Metrics for Software Requirements Traceability based on Output Deployment from Software Quality Perspective

Chan Hoe Kim [1], Jong Bae Kim [2], Jung Won Byun [3]

[1.] Korea Forest Service, Ministry of Agriculture, Korea
chkim@forest.go.kr
[2.] Graduate School of Software, Soong Sil Univ. Korea
kjb123@ssu.ac.kr
[3.] Computer Science and Engineering, Soong Sil Univ, Korea
jimi010327@gmail.com

Abstract: Requirements management is a very important factor in the success of software. Especially, requirements traceability plays important roles in requirements engineering. We propose two metrics to evaluate whether a certain software is implemented in accordance with its requirements or not. Our paper has 16 software projects of several public organizations in South Korea as well as their results of statistical analysis. They are shown that our metrics are helpful to software and project quality.
[Chan Hoe Kim, Jong Bae Kim, Jung Won Byun. **Metrics for Software Requirements Traceability based on Output Deployment from Software Quality Perspective.** *Life Sci J* 2014;11(7):602-606]. (ISSN:1097-8135). http://www.lifesciencesite.com. 83

Keywords: Requirements traceability, Output Deployment, Software and Project quality

1. Introduction

As the scope of software increases, stakeholder is becoming diverse, and knowledge of stakeholder is increasing, and expectation on software is increasing [1]. The scope of software is determined by requirements, and software increases in its scope with many requirements and becomes complex with complicated requirements [2]. As the scope of software increases, the management cost to identify the incorporation of requirements in software increases [3]. Namely, requirements management is a very important factor in the success of software, and its importance further increases as the scope increases.

The requirements traceability is an issue that has been emphasized since the past, and there are various studies and products that are related this [4]. However, requirements are not being traced and managed for various problems. Its examples include the practice of business implementation by non-expert at the site of information project implementation, the need to manage duplicated requirements from multi-stage service development, the theoretical complexity in its practical application, etc. [5].

We propose requirements traceability metrics to evaluate proper application of requirements to software and confirm that software projects in progress are being properly implemented. First of all, it visualizes the requirements deployment for each software development activities and clarify the scope of the method and metrics proposed in this study. Secondly, it presents a method of evaluating the incorporation, namely, deployment of particular requirements in artifacts of next activity. Thirdly, it proposes metrics for determining the deployment level of requirements based on the requirements deployment status.

Based on the case application of 16 public organizations, it shows the applicability of the proposed requirements traceability metrics. In addition, it presents the validity of proposed metrics through the comparison between the values of metrics obtained based on this study and the results of software/project quality evaluation.

2. Related Works

Gotel and Finkelstein [6] defined the requirements traceability as "the ability to explain and identify the life cycle of requirements. There are forward tracing and backward tracing in analysis, design, development, test, and operation". In other words, the requirements traceability refers to checking from requirements elicitation and how they are deployed throughout all stages.

Requirements traceability model that has advanced based on the definition of requirements tracing was established in considering various aspects of requirements. The first aspect is the “source” aspect of considering the changes in requirements themselves. This has advanced into requirements traceability model that becomes the main foundation of software version & configuration management system [7]. Permanency of requirements has been ensured through long-term tracing of the changes in requirements, which has positive effects on the cost

compared to value on the implementation of version & configuration management tool.

Another traceability model has advanced in the "stakeholder" aspect. It is a traceability model based on the assumption that requirements come from the relationship between stakeholder [8]. Such model in the aspect of stakeholder examines from whom requirements are created and utilized, and introduces various attributes of requirements [9].

In engineering perspective, requirements traceability model emphasizes the "process" aspect. In the study [10], requirements refer to the perspective on the procedure through which they are created and utilized subsequently for software to become implemented. This is an important aspect that connects the "source" and "stakeholder" described earlier, and various methodologies in software engineering are reflecting such aspect [11].

Requirements traceability model can be summarized into the "source" aspect model that is the change in requirements themselves, the "stakeholder" aspect model from whom requirements are created and utilized, and the "process" aspect model on the procedure being created and utilized. There is the last aspect that has not been presented specifically even though being mentioned in existing studies. Namely, a specific "artifact" is materialized from "source" through the "process" of various procedures by "stakeholder." The requirements traceability metrics presented in this study is different from the existing studies in the sense that it considers the "artifact" aspect.

3. Metrics for Measuring Requirements Traceability

The forest herbs species in the oak and pine forests belongs to 21 families. The total number of species present in the oak forest and pine forest was 32 and 41, respectively.

