UNIVERSITI PUTRA MALAYSIA

METHOD AND TOOL TO GENERATE REQUIREMENT TRACEABILITY MATRIX FOR SCRUM DEVELOPMENT METHODOLOGY

GUNAVATHI DURAISAMY

FSKTM 2014 27

METHOD AND TOOL TO GENERATE REQUIREMENT TRACEABILITY MATRIX FOR SCRUM DEVELOPMENT METHODOLOGY

By

GUNAVATHI DURAISAMY

Thesis Submitted to the School of Graduate Studies,
Universiti Putra Malaysia, in Fulfillment of the
Requirement for the Degree of Master of Science

October 2014

© COPYRIGHT UPM

All material contained within the thesis, including without limitation text, logos, icons, photographs and all other artwork, is copyright material of Universiti Putra Malaysia unless otherwise stated. Use may be made of any material contained within the thesis for non-commercial purposes from the copyright holder. Commercial use of material may only be made with the express, prior, written permission of Universiti Putra Malaysia.

Copyright © Universiti Putra Malaysia

© COPYRIGHT UPM

DEDICATION

To my parents, family members and friends for the endless support, encouragement and patience, and also those who made this study possible.

© COPYRIGHT UPM

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfillment of the requirement for the degree of Master of Science

METHOD AND TOOL TO GENERATE REQUIREMENT TRACEABILITY MATRIX FOR SCRUM DEVELOPMENT METHODOLOGY

By

GUNAVATHI DURAISAMY

October 2014

Chairman: Associate Professor Rodziah Atan, PhD

Faculty: Computer Science and Information Technology

Requirement traceability matrix is a table that captures the complete user and system requirement for a system. It helps to trace from requirement till testing in order to verify that the requirement is fulfilled. In SCRUM development methodology, requirement traceability matrix is used to capture the linkage of user stories between product backlog and sprint backlog documents. The linkages between the requirements are retrieved through these two documents. However, unstructured format of both documents do not help in requirement traceability. Thus, requirement traceability has become an issue for SCRUM practitioners especially for system development and maintenance phases. Therefore, this study introduce structured format of available artifacts which can create and maintain traceability link between those documents and develops a tracing tool to generate requirement traceability matrix automatically. Both the documents used in this study have to be prepared by using the proposed structured format and the developed traceability tool is able to generate the requirement traceability matrix automatically. Two case studies have been used for pre-test and post-test experiments. The result shows that the introduced structured format is very useful and it has increased the efficiency of retrieving the matrix far better than manual process. By using the proposed method and tool, it’s statistically shows significant result of time saving, completeness and correctness to generate requirement traceability matrix. Thus, the proposed method and the developed tool help in achieving requirement traceability in SCRUM methodology.

© COPYRIGHT UPM

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia
sebagai memenuhi keperluan untuk ijazah Master Sains

KAEDAH DAN ALAT UNTUK MENJANA MATRIK KEBOLEHKESANAN KEPERLUAN BAGI METODOLOGI PEMBANGUNAN SCRUM

Oleh

GUNAVATHI DURAISAMY

October 2014

Pengerusi: Profesor Madya Rodziah Atan, PhD

Fakulti: Sains Komputer dan Teknologi Maklumat

Matrik keperluan kebolehkesanan adalah jadual yang merekodkan keperluan pengguna dan sistem yang lengkap. Ia membantu mengesan daripada keperluan pengguna sehingga ke ujian sistem untuk mengesahkan bahawa kehendak telah dipenuhi. Dalam metodologi pembangunan SCRUM, matrik keperluan kebolehkesanan digunakan untuk merekod kaitan antara cerita pengguna dari 'product backlog' dan 'sprint backlog'. Hubungan antara keperluan akan diambil daripada kedua-dua dokumen ini. Walau bagaimanapun, format kedua-dua dokumen yang tidak berstruktur tidak dapat membantu dalam keperluan kebolehkesanan. Oleh itu, keperluan kebolehkesanan telah menjadi isu untuk pengamal SCRUM terutamanya untuk fasa pembangunan and penyelenggaraan sistem. Kajian ini memperkenalkan format berstruktur yang dapat menjana dan mengekalkan pautan kebolehkesanan antara dokumen yang sedia ada, serta membangunkan alat pengesanan untuk menjana matrik keperluan kebolehkesanan secara automatik. Kedua-dua dokumen yang digunakan dalam kajian ini perlu disediakan dengan menggunakan format berstruktur yang dicadangkan dan alat pengesanan yang dibangunkan mampu menjana matrik keperluan kebolehkesanan secara automatik. Dua kajian kes telah digunakan untuk pra-ujian dan pasca- ujian eksperimen. Hasilnya menunjukkan bahawa format berstruktur yang diperkenalkan adalah sangat berguna dan ia telah meningkatkan kecekapan dalam menjana matrik keperluan kebolehkesanan jauh lebih baik daripada proses manual. Dengan menggunakan kaedah dan alat pengesanan yang dicadangkan, pembuktian statistik menunjukkan keputusan yang signifikan dalam penjimatan masa, ketepatan and kelengkapan untuk menjana matrik keperluan kebolehkesanan. Keperluan kebolehkesanan boleh dicapai dalam metodologi pembangunan SCRUM melalui dokumentasi yang berstruktur seperti yang dicadangkan dan melalui alat pengesanan yang telah dibangunkan.

