Traceability Method for Software Engineering Documentation

Nur Adila Azram[1] and Rodziah Atan[2]

[1] Department of Information System, Universiti Putra Malaysia, Company Serdang, Selangor, Malaysia

[2] Department of Information System, Universiti Putra Malaysia, Company Serdang, Selangor, Malaysia

Abstract

Traceability has been widely discussed in research area. It has been one of interest topic to be research in software engineering. Traceability in software documentation is one of the interesting topics to be research further. It is important in software documentation to trace out the flow or process in all the documents whether they depends with one another or not. In this paper, we present a traceability method for software engineering documentation. The objective of this research is to facilitate in tracing of the software documentation.

Keywords: *Traceability, Method, Software Engineering, Software Documentation.*

1. Introduction

Traceability can be described as the ability to track and follow processes that links and depends with one another to complete a certain job. It is one of important criteria in software engineering quality factors in which it describes and follows the life of a software objects in term of modeling the relations between software artifacts explicitly [1]. Traceability increases the understandability in complexity of relations and dependencies among objects generated in software development. It is important in facilitates system developers to keep track on their system development from the requirements until the implementations especially on large and complex system.

Furthermore, traceability essential in tracking system flows for maintenance purpose. Issues on traceability have been widely discussed by researchers in software engineering. There are many approaches, method, framework or tool that has been proposed by researchers for solving problems regarding traceability in software engineering areas such as software design and testing. Traceability in software engineering documentation also important so that people involves in the software system can trace everything from the documentation for many purpose such as maintenance.

In this paper, we presented a traceability method for software engineering documentation tracing. The method use is based on a previous research. The remaining of this paper is structured as follows. In section 2, we make a review on related topic of the research. In section 3, we present the methodology. In section 4, we present the experimental setup for software engineering case studies documentation. Finally, in Section 5, we conclude the work.

2. Review on Related Topic

The review is divided into to sections which are:-

2.1) Software Engineering Traceability
2.2) Software Documentation

2.1 Software Engineering Traceability

In the context of software engineering, traceability has been an important aspect of software development that is often required by various professional standards and government agencies [2]. It can be defined as the ability to describe and follow the life of a software object and a means for modeling the relations between software artifacts in an explicit way [1]. Developers use traceability in keeping track their development activities so that any changes made can be track down.

From the review, one research use an approach which is usage-centered traceability process that utilizes UML class diagrams to define traceability strategies for a project and then visually represents trace queries as constraints upon subsets of the model [3]. Visual Trace Modeling Language (VTML) designed to increase the benefits of tracing, through making it more accessible to software developers and other project stakeholders. Their approach utilizes typical UML class diagrams to model trace queries as a set

of constraints enforced onto a division of traceability meta-model.

Research by Jane Cleland-Huang, Jane Huffman Hayes, J. M. Domel (2009) propose a model-based traceability approach to help developers in getting advantages from traces they develop and also planning trace strategies in a graphical modeling environment [4]. The approach proposed consists of four different layers which are strategic layer, document management layer, stored query layer and executable layer. All of the elements for this approach are represented in XML. The domain for demonstrating this approach is a traffic simulator project that composed of requirements, UML class diagrams, code, test cases and test case results.

In another research, they presented a rule-based approach to support the automatic generation of traceability relations between documents which are different parts of the requirements documentations of a software system and analysis object models (AOM) that specify the main entities in the application domain of a system [5]. The generation is done based on two traceability rules that are RTOM rules and IREQ rules. This approach is evaluated by implementation of a prototype traceability tool that implemented in Java and uses the XML parser for Java. They used the metrics of precision and recall for the traceability evaluation which are different from this research.

Another review is made on a research that presents an approach to customize traceability to the situation at hand [1]. This approach is done by scope the traces to be maintained to the activities that stakeholders must do. Core traceability paths that link activities to be supported are defined. They use the domain of software product line (SPL) engineering to test the approach. This approach has been selected in the research as the software engineering traceability method for software engineering documentation tracing.

2.1 Software Documentation

Software documentation has been one of the important things in software development. It may be described as any artifact intended to communicate information on the software system [6]. Documentation has been well-known on the list of recommended practices to improve development and help maintenance [7]. It has been one of the oldest recommended practices. There have been many problems regarding documentation such as nonexistent or of poor quality [8, 9]; over abundant and without a definite objective [10] and tracking of process or flow between various documents that depends with one another.

