A DOCUMENT-BASED TRACEABILITY MODEL FOR TEST MANAGEMENT

AZRI BIN AZMI

UNIVERSITI TEKNOLOGI MALAYSIA

A DOCUMENT-BASED TRACEABILITY MODEL FOR TEST MANAGEMENT

AZRI BIN AZMI

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

DECEMBER 2017

Specially dedicated to all my family and relatives.

ACKNOWLEDGEMENT

First of all, I am thankful to God for giving me the opportunity and strength to complete my PhD. I would like to take this opportunity to thank my main supervisor, Prof. Dr. Suhaimi Bin Ibrahim for his encouragement, constant support, advice and inspiration throughout this research.

I would also like to thank my wife and kids for their patience during the course of my PhD. I am also grateful to all my friends who kept me motivated during my research especially to Dr. Othman, Dr. Saiful, Mdm. Haslina and Dr. Nazri Kama.

My thanks also go to all UTM AIS staff and individuals who have been involved directly or indirectly in the project. Lastly, my appreciation also goes to the experts for evaluating my model and prototype.

ABSTRACT

Software testing has became more complicated in the emergence of distributed network, real-time environment, third party software enablers and the need to test system at multiple integration levels. These scenarios have created more concern over the quality of software testing. The quality of software has been deteriorating due to inefficient and ineffective testing activities. One of the main flaws is due to ineffective use of test management to manage software documentations. In documentations, it is difficult to detect and trace bugs in some related documents of which traceability is the major concern. Currently, various studies have been conducted on test management, however very few have focused on document traceability in particular to support the error propagation with respect to documentation. The objective of this thesis is to develop a new traceability model that integrates software engineering documents to support test management. The artefacts refer to requirements, design, source code, test description and test result. The proposed model managed to tackle software traceability in both forward and backward propagations by implementing multi-bidirectional pointer. This platform enabled the test manager to navigate and capture a set of related artefacts to support test management process. A new prototype was developed to facilitate observation of software traceability on all related artefacts across the entire documentation lifecycle. The proposed model was then applied to a case study of a finished software development project with a complete set of software documents called the On-Board Automobile (OBA). The proposed model was evaluated qualitatively and quantitatively using the feature analysis, precision and recall, and expert validation. The evaluation results proved that the proposed model and its prototype were justified and significant to support test management.

ABSTRAK

Pengujian perisian menjadi semakin rumit dengan kemunculan rangkaian teragih, persekitaran masa-nyata, pembekal perisian pihak ketiga dan keperluan untuk menguji sistem pada pelbagai peringkat penggabungan. Senario-senario ini telah mencetuskan keprihatinan terhadap kualiti pengujian perisian. Kualiti perisian menjadi semakin kurang akibat aktiviti pengujian yang tidak cekap dan tidak berkesan. Salah satu kelemahan utama adalah berikutan penggunaan pengurusan ujian yang tidak efektif untuk menguruskan dokumen perisian. Dalam dokumentasi, adalah sukar untuk mengesan dan menjejak pepijat dalam beberapa dokumen berkaitan dimana jejak adalah kebimbangan utama. Pada masa kini, pelbagai kajian telah dijalankan pada pengurusan ujian, namun sangat sedikit memberi tumpuan kepada jujuk dokumen khususnya untuk menyokong penyebaran ralat berkenaan dengan dokumentasi. Objektif tesis ini adalah untuk membangunkan satu model jejak baharu yang menggabungkan dokumen kejuruteraan perisian untuk menyokong pengurusan ujian. Artifak-artifak itu merangkumi keperluan, reka bentuk, kod sumber, huraian ujian dan hasil ujian. Model yang dicadangkan ini dapat menangani masalah jejak perisian dalam kedua-dua jejak ke hadapan dan ke belakang dengan melaksanakan penuding berbilang dwi arah. Platform ini membolehkan pengurus ujian mengemudi dan menangkap satu set artifak yang berkaitan untuk menyokong proses pengurusan ujian. Prototaip baharu telah dibangunkan untuk memudahkan pemerhatian jujuk perisian pada semua artifak berkaitan di seluruh kitaran hayat dokumentasi. Model yang dicadangkan kemudiannya diaplikasikan ke atas satu kajian kes projek pembangunan perisian yang selesai dengan satu set lengkap dokumen perisian yang dipanggil *On-Board Automobile (OBA).* Model yang dicadangkan telah dinilai secara kualitatif dan kuantitatif menggunakan analisis ciri, *precision* dan *recall*, dan pengesahan pakar. Keputusan penilaian membuktikan bahawa model yang dicadangkan dan prototaipnya adalah wajar dan penting untuk menyokong pengurusan ujian.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	ii
	DEDICATION	iii
	ACKNOWLEDGEMENT	iv
	ABSTRACT	v
	ABSTRAK	vi
	TABLE OF CONTENTS	vii
	LIST OF TABLES	xiii
	LIST OF FIGURES	xv
	LIST OF ABBREVIATIONS	xix
	LIST OF APPENDICES	xx
1	**INTRODUCTION**	**1**
	1.1 Overview	1
	1.2 Background of the Problem	3
	1.3 Statement of the Problem	6
	1.4 Objectives of the Study	7
	1.5 Scope of the Research	7
	1.6 Significance of the Study	8
	1.7 Thesis Outline	9
2	**LITERATURE REVIEW**	**11**
	2.1 Introduction	11
	2.2 Software Testing	11

2.2.1 Discussion on the Motivation of Software Testing 13

2.3 Software Test Management 14

2.3.1 Activities of Test Management 15

2.3.2 Current Researches on Test Management 18

2.3.2.1 TAI Approach 18

2.3.2.2 Sigrid's Approach 19

2.3.2.3 Agent-based Test Management Approach 20

2.3.2.4 Architecture Centric Approach 22

2.3.2.5 Summary of Test Management Approaches 23

2.3.3 Test Management Issues 24

2.3.4 Test Management Summary 28

2.4 Software Documentation 28

2.4.1 Software Testing Documentations versus Test Management 30

2.4.2 Software Documentation Issues 31

2.4.3 Software Documentation Summary 35

2.5 Software Traceability 35

2.5.1 Introduction of Software Traceability 35

2.5.2 Dimension of Traceability 36

2.5.2.1 Forward and Backward Traceability 37

2.5.2.2 Implicit and Explicit Traceability 37

2.5.2.3 Intra-Level and Inter-Level Traceability 38

2.5.2.4 Material and Immaterial Traceability 39

2.5.2.5 Pre-RS Traceability and Post-RS Traceability 39

2.5.3 The Needs for Traceability 39

2.5.4 Traceability Approach versus Traceability

		Model	40
	2.5.5	Some Studies on Existing Traceability Approaches	41
		2.5.5.1 Information Retrieval Traceability	42
		2.5.5.2 Event-based Traceability	44
		2.5.5.3 Goal Centric Traceability	44
		2.5.5.4 Scenario-Based Traceability	45
		2.5.5.5 Rule-Based Traceability	46
		2.5.5.6 Evaluation and Comparative Study of Traceability Approaches	47
	2.5.6	Some Studies on Existing Traceability Model	50
		2.5.6.1 Inter Requirement Traceability Model	52
		2.5.6.2 Coding Phase Requirements Traceability Model	53
		2.5.6.3 Total Traceability Model	54
		2.5.6.4 End-to-End Traceability Model	55
		2.5.6.5 TraceabilityWeb Model	56
		2.5.6.6 Evaluation of Software Traceability Models	58
		2.5.6.7 Comparative Study and Criteria for Model Development	59
	2.5.7	Discussion of Software Traceability and Documentation	65
	2.5.8	Issues in Traceability	66
	2.5.9	Summary of Software Traceability	71
	2.6	Summary of the Chapter	71
3	**RESEARCH METHODOLOGY**		**73**
	3.1	Introduction	73
	3.2	Research Design Strategies and Research Process	73

3.3 Theoretical Framework 74
3.4 Operational Framework 75
3.5 Research Procedure and Activities 78
3.5.1 Phase 1: Literature Review 78
3.5.2 Phase 2: Modelling 79
3.5.3 Phase 3: Design and Development 80
3.5.4 Phase 4: Evaluation 80
3.5.4.1 Qualitative Method – Feature Analysis 81
3.5.4.2 Feature Analysis 82
3.5.4.3 Features Rating 83
3.5.4.4 Level of Importance 84
3.5.5 Feature Set and Overall Scores 85
3.5.5.1 Quantitative Method – Precision and Recall 87
3.5.5.2 Qualitative Method – Expert Validation 89
3.5.5.3 Questionnaire 90
3.6 Instrumentation 91
3.6.1 Case Study 92
3.6.2 OBA Overview 92
3.6.2 OBA Interface 93
3.6.2.1 Driving Station 93
3.6.2.2 Control Panel (MMI) 94
3.6.2.3 Transmission Shaft 95
3.6.2.4 Throttle 95
3.7 Assumptions and Limitations 95
3.8 Summary of Research Methodology 96

