(94)

LIBRARY
UNIVERSITY OF MORATUWA. SRI LANKA
MORATUWA

A SCALABLE SOFTWARE QUALITY VERIFIER FRAMEWORK

A dissertation submitted to the
Department of Electrical Engineering, University of Moratuwa
in partial fulfillment of the requirements for the
degree of Master of Science

by

PARANA WIDANARALALAGE DILEEPA CHIRANTHANA JAYATHILAKE

University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Supervised by: Dr. Amal Shehan Perera

Department of Electrical Engineering
University of Moratuwa, Sri Lanka

February 2011

96798

621.3 "11"
621.3(043)

University of Moratuwa
96798

96798

UNIVERSITY OF MORATUWA
LIBRARY
SRI LANKA

DECLARATION

The work submitted in this dissertation is the result of my own investigation, except where otherwise stated.
It has not already been accepted for any degree, and is also not being concurrently submitted for any other degree.

UOM Verified Signature

....

P.W.D.C. Jayathilake
18/02/2011

I endorse the declaration by the candidate.

UOM Verified Signature

....

Amal Shehan Perera

University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

TABLE OF CONTENTS

DECLARATION i

Table of Contents ii

Abstract v

Table of Figures vi

Chapter 1 1

1.1 Software Outsourcing Business 1

1.2 Issues with Outsourcing 2

1.3 Value of Quality 2

1.4 Existing Software Quality Verification Methods 4

1.5 Value of a Quality Verification Framework for an outsourcing company 4

Chapter 2 5

2.1 Evaluation Contexts 5

2.2 Output of Quality Evaluation 6

2.3 Components and Procedures to be implemented 6

Chapter 3 8

3.1 Modules and components of the system 8

3.1.1 Hook Program 8

3.1.2 Commit Manager 8

3.1.3 Offline Quality Analyzer 10

3.1.4 Product Quality Analyzer 11

3.1.5 Configurations 12

3.1.6 Scripting Language 12

3.2 Users of the system 13

3.2.1 Technical Lead 13

3.2.2 Project Manager 14

3.2.3 Developer 14

3.2.4 Due-diligence Engineer 14

Chapter 4 15

4.1 Svn hook 15

4.2 Commit Manager 16

4.3 Tools and Wrappers 16

4.4 Tool configuration 16

4.5 Output of the system 17

4.6 Usability considerations 18

Chapter 5 19

5.1 Hardware/Software Infrastructure 19

University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

5.2 Execution Flow 20
5.3 Adding a new project to the code quality evaluation system 22
5.4 Output of the system 23
5.5 Pilot project run 26
Chapter 6 28
6.1 Problems associated with software application analysis 28
6.1.1 Log file interpretation 29
6.1.2 Making inferences from interpreted information 30
6.1.3 Documenting lessons learnt from analysis 30
6.1.4 Presenting information to different levels of people 30
6.2 Framework Goals 30
6.2.1 Capability to handle huge log files 30
6.2.2 Configurable syntax 31
6.2.3 Flexibility 31
6.2.4 Knowledge representation easily decodable by both humans and machines 31
6.2.5 Short learning curve 31
6.2.6 Promote reusable patterns 31
6.2.7 Scalability 31
6.3 Framework Concepts 31
6.3.1 Represent knowledge as mind maps 31
6.3.2 Use familiar data types 32
6.3.3 A simple scripting language 32
6.3.4 Inferences expressed as 'scriptlets' 33
6.3.5 Metadata 33
6.4 Implementation of the framework 34
6.4.1 Data types 34
6.4.2 Functions 35
6.4.3 File Manipulations 36
6.4.4 Metadata 37
6.4.5 Parser 37
6.4.6 Execution Engine 39
6.4.7 Control Code 41
6.4.8 Memory Manager 42
6.4.9 Tools 42
6.5 Usage 42
6.6 Proof of Concept Implementation 43
Chapter 7 45
7.1 Code Quality Evaluation Tools 45
7.1.1 FxCop 45