© COPYRIGHT UPM

ACKNOWLEDGEMENTS

Thanks to the Almighty God, for giving me the strength and commitment to successfully complete this study. First of all, thanks to my supervisor, Assoc. Prof. Dr. Rodziah Atan who gives me full support, courage, ideas, advices and knowledgeable supervision throughout this study.

Special thanks to my strongest source of motivation and inspiration throughout the entire process of study: my respectful mother, Nallamah. The never ending supports from my siblings, Kalaiwani and Thilaghawati. The support and understanding from my friends Dhanam and Jayanthi. The courage and motivation from my superior at work Jenny Ng.

Next, I would like to thank management of IXEON MSC for the support and understanding and also give me the change and opportunity to conduct both my experiments in their organization.

Last but not least, I would like to dedicate my sincere gratitude and appreciation to all my friends and all who is directly or indirectly has contributed to the success of this study. Cherish for all your support, guidance and help.

© COPYRIGHT UPM

I certify that an Examination Committee has met on 8 October 2014 to conduct the final examination of Gunavathi a/p Duraisamy on her thesis entitled "METHOD AND TOOL TO GENERATE REQUIREMENT TRACEABILITY MATRIX FOR SCRUM DEVELOPMENT METHODOLOGY" in accordance with the Universities and University Colleges Act 1971 and the Constitution of the Universiti Putra Malaysia [P.U. (A) 106] 15 March 1998. The Committee recommends that the student be awarded the Master of Science.

Members of the Thesis Examination Committee were as follows:

Abu Bakar bin Md Sultan, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Hazura bt. Zulzalil, PhD
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Internal Examiner)

Yusmadi Yah bt Jusoh, PhD
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Internal Examiner)

Wan Mohd. Nasir Wan Kadir, PhD
Associate Professor
Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia
(External Examiner)

ZULKARNAIN ZAINAL, PhD
Professor and Deputy Dean
School of Graduate Studies
Universiti Putra Malaysia

Date:

© COPYRIGHT UPM

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfillment of the requirement for the degree of Master of Science. The members of the Supervisory Committee were as follows:

Rodziah Atan, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Abdul Azim Abdul Ghani, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

BUJANG KIM HUAT, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date:

© COPYRIGHT UPM

Declaration by graduate student

I hereby confirm that:

- this thesis is my original work;
- quotations, illustrations and citations have been duly referenced;
- this thesis has not been submitted previously or concurrently for any other degree at any other institutions;
- intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from supervisor and the office of Deputy Vice-Chancellor (Research and Innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software.

Signature: ________________________ Date: _________________

Name and Matric No.: ____

© COPYRIGHT UPM

Declaration by Members of Supervisory Committee

This is to confirm that:

- the research conducted and the writing of this thesis was under our supervision;
- supervision responsibilities as stated in the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) are adhered to.

Signature: ____________________

Name of
Chairman of
Supervisory
Committee: ____________________

Signature: ____________________

Name of
Member of
Supervisory
Committee: ____________________

© COPYRIGHT UPM

TABLE OF CONTENTS

© COPYRIGHT UPM

© COPYRIGHT UPM

LIST OF TABLES

© COPYRIGHT UPM

© COPYRIGHT UPM

© COPYRIGHT UPM

LIST OF FIGURES

© COPYRIGHT UPM

© COPYRIGHT UPM

LIST OF ABBREVIATIONS

CMMI	Capability Maturity Model Integrated
UML	Unified Modeling Language
XP	eXtreme Programming
CASE	Computer Aided Software Engineering
MTTC	Mean Time To Close
ERD	Entity Relationship Diagram
PHP	Hypertext Preprocessor
HTTP	Hypertext Transfer Protocol
RDBMS	Relational Database Management System
RAM	Random Access Memory
UI	User Interface
RTM	Requirement Traceability Matrix
UAT	User Acceptance Test
PMO	Project Management Officer
SIT	System Integration Testing
VCR	Visual Custom Report
SAMS	Sports Administrative Management System

© COPYRIGHT UPM

LIST OF APPENDIES

© COPYRIGHT UPM

© COPYRIGHT UPM

CHAPTER 1

INTRODUCTION

1.1 Introduction

Software production increases in conjunction with fastest growing computing technology. It is getting larger day by day with increased number of artifacts along with the source codes. Software development processes have to go through several development life cycles since it is a complex endeavor. Each life cycle focuses on different level of software development activity such as planning, requirement gathering, requirement elicitation, analysis, design, development, implementation, validation and maintenance. Different kind of documentation is created for each level by different people and it requires a high level of maintenance to ensure these documents are up-to-date.

Varieties of studies conducted to measure the success rate of software development projects and numerous software development methodologies were introduced to achieve delivering software on time. However, projects are often delivered over schedule with many quality issues. Agile development methodology is new and one of the preferred methodology in recent years (Scott, 2011) as it focuses on improving the development process. It is designed in a way that the end product delivered continuously with frequent changes.