This study presents the requirements traceability metrics in the artifact aspect of software requirements. Namely, it will be considered that software requirements and outputs (artifact) that need traceability have already been provided. In addition, a validity of these requirements and output is not included in the scope of this study. From the artifact aspect, requirements are implemented and specified as various outputs through the software life cycle process. In this study, this will be referred to as the 'deployment of requirements'.

3.1 Software Life Cycle and Deployment of Requirements

Software requirements are important data that are created in early stage of software life cycle and affects until the late stage [12]. Namely, this means that they affect the entire stage of software life cycle. Requirements can be seen as the object of improvement or management in each stage of software life cycle.

ISO 12207:2008 [13] presents the standard of software life cycle. It presents the overall road map of software product and service from acquisition, supply and development to operation, maintenance and disposal. Software life cycle process is composed by dividing into system context processes group and software specific processes group. Since the scope of this study is limited to software, processes in which requirements are used are the main target among the processes in the software specific processes group. This standard lists the activities and tasks in each process, and indicates main input & output. In this standard, software requirements are created through software requirements analysis process. Every requirement needs to be assigned as software component in software architectural design process, which is specialized through software detail design process and developed into artifact through software implementation process. The following software qualification testing process checks the requirements fulfillment by artifact created earlier to finalize the completion of software. Namely, software requirements affect the entire process from design and implementation to test after being created, and it is important to trace and manage them.

Based on this perspective, requirements can be deployed in various activities, and the level of incorporation is gradually specializing. Effects of requirements on other output could be indirect or direct. For example, requirements have direct effects in design stage. However, requirements have indirect effects (by constructing the output of the design stage) in implementation stage. Lastly, requirements have indirect effects (by testing the output of the implementation stage) in test stage, and have direct effects (that could not be incorporated in the implementation stage).

If a requirement does not have effects in any of the design, implementation and test stages, it is thought that this requirement has not been deployed. For requirements that have not been deployed, follow-up tasks are needed. As an example of such tasks, previous stage needs to be performed once again or it should be specified that the requirement will not be deployed in this release.

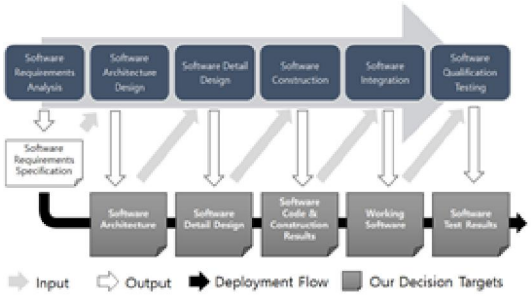
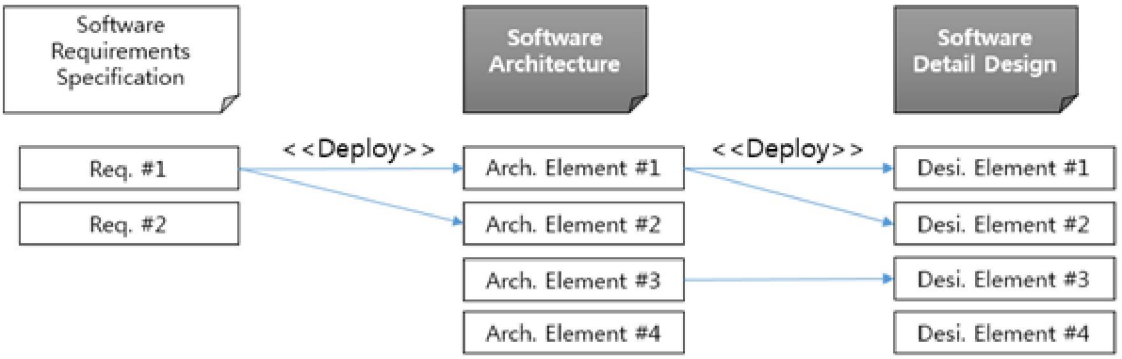


Figure 1. Requirements Deployment Concept & Deployment Decision Input

In the case of code created through implementation stage, however, it could be indirectly affected by requirement (indirectly to the output of design stage). Namely, tracing of reference point is needed. For example, if R1 has been deployed in the design document D1 and D1 has been deployed to the implementation code C1, it means that R1 has been deployed as C1. Namely, for at least two outputs of A and B (A is called input and B is called output based on creation time), the case in which an item of input A is connected to the items of output B for at least 1 is referred to as having been deployed.



