A PRELIMINARY STUDY OF SOFTWARE TRACEABILITY REFERENCE MODEL USING FEATURE MODELING

MUHAMMAD IRSYAD BIN ABDULLAH

This Project Report Submitted In Partial Fulfillment of the Requirement for the Degree in the Master of Computer Science (Real Time Software Engineering)

Faculty of Computer Science & Information System
Universiti Teknologi Malaysia

NOVEMBER, 2007

Special dedication to...

My family, as a token of love and appreciation

To my loving parents… who placed me on the right path and taught me the value of knowledge

To Haliyusswanee, wife, friend, inspiration and blessing.

And to all my unforgettable friends

“THANK YOU FOR BEING PART OF MY LIFE”

ACKNOWLEDGEMENT

"BISMILLAHIRRAHMANIRRAHIM"

Special thanks to my supervisor, En Mohd Nazri Kama who gave lots of supports for me in preparing this report and also giving me so much precious experiences and guiding me to the success of delivering the research. To all CASE lecturers for serving with software engineering knowledge that definitely benefit me in my future profession. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family members.

THANKS.

ABSTRACT

Traceability, a key aspect of any engineering discipline, enables engineers to understand the relations and dependencies among various artifacts in a system. It is a well-known fact that even in organizations and projects with mature software development processes, software artifacts created as part of these processes end up to be disconnected from each other. From a software engineer's perspective, it therefore becomes essential to establish and maintain the semantic connections among these artifacts. The missing traceability among software artifacts becomes a major challenge for many software engineering activities. As a result, during the comprehension of existing software systems, software engineers have to spend a large amount of effort on synthesizing and integrating information from various sources to establish links among these artifacts. Existing research in software traceability focuses on reducing the cost associated with this manual effort by developing automatic assistance in establishing and maintaining traceability links among software artifacts. This research is based on the premise that a more effective and unified solution to manage traceability semantic link information can be achieved by considering a feature model as the traceability reference model. The aim of this research is to propose a software traceability reference model that can store traceability links information using the concept of feature modeling. It identifies the traces of software components of various software artifacts such as design and requirements and stores them in hierarchical form.

ABSTRAK

Jejakan adalah salah satu kunci utama dalam mana-mana bidang kejuruteraan yang membolehkan jurutera memahami hubungan dan kebergantungan pelbagai jenis artifak perisian dalam sesebuah sistem. Satu fakta yang jelas iaitu dalam mana-mana organisasi atau projek, untuk proses pembangunan perisian yang matang, artifak perisian yang terhasil dari proses ini pada akhirnya akan terpisah di antara satu sama lain. Dari perspektif seorang jurutera perisian, adalah sangat penting untuk menghasikan dan mengekalkan hubungan semantik di antara artifak ini. Kehilangan jejak di antara artifak perisian telah menjadi cabaran utama dalam kebanyakan aktiviti kejuruteraan perisian. Oleh itu, dalam fasa analisis, jurutera perisian terpaksa menggunakan usaha yang banyak untuk membuat sintesis dan integrasi terhadap maklumat dari pelbagai sumber dalam sistem yang sedia ada untuk menghasilkan hubungan antara artifak ini. Penyelidikan yang sedia ada dalam bidang jejakan perisian lebih memfokus kepada pengurangan kos yang berkaitan dengan usaha manual ini dengan membangunkan bantuan secara automatik dalam menghasilkan dan mengekalkan hubungan jejak di antara artifak perisian. Penyelidikan ini berdasarkan kajian bahawa penyelesaian yang lebih efektif dan teratur boleh tercapai melalui penggunaan model ciri-ciri sebagai rujukan model jejak. Tujuan penyelidikan ini adalah untuk membuat usul terhadap satu rujukan perisian jejak yang boleh menyimpan maklumat hubungan jejak dengan menggunakan konsep model ciri-ciri. Ianya akan mengenalpasti jejak komponen perisian dari pelbagai artifak perisian seperti fasa rekabentuk dan fasa keperluan dan menyimpan ianya dalam bentuk hirarki.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter provides an introduction to the research work presented in this thesis. It describes the research overview that motivates the introduction of *a preliminary study of software traceability reference model using feature modeling*. This is followed by a discussion on the research background, problems statements, objectives and importance of the study. Finally, it briefly explains the scope of work and structure of the thesis.