Agile is a philosophy was initiated by 17 signatories in 2001. A group of software practitioners and consultants (Lee & Xia, 2010; Cockburn, 2006; *Beck et al., 2001*) publish four agile manifestos as below. More details about these manifestos are described in Appendix A.

"We are uncovering better ways of developing software by doing it and helping others do it. We value:

1. Individual and interaction over process and tools.
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

That is, while there is value in the items on the right, we value items on the left more." (Agile Alliance, 2001b).

Documentation is one of the important activities in software engineering. Software documents explain how the software operates or how to use it and it means differently to people in different role. Large amount of associated documents are being generated for large software development projects. The common types of documentation being prepared are requirement, design, technical, end user and marketing specification. Creating and maintaining these documentations need a higher amount of costs. Therefore, the software development team has to give an equal attention and effort for documentation as of to the software development.

However, this is different in organization that is practicing agile methodology. Agile methodology focuses more on working software than the comprehensive documentation, but it doesn't mean that agile not producing any document. Agilists create documentations only when there is a need at appropriate point in the life cycle. Therefore tracing requirement from the available document is hard and sometimes impossible for developers (Cleland-Huang, 2012).

Traceability is defined as "the degree to which a relationship can be established between two or more products of the development process, especially products having predecessor-successor or master-subordinate relationship to one another" by IEEE (2004). Requirement traceability is "the ability to describe and follow the life of a requirement, both forward and backward direction, ideally through the entire system lifecycle" (Mäder & Gotel, 2012; Cleland-Huang, Gotel, Hayes, Mäder, & Zisman, 2014; Hemalatha & Prakash, 2013). Requirement can be traced in two directions (Mäder & Gotel, 2012; Hemalatha & Prakash, 2013):

i. Backward traceability: "following the requirement back to its origin".
ii. Forward traceability: "tracing the requirement to the modules and functions by which it is implemented".

In case of traceability is missing among the requirement or documentation being prepared, the main issue or problem faced by project team is the hardness to get the product to meet the original or core requirement in sequence. When there is a missing linkage in traceability between requirement and design specification, there are possibilities of delivering an

© COPYRIGHT UPM

end product which does not met the intended outcome. This will lead to low quality in the delivered software.

1.2 SCRUM Methodology

SCRUM defined by Ken Schwaber as “a process that accepts that the development process is unpredictable, formalizing the “do what it takes” mentality, and has found success with numerous independent software vendors.” (Schwaber & Beedle, 2002). SCRUM has become increasingly popular in the past decade (Schwaber & Beedle, 2002). It practices time-boxed process as:

i. Iterative: the product is produced during the small cycles called iterations.
ii. Incremental: the functionality of the product increase during each iteration by adding new properties.

Communication among team members, team collaboration in completing the task, faster way of exchanging information between team members, teamwork, workable end product and flexibility are the key attributes of SCRUM which differ from other traditional methodologies. SCRUM allows frequent changes in specification as per end user requirement.

1.2.1 SCRUM Documentation

As the second manifesto of agile methodology (working software over comprehensive documentation) is being concerned, document or artifacts produced in SCRUM methodology are limited to:

1. Product Backlog: This document contains the whole list of business and technical functionality to be developed.
2. Sprint Backlog: Details for each item from product backlog are logged in this document. The list consists of business and technology features, enhancements and defect which are planned for current iteration.
3. Burndown Chart: Remaining hours to complete for each of the item in sprint backlog are graphed in this chart.

© COPYRIGHT UPM

1.2.2 Traceability in SCRUM

As the requirement changes frequently at the end of each iteration, these changes have to be considered to be included in the next iteration. When there is no proper documentation has been prepared, the sprint backlog and product backlog have to be updated with the current changes required or take place as well as with the existing requirement list. At this point of time, SCRUM team faces requirement traceability linkage issues to see the impacted requirements which are already implemented and to be implemented.

1.3 Problem Statement

Traceability is essential in assuring that system is conform to requirements with terms are defined and used consistently which corresponding to the structures of models (Alexander, 2002). Even though, various standards and government agencies (Asuncion, François & Taylor, 2007; Leffingwell & Widrig, 2002) encouraging and mandating traceability, many organization take it infeasible to incorporate traceability into their practices (Alexander, 2002). Currently the information about implemented requirements, code changes or test results are traced manually (Azmi, Ibrahim & Mahrin, 2011) and it is time consuming (Brinkkemper, 2004).

Traceability links are typically stored "in a trace matrix" that is constructed manually by team members during system development (Port, Nikora, Hihn & Huang, 2011; Borg, Runeson & Ardö, 2013). Building and maintaining complete and accurate trace matrices is "arduous and effort consuming" and so practitioners often fail to implement consistent and effective traceability processes (Espinoza & Garbajosa, 2011; Cleland-Huang, 2012). The "sheer number of artifacts" produced in a project, the differing levels of formality and specifically between various artifact types and the complex interrelationships between artifacts (Mäder & Gotel, 2012; Alexander, 2002) are the main reason of traceability links problem. SCRUM documents are only limited to product backlog and sprint backlog, and codes are treated as the final specification. Limited content of the documents in SCRUM leads to limited references in requirement traceability.

