FACULTY OF SCIENCES

Optimising Continuous Integration using Test Case Prioritisation

Pieter De Clercq
Student number: 01503338

Supervisors: Prof. dr. Bruno Volckaert, Prof. dr. ir. Filip De Turck
Counsellors: Jasper Vaneessen, Dwight Kerkhove

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de informatica

Academic year 2019-2020

GHENT UNIVERSITY

Admission

Acknowledgements

Completing this thesis would not have been possible without the help and support of many people, some of which I want to thank personally.

First of all I want to thank prof. dr. Bruno Volckaert and prof. dr. ir. Filip De Turck for allowing me to propose my own subject and for their prompt and clear responses to every question I have asked. I especially want to thank you for giving me the permission to insert a two-week hiatus during the Easter break, so I could help out on the UGent Dodona project.

Secondly I want to express my gratitude towards my counsellors Jasper Vaneessen and Dwight Kerkhove, for steering me into researching this topic, as well as their guidance, availability, and willingness to review every intermediary version of this thesis.

Furthermore I want to thank my parents, my brother Stijn and my family for convincing me and giving me the possibility to study at the university, to support me throughout my entire academic career and to provide me the opportunity to pursue my childhood dreams.

Last, but definitely not least, I want to thank my amazing friends, a few of them in particular. My best friend Robbe, for always being there when I need him, even when I least expect it, for both supporting as well as protecting me against my sometimes unrealistic ideas and ambition to excel. Helena for never leaving my side, for always making me laugh even when I don't want to, and most importantly to remind me that I should relax more often. Jana for my daily dose of laughter and fun, and for her inexhaustible positivity. Tobiah for the endless design discussions and for outperforming me in almost every school project, encouraging me to continuously raise the bar and never give up. Finally I want to thank Doortje and Freija for answering my mathematical questions, regularly asking about my thesis progression and thereby motivating me to persevere.

Thank you.

Pieter – Ghent, 2020

Summary

ChapterSummary Summary in English will come here.

Samenvatting

Nederlandse samenvatting komt hier.

Optimising Continuous Integration using Test Case Prioritisation

Pieter De Clercq

Supervisor(s): Prof. dr. B. Volckaert, Prof. dr. ir. F. De Turck, J. Vaneessen, D Kerkhove

Abstract—**This abstract is very abstract.**

Keywords—**words, will, appear, here, soon**

I. Introductie

Things will appear here. [1]

References

[1] Michael Cusumano, Akindutire Michael, and Stanley Smith, "Beyond the waterfall : software development at microsoft," 02 1995.

Optimaliseren van Continue Integratie door middel van Test Prioritering

Pieter De Clercq

Supervisor(s): Prof. dr. B. Volckaert, Prof. dr. ir. F. De Turck, J. Vaneessen, D Kerkhove

Abstract—**Dit abstract is super abstract.**

Trefwoorden—**woorden, komen, hier**

I. Introductie

Dingen komen hier. [1]

Referenties

[1] Michael Cusumano, Akindutire Michael, and Stanley Smith, "Beyond the waterfall : software development at microsoft," 02 1995.

Vulgarising summary

Vulgarising summary will come here.

Contents

Glossary

CI Continuous Integration. 2

TCP Test Case Prioritisation. 2

TCS Test Case Selection. 2

TSM Test Suite Minimisation. 2

VCS Version Control System. 2

Chapter 1

Introduction

Given the complexity and rapid pace at which software is being built today, it is inevitable that at some point bugs will emerge. These bugs can either be introduced by a malfunctioning new feature, or by breaking existing functionality (*a regression*). In order to detect bugs in an application before its customers do, an adequate *testing infrastructure* is required.

This testing infrastructure consists of multiple *test cases*, collectively referred to as the *test suite* of the application. The quality of a test suite can be assessed in multiple ways. The first and most used option is to measure which fraction of the source code is tested by at least one test case, a ratio which is expressed as the *coverage* of the application. Another possibility is to apply transformations to the source code and validate whether or not this results in a failed test case, a process indicated as *mutation testing*.

Ideally, this testing process should be automated and performed after every change to the source code. This is generally a very time-consuming occupation, and as such has led to the creation of various automation frameworks and tools, collected under the name of Continuous Integration (CI). Common examples of CI practices are automatically running the test suite and estimating the code coverage after every pushed change to the Version Control System (VCS).

However, applying these practices and maintaining a qualitative test comes at a cost. After every addition or modification to the source code, at least one new test case must be introduced to validate its correctness. As a result of the speed at which the source code tends to grow, the test suite suffers from severe scalability issues. While it is desired and ideally required to execute every single test case in the test suite, there are examples known to literature where this is not possible since this incurs an increasing delay in the development process, which in turn results in economic loss.

Three approaches can be taken towards resolving this issue by reducing the time occupied by waiting for the test results: Test Suite Minimisation (TSM), Test Case Selection (TCS) and Test Case Prioritisation (TCP). The main subject of this thesis will be to implement a framework for TCP. To accomplish this, the next chapter will introduce

important concepts which are used in modern software engineering. **??** will elaborate on the aforementioned approaches and present accompanying algorithms. The implementation details of the new framework will be discussed in **??**. Afterwards, **??** will evaluate the performance of this framework and provide insights to the characteristics of a typical test suite. More specifically, this chapter will research the probability of (repeated) test failure and the average duration of a test run. Finally, **??** will present additional ideas and improvements to the framework.

Chapter 2

Software Engineering

Chapter 3

Related work

Chapter 4

Proposed framework: VeloCIty

Chapter 5

Evaluation

Chapter 6

Conclusion

List of Figures

List of Listings

List of Tables