

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

A STUDY OF SOFTWARE COMPONENT SYSTEM EVOLUTION

A THESIS PRESENTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
DOCTOR OF PHILOSOPHY
IN
COMPUTER SCIENCE
AT MASSEY UNIVERSITY, PALMERSTON NORTH,
NEW ZEALAND.

Graham Jenson

2013

Abstract

There are an estimated 20 million users of the Ubuntu operating system and millions of users of the Eclipse integrated development environment. Ubuntu and Eclipse systems are constructed from components, called packages and bundles respectively, and can be changed by adding or removing components to and from their systems. Over time these systems will be continually changed to adapt to their software environment, accommodate new user requirements, fix errors and/or prevent errors from occurring in the future. This continual change is called the component system evolution process.

Using a developed simulation this thesis investigates the reduction of negative effects during the component system evolution process. The primary negative effects that are focused on are the amount of change made to the system, and the out-of-dateness of the system. The simulation was created by modelling the evolution of component systems and executed using a developed implementation. Various experiments that simulate an Ubuntu system evolving over a year were conducted, and the change and out-of-dateness of these systems measured. These experiments resulted in two novel approaches that can be used to reduce change and out-of-dateness during evolution. Therefore, this research could be used to reduce negative effects on millions of evolving component systems.

Acknowledgements

I would like to thank my supervisors Jens Dietrich and Hans Guesgen. You have wisely directed me throughout this project.

I would like to thank my office mates Jevon Wright and Fahim Abbasi. You have been the continual distraction that allowed me to keep my sanity.

I would like to thank Stephen Marsland, Giovanni Moretti, Michele Wagner, Catherine McCartin and Patrick Rynhart. You provided a community that made this task less daunting.

I would like to thank my parents, Georgette and Deryk. You gave me my curiosity and persistence.

Most importantly, I would like to thank Yuliya Bozhko. The completion of this thesis is entirely due to your constant support and love.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivation	2
1.2 Objective	3
1.3 Research Method	3
1.3.1 Dataset	4
1.3.2 Methodology	4
1.4 Contributions	6
1.5 Thesis Overview	7
2 Background	9
2.1 Software Evolution and Component-Based Software Engineering	10
2.1.1 Software Evolution	10
2.1.2 Component-Based Software Engineering	12
2.1.3 Unix and GNU/Linux Modular Operating Systems	14
2.2 Component Evolution vs. Component System Evolution	15
2.2.1 Component Evolution	16
2.2.2 Component System Evolution	17
2.3 What is a Software Component?	18
2.3.1 The Definition of Software Component in this Thesis	20
2.4 Component Models	21
2.4.1 OSGi	21
2.4.1.1 Bundle Layer	22
2.4.1.2 Service Layer	22
2.4.1.3 OSGi Change	23
2.4.1.4 OSGi Bundle Repository	24
2.4.2 Eclipse Plugins	25
2.4.2.1 Eclipse Change	26

2.4.2.2	Eclipse P2	27
2.4.3	Fractal	28
2.4.3.1	Fractal Change	30
2.4.4	Maven	31
2.4.4.1	Maven Change	32
2.4.5	Debian Packages	32
2.4.5.1	dpkg	33
2.4.5.2	apt-get	33
2.4.6	SOFA 2.0	34
2.4.6.1	SOFA Change	34
2.4.7	Common Upgradeability Description Format	35
2.4.7.1	Mancoosi MPM	36
2.4.8	Comparison	37
2.5	Summary	39
3	Formal Model of Component System Evolution	41
3.1	CoSyE Model of Component System Evolution	42
3.1.1	Evolution Problem	43
3.1.1.1	Complexity of an Evolution Problem	44
3.1.2	Constraints and Requests	45
3.1.3	Component System Evolution	47
3.1.4	Multiple Criteria Preferences	50
3.2	CUDF* Language	52
3.2.1	CUDF	53
3.2.1.1	CUDF Language	53
3.2.1.2	Package Description	54
3.2.1.3	Preamble	55
3.2.1.4	Request	56
3.2.2	CUDF Example	57
3.2.3	Mancoosi Optimisation Format	59
3.2.4	CUDF*	60
3.2.5	CUDF* Example	61
3.3	Summary	64
4	User Model	65
4.1	User Survey	66
4.1.1	Questions	66
4.1.2	Results	67
4.1.3	Progressive vs. Conservative Users	68

4.2	SimUser model	69
4.2.1	Variables and Assumptions	69
4.2.2	CUDF* Document Creation	71
4.3	SimUser Data Collection and Conversion	72
4.3.1	Collecting the Components	73
4.3.2	Probability a component will be selected	74
4.4	SimUser Validation	75
4.4.1	Randomness of SimUser	75
4.4.2	Limitations of SimUser	77
4.4.3	Perspective of the Ubuntu Repository	78
4.5	Summary	79
5	Resolving CoSyE Instances	81
5.1	Boolean Lexicographic Optimization Problem	82
5.1.1	Boolean Satisfiability Problem (SAT)	83
5.1.1.1	Pseudo-Boolean Extension of SAT to SAT+PB	84
5.1.2	Boolean Lexicographic Optimization	86
5.1.3	Mapping CoSyE Instance to BLO Problems	87
5.1.4	Evolution Preference Order Mapping	89
5.2	Criteria	90
5.2.1	Change Criteria	90
5.2.2	Out-of-date Criteria	91
5.2.3	One Version per Package Criteria	93
5.3	Solving a BLO Problem	93
5.3.1	Davis-Putnam-Logemann-Loveland Algorithm for SAT Solvers	94
5.3.1.1	Unit Propagation	95
5.3.1.2	Decide	95
5.3.1.3	DPLL Advancements	96
5.3.2	Iterative Strengthening	96
5.3.3	Lexicographic Optimisation	98
5.3.4	Resolving a CoSyE instance	99
5.4	GJSolver	100
5.4.1	GJSolver Implementation	100
5.4.1.1	SAT4J	101
5.4.2	Validation of GJSolver	102
5.4.2.1	Mancoosi International Solver Competition	102
5.4.2.2	Tracks and Scoring	103
5.4.2.3	MISC Live	104
5.4.2.4	MISC	104

5.4.2.5	Analysis	105
5.5	Simulation Validation	105
5.5.1	Comparison to <code>apt-get</code> Logs	106
5.5.2	Comparison to Virtual System	107
5.6	Summary	108
6	Experiments, Results and Analysis	109
6.1	Users Choices to Upgrade and Install	110
6.1.1	Boundary Cases	110
6.1.1.1	Results and Analysis	111
6.1.2	Failures	113
6.1.2.1	Hard Failures	114
6.1.2.2	Soft Failures	114
6.1.3	Upgrade Probability Effects	115
6.1.4	Install Probability Effects	118
6.2	Reduction of Out-of-dateness During Evolution	119
6.3	Reduction of Change During Evolution	121
6.4	Realistic Evolution	123
6.4.1	Extracting Information from the User Submitted Logs	124
6.4.2	Simulation of Realistic Users	125
6.5	Answers	129
6.5.1	User Choices	129
6.5.2	Reduce Out-of-dateness	130
6.5.3	Reduce Change	130
6.5.4	Real Users	131
6.6	Summary	131
7	Conclusion	133
7.1	Thesis Validation	134
7.2	Future Research	135
7.3	Closing Remark	136
A	CUDF* Parsing to CoSyE Instance	137
A.1	CUDF* BNF Grammar	137
A.2	Additional Stanza Constraints	139
A.3	Parsing	140
A.4	Components	140
A.5	Features	141
A.6	Package formula	142

A.6.1	Sets of Package Formula	143
A.7	System Constraints	143
A.7.1	Keep Constraints	143
A.7.2	Dependency Constraint	144
A.7.3	Conflict Constraint	145
A.8	Request	145
A.8.1	Install	146
A.8.2	Remove	146
A.8.3	Upgrade	146
A.9	Criteria	148
B	Full Criteria Mapping	149
B.1	-changed	150
B.2	-removed	150
B.3	-new	151
B.4	-uptodatedistance	151
B.5	-ovpp	152
B.6	-unstable(d)	152
	Bibliography	153

List of Tables

2.1	Summary of presented component models' constraints.	37
2.2	Summary of presented component models' constraints description formats.	38
2.3	Summary of presented component models' properties.	38
4.1	Summary of the survey respondents types and frequencies of interactions with their package managers.	67
5.1	Example of the mapping from CoSyE criterion to a PB criterion.	90
5.2	Mapping of the change, removed and new criteria between MOF, CoSyE and PB	91
5.3	Mapping of the up-to-date distance criterion between MOF, CoSyE and PB	93
5.4	Mapping of the one version per package criterion between MOF, CoSyE and PB	93
5.5	Results from MISC 2011, (score (less is better) : time in seconds)	105
5.6	Comparison of installed packages between the collected <code>apt-get</code> logs and the simulation.	107
5.7	Comparison of upgraded packages between the virtual Ubuntu 11.10 system and the simulation.	107
6.1	Configuration of users that are the boundary cases in the simulation.	110
6.2	Configuration of users with variable probability to upgrade.	115
6.3	Configuration of users with variable probability to install a component.	118
6.4	Configuration of progressive user that always upgrades.	119
6.5	Mapping of the unstable criterion between MOF, CoSyE and PB	121
6.6	Configuration of users using the unstable criterion	122
6.7	Reduced change compared to the estimated reduced change of using the unstable criterion.	123
6.8	Configuration of "realistic" users extracted from the submitted <code>apt-get</code> logs	125
6.9	The mean final UTpC of the simulated "realistic" users using <code>apt-get</code> and progressive criteria	127

6.10	The mean total change of the simulated “realistic” users using <code>apt-get</code> and progressive criteria	127
6.11	The mean estimated reduced change when using the unstable criterion for the simulated “realistic” users using <code>apt-get</code> and progressive criteria	129
A.1	CUDF* Preamble properties	139
A.2	CUDF* Package Description properties	139
A.3	CUDF* Request properties	139
B.1	Mapping of the change, removed, new, one version per package, up-to-date distance and unstable criteria between MOF, CoSyE and PB . . .	149

List of Figures

2.1	Example of OSGi Meta-data	22
2.2	Example of OSGi Declarative Services meta-data	23
2.3	Example of OSGi Bundle Repository meta-data	24
2.4	Example of an Eclipse Plugin plugin.xml meta-data file	26
2.5	Example of an Eclipse Plugin extension point schema file	27
2.6	Example of a simple Fractal ADL file	28
2.7	Example of a hierarchical Fractal ADL file	29
2.8	Example of a Maven POM file	31
2.9	Debian Control file for Text Editor	32
2.10	Example of a Debian Control File for Spell Checker	32
2.11	Example of a SOFA ADL Files	35
2.12	Example of a CUDF file	36
3.1	The relationships between CUDF* and CoSyE	42
3.2	Structure of the CUDF stanzas	54
3.3	Structure of a CUDF stanza	54
3.4	Example of Package Description Stanza	55
3.5	Example of preamble stanza demonstrating the extendable CUDF language	56
3.6	Example of CUDF request stanza	57
3.7	Example of a CUDF document	58
3.8	Structure of the CUDF* stanzas	61
3.9	Example of a CUDF* document	62
4.1	Relationships between the SimUser model and CUDF*	65
4.2	The process to create a CUDF* document from a SimUser instance. . .	71
5.1	The relationships between the BLO problem and the CoSyE model . . .	82
5.2	The DPLL algorithm	94
5.3	The Unit Propagation algorithm	95
5.4	The Iterative Strengthening algorithm	97
5.5	The Lexicographic Iterative Strengthening algorithm	98
5.6	The algorithm to resolve a CoSyE instance	100

5.7	The relationships within the GJSolver implementation	100
6.1	The UTTPdC of the simulated boundary case users.	111
6.2	The total change of the simulated boundary case users.	112
6.3	The UTTPdC of the variable upgrade users.	116
6.4	The mean UTTPdC against of the frequency a user upgrades ($1/u$)	116
6.5	The mean total change of the variable upgrade users.	117
6.6	The mean total change of the variable install users	119
6.7	The UTTPdC of the simulated progressive user.	120
6.8	The UTTPdC of the simulated users using the unstable criterion	122
6.9	An extract of an <code>apt-get</code> log file	124
6.10	The upgrade and install probabilities of the submitted <code>apt-get</code> logs and the “realistic” users	125
6.11	The UTTPdC for the simulated “realistic” users. Top: the <code>apt-get</code> criteria users. Bottom: the progressive users.	126
6.12	The mean total change for the simulated “realistic” users. Top: the <code>apt-get</code> criteria users. Bottom: the progressive users.	128
A.1	CUDF* BNF Grammar	138