Developers view traceability as a heavyweight and burdensome activity (Cleland-Huang, 2012; *Zhang et al.*, 2010). To understand the requirement,

© COPYRIGHT UPM

developers are depending on the documentations. Without documentation, source codes are the only reference point for them which consume their time more and they tend to make mistakes easier (Azmi, Ibrahim & Mahrin, 2011). Traceability is “the ability to link between various artifacts in software development phases linking requirements, design, source code and testing artefacts” (De Lucia, Marcus, Oliveto & Poshyvanyk, 2012; Ramesh & Jarke, 2001). Unfortunately, difficulties in using and maintaining traceability links causing many organizations failed to implement effective traceability. On top of that, developers rarely apply formal requirement specification techniques in practice while creating or maintaining requirement documentation. Due to this there are no proper linkage and identification in the documents in order to create or maintain the traceability links in SCRUM.

All three problems which are, creating and maintaining traceability links among documents, time consuming in current manual method and informal structure of the document showed the necessity of proposed solution of this research. This study proposed a method to create and maintain links between product backlog and sprint backlog documents which eventually will reduce the time taken in generating requirement traceability matrix.

1.4 Research Question

Below are the research questions identified for the stated research problems:

1. How to structure the existing format and content?
2. How to create traceability reference links among the documents?
3. How to retrieve requirement traceability matrix efficiently?

1.5 Objective

In this research, impact and benefits of traceability in SCRUM software development methodology will be studied to enable the development team to create and maintain requirement traceability links.

Therefore, the study aims to achieve the following objectives:

© COPYRIGHT UPM

1. To propose a method that is able to effectively create and maintain requirement traceability links in SCRUM.
2. To validate and demonstrate the applicability of the proposed method by developing its supported tool and conducting experiments.

1.6 Scope and Assumption

This study will focus on Agile – SCRUM methodology for requirement traceability among functional requirement and design. The requirement traceability will be based on the documents prepared by selected SCRUM team which are product backlog and sprint backlog.

It is presumed by this study that the documents used are only SCRUM documents prepared by organization which currently practices the SCRUM development methodology.

1.7 Significance of the study

As this research aims to develop a requirement traceability method and tool for SCRUM methodology, the outcome of this study is to reduce the time taken to retrieving requirement traceability matrix automatically whenever needed by SCRUM team members. By achieving this, it would be beneficial for the organization who practicing SCRUM to maintain the bidirectional traceability among requirement even with limited documentation.

1.8 Definition of Terms

This part describes the several definitions that are used in this research. These definitions are by no means comprehensive, but it will provide a focal point for terms to be used in the following chapters. These definitions are as follows:

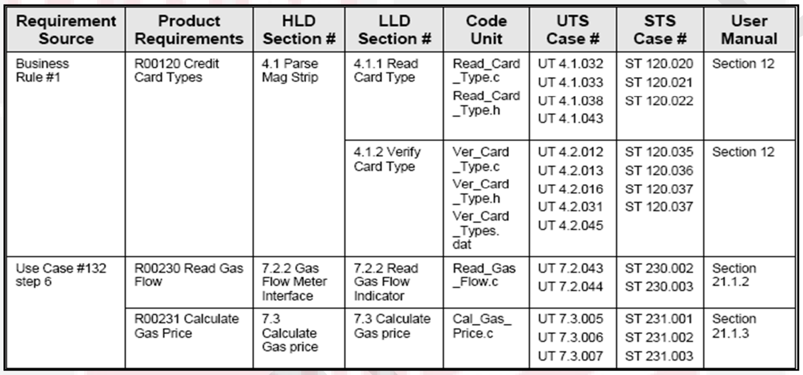
© COPYRIGHT UPM

Definition 1: Requirement Traceability

"A characteristics of a system in which the requirements are clearly linked to their sources and to the artifacts created during the system development life cycle based on these requirements". (Mäder & Gotel, 2012; Hemalatha & Prakash, 2013; Ramesh & Jarke, 2001).

Definition 2: Requirement Traceability Matrix

A table consists of features or requirement traces in both forward and backwards direction. A sample of requirement traceability matrix is shown in Figure 1.1.



