International Journal of Advanced Smart Convergence Vol.9 No.4 167-172 (2020)
http://dx.doi.org/10.7236/IJASC.2020.9.4.167

IJASC 20-4-21

Current issues on Requirement Traceability Mechanism for Software Organization of the 4th Industrial Revolution

Janghwan Kim[1], R. Young Chul Kim[2†]

[1] M.S., Software Engineering Laboratory, Department of Electronics and Computer Engineering, Hongik University, Korea
[2] Professor, Software Engineering Laboratory, Department of Software and Communication Engineering, Hongik University, Korea
{janghwan[1], bob[2]}@selab.hongik.ac.kr

Abstract

In the 4th industrial revolution, there are many projects for diverse software applications of smart city environments. Most of the stakeholders focus on considering software quality for their developed software. Nobody doesn't guarantee requirement satisfaction after complete development. At this time, we can only work on user acceptance testing for requirement satisfaction on frequently changing requirements. Why keeps the requirement traceability? This traceability is to identify risks related to requirements, to assure correct software development based on customer requirements. To solve this, we are researching how to implement requirement traceability across each artifact's relationship to each activity of a whole development lifecycle.

Key words: *Requirement traceability, requirement satisfaction, customer requirements, frequent requirement changes*

1. Introduction

Currently, there are the needs for diverse and huge software in the 4th industrial revolution and smart military software area. Until now, many researchers have focused on how to make requirement specifications rightly, how to do correct design and build software Architecture, how to use software development tools, and how to test with test tools. But still, we have a problem with software quality. How to improve software quality? Most of the researchers tried to enhance software maturity with authentication such as Software Process (SP), SPICE, and CMMI, to adapt software process and methodology for development, to improve quality through Good Software (GS), TMMI, Test Process, and Testing tools, to reduce errors with static and dynamic analysis, and to focus on requirement engineering and design [1]. We are also focusing on identify code with quality indicators by static analysis and software visualization based on reverse engineering. One of the more important things of quality issues is the software process. Therefore we consider both forward and backward ways for the software development lifecycle, and also need to have relationships across activities and artifacts at all phases of the lifecycle. Current problems have that 1) nobody provides a standard guideline of traceability, 2) traceability tools cannot trace artifacts of all activities even if there exist some tools such as RTM,

Manuscript Received: November. 20, 2020 / Revised: November. 25, 2020 / Accepted: November. 29, 2020
Corresponding Author: bob@selab.hongik.ac.kr
Tel: +82-44-860-2477
Professor, Software Engineering Laboratory, Department of Software and Communication Engineering, Hongik University, Korea

Teamwork/RQT, RDD-100, STS, and RTrace, and 3) one needs to define the granularity of a traceability unit. Why do we need to have requirement traceability? Ramech and Jarke reported saving 80 % of development costs for systems with the reference model of traceability [4]. But the U.S Department of Defense spends 4% of its IT costs on traceability. In this paper, we provide detailed guidelines for requirement traceability, mention still not enough to use even if diverse tools used in software industries, and describe this traceability model with empirical analysis.

This paper is organized as follows. Chapter 2 describes related works. Chapter 3 mentions the Requirement Traceability Mechanism on Software Process. Chapter 4 shows both Forward and Backward Traceability mechanisms on the Process. Finally, the conclusion is discussed.

2. Related Works

For the first time, in the NATO working conference in 1968, one mentioned the requirements for a methodology of software designs, and also reported on how to ensure that the system is developed based on the original design. In 1976, Boehm mentioned the traceability of future works in software engineering. By the 1980~1990s, they began national and international standards such as DOD-STD-2167A for software and system development [2]. After 1994, Gotel and Finkelstein analyzed requirement traceability, which is the difference between pre- and post- requirements specification traceability. The problem of pre-requirements specification traceability is how to make requirements specification before requirement production rightly [2,3]. Gotel, etc., defined terminology and concept of requirement traceability as “the ability to describe and follow the life of a requirement, in both forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent development and use, and through periods of ongoing refinement and iteration in any way of these phases” [2,3,4]. Ramesh and Jarke [5] mentioned about reference model for requirement traceability, which saved up to 80 percent in development costs for systems. But they have more than 18 types of traceability links for their reference model on empirical studies. Previous traceability models focused on other version and configuration management systems on the documentation management of traceability.