3. Methodology

The method used in software engineering documentation tracing is based on a previous research by [1] that used the example of collaborative writing (CW) in software product line (SPL) engineering domain. There are three SPL engineering traceability issues that have been presented by them which are traceability links between feature and structural models, traceability links between SPL level and product level models and lastly traceability links between generic and concrete structural models. To be more specific, the description of each issue is in Table 1. They presented the framework of the tracing which is shown in Figure 1. From the framework, the tracing is done from feature model to structural model (a1 and b2) and from SPL level and Product level (a2 and b1).

Table 1: Description of SPL engineering traceability issues

Issue	Description
1	Clear documentation of the dependency between feature model of SPL and products.
2	The documentation of where SPL commonality is realized in the different products.
3	Documentation of the generic models embodied by the concrete structural solutions develops for the product line.

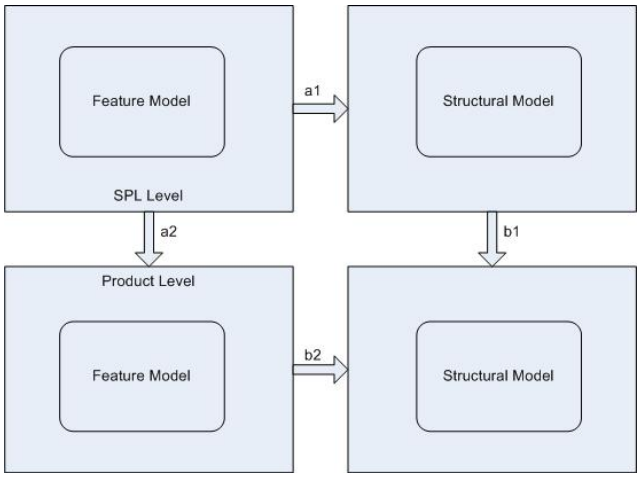

Fig. 1 Framework of previous research.

Based on the issues, we proposed the traceability metrics and attributes that are significant to our research for verifying and validating the tracing result. The proposed metrics are shown in Table 2 while the attributes are shown in Table 3. We take the concept of the tracing from the framework and modify it to suit in our own domain. The modified framework is shown in Figure 2. From the modified framework, the tracing is done from Document 1 to Document 2, Document 2 to Document 3 and Document 1 to Document 3.

Table 2: The proposed traceability metrics

Metric	Description
Dependency	Dependency of keyword from one document to another.
Commonality	Similarity of keyword within a document or from one document to another.
Structural Solution	Standard documentation followed.

Table 3: The proposed attributes

Attribute	Description
Number of document used	Total number of document used for tracing.
Number of document linked	Total number of document that linked with one another.
Dependency of document	Percentage of dependency between documents.
Number of keyword used	Total number of keyword used for tracing.
Number of common keyword	Total number of common keyword used in each document.
Keyword linked	Percentage of keyword that linked between documents.
Categorization of document	Type of document.
Scope of document	Total range cover by document following standard documentation.
Document compliance	Document compliance percentage following standard documentation

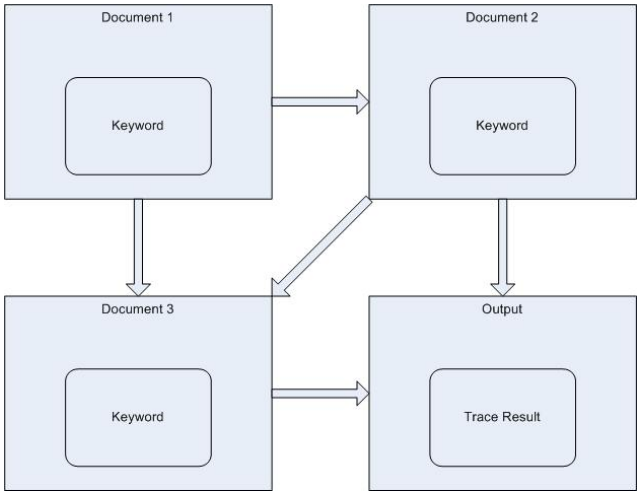

Fig. 2 The modified framework.

4. Experimental Setup

The experiment is set up to test and validate the method that we proposed. We use two software engineering case studies which are e-Perubatan and e-Exchange system. These two systems is a web-based application which developed for in-house usage. We use two documents (Software Requirement Specification and Software Detail Design) and source code files from the two case studies for our experiment. The illustration of the tracing following the modified framework is shown in Figure 3.