4 DESIGN AND DEVELOPMENT OF THE PROPOSED MODEL 97
4.1 Overview 97
4.2 Rationale of the DBT Model 97

	4.3	Construction of the DBT Model Component	99
	4.4	Definition of the DBT Model	100
		4.4.1 Extractor	102
		4.4.2 XML Repository	103
		4.4.3 Analyzer	103
		4.4.4 Traceability Engine	104
		4.4.5 Traceability Repository	104
		4.4.6 Document Generator	105
	4.5	The DBT Architecture	105
	4.6	The DBT Process	106
	4.7	Design and Development of DBT	109
		4.7.1 DBT Use Cases	109
		4.7.2 DBT Class Diagram	110
		4.7.3 DBT Package Diagram Details	111
		4.7.3.1 Extractor Package	113
		4.7.3.2 Analyzer Package	115
		4.7.3.3 Traceability Engine Package	119
		4.7.3.4 Report Generator Package	126
		4.7.3.5 Queries Package	127
		4.7.4 DBT Relational Database	127
		4.7.4.1 Perform Trace Links Between Artefacts	130
	4.8	DBT Graphical User Interface	134
	4.9	Discussion on Design and Development of the Proposed Model	140
	4.10	Summary	141
5	**EVALUATION OF THE PROPOSED MODEL**		**142**
	5.1	Overview	142
	5.2	Features Mapping for Evaluation of Proposed Model	142
	5.3	Results of Feature Analysis	146
		5.3.1 Feature Set 1	146

		5.3.2 Feature Set 2	150
		5.3.3 Feature Set 3	152
		5.3.4 Feature Set 4	153
		5.3.5 Feature Set 5	155
		5.3.6 Summary Result of Feature Analysis	157
	5.4	Case Study - OBA	161
		5.4.1 OBA Functionalities	162
		5.4.2 Traceability in OBA	164
		5.4.3 OBA Dataset	165
	5.5	Quantitative Evaluation Method – Precision and Recall	167
		5.5.1 Traceability Between STD and SRS	167
		5.5.2 Traceability Between SDD and SRS	170
	5.6	Expert Validation	173
		5.6.1 Questionnaire Result	173
	5.7	Discussion on the Evaluation Results	179
	5.8	Summary	181
6	**CONCLUSION**		**182**
	6.1	Summary and Achievement	182
	6.2	Contributions of the Research	184
	6.3	Limitation and Feature Works	185
REFERENCES			**187**
Appendices A - E			207 - 235

LIST OF TABLES

TABLE NO.	TITLE	PAGE
2.1	Test management approach comparison	23
2.2	Test management issues	26
2.3	Software documentation issues	33
2.4	Evaluation of traceability approaches	48
2.5	Comparative study of traceability approaches	51
2.6	Comparative result of traceability models	58
2.7	Comparative study of traceability models	62
2.8	Criteria for model of component	63
2.9	Traceability issues	67
3.1	Operational framework	77
3.2	Method proposed by DESMET	81
3.3	Scoring scale	83
3.4	Level of importance of a feature with multiplier	84
3.5	Features and sub-features used in the analysis	85
3.6	Feature set weighting	87
3.7	Usability attributes adoption	91
4.1	Selection of component for DBT model	99
4.2	List of table in DBT	128

5.1	Selection of features for software development tool evaluation	145
5.2	Comparison of overall scores details of feature analysis	148
5.3	Feature set scores and overall scores comparison	158
5.4	OBA documentation set	166
5.5	OBA component/work product	166
5.6	OBA tracing activities	167
5.7	Tracing between STD and SRS results	169
5.8	Tracing between SDD and SRS results	171
5.9	Experts profile	174
5.10	Cross tabulation percentage of gender and qualification	174
5.11	Cross tabulation of working experience	174
5.12	List of questions and usability criteria	175
5.13	Mean and standard deviation calculation	176
5.14	Comments/suggestions received from the experts	178
5.15	5-level mean analysis	178

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
2.1	The V-model with software testing life cycle	13
2.2	System architecture of TAI	19
2.3	The test preparation phase in test management system	20
2.4	Adaptive test management system	21
2.5	Structure element service diagram	23
2.6	Software documentation in SDLC	29
2.7	Dimension of traceability	38
2.8	Inter-requirements traceability model	53
2.9	Coding phase requirements traceability model	54
2.10	Total traceability model	55
2.11	End-to-end traceability model	56
2.12	TraceabilityWeb model	57
2.13	The proposed research area	72
3.1	Traceability theoretical framework	75
3.2	Flowchart of research process	76
3.3	Research procedure and activities	78
3.4	General concepts of precision and recall	88
3.5	OBA interfaces – context diagram	94
4.1	Proposed document-based traceability (DBT) model	101

4.2	Multi linked-list implementation	104
4.3	The architecture of DBT	106
4.4	The process of DBT	108
4.5	DBT use cases	110
4.6	DBT class diagram	112
4.7	DBT packages (static organization)	113
4.8	DBT extractor package	114
4.9	DBT analyzer package	115
4.10	Lucene indexing process	118
4.11	DBT traceability engine package	120
4.12	Multi-bidirectional traceability links between artefacts	120
4.13	Part of SRS document - capturing ID and term	122
4.14	Part of SDD document - capturing SDD ID	123
4.15	Part of SDD document – capturing SDD ID and term	123
4.16	Mapping the source code term with SDD ID – SDD document	124
4.17	Part of STD document – mapping STD ID	125
4.18	Searching for ID in STD for traceability	125
4.19	Report generator package	126
4.20	Queries package	127
4.21	DBT entity relationship diagram (ERD)	129
4.22	STD relation (table) with partial data	130
4.23	SRS relation (table) with partial data	131
4.24	STDxSRS relation (table) with partial data	131
4.25	Links between STD and SRS from view of set	131
4.26	Macro icon for DBT in Microsoft Word	134

4.27	STR (partial) - operate cruise control result	135
4.28	STR (partial) – set car calibration result	136
4.29	The failed result change into red and highlighted (set car calibration)	136
4.30	The failed result change into red and highlighted (cruise control)	137
4.31	Failed summary report	137
4.32	SRS data interface	138
4.33	STD data interface	139
4.34	SRS to STD traceability interface	139
5.1	Mapping of features for evaluation of proposed model (sub-features not shown)	144
5.2	Feature set F1 level of scores (traceability)	149
5.3	F1 percentage of feature set score	149
5.4	Feature set F2 level of scores (document management)	151
5.5	F2 percentage of feature set score	151
5.6	Feature set 3 – level of score (defect management)	152
5.7	F3 percentage of feature set score	153
5.8	Feature set 4 – level of score (report)	154
5.9	F4 percentage of feature set score	154
5.10	F5 feature set score for activity support	156
5.11	F5 percentage of feature set score	156
5.12	Multiple metric graph for feature set score	159
5.13	Multiple metric graph for overall % score using feature set weightings	159
5.14	% feature set score versus models	160
5.15	OBA sub-system component	161

5.16	Auto-cruise control state	162
5.17	Letter from client	163
5.18	Requirements document	163
5.19	OBA traceability hierarchy (bidirectional relationship)	165
5.20	Precision and recall graph for STR vs SRS	170
5.21	Precision and recall graph for SDD vs SRS	172
5.22	DBT leads to time efficiency	177
5.23	DBT easy to use	178

LIST OF ABBREVATIONS

CSCI	-	Computer Software Configuration Item
DAS	-	Driving Assistance System
DBT	-	Document-based Traceability
DoD	-	Department of Defense
ERD	-	Entity Relationship diagram
IEEE	-	Institute of Electrical and Electronics Engineers
IRS	-	Interface Requirements Specification
ISO	-	International Organization for Standardization
OBA	-	On-Board Automobile
SDCP	-	Safe Drive Control Panel
SDD	-	Software Design Document
SDLC	-	Software Development Life Cycle
SDP	-	Software Development Plan
SRS	-	Software Requirements Specification
STD	-	Software Test Description
STR	-	Software Test Result
UML	-	Unified Modeling Language
UTM	-	Universiti Teknologi Malaysia
XML	-	Xtensible Markup Language

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	OBA Data Set (Data Extraction – Traceability)	211
B	OBA Data Set (Data Description)	218
C	OBA Documents	222
D	Expert Validation Questionnaire / Questionnaire Answer Sheet	223
E	List of Publications	239

CHAPTER 1

INTRODUCTION

1.1 Overview

Software testing is a vital phase in software development life cycle preceding the software maintenance. Software testing has increased colossal significance in the present competitive world of innovation, complexity and challenging age of which software is expected to be more efficient and reliable (Kassab *et al*., 2016). Software testing activities are carried out throughout the software development lifecycle (SDLC) that involves several phases towards the end of the test summary (Spillner *et al*., 2014). In general, software testing can be divided into four categories; unit testing, integration testing, system testing and acceptance testing (Jorgensen, 2014). As testing is an important platform to ensure software quality and conformity involving many staff and documentations, it is quite hard to effectively manage these activities at a time (Naik and Tripathy, 2011). Test management is dedicatedly engaged to manage all these activities and to find ways to reduce the complexities.