Requirements	Status
Req. #1	Arch. #1 Arch. #2 Desi. #1
Req. #2	-

Arch. Element	Status
Arch. #1	**Req. #1** Desi. #1 Desi. #2
Arch. #2	**Req. #1**
Arch. #3	Desi. #3
Arch. #4	-

Desi. Element	Status
Desi. #1	**Req. #1** Arch. #1
Desi. #2	**Req. #1** Arch. #1
Desi. #3	Arch. #3
Desi. #4	-

Figure 2. Example of Requirements Deployment Status

3.2 Requirements Deployment Decision Method

For the purpose of deciding the effects of requirements in the design, implementation and test stages, stakeholder of each stage needs to record the deployment of requirements. Such process of referred to as requirements tracing, and requirements deployment status needs to be recorded through the tracing technique limited to the focus of this study. An example of deployment status is as shown in [Figure 2]. In this deployment status, each requirement and output attribute needs to be included. Cross reference becomes possible through the deployment status attribute.

Deployment decision can be determined very simply. It is to check the record status of the reference point of other output in the status attribute.

Arch. Element	Status	Is Deploy?
Arch. #1	**Req. #1** Desi. #1 Desi. #2	O (Req. #1)
Arch. #2	**Req. #1**	O (Req. #1)
Arch. #3	Desi. #3	X
Arch. #4	-	X
Desi. #1	**Req. #1** Arch. #1	O (Req. #1)
Desi. #2	**Req. #1** Arch. #1	O (Req. #1)
Desi. #3	Arch. #3	X
Desi. #4	-	X

Figure 3. Deployment Status Decision

3.3 Proposal of Metrics for Displaying Requirements Traceability

In this study, two metrics are proposed for displaying the requirements traceability. The first

metric is requirements deployment level for measuring the effects of a requirement on various outputs. The second metric is requirements deployment density for specifying the deployment of a requirement in specifics.

$$f_{d\text{-}count}(A_{P1} \mid A_{Pn}) = Count\ (\ fd(A_{P1}^{x}, A_{Pn\text{-}1}) \mid \sum fd(A_{P1\text{-}x}, A_{Pn\text{-}m}) == true\ ),$$

However, $fd(A_{P1}^{x}, A_{Pn\text{-}k}) = \{true, false\}$,
this indicates that item A_{P1}^{x} (requirement) in the order of x of output A_{P1} (requirements specification) in the P1 stage has been deployed to output $A_{Pn\text{-}k}$ in the order of k of output A_{Pn} in the Pn stage.

M indicates the number of every output of Pn.
In $\sum$, $m = 1 \sim M$,
$A_{Pn\text{-}m}$ indicate output in the order of m of A_{Pn}.

3.3.1 Requirements deployment level

The level of association of items included in input to output needs to be identified. Since the every item input needs to be deployed to output, this deployment level will be expressed as a certain percentage (0 ~ 1), and values close to 1 indicate that every item of input has been deployed to output. This can be expressed through the following [Formula 1].

$$Deployment\ Level\ (A_{p1}, A_{p2}) = f_{d\text{-}count}(A_{P1} \mid A_{P2}) / f_{t\text{-}count}(A_{P1})$$

However, A is an artifact, A_{Pn} is an artifact in n^{th} activity Pn, (e.g.) analysis, design, implementation, test, etc.

$f_{t\text{-}count}(X)$ is the total number of items of artifact X (e.g.) $f_{t\text{-}count}$(requirements specification) is the number of included requirements

$f_{d\text{-}count}(X \mid Y)$ is the number of items included in artifact X designated as reference point to artifact Y
(e.g.) $f_{d\text{-}count}$(requirements specification | GUI specification) is the number of requirements specified with requirements (items of requirement specification) in the reference point of each item of GUI specification.

Since this study presents requirement traceability evaluation technique, X is fixed as requirements specification. In the case of P1, it is restricted as the analysis stage of making requirements specification. In the Pn stage of inputting requirements specification, however, various outputs can exist. Namely, APn indicates that not one output but various outputs can exist. Since the concept of deployment metric is to determine the deployment of requirements, formula fd-count(AP1 | APn) is expanded as follows in this case.

[Formula 2] shows the "deployment level" of input associated with every output at a particular stage. The range is 0 ~ 1 and values close to 1 indicate that every artifact (input) of P1 has been deployed to every artifact (output) of stage P2. For example, the result of 0.97 from computing the deployment metric of stage P2 for 100 requirements indicates that 97 requirements have been deployed and specified to stage P2. In other words, it shows that 3 requirements have not been deployed.