Requirement Source	Product Requirements	HLD Section #	LLD Section #	Code Unit	UTS Case #	STS Case #	User Manual
Business Rule #1	R00120 Credit Card Types	4.1 Parse Mag Strip	4.1.1 Read Card Type	Read_Card _Type.c Read_Card _Type.h	UT 4.1.032 UT 4.1.033 UT 4.1.038 UT 4.1.043	ST 120.020 ST 120.021 ST 120.022	Section 12
			4.1.2 Verify Card Type	Ver_Card _Type.c Ver_Card _Type.h Ver_Card _Types. dat	UT 4.2.012 UT 4.2.013 UT 4.2.016 UT 4.2.031 UT 4.2.045	ST 120.035 ST 120.036 ST 120.037 ST 120.037	Section 12
Use Case #132 step 6	R00230 Read Gas Flow	7.2.2 Gas Flow Meter Interface	7.2.2 Read Gas Flow Indicator	Read_Gas _Flow.c	UT 7.2.043 UT 7.2.044	ST 230.002 ST 230.003	Section 21.1.2
	R00231 Calculate Gas Price	7.3 Calculate Gas price	7.3 Calculate Gas price	Cal_Gas_ Price.c	UT 7.3.005 UT 7.3.006 UT 7.3.007	ST 231.001 ST 231.002 ST 231.003	Section 21.1.3

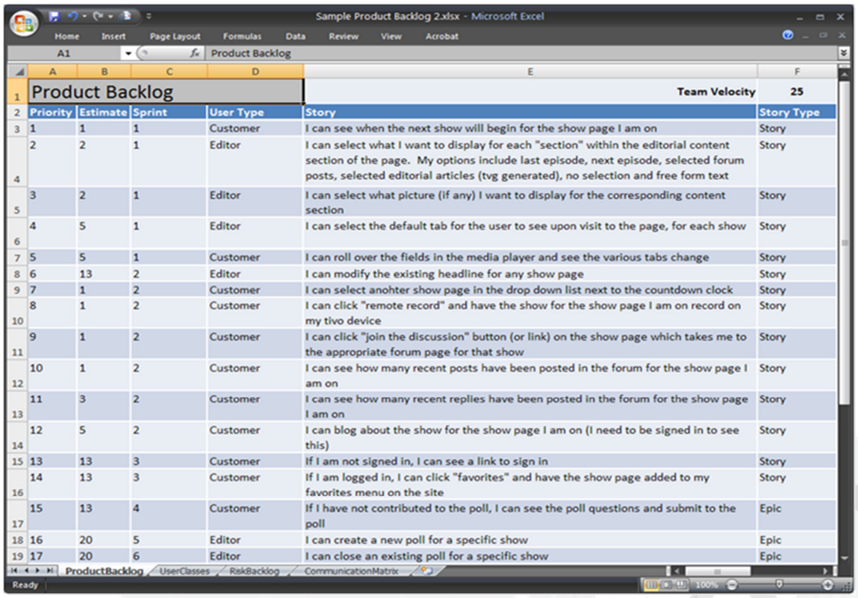
Source of sample image: www.westfallteam.com

Figure 1.1: Sample Requirement Traceability Matrix

Definition 3: Product Backlog

Product backlog contains list of features and description of the functionality. Product owner have to maintain this documents and responsible to prioritize the product backlog items. It comprises features, bugs, technical work and knowledge acquisition as shown in Figure 1.2.

© COPYRIGHT UPM



Sample Product Backlog 2.xlsx - Microsoft Excel

Product Backlog

Team Velocity 25

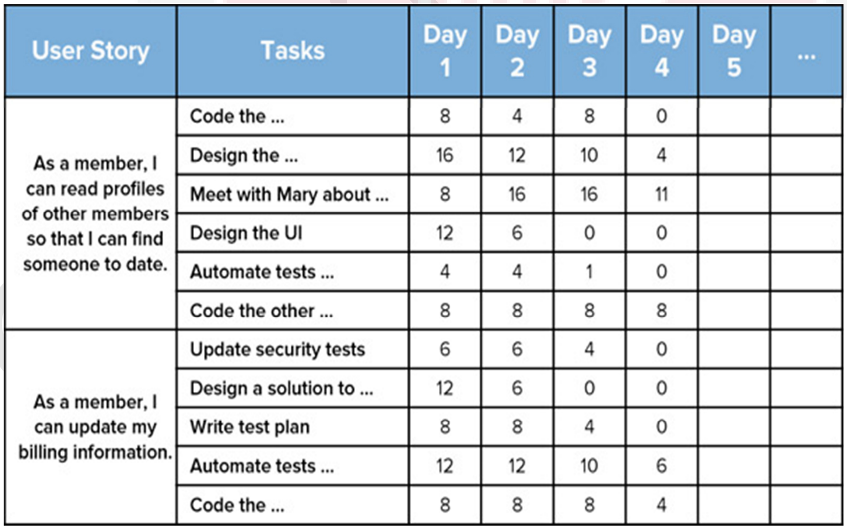
Priority	Estimate	Sprint	User Type	Story	Story Type
1	1	1	Customer	I can see when the next show will begin for the show page I am on	Story
2	2	1	Editor	I can select what I want to display for each "section" within the editorial content section of the page. My options include last episode, next episode, selected forum posts, selected editorial articles (tvg generated), no selection and free form text	Story
3	2	1	Editor	I can select what picture (if any) I want to display for the corresponding content section	Story
4	5	1	Editor	I can select the default tab for the user to see upon visit to the page, for each show	Story
5	5	1	Customer	I can roll over the fields in the media player and see the various tabs change	Story
6	13	2	Editor	I can modify the existing headline for any show page	Story
7	1	2	Customer	I can select anohter show page in the drop down list next to the countdown clock	Story
8	1	2	Customer	I can click "remote record" and have the show for the show page I am on record on my tivo device	Story
9	1	2	Customer	I can click "join the discussion" button (or link) on the show page which takes me to the appropriate forum page for that show	Story
10	1	2	Customer	I can see how many recent posts have been posted in the forum for the show page I am on	Story
11	3	2	Customer	I can see how many recent replies have been posted in the forum for the show page I am on	Story
12	5	2	Customer	I can blog about the show for the show page I am on (I need to be signed in to see this)	Story
13	13	3	Customer	If I am not signed in, I can see a link to sign in	Story
14	13	3	Customer	If I am logged in, I can click "favorites" and have the show page added to my favorites menu on the site	Story
15	13	4	Customer	If I have not contributed to the poll, I can see the poll questions and submit to the poll	Epic
16	20	5	Editor	I can create a new poll for a specific show	Epic
17	20	6	Editor	I can close an existing poll for a specific show	Epic