1.2 Introduction to Requirement Engineering

Systematic requirements analysis which is also known as *requirements engineering* is critical to the success of a development project (Pierre Bourque, Robert Dupuis, 2005). The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended. It is the process of discovering that purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation. There are a number of inherent difficulties in this process. Stakeholders (including paying customers, users and developers) may be numerous and distributed. Their goals may vary and conflict, depending on their perspectives of the environment in which they work and the tasks they wish to accomplish. Their goals may not be explicit or may be difficult to articulate, and,

inevitably, satisfaction of these goals may be constrained by a variety of factors outside their control. Conceptually, requirements analysis includes five types of activity (Bashar Nuseibeh, Steve Easterbrook, 2000):

1) *eliciting* requirements,
2) *modelling* and *analyzing* requirements,
3) *communicating* requirements,
4) *agreeing* requirements, and
5) *evolving* requirements.

Requirement Engineering is not only a process of discovering and specifying requirements, it is also a process of facilitating effective *communication* of these requirements among different stakeholders. The way in which requirements are documented plays an important role in ensuring that they can be read, analyzed, (re-) written, and validated. What is increasingly recognised as crucial, however, is *requirements management* – the ability, not only to write requirements but also to do so in a form that is readable and traceable by many, in order to manage their evolution over time. However there are challenges in which requirements traceability is a crucial approach to address this issue. It is another major factor that determines how easy it is to read, navigate, query and change requirements documentation.

1.3 Background of the Problem

Software traceability is fundamental to both software development and maintenance. It shows the ability to trace information from various resources that requires special skill and mechanism to manage it. In this research problem, it focuses on software traceability reference model to support traceability semantic discovery and classification. Following are some major issues to the research problem which related to traceability link information.

1.3.1 Inconsistencies of Traceability Information

Software traceability linkage or traceability link is a relationship between two or more artifacts or entities. It usually contains traceability information which can be used to describe the link's semantics. Links are classified as certain types in order to facilitate and enhance the understanding of the use of a link. However, the capture and use of traceability information of different perspectives will cause wide variations in the format and content of traceability information. In addition, semantics of a given linkage as viewed by different stakeholders would also have different meaning and understanding. This will lead to the inconsistencies of traceability information.

1.3.2 Excessive Traceability

Appleton as described by Jane Cleland-Huang has identified nine specific problems with traditional traceability practices and labeled them as 'gripes.' These gripes included the unnecessary creation of trace artifacts and the almost inevitable failure to accurately maintain them (Jane Cleland-Huang, 2006); the focus on upfront activities and comprehensive documentation which meant that the important task of writing code and delivering executable product was delayed and had a negative impact on production performance; creating an illusion that real work is being done while in fact time is being wasted developing the trace matrix; focusing on comprehensive documentation rather than the real deliverable of working software; creation of overhead to the change process itself which actually makes change more difficult to implement.

1.4 Statement of the Problem

This research is intended to deal with the traceability link information problems as discussed above. The main question is "*How to define a software traceability reference model that can provide a unified format for representing semantic information of traceability links?"*

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

i. Is there any references model to represent semantic information of traceability links?
ii. What is the suitable format to be used for representing semantic information of traceability links?

1.5 Objectives of the Study

The above problem statement serves as a premise to establish a set of specific objectives that will constitute major milestones of this research. The objectives of this research are listed as follows:

1) To investigate current software traceability approach.
2) To review state-of-the-art software traceability approach.
3) To propose preliminary software traceability reference model as a way to represent semantic information of traceability links.

1.6 Importance of the Study

Any approach to traceability faces significant challenges. According to Ramesh et al. [3], the U.S. Department of Defense spends billions of dollars each year collecting and maintaining traceability information. Often without getting an

adequate value for this money as traceability in many organizations is haphazard, the standards provide little guidance, and the models and mechanisms vary to a large degree and are often poorly understood. Not surprisingly, the market for requirements traceability tools is booming even though current tools support only rather simple traceability models and services. Previous models of requirements traceability focus on different aspects of requirements traceability. Version and configuration management systems focus on the *source* aspect (i.e., physical artifacts where traceability information is maintained), emphasizing the document management. Additionally, software development produces a highly heterogeneous set of software artifacts and it is difficult to create links that can span across multiple document formats. In addition, this heterogeneity of document formats is caused by the use of a heterogeneous set of document editors. Multiple editors make it difficult to automate the creation of traceability information because it is difficult to get all of the editors to work together. Indeed, traceability is generally acknowledged to be a highly manual, laborious process.

As such, the focus of this research is to develop a unified reference model to store traceability information links. It can also aid the maintenance of this type of information by creating it in an organized fashion.

1.7 Scope of Work

Software traceability can be applied to some applications such as consistency-checking, defect tracking, cross referencing and reuse (Ramesh and Jarke, 2001). The techniques and approaches used may differ from one another due to different objectives and feature requirements. Some of these approaches are geared toward system development while others are designed for system maintanence.