3.3.2 Requirements Deployment Density

Considering various situations of deployment, there are limitations in measuring the tracing level with deployment level only. One of the limitations is a case in which a requirement is deployed to multiple outputs of the next stage. In such case, measuring with deployment level alone will lead to deployment of requirements to multiple outputs, which cannot be reflected by deployment level.

For the purpose of improve such situation, concept of density of the requirements deployment is introduced. In this study, 'deployment density' refers to the deployment metric of particular requirements as outputs in subsequent stage. It can be expressed as [Formula 3].

$$Deployment\ Density\ (Rx, Pn) = f_{a\text{-}count}(Pn \mid Rx) / f_{a\text{-}count}(Pn)$$

However,
Pn is a particular stage,
Rx is particular requirement,
$f_{a\text{-}count}(Pn)$ is the number of outputs in stage Pn,
$f_{a\text{-}count}(Pn \mid Rx)$ is the number of outputs deployed with Rx in stage Pn

The range of deployment density is above 0, and Deployment Density (Rx) = 0 indicates that requirement Rx has not been deployed, and 1 indicates that requirement Rx has been deployed to one output. For example, the result of 2 from computing the deployment density of requirement R1 at stage Pn indicates that R1 has been deployed in two outputs out of the entire outputs of Pn.

The mean of the deployment density of every requirement is called "mean deployment density". For example, the result of 1.41 mean

deployment density at stage Pn with 3 outputs on 100 requirements indicates that a requirement is deployed to 141% of output items of Pn (1.41 items) in general.

4. Conclusion

This study proposed requirements traceability level measurement metrics from output centered perspective. The following limitations were identified while conducting this study. The proposed traceability metrics was found to have correlation with software quality, but specific plan to improve software quality has not been presented. Namely, the issue of requirements without traceability can be presented, but it has not presented a solution to such issue.

As for future study, the limitations mentioned earlier need to be resolved. First, it is necessary to present a solution when traceability issue has been identified and reveal that this can be helpful in improving traceability and software quality. Second, it is necessary to ensure the generality of the proposed study result through case studies on non-public projects.

Corresponding Author:
Prof. JongBae Kim
Graduate School of Software
SoongSil University
Seoul, Korea
E-mail: kjb123@ssu.ac.kr

References

1. Pillai, K and Sukumaran Nair, V.S., “A Model for Software Development Effort and Cost Estimation”, IEEE Trans. on Software Eng., Vol.23, No.8, pp.485-497, 1997.
2. Karl Wigers, Software Requirements, Microsoft Press, 2003.
3. Lagerström, Robert, et al. "Identifying factors affecting software development cost and productivity." Software Quality Journal, Vol.20, No.2, pp.395-417, 2012.
4. Carrillo de Gea, Juan M., et al. "Requirements engineering tools: Capabilities, survey and assessment." Information and Software Technology, Vol.54, No.10, pp.1142-1157, 2012
5. Torkar, Richard, et al. "Requirements traceability: A systematic review and industry case study." International Journal of Software Engineering and Knowledge Engineering, Vol.22, No.3, pp.385-433, 2012.
6. O. Gotel and A. Finkelstein, Analysis of the Requirements Traceability Problem,º Proc. First Int'l Conf. Requirements Eng., pp. 94±101, 1994.
7. R. Conradi and B. Westfechtel, Version Models for Software Configuration Management, ACM Computing Surveys, vol. 30, pp. 232-282, 1998.
8. E. Yu and J. Mylopoulos, Understanding 'Why' in Software Process Modeling, Analysis and Design, Proc. 16th Int'l Conf. Software Eng., pp. 159-168, 1994.
9. Seo Jung-ho, Study on the Priority Selection and Classification Guideline of Requirement Based Output Properties, Soongsil University, Master's Degree Thesis, 2009.
10. K. Pohl, Process Centered Requirements Eng. Somerset, U.K.: John Wiley Research Studies Press Ltd., 1996.
11. Software Engineering Research Team, MaRMI-III Development Methodology, Electronics and Telecommunications Research Institute, 2003
12. Karl Wiegers, “Software Requirements”, MS Press, 2003.
13. ISO/IEC 12207, International Standard Organization, 2008.
14. Roger S. Pressman, Software Engineering – A Practitioner’s Approach, McGraw-Hill, 2009.
15. National Information Society Agency, Electronic Government Project Quality Management Manual, Ministry of Public Administration and Security, 2013.

7/1/2013