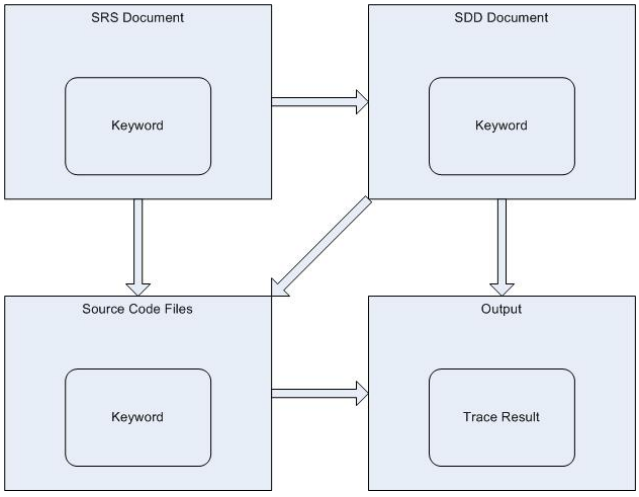

Fig. 3 Illustration of the tracing.

First of all, we need to determine what we want to trace. We called the thing that we want to trace as keyword. The keywords that we have determined will be traced down manually in SRS document first. Then we trace them in SDD document and source code files. The result of the tracing will show the total number of each keywords traced in each documents. Then we proceed we automated tracing using a trace tool that we developed.

The results from manual and automated tracing are compared by calculating the percentage of tracing and the percentage of differential rate between them. Table 4 shows the tracing result for e-Exchange SRS document and Table 5 shows the tracing result for e-Exchange SDD document.

Table 4: Tracing result for e-Exchange SRS document

KEYWORD	SRS		TRACING %	DIFFERENCE %
	MANUAL TRACE	AUTO TRACE		
User ID	8	8	100%	0%
Password	28	28	100%	0%
IOCA	63	63	100%	0%
student	97	96	99%	1%
Administrator	17	16	94.1%	5.9%
Application	42	42	100%	0%
Exchange Program	21	20	95.2%	4.8%
update	8	8	100%	0%
delete	3	3	100%	0%
insert	1	1	100%	0%

Table 5: Tracing result for e-Exchange SDD document

KEYWORD	SRS		TRACING %	DIFFERENCE %
	MANUAL TRACE	AUTO TRACE		
User ID	0	0	-	-
Password	12	12	100%	0%
IOCA	19	19	100%	0%
student	54	56	96.4%	3.6%
Administrator	8	8	100%	0%
Application	24	25	96%	4%
Exchange Program	12	12	100%	0%
update	6	6	100%	0%
delete	1	1	100%	0%
insert	1	1	100%	0%

The result of the tracing are verified and validated with the metrics and attributes that we have proposed. Metric 1 verified the dependency of keywords between SRS-SDD, SDD-SC and SRS-SC. Metric 2 verified the similarity of keywords from SRS to SDD to SC. And lastly Metric 3 verified the compliance percentage of SRS and SDD documents with the standard guideline in which we use the IEEE standard guideline. We did not consider source code files for compliance percentage because of the difference programming language used by both case studies.

Figure 4 show the verification result of Metric 1 for e-Exchange SRS-SDD keywords dependency. It shows that most of the keywords have 100% percentage linked between SRS and SDD which mean they are highly dependent between SRS and SDD. Figure 5 show the verification result of Metric 2 for e-Exchange case study. It shows the number of keywords that are the same with the actual keyword we used in the tracing. Figure 6 show the verification result of Metric 3 for e-Exchange SRS and SDD compliance percentage. It shows the compliance percentage of more than 70% which mean they are highly following the standard.

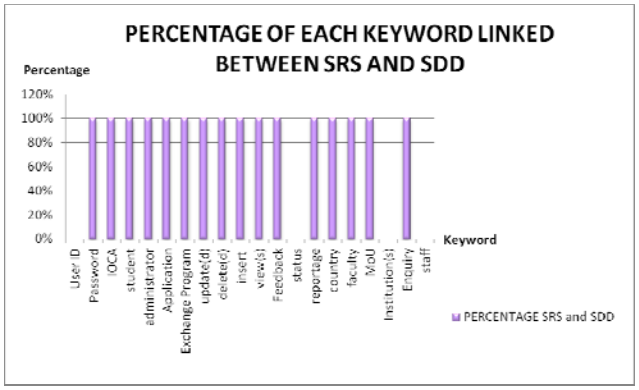

Fig. 4 Verification result for Metric 1.