Test management is the process of organizing testing and validating the software. Effective test management is a vital part of developing high quality software product (Kukreja *et al*., 2015). Through well-planned and well-managed testing processes, the team can ensure that they are producing the high quality software. The team is led by test manager who has the responsibility to manage risk, reviews, assessments and audits. The function of a software test manager is to

effectively and efficiently lead the testing team. To fulfil this role, the leader must comprehend the order of testing and how to successfully execute a testing procedure while satisfying the customary administration part of a manager (Shuja and Krebs, 2007). The role includes quality and test advocacy, asset arranging and administration, and determination of issues that block the test effort.

One of the main challenges in test management is to manage software documentations. Documentation in software engineering is an artefact with the purpose to share the information of which systems it belongs to. Test documentation is identified as the pivotal point as stated by the IEEE829:2010 in order to manage and to report test contents (Sidek, Noraziah and Wahab, 2011). In other perspective, test maturity model takes documentation as an important measure to associate test management to software test process improvement (Van Veenendaal and Cannegieter, 2010). In other words, for test management to remain useful throughout the phases with acceptable maintenance features built-in is to adopt a good documentation model.

One of the activities involved in test management is traceability. Traceability is the ability of linking various artefacts in software development life cycle in forward and backward way (Schwarz, 2012). In the test management, traceability is used to track the bugs back to the corresponding version of requirements. Traceability has been proven to increase the effectiveness or the efficiency of test management.

1.2 Background of the Problem

Nowadays software is becoming more complex. It consists of diverse components with distributed locations, complex algorithms, on varieties of platforms, many sub-contractors with different kind of development methodologies. Complexity brings the fact that no software parts are indistinguishable. A software can be considered as good and high quality if it has a vigorous software testing. Software testing starts as early as software development begins with an enormous testing activities (Parizi *et al.*, 2014). These activities in software testing need to be planned and managed properly; especially the defects or bugs are found during testing. Each of the defects found needs to be traced to the corresponding requirements. This practice is called software traceability.

Traceability is a vital part of software development and maintenance and broadly recognized as a key to quality of the software (Zhang *et al*., 2016). It is used to capture the link between software artefacts. It is required for the development of safety-critical systems such as in domain of an aerospace (ISO12207, DO-178B), railway (EN50128) and etcetera (Bouillon *et al.*, 2013). In addition, several international quality standards recommended traceability such as IEEE 1291, ISO 9000ff, ISO 15504 and SEI, CMM/CMMI (Wiederseiner *et al*., 2011).

Currently there are many researchers working on traceability. This is due to the arise of many problem in the industries (Mustafa and Labiche, 2015). Though traceability is proven to be having great impact on software project, there are still a lot of problems such as it is an error prone and time consuming (Marques *et al*., 2015b), cost-intensive (Maro *et al*., 2016; Regan *et al*., 2012), laborious (Shao *et al*., 2013; Kamalabalan *et al*., 2015), ad-hoc traceability without strategy (Bouillon, Mäder and Philippow, 2013) and difficult (Regan *et al.,* 2012). There are a few researches on traceability regarding to testing artefacts such as unit testing and class (Qusef *et al.*, 2010), test artefacts and code (Wiederseiner *et al*., 2011), test cases

and requirements (Noack *et al*., 2014), bugs and test cases (Kaushik *et al*., 2011), design and test (Lormans and van Deursen, 2009). Although studies have shown an increase in testing traceability, the research focuses on test result, bugs and test cases is still vague (Garousi, Eskandar and Herkiloğlu, 2016). Research has revealed that poor traceability can be an essential contributing factor to software project failure (Parizi *et al*., 2014). Though, notwithstanding the available commercial tools to support traceability, the actual practice of traceability remains poorly documented (Cleland-Huang *et al.*, 2012; Maro *et al*., 2016).

A poorly documented traceability would jeopardize the quality of the software product especially in the critical-safety system. Software engineers depend on system documentation as a guide in comprehension of the practical, architectural design, and the usage of subtle elements of complex applications. Software engineers are compelled to depend exclusively on source code when the documentation does not exist. This is a failure-prone process and a time consuming (Roth *et al*., 2013), particularly when one considers the amount of information adaptation and domain mapping that is required to comprehend the architecture of a multi-function software system. There are various inadequacies in current project documentation methods (de Graaf *et al*., 2016). Since the initial days of software development some of these insufficiencies have existed, for example the absence of consistency between the source code and documentations. Other deficiencies have only recently become apparent as vital issues, such as the intricacy in incorporating existing documentations with newly created artefacts (Herwig, 2014). Numerous studies have demonstrated that documentation regularly experiences the accompanying issues:

(i) Nonexistent or of low quality (Alaranta and Betz, 2012; McBurney, 2015)

(ii) Out-dated (McBurney, 2015; Garousi *et al*., 2013; Satish and Anand, 2016)

(iii) Over abundant and without a definite objectives / incomplete (Parnas, 2011; Dautovic, 2011)

(iv) Difficult to access and manage (for instance when the records are scattered on different computers or in distinctive format: diagrams and text) (Choudhury and Thushara, 2014)

(v) Difficult to trace / Lack of traceability (Satish and Anand, 2016; Plosch, Dautovic and Saft, 2014)

The key point solution to the above problems is not the documentation itself, but how to manage the documentation. One or more types of documentation may be made available at each testing phase. The contents of document may reflect some duplication while others are disintegrated that make it difficult for test manager to access, update and control the visibility of current status of testing (Khan and Mattsson, 2012). Currently many researchers have been working on the software documentation however very few are working on the importance of test documentation as a way forward to support test management (Donald, 2013).

Despite this, test documentation is not given due respect by many testers (Andrade *et al,.* 2013). Test documentation is treated as a time consuming task that not many people would like to get involved with. Some organisations give less attention on documentation with reason being the lack of staffing (Khan and Mattsson, 2012). Worse, the distribution of man power allocated to testing activities is not justified in that it is far less than the allocation assigned to the development activities (Treude, Robillard and Dagenais, 2015). This gives more strong reason to why there is a need to have a special emphasis on the need of test documentations and the way to manage them.

Based on the evidence mention, there are fewer endeavours done to manage document traceability in software testing artefacts. Hence, the need to develop new traceability model that support test management is crucial.

1.3 Statement of the Problem

There is a need to establish integration amongst documentation such that all can be made accessible and easy to manage. Secondly as different organization may adopt different test documentation, it is necessary to make a survey to understand the most relevant information that is practically used and adopted by the industries. Thirdly, the existing software test documentations are difficult to manage. Thus, there is a need to propose a special mechanism or model to manage software testing documentation in integration. The key solution to above problems is to establish an effective traceability model to support software testing documentations.

This research investigates the need for customized software testing documentation and formulates a software traceability model to support documentation in software testing. The main research question is "*How to design and implement an effective software engineering documentation model based on Software Engineering Standards using traceability model to support Test Management?"*

The sub questions of the main research questions are as follows:

(i) **RQ1** : Why the existing software engineering documentation are not fully adopted by test management and why the existing traceability model still not able to manage the link between the artefacts?

(ii) **RQ2** : What is the effective way to help test management in maintaining a software traceability within a software engineering documentation?

(iii) **RQ3** : How to provide traceability links between artefacts that will support test management ?

(iv) **RQ4** : How to evaluate the usability of the proposed model to support test management at some significant degree?

1.4 Objectives of the Study

The research objectives are mentioned based on the problem statement, are as follows:

(i) To study and investigate current issues in software traceability associated to software documentation and test management.

(ii) To formulate a new traceability model that integrates all software engineering artefacts within a repository to support test management.

(iii) To design and develop the prototype of the proposed document-based traceability model.

(iv) To evaluate the effectiveness and the efficiency of the proposed model.

1.5 Scope of the Research

The scope of this study covers the following:

(i) This research focuses on traceability for software testing and its associated components. This will involve the study on system level of software testing (unit, integration, system, and acceptance) but not on types of testing (example – smoke, security, performance, regression, compliance etc.)

(ii) The testing documents will be used are Software Test Description (STD) and Software Test Result (STR). No other testing documents will be used.

(iii) Software engineering documents will be used besides software testing documents are Software Requirements Specifications (SRS), Software Design Document (SDD) and source code.

(iv) This is not a bug tracking system. It just uses documentation to highlight the bugs inside the document.

1.6 Significance of the Study

Requirement traceability has been shown to give numerous advantages to organization that make utilization of traceability methods. This is the reason traceability is an imperative part of numerous standards for software development, such as the CMMI , ISO 9001:2000 and ISO/IEC 15504/SPICE (Gotel *et al*., 2012). Disregarding the advantages that traceability offers to the software engineering industries, its practice confronts numerous difficulties (Kannenberg and Saiedian, 2009; Cleland-Huang *et al.,* 2014). These difficulties can be distinguished under the zones of cost in terms of endeavor and time, the trouble of keeping up traceability through change, tool support, distinctive perspective focuses on traceability by diverse stakeholders, hierarchical issues and legislative issues, and poor documentation.

On the other hand, documentation plays a vital role in software development and maintenance. Typical software system documentation consists of different type of artefacts, ranging from source code, requirements, architecture design, testing and many more. Good software documentation provides multiple views of a system at different abstraction level and using different formats. As the quantity and variety of information about software system develops, so does the requirement for supporting consistency and traceability among distinctive levels of abstraction for engineers (Nair *et al.*, 2013).