Source of sample image: https://www.mitchlacey.com/intro-to-agile/the-product-backlog

Figure 1.2: Sample Product Backlog

Definition 4: Sprint Backlog

Sprint backlog consist of list of task to be completed for particular sprint or iteration as shown in Figure 1.3. Team selects the product backlog items and identifies the tasks to be done. After estimate the effort needed, the task will be assigned to the team members.



User Story	Tasks	Day 1	Day 2	Day 3	Day 4	Day 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the ...	8	4	8	0		
	Design the ...	16	12	10	4		
	Meet with Mary about ...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate tests ...	4	4	1	0		
	Code the other ...	8	8	8	8		
As a member, I can update my billing information.	Update security tests	6	6	4	0		
	Design a solution to ...	12	6	0	0		
	Write test plan	8	8	4	0		
	Automate tests ...	12	12	10	6		
	Code the ...	8	8	8	4		

Source of sample image: http://www.mountaingoatsoftware.com/agile/scrum/sprint-backlog/

Figure 1.3: Sample Sprint Backlog

© COPYRIGHT UPM

1.9 Organization of thesis

This thesis is organized into six (6) chapters. It starts with Chapter 1 which covers the introduction, background of the research, problem statements, objectives, scope significance of the study and definition of terms.

Chapter 2 describes the literature review of this project. The agile methodology, SCRUM, traceability methods, requirement traceability, traceability tool and documents which are related to this study will be described in this chapter.

Chapter 3 describes the research methodology used in this study. The step by step activities has been explained in detail in this chapter.

Chapter 4 describes the system design and implementation of the developed traceability tool. In this chapter, the detail design of the tool has been drawn and explained.

Chapter 5 shows the results and discussion on the developed tool based on the metric and attribute set. The result gathered for per-test and post-test for both case studies.

Chapter 6 summarizes the overall study as a conclusion and some outline for the future work.

© COPYRIGHT UPM

REFERENCES

Agile Alliance (2001a). Principles behind the agile manifesto. Retrieved November 11, 2006 from http://agilemanifesto.org/principles.html.

Agile Alliance (2001b). Manifesto for agile software development. Retrieved November 11, 2006 from http://agilemanifesto.org.

Ali, N., Gueneuc, Y., & Antoniol, G. (2013). Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. Software Engineering, IEEE Transactions on, 39(5), 725-741.

Alexander, I. (2002, September). Towards automatic traceability in industrial practice. In Proc. of the 1st Int. Workshop on Traceability (pp. 26-31).

Ambler, S. (2004) The Object Primer: Agile Model-Driven Development with UML 2.0

Antoniol, G., Canfora, G., Casazza, G., De Lucia, A. (2000) "Information Retrieval Models for Recovering Traceability Links between Code and Documentation", Proceedings of the Internaltional Conference on Software Maintenance, pp. 40-49.

Antoniol, G., Canfora, G., Casazza, G., De Lucia, A, Merlo, E. (October, 2002) "Recovering Traceability Links between Code an Documentation", IEEE Transactiona on Software Engineering,Vol. 28, No. 10, pp. 970-983

Appleton, B. (2005) ACME Blog: Traceability and TRUST-ability. Accessed June 2011 from http://bradapp.blogspot.com/2005/03/traceability-and-trust-ability.html

Appleton, B., Cowham, R. & Berczuk, S. (2007) Lean traceability: A smattering of strategies and solutions, CM Crossroads (Configuration Management)

Asuncion, H. U., François, F., & Taylor, R. N. (2007, September). An end-to-end industrial software traceability tool. In Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (pp. 115-124). ACM.

Avik, S., Amit, P., & Clay, W. (2007) On Generating EFSM models from Use Cases. Proceedings of the 29th International Conference on Software Engineering Workshops May 20-26 2007; p.97

Azmi, A., Ibrahim, S., & Mahrin, N. R. (2011). Formulating a Software Traceability Model for Integrated Test Documentation. In International

© COPYRIGHT UPM

Conference on Computer Engineering and Technology, 3rd (ICCET 2011). ASME Press.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grennung, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). Manifesto for Agile Software Development. http://AgileManifasto.org March 2002

Beck, K., Andres, C. (2004) Extreme programming explained:embrace change, 2nd edn. Addison-Wesley, Boston, MA ISBN:0321278658

Bennet, S., McRobb, S. & Farmer, R. (2002) Object oriented system analysis and design using UML (2nd ed). Backshire: McGraw – Hill Education

Borg, M., Runeson, P., & Ardö, A. (2013). Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability.Empirical Software Engineering, 1-52.