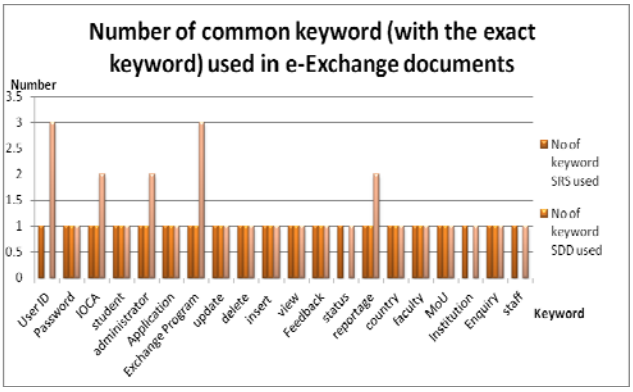

Fig. 5 Verification result for Metric 2.

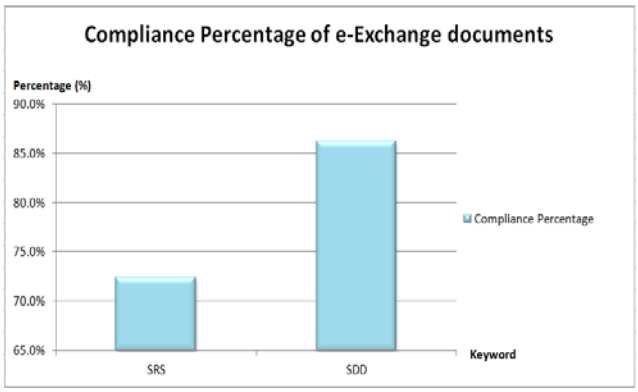

Fig. 6 Verification result for Metric 3.

5. Conclusions

In this paper, we present a traceability method for software engineering documentation. We proposed the traceability metrics and attributes for verifying and validating the tracing result. We use documentation from two software engineering case studies to implement this method using both manual tracing and automated trace tool. Evaluation of the tracing result from the experiment shows an acceptable tracing result. Thus, we are going to implement this method in a different domain which is in food ingredient documentation and evaluate whether we can get the same satisfactory result as in software engineering documentation.

References

[1] P. Lago, H. Muccini, H.v. Vliet, "A scoped approach to traceability management", The Journal of System and Software 82, 2008, pp. 168-182.
[2] H.C. Asuncion, F. Francois, R.N. Taylor, "An End-to-end Industrial Software Traceability Tool", Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 2007, pp.115-124.
[3] P. Mäder and J. Cleland-Huang, "A Visual Traceability Modeling Language", In Proceedings of ACIVS (2). 2010, 226-240.
[4] G. Spanoudakis, A. Zisman, E. Perez-Minana, P. Krause, "Rule-based Generation Requirements Traceability Relations", The Journal of Systems and Software 72, 2003, pp. 105 – 127.
[5] J. Cleland-Huang, J.H. Hayes, J.M. Domel, "Model-based Traceability", TEFSE '09 Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, 2009, pp.6-10.
[6] S C. B. de Souza, N Anquetil, K. M. de Oliveira, "A Study of the Documentation Essential to Software Maintenance", In SIGDOC'05: Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting & Designing for Pervasive Information, 2005,pp. 68-75.
[7] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey", In *DocEng '02:* Proceedings of the 2002 ACM symposium on Document engineering, 2002, pp. 26–33.
[8] L. C. Briand. "Software documentation: How much is enough", In Proceedings of the Seventh European Conference on Software Maintenance and Reengineering (CSMR'03), 2003, pp.13–17.
[9] S. Huang and S. Tilley, "Towards a documentation 74 maturity model", In SIGDOC '03: Proceedings of the 21st annual international conference on Documentation, 2003, pp. 93–99.
[10] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey", In DocEng '02: Proceedings of the 2002 ACM symposium on Document engineering, 2002, pp. 26–33.

Nur Adila Azram is a Master degree student of Department of Information System at Universiti Putra Malaysia. She received Bachelor of Computer Science major in Software Engineering from University Putra Malaysia in 2010. She currently continued her study in Master of Science focusing on Software Engineering at University Putra Malaysia. Her research interests are focusing on software engineering traceability method and food traceability.

Rodziah Atan is a Associate Professor of Department of Information System at Universiti Putra Malaysia. She received her PhD from University Putra Malaysia in 2005. Her field of interest is software process modeling and pursuing for new knowledge in bioinformatics visualization tools, cloud environment and other software engineering related areas especially software design.