A survey conducted by (Bouillon, Mäder and Philippow, 2013; Mustafa and Labiche, 2017) shows that traceability between requirements and others artefacts (especially testing) was rarely maintained in practice. Meanwhile, research conducted by (Regan *et al.*, 2012) indicates the needs of documentation in practice, and the tools and technologies used to maintain, verify and validate such documents.

Clearly, traceability is very important to trace the link between artefacts involved in software development and maintenance of a software system.

1.7 Thesis Outline

This thesis discusses the issues concerning to traceability that relate to testing artefacts which support test management. It highlights the problems and limitation of software documentation, test management, traceability and the similarity link between them. This thesis is organized as follows:

Chapter 2: Discusses in general about software testing, followed by test management and it approaches and issues. This chapter discusses about software documentation and the problems/issues. This chapter highlights the traceability approaches, traceability models and issues. A comparison study was tabulated and identifies the limitations and issues.

Chapter 3: Describes the research methodology in this research. It explains the resign design, procedure and activities which are used in this research. This chapter also discusses on the evaluation method, instrumentation, case study, assumptions and limitation that have been adopted and observed in this research.

Chapter 4: Presents a conceptual of the proposed model. It also describes the detailed component of the proposed model including the architecture and the process. This chapter explains the development of the proposed model in the UML notation.

Chapter 5: Elucidates the evaluation of the proposed model in terms of effectiveness, efficiency and satisfactory. The quantitative and qualitative method is apply; feature analysis, precision and recall, and expert validation. The results are based on customer perception and metric calculation.

Chapter 6: This chapter concludes the research by describing the research achievement and contributions. The last part explains the limitations and suggestions for future works.

REFERENCES

Abualigah, L.M.Q. and Hanandeh, E.S. (2015). Applying Genetic Algorithms To Information Retrieval Using Vector Space Model. *International Journal of Computer Science, Engineering and Applications.* 5**,** 19.

Abubakar, H.I., Hashim, N.L. and Hussain, A. (2015). Verification Process of Usability Evaluation Model for M-banking Application. *Recent Advances in Computer Science Proceedings of the 14th International Conference on Applied Computer and Applied Computational Science (ACACOS '15)*. 23-25 April. Kuala Lumpur: WSEAS, 182–189.

Aizenbud-Reshef, N., Nolan, B. T., Rubin, J. and Shaham-Gafni, Y. (2006). Model traceability. *IBM Systems Journal.* 45, 515–526.

Alaranta, M. and Betz, S. (2012). Knowledge Problems in Corrective Software Maintenance - A Case Study. *2012 IEEE 45th Hawaii International Conference On System Science (HICSS).* 4-7 June. Maui, Hawaii, USA: IEEE, 3746–3755.

Alexander, I. (2002). Towards Automatic Traceability in Industrial Practice. *Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering*. 23-27 September. Edinburgh, Scotland: ACM, 26–31.

Aljahdali, S., Hussain, S. N., Hundewale, N. and Poyil, A. T. (2012). Test Management and Control. *2012 IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS)*. 22-24 June. Beijing, China: IEEE, 429–432.

Anand, T. and Mani, V.S. (2015). Practices to Make Agile Test Teams Effective: Challenges and Solutions. *2015 IEEE 10th International Conference on Global Software Engineering Workshops*. 13-16 July. Ciudad Real, Spain: IEEE, 7–11.

Andrade, J., Ares, J., Martínez, M.-A., Pazos, J., Rodríguez, S., Romera, J. and Suárez, S. (2013). An Architectural Model for Software Testing Lesson Learned Systems. *Information and Software Technology.* 55, 18–34.

Anquetil, N., Sousa, A., Kulesza, U., Rummler, A., Mitschke, R., Moreira, A. and Araújo, J. (2010). A Model-driven Traceability Framework for Software Product Lines. *Software & Systems Modeling.* 9, 427–451.

Antoniol, G., Canfora, G., Casazza, G. and De Lucia, A. (2000). Information Retrieval Models for Recovering Traceability Links Between Code and Documentation. *Proceedings 2000 International Conference on Software Maintenance (ICSM '00)*. 11-14 October. San Jose, CA, USA: IEEE, 40–49.

Antoniol, G., Canfora, G., Casazza, G., De Lucia, A. and Merlo, E. (2002). Recovering Traceability Links Between Code and Documentation. *IEEE Transactions on Software Engineering*. 28(10), 970–983.

Asuncion, H. and Taylor, R.N. (2007). Establishing the Connection Between Software Traceability and Data Provenance. *Institute for Software Research, University of California, Irvine, Technical Report UCI-ISR-07-9*.

Asuncion, H.U., Asuncion, A.U. and Taylor, R.N. (2010). Software Traceability with Topic Modeling. *2010 ACM/IEEE 32nd International Conference on Software Engineering.* 2-8 May. Cape Town, South Africa: ACM/IEEE, 95–104.

Asuncion, H.U., François, F. and Taylor, R.N. (2007). An End-to-end Industrial Software Traceability Tool. *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. 3-7 September. Dubrovnik, Croatia: ACM, 115–124.

Asuncion, H.U. and Taylor, R.N. (2012). Automated Techniques for Capturing Custom Traceability Links Across Heterogeneous Artifacts. In Cleland-Huang. J, Gotel. O and Zisman. A (Ed.) *Software and Systems Traceability*. (pp. 129–146). London: Springer-Verlag.

Azram, N.A. and Atan, R. (2012). Traceability Method for Software Engineering Documentation. *International Journal of Computer Science Issues(IJCSI)*. 9, 216-220.

Baharuddin, R., Singh, D. and Razali, R. (2013). Usability Dimensions for Mobile Applications - A Review. *Research Journal of Applied Sciences, Engineering and Technology.* 5(6), 2225-2231.

Bajracharya, S., Ossher, J. and Lopes, C. (2014). Sourcerer: An Infrastructure for Large-Scale Collection and Analysis of Open-source Code. *Science of Computer Programming.* **7**9, 241–259.

Balaji, S. and Murugaiyan, M.S. (2012). Waterfall vs. V-Model vs. Agile: A Comparative Study on SDLC. *International Journal of Information Technology and Business Management.* 2, 26–30.

Bavota, G., De Lucia, A., Oliveto, R. and Tortora, G. (2014). Enhancing Software Artefact Traceability Recovery Processes with Link Count Information. *Information and Software Technology* **5**6(2), 163–182.

Bavota, G., Colangelo, L., De Lucia, A., Fusco, S., Oliveto, R. and Panichella, A. (2012). TraceME: Traceability Management in Eclipse. *International Conference on Software Maintenance*. 23-28 September. Riva del Garda, Italy: IEEE, 642–645.

Beecham, S., Hall, T., Britton, C., Cottee, M. and Rainer, A. (2005). Using an Expert Panel to Validate a Requirements Process Improvement Model. *Journal of Systems and Software.* **7**6, 251–275.

Bertolino, A. (2003). Software Testing Research and Practice. In Börger E., Gargantini A. and Riccobene E. (Ed.) *Abstract State Machines 2003 - Lecture Notes in Computer Science, vol 2589* (pp. 1-21). Berlin: Springer-Valag Heidelberg.

Bevan, N., Carter, J. and Harker, S. (2015). ISO 9241-11 Revised: What Have We Learnt About Usability Since 1998? *The 17th International Conference on Human-Computer Interaction*. 2-7 August. Los Angeles, CA, USA: Springer, 143–151.

Blaauboer, F., Sikkel, K. and Aydin, M.N. (2007). Deciding to Adopt Requirements Traceability in Practice. In Krogstie J., Opdahl A. and Sindre G. (Ed.) *Advanced Information Systems Engineering. CAiSE 2007. Lecture Notes in Computer Science, vol 4495* (pp. 294-308) Heidelberg Berlin: Springer-Verlag.

Black, R. and Jamie, L.M. (2014). *Advanced Software Testing-Vol. 2: Guide to the ISTQB Advanced Certification as an Advanced Test Manager*. Vol. 2. Santa Barbara, California: Rocky Nook, Inc.

Borg, M., Gotel, O.C. and Wnuk, K. (2013). Enabling Traceability Reuse for Impact Analyses: A Feasibility Study in a Safety Context. *2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*. 19 May. San Francisco, California, USA: IEEE, 72–78.

Bouillon, E., Mäder, P. and Philippow, I. (2013). A Survey on Usage Scenarios for Requirements Traceability in Practice. In Doerr J. and Opdahl A.L. (Ed.) *Requirements Engineering: Foundation for Software Quality. REFSQ 2013. Lecture Notes in Computer Science, vol. 7830* (pp. 158-173) Heidelberg Berlin: Springer-Verlag.

Buckley, J., Mens, T., Zenger, M., Rashid, A. and Kniesel, G. (2005). Towards a Taxonomy of Software Change. *Journal of Software Maintenance and Evolution.* 1**7,** 309–332.