In that time, researchers identified requirement traceability as a quality factor-like nonfunctional requirements.

For U.S Government, MIL-STD-2167A and MIL-STD-498 required the development of requirement traceability documents [2].

Palmer [2,4] mentioned that traceability keeps the relationships across software requirements, design, and implementation, which shows design to meet the requirements and helps identify those requirements against design.

The complete traceability can determine the costs and schedules of requirement changes rather than recognizing the change at all phases. Also, a test procedure can be modified when errors are discovered if traceable to requirements or design [3]. Especially, requirement management is one activity of requirement engineering, which keeps information change without connecting with the late stage. Therefore, requirement traceability keeps a clear relationship between each stage of development with the defined granularity of each artifact based on the context of traceability. Otherwise, stakeholders don’t recognize what’s a problem to develop a system to keep a little mistake from the requirement, analyze them, and to implement a system finally, and how much have error propagation from the requirement, analysis, design, and to implementation stage during the development lifecycle. In figure 1, we show the process of error propagation to work each activity at each phase of the software lifecycle. For example, like in Figure 1, though a customer requests to develop a system with the requirements of creating an ‘a lady’ system, even developers work so much to

implement it with a little design mistake. Testers do exhaustively test the system like an 'a strong man' system. None can guarantee to satisfy the developed system that they can result in complete testing of it. This is why we need across the relationship between them. We suggest the requirement traceability mechanism for clearly recognizing what stakeholders should include in the process. Figure1 shows the software lifecycle without any cross-relationship between each phase, that is, requirement traceability mechanism.

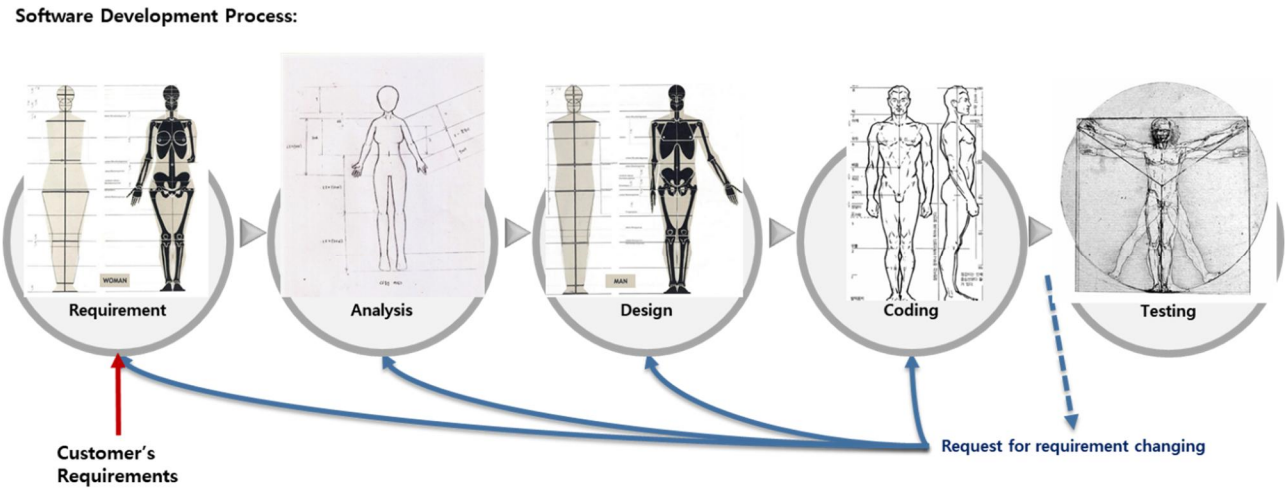


Figure 1. Software Life cycle without Traceability Mechanism

3. Requirement Traceability Mechanism on Software Process

In most software organizations they just use project management, software engineering development tool, requirement management, system integration, configuration management, maintenance tools for software management. But we argue about keeping requirement traceability on software process.

