the conflict. In three-way merging, also the common ancestor is used, which is more powerful [8].

Semistructured Merging. Semistructured Merging. Furthermore, there exist merge tools that uses another approach than textual merging, such as syntactic- and semantic merging, which have language specific knowledge [9] and does not only compare lines of text. A combination of both textual merging, syntactic and semantic merging is called semistructured merging [7]. Other studies have proved that the use of semistructured merge decreases the number of conflicts significantly. The study "Assessing Semistructured Merge in Version Control Systems: A Replicated Experiment", proves that semistructured merge can reduce the number of conflicts by 55% [10]. These works are important to us. But instead of proposing a new conflict-resolution technique, we are interested in how developers resolve conflicts arising from different variants of features or projects.

Conflict Patterns. During merging, several types of conflict patterns might occur. In her study, Accioly [11] identifies numerous conflict patterns. This list will be used throughout our study both to learn about which conflict patterns exist, and later using them when developing the analyzing tool. While Accioly focuses conflict patterns, we strive to identify conflict-resolution patterns.

Mining. To automatically analyze projects on GitHub, a high number of requests to GitHub has to be performed. When querying requests to GitHub, GitHub has a limit of 5000 requests per hour [12].

2.1 Figure

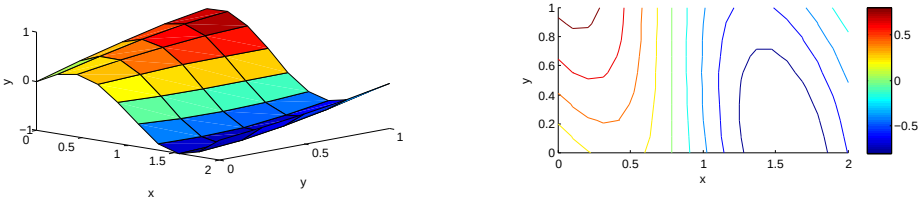


Figure 2.1: Surface and contour plots showing $z(x, y) = \sin(x + y)\cos(2x)$.

2.2 Equation

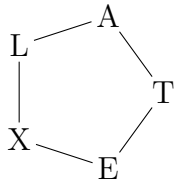
$$f(t) = \begin{cases} 1, & t < 1 \\ t^2 & t \geq 1 \end{cases} \tag{2.1}$$

2.3 Table

Table 2.1: Values of $f(t)$ for $t = 0, 1, \ldots 5$.

t	0	1	2	3	4	5
$f(t)$	1	1	4	9	16	25

2.4 Chemical structure



2.5 List

1. The first item
   (a) Nested item 1
   (b) Nested item 2
2. The second item
3. The third item
4. ...

2.6 Source code listing

```
% Generate x- and y-nodes
x=linspace(0,1); y=linspace(0,1);

% Calculate z=f(x,y)
for i=1:length(x)
  for j=1:length(y)
    z(i,j)=x(i)+2*y(j);
  end
end
```

2.7 To-do note

The `todo` package enables to-do notes to be added in the page margin. This can be a very convenient way of making notes in the document during the process of writing. All notes can be hidden by using the option *disable* when loading the package in the settings.

Example of a to-do note.

3
Prestudy

3.1 Repositories to analyze

Since we are going for a quantitative analysis, we want to analyse fairly big projects that contain many commits and many forks along with many branches. This is because we want to cover as much of variant possibilities as possible to gain the most accurate result. To satisfy these requirements, the 20 top starred Java repositories on GitHub were chosen.

The projects listed in Table (no) were cloned so that Git commands could be used to analyze the repositories. Elasticsearch was chosen as the project for our initial analysis since it has a vast number of commits (more than 20000) and forks.

Elasticsearch is a distributed search engine used for analysing data in realtime.

(As of 23/3-16)

Name	Commits	Branches	Forks
Elasticsearch	20712	46	5229
Android-async-http	856	3	4024
Android-best-practices	201	1	1696
Android-universal-image-loader	1025	3	5640
Curator	1050	9	304
Eventbus	404	5	2493
Fresco	494	3	2453
Guava	3372	4	1862
Iosched	129	2	4071
Java-design-patterns	1196	6	3495
Leakcanary	238	15	1291
Libgdx	12247	4	4479
Okhttp	2449	37	2518
React-native	5707	23	5609
Retrofit	1285	21	2081
Rxjava	4630	24	1919
Slidingmenu	336	8	5306
Spring-framework	11825	10	6860
Storm	1764	44	1760
Zxing	3203	3	4730

4

Methods

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5

Results

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

6
Conclusion

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bibliography

[1] Frisk, D. (2015) A Chalmers University of Technology Master's thesis template for LaTeX. Unpublished.

A

Appendix 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.