In this scope of research, it needs to explore a software traceability specifically to discover and classifed tracebility link using a predetermined reference model. The proposed model and techniques used should allow the implementation of respository to support different software artifacts. It also is assumed that the software

system feature has been identified. However, this research does not concern with the the implementation and testing of the proposed model.

1.8 Thesis Outline

This thesis covers some discussions on the specific issues associated to software traceability for impact analysis and understanding how this new research is carried out. The thesis is organized in the following outline.

Chapter 2: Discusses the literature review of the requirement engineering, traceability and feature modelling. Few areas of interest are identified from which all the related issues, works and approaches are highlighted. This chapter also discusses some current techniques of software traceability. Next, is a discussion on feature models as described by FODA method (Kang et al,1990) and by (Czarnecki et al, 2000).

Chapter 3: Provides a research methodology that describes the research design and formulation of research problems and validation considerations. This chapter leads to an overview of data gathering and analysis. It is followed by some research assumptions.

Chapter 4: Discusses the detailed model of the proposed software traceability reference model. A set of formal notations are used to represent the conceptual model of the software traceability. It is followed by some approaches and mechanisms to achieve the model specifications.

Chapter 5: Presents the initial result based on the hypothetical analysis.

.

Chapter 6: The statements on the research achievements, contributions and conclusion of the thesis are presented in this chapter. This is followed by the research limitations and suggestions for future work.

REFERENCES

Antoniou, G. (1998). The role of nonmonotonic representations in requirements engineering. *International Journal of Software Engineering and Knowledge Engineering*, 8(3): 385-399.

Bashar Nuseibeh, Steve Easterbrook (2000). Requirements Engineering: A Roadmap, *Proceedings of the Conference on the Future of Software Engineering*, Ireland.

Czarnecki, K., Eisenecker, U., *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, New York (2000)

Frakes, W., Prieto-Diaz, R., Fox, C.: DARE: *Domain Analysis and Reuse Environment, Annals of Software Engineering 5* (1998) 125-151

Griss, M. L., Favaro, J., d'Alessandro, M.: Integrating Feature Modeling with the RSEB, *Proc. Fifth International Conference on Software Reuse*, Victoria, BC, Canada (1998) 76-85

Griss, M.: Implementing Product-Line Features by Composing Aspects, In Proceedings of the First Software Product Line Conference (SPLC), August 28-31, 2000, Denver, Colorado, USA, Patrick Donohoe (Eds.), *Software Product Lines: Experience and Research Directions*, Kluwer Academic Publishers, Norwell, Massachusetts (2000) 47-70

Gotel, O. & Finkelstein, A. (1994). An Analysis of the Requirements Traceability Problem. *1st International Conference on Requirements Engineering (ICRE'94)*, Colorado Springs, April 1994, pp. 94-101.

Hoffman, M.A. (2000). *A methodology to support the maintenance of object-oriented systems using impact analysis*. Louisiana State University: Ph.D. Thesis.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *Unified Software Development Process.* USA: Addison-Wesley.

Jane Cleland-Huang (2006), "Just Enough Requirements Traceability", *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, De Paul University.

Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study, *Technical Report CMU/SEI-90-TR-21*, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University (1990)

Kwanwoo Lee, Kyo C. Kang, and Jaejoon Lee, (2000), Concepts and Guidelines of Feature Modeling for Product Line Software Engineering, Korea

Lazaro M. and Marcos E. (2005). Research in Software Engineering: Paragidms and Methods. PHISE'05.

Lee, K., Kang, K., Chae, W., Choi, B.: Feature-Based Approach to Object-Oriented Engineering of Applications for Reuse, *Software-Practice and Experience* 30, 9 (2000) 1025-1046

Marcus A. and Meletic J.I. (2003). Recovering ocumentation-to-Sourcce-Code Traceability Links Using Latent Semantic Indexing. *Proceedings of the Twenty Fifth International Conference on Software Engineering*. May 3-10. USA: IEEE Computer Society. 125-135.

Pierre Bourque, Robert Dupuis (2005). Chapter 2: Software Requirements. *Guide to the software engineering body of knowledge*. Los Alamitos.

Pashov, I.: *Feature Based Method for Supporting Architecture Refactoring and Maintenance of Long-Life Software Systems*. PhD Thesis, Technical University Ilmenau, 2004 (submitted).

Ramesh, B.; Jarke, M.: *Toward Reference Models for Requirements Traceability*. IEEE Transactions on Software Engineering, Volume 27, Issue 1 (January 2001) pp. 58 – 93

Orlena C. *Z.* Gotel & Anthony C. W. Finkelstein (1994). An Analysis of the Requirements Traceability Problem, *International Journal of Software Engineering and Knowledge Engineering*,