When considering requirement traceability, we first consider as follows: 1) what kind of granularity of each artifact at each stage, 2) what kind of relationships such as the 1-to-1, the 1-to-many, the many-to-many relationship between each stage, and 3) how to keep both forward and backward traceability based on the process. Figure shows the whole relationships between artifacts of each stage.

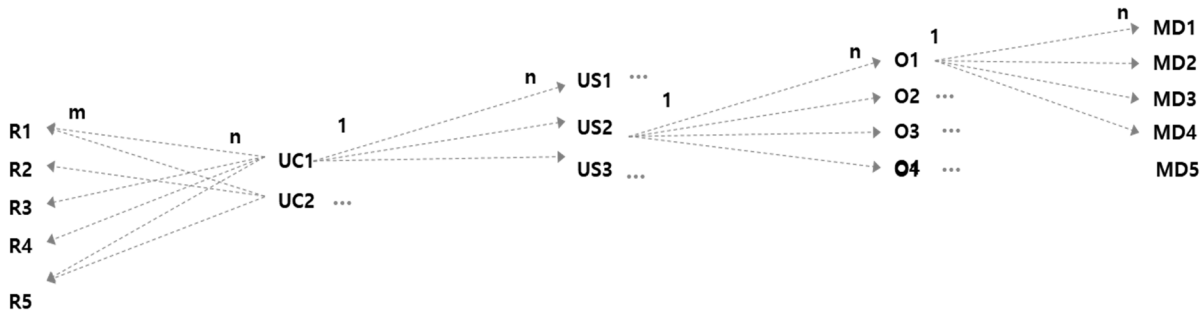


Figure 2. The whole Relationships between Artifacts of each stage

Therefore, we suggest how to make traceability mechanism on object-oriented software engineering environment as follows:

1) Granularity: in the development aspect, we consider the granularity of requirement, use case, use case scenario, object, and methods.

2) Across relationships: in the software development lifecycle, we recognize to have many-to-many relationships between requirement and use cases, and also one-to-many relationships between other stages.

In figure 3, we consider requirement traceability keep a clear relationship between each stage of development with the defined granularity of each artifact based on the context of traceability as follows:

1) Requirement Documents: define user requirements, which have a detailed functional requirement document.