Captain, F.A. (2013). *Six-Step Relational Database Design^{TM}: A Step by Step Approach to Relational Database Design and Development(2nd ed)*. CreateSpace Independent Publishing Platform.

Celebic, B., Breu, R. and Felderer, M. (2016). Traceability Types for Mastering Change in Collaborative Software Quality Management. In Steffen B. (Ed.) *Transactions on Foundations for Mastering Change I. Lecture Notes in Computer Science, vol 9960* (pp. 242-256) Cham, Switzerland: Springer.

Chang, C.K. and Christensen, M. (2003). Event-based Traceability for Managing Evolutionary Change. *IEEE Transactions on Software Engineering.* 29, 796–810.

Chen, X., Hosking, J. and Grundy, J. (2012). Visualizing Traceability Links Between Source Code and Documentation. *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* 30 September - 4 October. Innsbruck, Austria: IEEE, 119–126.

Chomal, V.S. and Saini, J.R. (2014). Significance of Software Documentation in Software Development Process. *International Journal of Engineering Innovation and Research.* 3, 410-416.

Chomal, V.S. and Saini, J.R. (2015). Software Template for Evaluating and Scoring Software Project Documentations. *International Journal of Computer Applications,* 116, 15-27.

Choudhury, J. and Thushara, B. (2014). Software Documentation in a Globally Distributed Environment. *2014 IEEE 9th International Conference on Global Software Engineering*. 18-21 August. Shanghai, China: IEEE, 90–94.

Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E. and Christina, S (2005). Goal-centric traceability for managing non-functional requirements. *Proceedings of the 27th International Conference on Software Engineering,* 15-21 May. St. Louis, MO, USA: ACM, 362–371.

Cleland-Huang, J., Gotel, O. and Zisman, A (2012). *Software and Systems Traceability*. London: Springer.

Cleland-Huang, J., Gotel, O. C. Z., Huffman Hayes, J., Mäder, P. and Zisman, A. (2014). Software Traceability: Trends and Future Directions. *Proceedings of the on Future of Software Engineering (FOSE2014)*. 31 May - 7 June. Hyderabad, India: ACM, 55–69.

Cleland-Huang, J. (2005). Toward improved traceability of non-functional requirements. *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering TEFSE '05*. 8 November. Long Beach, California: ACM, 14–19.

Cleland-Huang, J., Settimi, R., Chuan Duan, *et al.* (2005). Utilizing supporting evidence to improve dynamic requirements traceability. *13th IEEE International Conference On Requirements Engineering, 2005. Proceedings.* 29 August - 2 September. Paris, France: IEEE, 135–144.

Coronel, C. and Morris, S. (2014). *Database Systems: Design, Implementation, & Management* (11th ed). United States: Cengage Learning.

Dautovic, A. (2011). Automatic Assessment of Software Documentation Quality. *2011 26th IEEE/ACM International Conference on Automated Software Engineering*. 6-10 November. Lawrence, Kansas, USA: ACM/IEEE Computer Society, 665–669.

Davis, A.M. (1990). *Software Requirements: Analysis and Specification*. Upper Saddle River, NJ, USA: Prentice Hall Press.

Delacroix, L. (2014). *Longman Dictionary of Contemporary English*. London, United Kingdom: Pearson Longman.

Dennis, A., Wixom, B.H. and Roberta, M.R. (2015). *Systems Analysis and Design: An Object-Oriented Approach with UML,* (5th ed) New Jersey: John Wiley & Sons.

Devarajan, K., Wang, G. and Ebrahimi, N. (2015). A Unified Statistical Approach to Non-Negative Matrix Factorization and Probabilistic Latent Semantic Indexing. *Machine learning.* 99, 137–163.

Diesendruck, L., Kooper, R., Marini, L. and McHenry, K. (2014). Using Lucene to Index and Search the Digitized 1940 us Census. *Concurrency and Computation: Practice and Experience.* 26, 2167–2177.

Donald, F. (2013). *Common Testing Problems: Pitfalls to Prevent and Mitigate*. Pittsburgh, Pennsylvania USA: Software Engineering Institute, Carnegie Mellon University.

Doraisamy, M., Ibrahim, S. and Mahrin, M.N. (2015). Metric Based Software Project Performance Monitoring Model. *2015 IEEE Confernece on Open Systems (ICOS)*. 24-26 August. Malacca, Malaysia: IEEE, 12–17.

Dubey, S.K., Gulati, A. and Rana, A. (2012). Integrated Model for Software Usability. *International Journal on Computer Science and Engineering.* 4, 429–437.

Egyed, A. (2003). A Scenario-driven Approach to Trace Dependency Analysis. *IEEE Transactions on Software Engineering.* 116–132.

Egyed, A. (2001). A Scenario-Driven Approach to Traceability. *Proceedings of the 23rd international conference on Software engineering,* 12-19 May. Toronto, Canada: ACM, 123–132.

Egyed, A., Graf, F. and Grünbacher, P. (2010). Effort and Quality of Recovering Requirements-to-Code Traces: Two Exploratory Experiments. *2010 18th IEEE International Requirements Engineering Conference (RE)*. 27 September - 1 October. Sydney, Australia: IEEE, 221–230.

Egyed, A. and Grünbacher, P. (2002). Automating Requirements Traceability: Beyond the Record & Replay Paradigm. *Proceedings of the 17th IEEE International Conference on Automated Software Engineering*. 23-27 September. Edinburgh, Scotland: ACM/IEEE, 163-171.

Eldh, S., Brandt, J., Street, M., Hansson, H. and Punnekkat, S. (2010). Towards Fully Automated Test Management for Large Complex Systems. *2010 Third International Conference on Software Testing, Verification and Validation*. 6-10 April. Paris, France: IEEE, 412–420.

Fernández-Sáez, A. M., Caivano, D., Genero, M. and Chaudron, M. R. (2015). On the Use of UML Documentation in Software Maintenance: Results from a

Survey in Industry. *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS).* 30 September - 2 October. Ottawa, ON, Canada: IEEE, 292–301.

Fleischer, D., Schwerdtfeger, J., Capaul, P. and Thiel, V. (2012). Evaluation of Open Source Tools for Test Management and Test Automation. *Seminararbeit DHBW Stuttgart*.

Flint, S. (2009). A Conceptual Model of Software Engineering Research Approaches. *Proceedings of the 2009 Australian Software Engineering Conference,* 14-17 April. Gold Coast, Australia: IEEE, 229–236.

Galvao, I. and Goknil, A. (2007). Survey of Traceability Approaches in Model-Driven Engineering. *Enterprise Distributed Object Computing Conference, EDOC'07.* 15-19 October 2007. Annapolis, USA: IEEE, 313–326.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J (1995). *Design Patterns: Elements of Reusable Object-Oriented*. India: Pearson Education

Gao, A. and Liu, Z. (2014). TAI: A Workflow-based Automated Testing Management System. *2014 5th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 27-29 June. Beijing, China: IEEE, 208–211.

Gao, L. (2011). Research on Implementation of Software Test Management. *2011 3rd International Conference on Computer Research and Development (ICCRD)*. 11-13 March. Shanghai, China: IEEE, 234–237.

Garcia, J.E. and Paiva, A.C. (2016). A Requirements-to-Implementation Mapping Tool for Requirements Traceability. *Journal of Software.* 11, 193–200.

Garousi, G., Garousi, V., Moussavi, M., Ruhe, G. and Smith, B. (2013). Evaluating Usage and Quality of Technical Software Documentation: An Empirical Study. *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*. New York, USA: ACM, 24–35.

Garousi, G., Garousi-Yusifoğlu, V., Ruhe, G., Zhi, J., Moussavi, M. and Smith, B. (2015). Usage and Usefulness of Technical Software Documentation: An Industrial Case Study. *Information and Software Technology.* 57, 664–682.

Garousi, V., Eskandar, M.M. and Herkiloğlu, K. (2016). Industry–academia Collaborations in Software Testing: Experience and Success Stories from Canada and Turkey. *Software Quality Journal.* 1–53.

Garousi, V. and Zhi, J. (2013). A Survey of Software Testing Practices in Canada. *Journal of Systems and Software.* 86, 1354–1376.

Geraci, A. (1991). *IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries*. Piscataway, NJ, USA: IEEE Press.

Goodrich, M.T., Tamassia, R. and Goldwasser, M.H. (2014). *Data Structures and Algorithms in Java* (6th ed). Hoboken, NJ: Wiley.

Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P. and Antoniol, G. (2012). The Quest for Ubiquity: A Roadmap for Software and Systems Traceability Research. *2012 20th IEEE International Requirements Engineering Conference (RE).* 24-28 September. Chicago, Illinois, USA: IEEE, 71–80.

Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P. and Mäder, P. (2012). Traceability fundamentals. In Cleland-Huang J., Gotel O. and Zisman A. (Ed.) *Software and Systems Traceability*. (pp. 3-22) London, United Kingdom: Springer-Verlag.

Gotel, O. and Finkelstein, A. (1994). An Analysis of the Requirements Traceability Problem. *Proceedings of the First International Conference on Requirements Engineering*. 18-22 April. Colorado Springs, CO, USA: IEEE, 94–101.

de Graaf, K. A., Liang, P., Tang, A. and Van Vliet, H. (2016). How Organization of Architecture Documentation Affects Architectural Knowledge Retrieval. *Science of Computer Programming.* 121, 75–99.

Graham, D. (2002). Requirements and Testing: Seven Missing-link Myths. *IEEE Software.* 19, 15–17.

Grimán, A., Pérez, M., Mendoza, L. and Losavio, F. ((2006). Feature Analysis for Architectural Evaluation Methods. *Journal of Systems and Software.* **7**9, 871–888.

Grossman, D.A. and Frieder, O. (2012). *Information Retrieval: Algorithms and Heuristics*. New York: Springer Science + Business Media, LLC.

Guo, J., Monaikul, N., Plepel, C. and Cleland-Huang, J. (2014). Towards an Intelligent domain-specific Traceability Solution. *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*. 15-19 September. Vasteras, Sweden: ACM, 755–766.

Hammad, M., Collard, M.L. and Maletic, J.I. (2011). Automatically Identifying Changes That Impact Code-to-Design Traceability During Evolution. *Software Quality Journal.* 19, 35–64.

Hassnain, M. (2015). A Comparative Study on Traceability Approaches in Software Development Life Cycle. *International Journal of Information Technology and Electrical Engineering.* 4, 1-4.

Hayes, J.H., Dekhtyar, A. and Osborne, J. (2003). Improving Requirements Tracing Via Information Retrieval. *Proceedings of the 11th IEEE International Conference on Requirements Engineering*. 8-12 September 2003. Monterey Bay, USA: IEEE, 138-147.

Hedberg, H. and Lappalainen, J. (2005). A Preliminary Evaluation of Software Inspection Tools, with the DESMET Method. *Fifth International Conference On Quality Software (QSIC 2005).* 19-20 September. Melbourne, Australia: IEEE, 45–52.

Herwig, V. (2014). Documentation of Software System. *Jounal of Electrotechnic and Computer Systems.* 13, 240–246.

Ibrahim, S., Idris, N. B., Munro. M. and Deraman, A. (2005). Implementing a Document-based Requirements Traceability: A Case Study. *IASTED International Conference on Software Engineering*. 15-17 February 2005. Innsbruck, Austria: Octa Press, 124–131.

Illes, T., Herrmann, A., Paech, B. and Rückert, J. (2005). Criteria for Software Testing Tool Evaluation. A Task Oriented View. *Proceedings of the 3rd World Congress for Software Quality*. 20-30 September. Munich, Germany: ISQI, 213–222.

ISO - International Organization for Standardization (2010). *ISO 9241-210:2010 - Ergonomics of Human-System Interaction - Part 210: Human-Centred Design for Interactive Systems.* Geneva, Switzerland: ISO.

Jaber, K., Sharif, B. and Liu, C. (2013). A Study on the Effect of Traceability Links in Software Maintenance. *IEEE Access.* 1, 726–741.

Järvelin, K. and Kekäläinen, J. (2000). IR Evaluation Methods for Retrieving Highly Relevant Documents. *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* . 24-28 July. Athens, Greece: ACM, 41–48.

Javed, M.A. and Zdun, U. (2014). The Supportive Effect of Traceability Links in Architecture-Level Software Understanding: Two Controlled Experiments. *2014 IEEE/IFIP Conference on Software Architecture (WICSA)*. 7-11 April. Sydney, Australia: IEEE, 215–224.

Jorgensen, P.C. (2014). *Software Testing: A Craftsman's Approach - Fourth Edition* (4th ed.) Boca Raton Florida: CRC Press.

Kamalabalan, K., Uruththirakodeeswaran, T., Thiyagalingam, G., Wijesinghe, D. B., Perera, I., Meedeniya, D. and Balasubramaniam, D.. (2015). Tool Support for Traceability of Software Artefacts. *Moratuwa Engineering Research Conference (MERCon).* 7-8 April. Moratuwa, Sri Lanka: IEEE, 318-323.

Kannenberg, A. and Saiedian, H. (2009). Why Software Requirements Traceability Remains a Challenge. *CrossTalk - The Journal of Defense Software Engineering*. 22(5), 14-19.

Kassab, M., DeFranco, J. and Laplante, P. (2016). Software Testing Practices in Industry: The State of the Practice. *IEEE Software.* PP, 1-10.

Kaushik, N., Tahvildari, L. and Moore, M. (2011). Reconstructing Traceability Between Bugs and Test Cases: An Experimental Study. *2011 18th Working Conference On Reverse Engineering.* 17-20 October. Lero, Limerick, Ireland: IEEE, 411–414.

Keenan, E., Czauderna, A., Leach, G., Cleland-Huang, J., Shin, Y. and Moritz, E. (2012). Tracelab: An Experimental Workbench for Equipping Researchers to Innovate, Synthesize, and Comparatively Evaluate Traceability Solutions. *Proceedings of the 34th International Conference on Software Engineering*. 2-9 June. Zurich, Switzerland: IEEE, 1375–1378.

Khan, A.S. and Mattsson, M.K. (2012). Management of documentation and maintainability in the context of software handover. *2012 8th International Conference On Computing Technology and Information Management (ICCM).* 24-26 April. Seoul, South Korea: IEEE, 238–243.

Kim, M. W., Kim, W. Y., Son, H. S. and Kim, R. Y. C. (2011). A Test Management System for Operational Validation. *International Conference on Advanced Software Engineering and Its Applications*. 8-10 December. Jeju Island, Korea: Springer, 305–313.

Kirova, V., Kirby, N., Kothari, D. and Childress, G (2008). Effective Requirements Traceability: Models, Tools, and Practices. *Bell Labs Technical Journal.* 12, 143–158.

Kitchenham, B., Linkman, S. and Law, D. (1997). DESMET: A Methodology for Evaluating Software Engineering Methods and Tools. *Computing & Control Engineering Journal.* 8, 120–126.

Kornecki, A.J. and Zalewski, J. (2005). Experimental Evaluation of Software Development Tools for Safety-Critical Real-Time Systems. *Innovations in Systems and Software Engineering.* 1, 176–188.

Kotonya, G. and Sommerville, I. (1998). *Requirements Engineering*: *Processes and Techniques.* Chichester, England: John Wiley & Sons.

Kritzinger, P.S. and Krüger, H. (2008). Software Traceability Using Latent Semantic Analysis and Relevance Feedback. *Technical Report CS08-01-00, Department of Computer Science, University of Cape Town.,* 391–402.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction*. Bostan, MA, USA: Addison-Wesley.

Kukreja, S., Singhal, A. and Bansal, A. (2015). A Critical Survey on Test Management in IT Projects. *2015 International Conference on Computing, Communication Automation (ICCCA)*. 15-16 May. Greater Noida, India: IEEE, 791–796.

Lázaro, M. and Marcos, E. (2005). Research in Software Engineering: Paradigms and Methods. *Proceedings of the CAiSE'05 Workshops, Vol. 2.* 13-15 June. Porto, Portugal: Springer, 517–522.

Leino, V. (2001). *Documenting Requirements Traceability Information: A Case Study*. Masters. Helsinki University of Technology.

Li, F.-S., Ma, W.-M. and Chao, A. (2008). Architecture Centric Approach to Enhance Software Testing Management. *2008 Eighth International Conference On Intelligent Systems Design and Applications, ISDA '0.,* 26-28 November. Kaohsiung, Taiwan: IEEE, 654–659.

Lodhi, A., Wind, S. and Turowski, K. (2013). Test Management Framework for Managing IT Projects in Industry. *2013 IEEE 10th International Conference on E-Business Engineering (ICEBE)*. 11-13 September. Coventry, United Kingdom: IEEE, 509–514.

Lohar, S., Amornborvornwong, S., Zisman, A. and Cleland-Huang, J. (2013). Improving Trace Accuracy Through Data-Driven Configuration and Composition of Tracing Features. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. 18-26 August. Saint Petersburg, Russian Federation: ACM, 378–388.

Lormans, M. and van Deursen, A. (2009). Reconstructing Requirements Traceability in Design and Test Using Latent Semantic Indexing. *Journal Software Maintenance Evol Res Pract - TUD-SERG-2007-007.* 1-32.

Louridas, P. (2011). Test Management. *IEEE Software.* 28, 86–91.

Madan, A. and Dubey, S.K. (2012). Usability Evaluation Methods: A literature Review. *International Journal of Engineering Science and Technology.* 1, 590–599.

Mäder, P., Philippow, I. and Riebisch, M. (2007). Customizing Traceability Links for the Unified Process. In Overhage S. *et al.* (Ed.) *Software Architectures, Components, and Applications. QoSA 2007. Lecture Notes in Computer Science, vol 4880* (pp. 53-71). Heidelberg, Berlin: Springer.

Maletic, J. I., Munson, E. V., Marcus, A. and Nguyen, T. N. (2003). Using A Hypertext Model for Traceability Link Conformance Analysis. *Proceeding of The 2nd International Workshop on Traceability in Emerging Forms of Software Engineering.* October. Montreal, Canada: ACM, 47-54.

Malz, C. and Jazdi, N. (2010). Agent-based Test Management for Software System Test. *2010 IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*. 28-10 May. Cluj-Napoca, Romania: IEEE, 1–6.

Marcus, A. and Maletic, J.I. (2003). Recovering Documentation-to-source-code Traceability Links Using Latent Semantic Indexing. *Proceedings of the 25th International Conference on Software Engineering*, 3-10 May. Portland, OR, USA: IEEE, 125–135.

Marcus, A., Xie, X. and Poshyvanyk, D. (2005). When and How to Visualize Traceability Links? *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering*, 8 November. Long Beach, CA, USA: ACM, 56-61.

Maro, S., Anjorin, A., Wohlrab, R. and Steghöfer, J. P. (2016). Traceability Maintenance: Factors and Guidelines. *Proceedings of the 31st IEEE/ACM*

International Conference on Automated Software Engineering. 3-7 September. Singapore: ACM, 414–425.

Marques, A., Ramalho, F. and Andrade, W.L. (2015a). Towards A Requirements Traceability Process Centered on the Traceability Model. *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 13-17 April. Salamanca, Spain: ACM, 1364–1369.

Marques, A., Ramalho, F. and Andrade, W.L. (2015b). TRL: A Traceability Representation Language. *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 13-17 April. Salamanca, Spain: ACM, 1358–1363.

Marshall, C., Brereton, P. and Kitchenham, B. (2015). Tools to Support Systematic Reviews in Software Engineering: A Cross-domain Survey Using Semi-structured Interviews. *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. 27-29 April. Nanjing, China: ACM, 26:1-26:6.

Marshall, C., Brereton, P. and Kitchenham, B. (2014). Tools to Support Systematic Reviews in Software Engineering: A Feature Analysis. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. 13-14 May. London, United Kingdom: ACM, 13:1-13:10.

Mazni, O., Sharifah-Lailee, S.-A. and Azman, Y. (2010). Agile Documents: Toward Successful Creation of Effective Documentation. *International Conference on Agile Software Development*. 1-4 June. Trondheim, Norway: Springer, 196–201.

McBurney, P.W. (2015). Automatic Documentation Generation via Source Code Summarization. *Proceedings of the 37th International Conference on Software Engineering-Volume 2*. 16-24 May. Florence/Firenze, Italy: ACM/IEEE 903–906.

McIntosh, C. (2013). *Cambridge Advanced Learner's Dictionary*. New York, USA: Cambridge University Press.

Meneely, A., Smith, B. and Williams, L. (2013). Validating Software Metrics: A Spectrum of Philosophies. *ACM Transactions on Software Engineering Methodology.* 21, 24:1–24:28.

Mustafa, N. and Labiche, Y. (2015). Modeling Traceability for Heterogeneous Systems. *10th International Joint Conference on Software Technologies (ICSOFT).* 20-22 July. Colmar, France: IEEE. 358–366.

Mustafa, N. and Labiche, Y. (2017). The Need for Traceability in Heterogeneous Systems: A Systematic Literature Review. *Computer Software and Applications Conference (COMPSAC),* 4-8 July. Turin, Italy : IEEE, 305–310.

Nagano, S., Ichikawa, Y. and Kobayashi, T. (2012). Recovering Traceability Links Between Code and Documentation for Enterprise Project Artifacts. *36th Annual Computer Software and Applications Conference*. 16-20 July. Izmir, Turkey: IEEE, 11–18.

Naik, K. and Tripathy, P. (2011). *Software Testing and Quality Assurance: Theory and Practice*. New Jersey: John Wiley & Sons.

Nair, S., de la Vara, J.L. and Sen, S. (2013). A Review of Traceability Research at the Requirements Engineering Conference RE@ 21. *21st IEEE International Requirements Engineering Conference*. 15-19 July. Rio de Janeiro, Brazil: IEEE, 222–229.

Namdar, S. and Mirakhorli, M. (2015). Toward Actionable Software Architecture Traceability. *Proceedings of the 8th International Symposium on Software and Systems Traceability*. 16-24 May. Florence/Firenze, Italy: IEEE/ACM, 36–42.

Narmanli, M. (2010). *A Business Rule Approach to Requirements Traceability*. Masters. Middle East Technical University.

Ng, E. and Mohan, V. (2015). *Lucene 4 Cookbook*. Birmingham, United Kingdom: Packt Publishing Ltd.

Noack, T., Karbe, T. and Helke, S. (2014). Reuse-Based Test Traceability: Automatic Linking of Test Cases and Requirements. *International Journal on Advances in Software.* **7**, 469–485.

Norman, G. (2010). Likert Scales, Levels of Measurement and the "laws" of Statistics. *Advances in health sciences education.* 15, 625–632.

Oliveto, R., Antoniol, G., Marcus, A. and Hayes, J. (2007). Software Artefact Traceability: The Never-Ending Challenge. *International Conference On Software Maintenance ICSM 2007. IEEE*. 2-5 October. Paris, France: IEEE, 485–488.

Omar, S.F. (2013). *A Software Traceability Approach To Support Requirement Based Test Coverage Analysis*. Masters. Universiti Teknologi Malaysia.

Papakitsos, E.C. (2014). An application of Cognitive Ergonomics to the Quality Assurance of Software Documentation. *International Journal of Academic Research in Computer Sciences and Electrical Engineering.* 1, 16–30.

Parizi, R.M., Lee, S.P. and Dabbagh, M. (2014). Achievements and Challenges in State-of-the-Art Software Traceability Between Test and Code Artifacts. *IEEE Transactions on Reliability.* 63, 913–926.

Parnas, D.L. (2011). Precise documentation: The key to better software. In Nanz S. (Ed.) *The Future of Software Engineering* (pp. 125-148). Berlin, Heidelberg: Springer, 125–148.

Perera, I., Meedeniya, D. and Bandara, M. (2015). A Traceability Management Framework for Artefacts in Self-adaptive Systems. *The 10th International Conference on Industrial and Information Systems (ICIIS)*. 18-20 December. Peradeniya, Sri Lanka: IEEE, 37–42.

Pinheiro, F.A. (2004). Requirements traceability. In do Prado Leite J.C.S. and Doorn J.H. (Ed.) *Perspectives on Software Requirements*. Boston, MA: Springer, 91–113.

Pinkster, I., Burgt, B. V. D., Janssen, D. and Veenendall, E. V. (2006). *Successful Test Management: An Integral Approach*. Netherlands: Springer.

Plosch, R., Dautovic, A. and Saft, M. (2014). The Value of Software Documentation Quality. *14th International Conference on Quality Software (QSIC)*. 4-5 August. Boston, USA: IEEE, 333–342.

Pohjoisvirta, L. (2013). *Choosing a Tool for Improved Software Test Management*. Masters. Tampere University of Technology.

Poston, R., Patel, J. and Dhaliwal, J.S. (2012). A Software Testing Assessment to Manage Project Testability. *20th European Conference on Information Systems (ECIS 2012)*. 11-13 June. Barcelona, Spain: ESADE, 1-12.

Qusef, A., Bavota, G., Oliveto, R., Lucia, A. D. and Binkley, D. (2013). Evaluating Test-to-code Traceability Recovery Methods Through Controlled Experiments. *Journal of Software: Evolution and Process.* 25, 1167–1191.

Qusef, A., Oliveto, R. and De Lucia, A. (2010). Recovering Traceability Links Between Unit Tests and Classes Under Test: An Improved Method. *International Conference On Software Maintenance (ICSM),* 12-18 September. Timisoara, Romania: IEEE, 1-10.

Raja, U.A. and Kamran, K. (2008). *Framework for Requirements Traceability*. Masters, Blekinge Institute of Technology, Sweden.

Regan, G., McCaffery, F., McDaid, K. and Flood, D. (2012). The Barriers to Traceability and Their Potential Solutions: Towards a Reference Framework. *38th EUROMICRO Conference On Software Engineering and Advanced Applications (SEAA).* 5-8 September. Izmir, Turkey: IEEE, 319–322.

Regan, G., Flood, D. and Mc Caffery, F. (2015). The Development and Validation of a Roadmap for Traceability. *International Conference on Software Process Improvement and Capability Determination*. 16-17 June. Gothenburg, Sweden: Springer, 45–57.

Rochimah, S., Wan Kadir, W.M. and Abdullah, A.H. (2011). Utilizing Multifaceted Requirement Traceability Approach: A Case Study. *International Journal of Software Engineering and Knowledge Engineering.* 21, 571–603.

Rongfa, T. (2012). Adaptive Software Test Management System Based on Software Agents. In Yanwen Wu (Ed.) *Advanced Technology in Teaching-Proceedings of the 2009 3rd International Conference on Teaching and Computational Science (WTCS 2009)* (pp. 1-9). Berlin: Springer.

Roth, S., Hauder, M., Farwick, M., Breu, R. and Matthes, F. (2013). Enterprise Architecture Documentation: Current Practices and Future Directions. *11th International Conference on Wirtschaftsinformatik*. 27th February - 1st March. Leipzig, Germany: Wirtschaftsinformatik Proceedings, 911-925.

Salem, A.M. (2006). Improving Software Quality Through Requirements Traceability Models. *International Conference On Computer Systems and Applications, 2006.* 8-11 March. Sharjah, United Arab Emirates: IEEE, 1159–1162.

Satish, C.J. and Anand, M. (2016). Software Documentation Management Issues and Practices: A Survey. *Indian Journal of Science and Technology.* 9(20), 1-7.

Schwarz, H. (2012) *Universal Traceability. A Comprehensive, Generic, Technology-Independent, and Semantically Rich Approach*. Berlin: Verlag Logos.

Schwarz, H., Ebert, J. and Winter (2009) A. Graph-based Traceability: A Comprehensive Approach. *Software and Systems Modeling, Springer*. 9(4), 473–492.

Shahid, M. (2014) *A Traceability Approach for Hybrid Coverage Analysis to Support Software Maintenance*. Doctor Philosophy. Universiti Teknologi Malaysia, Skudai.

Shahid, M. and Ibrahim, S. (2016). Change Impact Analysis with a Software Traceability Approach to Support Software Maintenance. *13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. 12-16 January. Islamabad, Pakistan: IEEE, 391–396.

Shao, J., Wu, W. and Geng, P. (2013). An Improved Approach to the Recovery of Traceability Links between Requirement Documents and Source Codes Based on Latent Semantic Indexing. *International Conference on Computational Science and Its Applications*. 24-27 June. Ho Chi Minh City, Vietnam: Springer, 547–557.

Sharma, H., Gupta, D. and Singh, R. (2015). Ranking Based Software Quality Assessment Using Experts Opinion. *International Conference On Computational Intelligence and Communication Networks (CICN)*. 12-14 December. Jabalpur, India: IEEE, 1436–1441.

Shuja, A.K. and Krebs, J. (2007). *IBM Rational Unified Process Reference and Certification Guide: Solution Designer (RUP)*. Indianapolis, Indiana, United States: IBM Press.

Sidek, R.M., Noraziah, A. and Wahab, M.H.A. (2011). The Preferable Test Documentation Using IEEE 829. *International Conference on Software Engineering and Computer Systems.* 27-29 June. Kuantan, Pahang, Malaysia: Springer, 109–118.

Siebra, C.A. and Lino, N.Q. (2014). Integration of Autonomic Mechanisms to a Test Management Solution. *9th International Conference On Software Engineering and Applications (ICSOFT-EA), 2014*. 29-31 August. Vienna, Austria: IEEE, 269–276.

Simon, D. and Simon, F. (2012). Integrating Test and Risk Management. *The 8th International Conference on Quality of Information and Communications Technology (QUATIC)*. 3-6 September. Lisbon, Portugal: IEEE, 97–102.

Sousa, A. L. (2008). *Traceability Support in Software Product Lines*. Bachelor Degree, Universidade Nova de Lisboa, Portugal.

Spanoudakis, G., Zisman, A., Pérez-Miñana, E. and Krause, P. (2004). Rule-based Generation of Requirements Traceability Relations. *Journal of Systems and Software.* 72, 105–127.

Spillner, A., Linz, T. and Schaefer, H. (2014). *Software Testing Foundations: A Study Guide for the Certified Tester Exam*. (4th ed.) Santa Barbara, California: Rocky Nook, Inc.

Spillner, A., Linz, T. and Schaefer, H. (2011). *Software Testing Foundations: A Study Guide for the Certified Tester Exam*. (3rd ed.) Santa Barbara, CA 93101, USA: Rocky Nook.

Stettina, C.J. and Kroon, E. (2013). Is There an Agile Handover? An Empirical Study of Documentation and Project Handover Practices Across Agile Software Teams. *International Conference On Engineering, Technology and Innovation (ICE) & IEEE International Technology Management Conference,* 24-26 June. Hague, Netherlands: IEEE, 1–12.

Tam, C. and Oliveira, T. (2016). Understanding the Impact of M-banking on Individual Performance: DeLone & McLean and TTF perspective. *Computers in Human Behavior.* 61, 233–244.

Thitisathienkul, P. and Prompoon, N. (2014). Quality Assessment Method for Software Development Process Document Based on Software Document Characteristics Metric. *9th International Conference On Digital Information Management (ICDIM),* 29 September - 1 October. Bangkok, Thailand: IEEE, 182–188.

Thomas, C. and Carolyn, B. (2015). *Database Systems, A Practical Approach to Design Implementation and Management*. (6th ed.) Essex, England: Pearson Education Limited.

Torkar, R., Gorschek, T., Feldt, R., Svahnberg, M., Raja, U. A. and Kamran, K. (2012). Requirements Traceability: A Systematic Review and Industry Case Study. *International Journal of Software Engineering and Knowledge Engineering.* 22, 385–433.

Treude, C., Robillard, M. and Dagenais, B. (2015). Extracting Development Tasks to Navigate Software Documentation. *IEEE Transactions on Software Engineering.* 41(6), 565-581.

Tsuchiya, R., Kato, T., Washizaki, H., Kawakami, M., Fukazawa, Y. and Yoshimura, K. (2013). Recovering Traceability Links Between Requirements

and Source Code in the Same Series of Software Products. *Proceedings of the 17th International Software Product Line Conference*. 26-30 August. Tokyo, Japan: ACM, 121–130.

Tulasi, A., Chittoor, R. and Mani, V.. (2016). System Testing Optimization in a Globally Distributed Software Engineering Team. *11th International Conference on Global Software Engineering (ICGSE).* 2-5 August. California, USA: IEEE, 99–103.

Umamaheswari, E. and Ghosh, K. (2014). Software Quality: Dual Experts Opinion and Conditonal Based Aggregation Method. *International Journal of Engineering and Technology (IJET).* 6, 1167–1175.

Van Loggem, B. (2014). Software Documentation: A Standard for the 21st Century. *Proceedings of the International Conference on Information Systems and Design of Communication*. 16-17 May. Lisbon, Portugal: ACM, 149–154.

Van Veenendaal, E. and Cannegieter, J.J. (2010). Test Maturity Model integration (TMMi). *TMMi Foundation*.

W. Linda (2006). *Bidirectional Requirements Traceability*. Westfallteam

Waitelonis, J., Exeler, C. and Sack, H. (2015). Linked Data Enabled Generalized Vector Space Model to Improve Document Retrieval. *NLP & DBpedia 2015 Workshop at 14th Int. Semantic Web Conferece.*. 11 October. Bethlehem, Pennsylvania, USA: CEUR-WS, 1-12.

Wang, Q., Xu, J., Li, H. and Craswell, N. (2013). Regularized Latent Semantic Indexing: A New Approach to Large-scale Topic Modeling. *ACM Transactions on Information Systems (TOIS).* 31, 5.

Weilkiens, T., Lamm, J. G., Roth, S. and Walker, M. (2015). B: The V-Model. *Model-Based System Architecture*. Hoboken, NJ, USA: John Wiley & Sons.

Wesiak, G., Al-Smadi, M. and GÜtl, C. (2012). Towards an Integrated Assessment Model for Complex Learning Resources: Findings from an Expert Validation. *15th International Conference On Interactive Collaborative Learning (ICL)*. 26-28 September. Villach, Austria: IEEE, 1–7.

Wiederseiner, C., Garousi, V. and Smith, M. (2011). Tool Support for Automated Traceability of Test/Code Artifacts in Embedded Software Systems. *The 10th International Conference on Trust, Security and Privacy in Computing and Communications*. 16-18 November. Changsha, China: IEEE, 1109–1117.

Wiegers, K. and Beatty, J. (2013). *Software Requirements*. (3rd ed.). Redmond, Washington: Pearson Education.

Wijesinghe, D. B., Kamalabalan, K., Uruththirakodeeswaran, T., Thiyagalingam, G., Perera, I. and Meedeniya, D.. (2014). Establishing Traceability Links Among Software Artefacts. *International Conference on Advances in ICT for Emerging Regions (ICTer)*. 10-14 December. Colombo, Sri Lanka: IEEE, 55–62.

Yu, L., Li, X. and Li, Z. (2011). Testing Tasks Management in Testing Cloud Environment. *The 35th Annual Computer Software and Applications Conference, COMPSAC 2011*. 18-20 July. Munich, Germany: IEEE, 76–85.

Zhang, C. and Zhan, S. (2013). Research and Implementation of Full-text Retrieval System Using Compass Based on Lucene. *Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering*. 23-25 August. Xi'an, China: Springer, 349–356.

Zhang, Y., Wan, C. and Jin, B. (2016). An Empirical Study on Recovering Requirement-to-code Links. *17th IIEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 30 May - 1 June. Shanghai, China: IEEE, 121–126.

Zhi, J., Garousi-Yusifoğlu, V., Sun, B., Garousi, G., Shahnewaz, S. and Ruhe, G. (2015). Cost, Benefits and Quality of Software Development Documentation: A Systematic Mapping. *Journal of Systems and Software.* 99, 175–198.

Zhi, J. and Ruhe, G. (2013). DEVis: A Tool for Visualizing Software Document Evolution. *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*. 27-28 September. Eindhoven, Netherlands: IEEE, 1–4.