Bose, B., Kurhekar, M. & Ghoshal, J. (2010) Agile Methodology in Requirements Engineering, SETLabs Briefings Online. http://www.infosys.com/research/publica-tions/agilerequirements-engineering.pdf.

Brinkkemper, S. (2004). Requirements engineering research the industry is and is not waiting for. In Proceedings of the 10th anniversary international workshop on requirements engineering: foundation for software quality (REFSQ'04), Riga, Latvia (pp. 41-54).

Cavano, J.P & Mc Call, J.A. (1978) A framework for the measurement of software quality. Software Engineering Notes; Vol. 3, No. 5, pp. 133-139

Christel, M.G., & Kang, K.C. (1992) Issues in Requirements Elicitation. Technical Report CMU/SEI-92-TR-012 ESC-TR-92-012

Cleland-Huang, J., Gotel, O., Hayes, J. H., Mäder, P., & Zisman, A. (2014, May). Software traceability: Trends and future directions. In Proc. of the 36th International Conference on Software Engineering (ICSE), Hyderabad, India.

Cleland-Huang, J. (2006) Just enough requirements traceability. COMPSAC **1**, 41–42

Cleland-Huang, J. (2012). Traceability in agile projects. In Software and Systems Traceability (pp. 265-275). Springer London.

Cleland-Huang, J., Berenbach, B., Clark, S., Settimi, R. & Romanova, E. (2007) Best practices for automated traceability. IEEE Comp. **40**(6), 27–35 ISSN:0018-9162

© COPYRIGHT UPM

Cleland-Huang, J., Change, C., Christensen, M. (September, 2003) Event-Based Traceability for Managing Evolutionary Change, IEEE Transactions on Software Engineering, Vol. 29, No. 9, pp. 796- 810.

Cockburn, A. (2002a). Agile Software Development. Boston, Addison-Wesley

Cockburn, A. (2006). Agile software development: the cooperative game. Pearson Education.

De Lucia, A., Marcus, A., Oliveto, R., & Poshyvanyk, D. (2012). Information retrieval methods for automated traceability recovery. In Software and Systems Traceability (pp. 71-98). Springer London.

De Lucia, A., Fasano, F., Oliveto, R. & Tortora, G. (2007) Recoveringtraceability links in software artifact management systems using information retrieval methods. ACM Trans. Softw. Eng. Methodol. **16**(4), 13. ISSN:1049-331

Domges, R. & Pohl, K. (1998) Adapting Traceability Environments to Project-Specific Needs. Communications of ACM; Vol. 41, No 21

Espinoza, A., & Garbajosa, J. (2011). A study to support agile methods more effectively through traceability. Innovations in Systems and Software Engineering, 7(1), 53-69.

Egyed, A. (2000) "A Scenario-Driven Aproach to Traceability", 23rd International Conference on Software Engineering, pp. 123-132.

Ghazarian, A. (2008, November). Traceability patterns: an approach to requirement-component traceability in agile software development. InProceedings of the 8th WSEAS International Conference on Applied Computer Science (ACS'08), Venice, Italy (pp. 236-241).

Gotel, O. & Finkelstein, A. (1994) An analysis of the requirements traceability problem. In: Proceedings of the International Conference on Requirements Engineering (RE), pp. 94–102. IEEE Computer Society, Springs, Colorado

Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J. & Mäder, P. (2012). Traceability fundamentals. In Software and Systems Traceability (pp. 3-22). Springer London.

Hayes, J.H., Dekhtyar, A. & Osborne, J. (2003) Improving requirements tracing via information retrieval. In: Proceedings of the IEEE International Conference on Requirements Engineering (RE), p. 138. IEEE Computer Society, Washington, DC. ISBN:0-7695-1980-6

© COPYRIGHT UPM

Hemalatha, T., & Prakash, N. (2013, December). An Emperical Study of Performance in the Requirement Traceability Links (RTL) Using Initial Mapping Technique. In International Journal of Engineering Research and Technology(Vol. 2, No. 12 (December-2013)). ESRSA Publications.

IEEE Std 610.12-1990 (2004) IEEE Standard Glossary of Software Engineering Terminology.

Jacobsson, M. (2009, January) Implementing traceability in agile software development. Master's Thesis, Lund Institute of Technology

Kalpana, S. & Jagadish, S. (2008) Moving from Waterfall to Agile. Agile 2008 Conference.

Kritzinger, P. S., & 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., pp. 391-402, 2008.

Lan, C. & Balasubramaniam, R (2008) Agile Requirements Engineering Practices: An Empirical Study. IEEE Software.