University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

7.1.2 StyleCop 46
7.1.3 Gendarme 46
7.1.4 CppCheck 47
7.1.5 Checkstyle 47
7.1.6 PMD 48
7.1.7 PMD-CPD 48
7.2 Binary Quality Analysis Tools 48
7.2.1 Apache JMeter 48
7.2.2 Microsoft Application Verifier 49
7.2.3 LeakDiag & LDGrapher 50
7.2.4 Process Monitor 50
7.2.5 Xperf 51
7.2.6 Application compatibility toolkit 51
7.3 Other tools 52
7.3.1 Cruise Control 52
7.3.2 Maven 52
7.3.3 Sonar 53
Chapter 8 55
8.1 Conclusions, Remarks and Discussion 55
8.2 Future Work 56
References 58
APPENDIX A: Scripting Language Syntax 59
APPENDIX B: Built-in functions in Product Quality Analyzer framework 62
B.1 Node functions 63
B.2 String functions 69
B.3 Integer functions 71
B.4 Boolean functions 72
B.5 List functions 72
B.6 General functions 73

University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

ABSTRACT

Outsourcing software development is a growing business that is proven to bring cost-effective and efficient solutions for varying demands of software product companies. Though it has proven its capability in bringing value to products to stay ahead in competition, few inherent problems are also identified in this practice. A prominent issue is how to verify the quality of the applications delivered by the vendor. Given that a critical bug in production can bring disasters, it is vital to the outsourcer to make sure that the deliverables from the vendor conform to a well defined set of quality guidelines. The work described here is the design and implementation of a scalable software quality verification framework on top of which, industrial grade automated quality verification systems can be built with minimum effort.

The framework is built to evaluate both software code and applications. Code level evaluation is done in two phases; when the developer tries to add code to the repository and a deeper test covering a wide range of problems in an offline context. The rules used for evaluation, actions on results and alerting can be customized in project level.

The framework provides a programming interface and a set of tools for application evaluation. The simple yet powerful programming interface creates ground for building a knowledgebase accumulating the experience of veterans. This is used in collaboration with modern tools to evaluate applications against their performance, security, memory and IO usage, etc.

A quality verification system built using the framework which was put into action in a commercial software project proved to add a significant value to the deliverables. An experiment done with the programming interface showed that powerful analysis systems can be built to both evaluate deliverables and aid in software due-diligence process.

University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

TABLE OF FIGURES

Figure 1.1 - Cost of remediation for a bug identified in various phases 3
Figure 3.1 - Modules and components of the system 9
Figure 3.2 - Use cases and actors of the system 13
Figure 4.1 - Commit time code quality verifier system 15
Figure 4.2 - Operation of commit time code quality verification system 17
Figure 4.3 - Output of Commit Manager for a commit containing rule violations 18
Figure 5.1 - Offline Code Quality Analyzer Architecture 19
Figure 5.2 - Execution flow in the code quality evaluation system 21
Figure 5.3 - Configuring rules for code quality evaluation 22
Figure 5.4 - High level information on results 24
Figure 5.5 - Names of violated rules and source files containing violations 24
Figure 5.6 - Rule violations pointed to the line in the source code 25
Figure 5.7 - part 1 of an email sent to developers by the system 25
Figure 5.8 - part 2 of an email sent to developers by the system 26
Figure 6.1 - Example Mind Map 32
Figure 6.2 - Architecture for Product Quality Analyzer 34
Figure 6.3 - Data types used in the framework 35
Figure 6.4 - Production rules for the parser 37
Figure 6.5 - An example script 38
Figure 6.6 - Algorithm for the parser 39
Figure 6.7 - Algorithm for the Execution Engine 40
Figure 6.8 - Example control code 41
Figure 6.9 - Example Usage Scenario 43

University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk