

An Empirical Investigation of Software Reliability Indicators

Sharifah Mashita Syed Mohamad (B.IT., M.CS.)

A dissertation submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Faculty of Engineering and Information Technology
University of Technology, Sydney

February 2012

CERTIFICATE OF AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Production Note:
Signature removed prior to publication.

Acknowledgement

- I would like to express my deepest and sincerest gratitude to my supervisors. Dr. Tom McBride, I greatly appreciate the encouragement, guidance and generous sharing of ideas and profound knowledge in software engineering throughout the research process. Thank you for your time and effort in assisting me. Professor Didar Zowghi, I admire your sharp argumentation and attention to details. Your guidance and insight into the whole process were truly invaluable.

Next, to my beloved husband, Syed and son, Arfaan, thank you for being very understanding, supportive and for all the sacrifices both of you did for me. I am forever indebted to my parents for their unconditional love, endless patience and caring support. I thank my parents-in-law too for their love and help. I greatly appreciate the logistical support given by my brother, Mahadi during the field work. To Dr. Sharifah Mastura, thank you for your moral support and endless inspiration. A special thanks to Athira for her help in the thesis writing. There are no words to express my gratitude for all the prayers from all family members.

This thesis would not been possible without the willingness and enthusiasm shown by all the participants involved in this project. Thank you for all of the trust and opportunity in conducting this research project. Thank you also to the operation manager and his team at the telecommunications company for providing us the defect data.

I am grateful to my thesis editor, Mr. Jules Baumann, for his great efforts in proofreading my work. I also would like to express my gratefulness to Dr. Richard Raban, the head of School of Software, and friends of Human Centered Technology Design (HCTD) in University of Technology, Sydney.

My appreciation goes as well to my sponsors, The Ministry of Higher Education, Malaysia and University of Science, Malaysia (USM). My thanks also go the academics at USM; Professor Rosni from the School of Computer Science, Professor Hamidi and Professor Badorul, both from the School of Civil Engineering. Finally, not to be missed, Dr. Kamal Zuhairi from the School of Electrical and Electronic Engineering for sharing his opinion on software testing.

Table of contents

1. Introduction	1
1.1 Research strategy	5
1.2 Research objectives and research questions	6
1.3 Contributions	8
1.4 Thesis overview	10
2. Evaluating Software Reliability	12
2.1 Reliability of open source and commercial software products	13
2.2 Software reliability	15
2.2.1 Fault, failure, defect and bug definitions	15
2.2.2 Software reliability definitions	15
2.2.3 Software reliability models	17
2.2.4 Summary	18
2.3 Software reliability growth models	18
2.3.1 Assumptions	21
2.3.2 Stabilisation phase	23
2.3.3 Summary	24
2.4 Rapidly evolving software	24
2.4.1 Open source software	25
2.4.2 Characteristics of software development	36
2.4.3 Testing early and often	39
2.4.4 The concept of rapidly evolving software	40
2.4.5 Summary	43
2.5 Software testing	44
2.5.1 Test levels	45
2.5.2 Test coverage and test adequacy criteria	47
2.5.3 Criteria for test completion	48
2.5.4 Summary	50
2.6 Conclusions	50
3. Exploratory and Confirmatory Studies	53
3.1 The case study methodology	55
3.1.1 Defect modelling	56
3.2 Ethical consideration	56
3.3 Stage 1: Exploratory Study	57
3.3.1 Research question number one	57

3.3.2	Sample selection	58
3.3.3	Data collection	59
3.3.4	Analysis and findings	62
3.3.5	Discussions	73
3.3.6	Conclusion	76
3.4	Stage 2: Confirmatory Study	77
3.4.1	Research question number two	77
3.4.2	Sample selection	78
3.4.3	Data collection	79
3.4.4	Analysis and findings	83
3.4.5	Discussion	102
3.5	Compilation of the six case studies and observations	104
3.6	Threats to validity	106
3.6.1	Internal validity	107
3.6.2	External validity	110
3.6.3	Construct validity	111
3.7	Discussion	112
3.7.1	Defect-based reliability indicators	112
3.7.2	Assumptions	113
3.8	Conclusions	114
4.	Alternative Reliability Indicators for Software Products	115
4.1	Test-based reliability indicators	116
4.2	Relationship between software reliability and test coverage	117
4.3	Focus Study: research objectives and research questions	119
4.4	Conclusion	122
5.	Focus Study: Research Design	123
5.1	Research design	123
5.1.1	Research methods in existing studies	123
5.1.2	Data collection method	126
5.1.3	Research Instrument	126
5.1.4	Research Ethics	131
5.1.5	Sample Selection	132
5.1.6	Conducting the Interview	135
5.1.7	External Validity	136
5.1.8	Construct Validity	136
5.1.9	Internal Validity	138

5.2	Conclusion	138
6.	Focus Study: Analysis and Findings	140
6.1	The analysis process	141
6.1.1	Content analysis	141
6.2	Initial and second round of interviews	142
6.3	Analysis of the initial interviews	143
6.3.1	Shortcomings and changes made to the interview questions	144
6.3.2	Early conclusions	147
6.4	Sample characteristics	149
6.4.1	Participant locations	149
6.4.2	Participant groups	149
6.4.3	Software domains	150
6.4.4	Team Size	152
6.4.5	Project release schedules	153
6.5	Test Coverage	154
6.5.1	Test coverage measure	154
6.5.2	Test evolution and maintenance	159
6.6	How much testing is enough	163
6.6.1	Levels of testing	163
6.6.2	Unit testing	165
6.6.3	Integration testing	171
6.6.4	System testing	174
6.6.5	Acceptance testing	179
6.6.6	Regression testing	182
6.6.7	Defect Analysis	185
6.6.8	Criteria for Test Completion	186
6.7	Test rigour	190
6.7.1	Test effectiveness	190
6.7.2	Test efficiency	194
6.7.3	Too many or too few defects	195
6.8	Reliability definitions	200
6.9	The use of testing	202
6.10	Discussion	205
6.10.1	Interpretation of testing approach	205
6.10.2	Test automation and practices	206
6.10.3	Continuous stabilisation of rapidly evolving software	207

6.10.4	Test coverage is used as a visible measure of project progress	208
6.10.5	Tacit and explicit measures	209
6.10.6	Open source software	210
6.11	Conclusions	210
7.	Applying test-based indicators to aid decisions about acquisition of open source software	211
7.1	Assessing the open source software reliability	211
7.2	Test-based indicators for Open Source X	212
7.2.1	Empirical evidence related to tests in open source	213
7.2.2	Description of the steps used	213
7.2.3	Empirical results	218
8.	Conclusions, Implications, Future Research and Concluding Remarks	222
8.1	Defect-based indicators of software reliability	222
8.2	Test-based indicators	224
8.2.1	Revisiting research question 3	224
8.3	Summarisation of the indicators with respect to software reliability	227
8.4	Implications of the study	228
8.4.1	Implications for research	229
8.4.2	Implications for practice	229
8.5	Future research	231
8.5.1	Open source software reliability measures	231
8.5.2	The value of tools and automation	231
8.6	Concluding remarks	232
	Bibliography	234
	Appendix A - The open source definition	248
	Appendix B – ODC description used in the inter-rater agreement study	251
	Appendix C – Introduction used in the inter-rater agreement study	257
	Appendix D – Results of the inter-rater agreement study	260
	Appendix E – Survey questions	261
	Appendix F – Survey questions (open source software)	269
	Appendix G – Survey introduction	271
	Appendix H – Consent form	272
	Appendix I – Initial survey data table (excerpt)	273

List of figures

Figure 2-1: Conceptual map of the background of the thesis	13
Figure 2-2: Concave and S-shaped reliability growth models (Wood, 1996)	19
Figure 2-3: Infeasible to fit a model at the time of release in Li <i>et al.</i> (2005a)	34
Figure 2-4: V-model of testing	45
Figure 3-1: Three-stage empirical research.....	53
Figure 3-2: The Case Study methodology for empirical stages 1 and 2	55
Figure 3-3: Structure of the exploratory study	58
Figure 3-4: Example of Concave growth modelling in SPSS.....	63
Figure 3-5: The distribution of defects over time of Open Source A.....	65
Figure 3-6: The distribution of defects over time of Open Source B.....	65
Figure 3-7: The distribution of defects over time of In-house C	66
Figure 3-8: Illustrating relative growth using ODC (Chillarege and Biyani, 1994, pg. 285)	69
Figure 3-9: The distribution of defects during 2007 of Open Source B by ODC defect-type	71
Figure 3-10: The distribution of defects during 2007 of In house C by ODC defect-type	71
Figure 3-11: ODC defect type distribution of open source per year 2007	72
Figure 3-12: ODC defect type distribution of In-house C per year 2007	73
Figure 3-13: Structure of the confirmatory study	78
Figure 3-14: The distribution of defects for 11 releases of Incremental A.	85
Figure 3-15: The distribution of defects for 15 releases of Incremental B.	85
Figure 3-16: Concave growth modelling of Incremental A version 0.6.....	86
Figure 3-17: S-shaped growth modelling of Incremental A version 0.6.	87
Figure 3-18: Inflection S-shaped growth modelling of Incremental A version 0.6.....	87
Figure 3-19: Concave growth modelling of Incremental B version 2.7.....	88
Figure 3-20: S-shaped growth modelling of Incremental B version 2.7.....	88
Figure 3-21: Inflection S-shaped growth modelling of Incremental B version 2.7.	89
Figure 3-22: Defect density vs. product size for Incremental A.	91
Figure 3-23: Defect density vs. product size for Incremental B.	92
Figure 3-24: The distribution of defects for 15 releases of Incremental C	94
Figure 3-25: Concave growth modelling of Incremental C version 0.5.3.....	95
Figure 3-26: S-shaped growth modelling of Incremental C version 0.5.3	95
Figure 3-27: Inflection S-shaped growth modelling of Incremental C version 0.5.3	96
Figure 3-28: Defect density vs. product size for Incremental C	97
Figure 3-29: The distribution of defects for 6 releases of Incremental D	98
Figure 3-30: Concave growth modelling of Incremental D version 2.2.1	99
Figure 3-31: S-shaped growth modelling of Incremental D version 2.2.1.....	100

Figure 3-32: Inflection S-shaped growth modelling of Incremental D version 2.2.1	100
Figure 3-33: The distribution of development defects and post-release defects for release 2.3 of Incremental D.....	101
Figure 3-34: Defect density vs. \bar{p} product size for Incremental D	102
Figure 3-35: Convex (and concave) pattern in the defect modelling of open source products developed by the community	104
Figure 3-36: Linear pattern in the defect modelling of open source products developed by a company	105
Figure 3-37: Defect density of two open source products developed by a company	106
Figure 3-38: Defect density of two open source products developed by the community	106
Figure 4-1: Concepts related to test-based reliability indicators.....	116
Figure 5-1: Methods in empirical studies on software testing	125
Figure 6-1: First part of the questionnaire	145
Figure 6-2: Second part of the questionnaire	146
Figure 6-3: Division of locations where the data was collected	149
Figure 6-4: Three groups of participants in the sample	150
Figure 6-5: Team size with testing practices in the sample.....	152
Figure 6-6: Distribution frequencies of project release schedules	153
Figure 6-7: Summary of the test coverage measure findings.....	157
Figure 6-8: Responses to the question of unit, integration and system testing.	164
Figure 6-9: Criteria for test completion as stated by participants of the development and independent software testing groups.....	189
Figure 6-10: Measures of test effectiveness.....	192
Figure 6-11: Practitioners' understanding of software reliability.....	200
Figure 7-1: Methods of assessing open source software reliability	212
Figure 7-2: Code coverage evolution in Open Source X (Zaidman <i>et al.</i> , 2008)	219
Figure 7-3: Production and test code evolution in Open Source X (Zaidman <i>et al.</i> , 2008)	220

List of tables

Table 2-1: Mathematical representation of the growth models.....	19
Table 2-2: Concave (also known as exponential) type of models.....	20
Table 2-3: S-shaped type of models.....	20
Table 2-4: Groups and roles of the open source software development community (Xu <i>et al.</i> , 2005).....	26
Table 2-5: The definitions of rapidly evolving and periodically stabilised software.....	42
Table 3-1: Open Source A details.....	60
Table 3-2: Open Source B details.....	61
Table 3-3: In-house C details.....	61
Table 3-4: Estimated parameters (b) and goodness of fit of the reliability growth models.....	66
Table 3-5: The ODC defect type and process associations (Chillarege <i>et al.</i> , 1992, pg. 947)....	67
Table 3-6: Incremental A details.....	79
Table 3-7: Incremental B details.....	80
Table 3-8: Incremental C details.....	82
Table 3-9: Incremental D details.....	83
Table 3-10: Defect density results of each release of Incremental A.....	90
Table 3-11: Defect density results of each release of Incremental B.....	91
Table 3-12: The differences between the software products.....	93
Table 3-13: The similarities of Incremental A and B.....	93
Table 3-14: Defect density results of Incremental C.....	96
Table 3-15: Defect density results of Incremental D.....	101
Table 3-16: The mean of defect counts.....	105
Table 3-17: Inter-rater reliability coefficient results between the researcher (main rater) and four raters.....	109
Table 3-18: Descriptions of software products under examination.....	111
Table 5-1: Mapping research questions and interview questions.....	129
Table 6-1: Software domains.....	151
Table 6-2: Unit testing findings.....	169
Table 6-3: Integration testing findings.....	173
Table 6-4: System testing findings.....	178
Table 6-5: Acceptance testing findings.....	181
Table 6-6: Regression testing findings.....	185
Table 6-7: Criteria for test completion as stated by participants of the development and independent software testing groups.....	187
Table 7-1: Open Source X release history.....	214
Table 8-1: Categorization of the indicators based on the stabilisation phase criterion.....	228

Terminology

Commercial-Off-The-Shelf (COTS)	A commercial software product that is developed for the general market, that is to be used by large numbers of people with a variety of users in different scenarios (Yang <i>et al.</i> , 2005 ; Boehm, 2006 ; Bishop <i>et al.</i> , 2007).
Commercial software	Proprietary software that is developed using in-house commercial development processes.
Open source software	In contrast to commercial software, open source software refers to software that is developed through open source software development. Open source software is software for which the source code is made freely available for others to access, use, copy, modify and redistribute (Raymond, 2000). This term is discussed in details in Section 2.4.1.
Open source software development	<p>A collaborative way of producing software by widely dispersed developers collaborating over the Internet (Stallman, 1999 ; O'Reilly, 1999 ; Raymond, 2000 ; Scacchi, 2002 ; Scacchi <i>et al.</i>, 2006 ; Crowston and Annabi, 2007 ; Wu <i>et al.</i>, 2007).</p> <p>The Open Source Initiative (2006) describes it as “a development method for software that harnesses the power of distributed peer review and transparency of process” among the open source community.</p>
Periodically stabilised software	<p>A term used to refer to software that does have a stabilisation phase in its development process.</p> <p>This term is introduced as a result of the investigation of this research and discussed in Section 2.4.4.</p>
Rapidly evolving software	<p>A term used to refer to software that does not have a stabilisation phase in its development process. A characteristic of this development method is that new features and functionality are constantly introduced at the same time that defects are being rectified.</p> <p>This term is introduced as a result of the investigation of this research and discussed in Section 2.4.4.</p>

Software reliability growth	<p>Software reliability growth is defined as an improvement in the reliability over a period of time (Lyu, 1996). Growth in reliability usually happens through freezing the functionality at a certain stage during system testing, and then fixing defects as they are detected. The rate of detection and fixing of outstanding defects and the overall decline in the number of outstanding defects with respect to time or (testing effort) indicates the level of reliability.</p> <p>A growth in reliability signs that the software had reached an adequate level of reliability before being deployed. It is observed during a stabilisation phase.</p>
Stabilisation phase	<p>A stabilisation phase is a time period where the software does not have any more features added (i.e. “feature freeze”). This feature freeze is enforced during which system testing is conducted and all the defects that are found are repaired. As a result of this testing-debugging process, a definite stabilising trend in the defect modelling can be observed - which indicates a growth in reliability of software.</p> <p>The phase is long enough to observe the growth and decline of exposed defects. Obviously, system testing will continue until no failure can arise from the defect fix. This term is discussed in Section 2.3.2.</p>
Test-first	<p>Test-first is a programming approach where software developers write unit tests (including functional tests) before writing the code.</p> <p>Case studies in the literature often use the term “test-first” when making comparison with the test-last programming approach (Siniaalto and Abrahamsson, 2007 ; Dybå and Dingsøy, 2008 ; Nagappan <i>et al.</i>, 2008). So, for the purpose of this research, the term ‘test-first’ is used throughout this thesis.</p>
Test-last	<p>In contrast to test-first, test-last is a “code and then test” programming approach as typically done in software development.</p>
Test rigour	<p>Test rigour is concerned with effectiveness and efficiency of testing. The term “effective testing” normally means the capability of finding defects, whereas “efficient testing” is defined as the capability of finding defects in a timely and cost effective manner (Bertolino, 2007). Test rigour is defined in Section 4.3.</p>
Test coverage	<p>A measure of how well a test suite tests a program. Test coverage can be measured in terms of code, requirements and design</p>

	coverage. Test coverage is defined in Section 4.3.
Test sufficiency	Test sufficiency covers the criteria for test completion or stopping criteria and release software. Test sufficiency is defined in Section 4.3.

Abstract

This thesis investigates how individuals and organisations, without technical skills, might determine the reliability of open source software given the increased use of such software. Software reliability is normally indicated by the growth and subsequent decline of defects in the software. A notable observation is that reliability growth models require a definitive stabilisation phase during which testing can reveal the growth and decline of defects as the indication of the increase in reliability of software. However, there is not necessarily a definitive stabilisation phase in the open source software development. More importantly, the presence or absence of the stabilisation phase is an attribute of a software development method and is not restricted to open source software. When software is developed without a definitive stabilisation phase, reliability growth models are not applicable because the conditions for their validity have not been achieved. Consequently, this thesis looks for alternative information based on tests to aid decision-making about software acquisition. Data was collected by conducting semi-structured interviews from 29 participants who were currently engaged in software development. The information of tests; coverage, sufficiency and rigours of tests concerns the testing that has been performed on the software product and gives expectations on how well the software product has been tested.