Lawrence, R. & Yslas, B. (2006) Three-way cultural change: Introducing agile with two non-agile companies and a non-agile methodology. Proceedings of AGILE Conference

Lee C., Guadagno L. & Jia X. (2003 October) An Agile Approach to Capturing Requirements Traceability, Proceedings of the 2nd International Workshop on traceability in Emerging Forms of Software Engineering (TEFSE 2003), Canada

Lee, G., & Xia, W. (2010). Toward agile: an integrated analysis of quantitative and qualitative field data. Management Information Systems Quarterly, 34(1), 7.

Leffingwell, D., & Widrig, D. (2002). The role of requirements traceability in system development. The Rational Edge, 2.

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

Letelier, P. (2002 September) A Framework for Requirements Traceability in UML-based projects. Proceedings of the 1st International Workshop on Traceability for Emerging Forms of Software Engineering (TEFSE'02), Edinburgh, UK.

Lindval, M. & Sandahl, K. (1996) Practical Implications of Traceability, Software Practice and Experience, vol. 26, no. 10, pp 1161-1180.

© COPYRIGHT UPM

Lindvall, M. & Sandahl, K. (1998) Traceability Aspects of Impact Analysis in Object Oriented Systems, Software Maintenance: Research and Practice, 10(1), 37-57

Lucia, A. D., Fasano, F., Oliveto, R., & Tortora, G. (2007). Recovering traceability links in software artifact management systems using information retrieval methods. ACM Transactions on Software Engineering and Methodology (TOSEM), 16(4), 13.

Mäder, P., & Gotel, O. (2012). Towards automated traceability maintenance.Journal of Systems and Software, 85(10), 2205-2227.

Maletz, B., Schnedl,J.G., Brisson,H., & Zamazal, K. (2007) A Holistic Approach for Integrated Requirements Modeling in the Product Development Process. The Virtual Vehicle-Research Center, Graz, Austria;1-10

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval (Vol. 1, p. 6). Cambridge: Cambridge university press.

Marcus, A. & Maletic J.I. (2003) Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing, Proceedings of 25th International Conference Software Engineering

Mohan, K. & Ramesh, B. (2002) Managing variability with Traceability in product and Service Families, Proceedings of the 35th Hawaii International Conference on System Sciences, IEEE

Paetsch, F., Eberlein, A. & Maurer, F. (2003) Requirements engineering and agile software development, Eighth International Workshop on Enterprise Security, Linz, Austria, 9 - 11 June.

Paulk, C. (2002) Agile Methodologies and Process Discipline. Institute for Software Research; Paper 3.

Paulk, C. (2001) Extreme Programming from CMM Perspective. IEEE Software; 18(6): 19-26.

Pinheiro, F.A.C. (2003) Requirements traceability. In: Sampaio do Prado Leite, J.C., Doorn J.H. (eds.), Perspectives on Software Requirements, vol. 753, pp. 93–113. Springer, Berlin

Port, D., Nikora, A., Hihn, J., & Huang, L. (2011, May). Experiences with text mining large collections of unstructured systems development artifacts at jpl. InSoftware Engineering (ICSE), 2011 33rd International Conference on (pp. 701-710). IEEE.

Prerna, G. & Rao, D.S. (2011) Best Practices to Achieve CMMI Level 2 Configuration Management Process Area through VSS tool.

© COPYRIGHT UPM

International Journal of Computer Technology and Applications; Vol 2(3): 542-558

Ramesh, B. & Jarke, M. (2001) Toward reference models for requirements traceability. IEEE Trans. Softw. Eng. **27**(1), 58–93

Richardson, J. & Green, J. (2004) Automating traceability for generated software artifacts. In: Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE '04), pp. 24–33. IEEE Computer Society, Washington, DC. ISBN:0-7695-2131-2

Sillitti, A. & Succi, G. (2005) Requirements Engineering for Agile methods. Engineering and Managing Software Requirements, Springer

Schwaber, K. (2009) SCRUM GUIDE. Scrum Alliance.

Schwaber, K. & Beedle, M. (2002) Agile Software Development with SCRUM. Prentice Hall, Englewood Cliffs

Schwaber, K., & Sutherland, J. (2011). The scrum guide. Scrum. org, October.

Schwarz, H., Ebert, J., & Winter, A. (2010). Graph-based traceability: a comprehensive approach. Software & Systems Modeling, 9(4), 473-492.

Scott, W. (2011). How Successful are IT Projects, Really?. Dr.Dobb's Journal.

Shelly G.B., Cashman, T.J. & Rosenblatt. (2001) System Analysis and Design (4th ed). Course Technology Thomson Learning, Boston

Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007, January). Distributed scrum: Agile project management with outsourced development teams. In System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on (pp. 274a-274a). IEEE.

Whitten, J.L., Betley, L.D., & Diltman, D.C. (2001) System Analysis and design method (5th ed). Boston: Mc-Graw-Hill Education

Weigers, K. (2003) Software Requirements, Redmond: Microsoft Press.

Zhang, Z., Arvela, M., Berki, E., Muhonen, M., Nummenmaa, J., & Poranen, T. (2010). Towards lightweight requirements documentation. Journal of Software Engineering and Applications, 3(09), 882.

© COPYRIGHT UPM