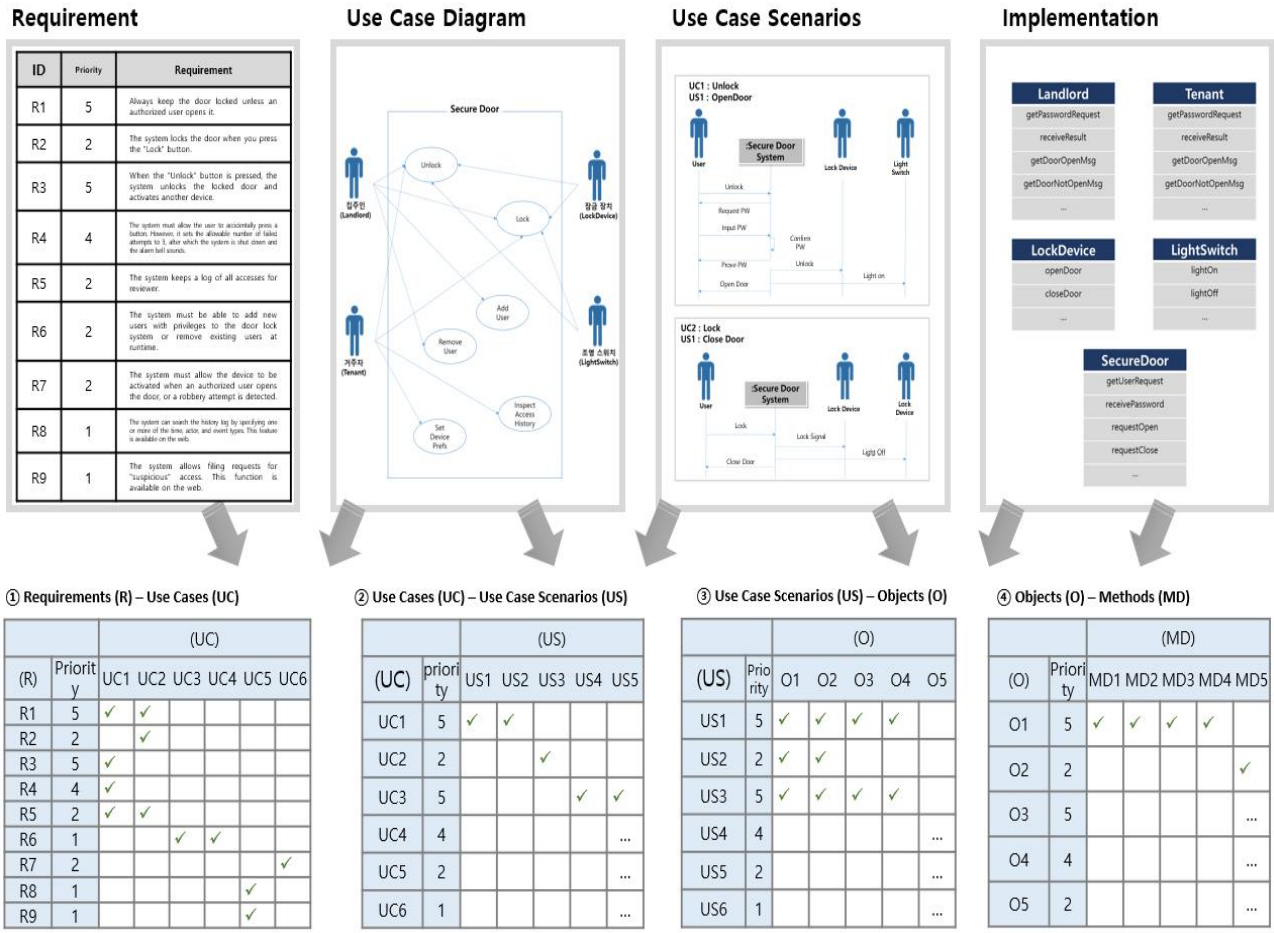
2) Use cases: define a result of the high-level analysis that shows a complete path of events in the system. The actor may be external entities which interact with the system

3) Use case Scenarios: show a detailed use case that represents all possible scenarios for functional requirements.

4) Objects: define to be in a particular use case scenario. The object should specify the name, attributes, and methods.

5) Methods: define to be in a particular object.

Figure 3 shows the whole relationships between artifacts and matrices.



		(UC)					
(R)	Priority	UC1	UC2	UC3	UC4	UC5	UC6
R1	5	✓	✓				
R2	2		✓				
R3	5	✓					
R4	4	✓					
R5	2	✓	✓				
R6	1			✓	✓		
R7	2						✓
R8	1					✓	
R9	1					✓	

		(US)				
(UC)	priority	US1	US2	US3	US4	US5
UC1	5	✓	✓			
UC2	2			✓		
UC3	5				✓	✓
UC4	4					...
UC5	2					...
UC6	1					...

		(O)				
(US)	Priority	O1	O2	O3	O4	O5
US1	5	✓	✓	✓	✓	
US2	2	✓	✓			
US3	5	✓	✓	✓	✓	
US4	4					...
US5	2					...
US6	1					...

		(MD)				
(O)	Priority	MD1	MD2	MD3	MD4	MD5
O1	5	✓	✓	✓	✓	
O2	2					✓
O3	5					...
O4	4					...
O5	2					...

Figure 3. The whole Relationships between Artifacts and Matrices

For a project, we should first consider clearly defined requirements, and prioritize them. When the developers consider developing the system, they may prioritize developing requirements. As we assign requirements to the order of priority in the traceability matrix, we can confirm the first development of requirements with this prior

information.

4. Forward and Backward Traceability

For forward and backward traceability, we suggest our proposed requirement traceability matrix mechanism. In this matrix, we just click a spot between requirement (R) and use case (UC), and then make both links on the requirement traceability matrix mechanism. Then we click a spot between use cases (UC) and use case scenarios (US), click a spot between use case scenarios (US) and objects (O), and also click a spot between objects (O) and methods (MD). Then automatically create both link relationships for the forward and backward traceability on software process in Figure 4.

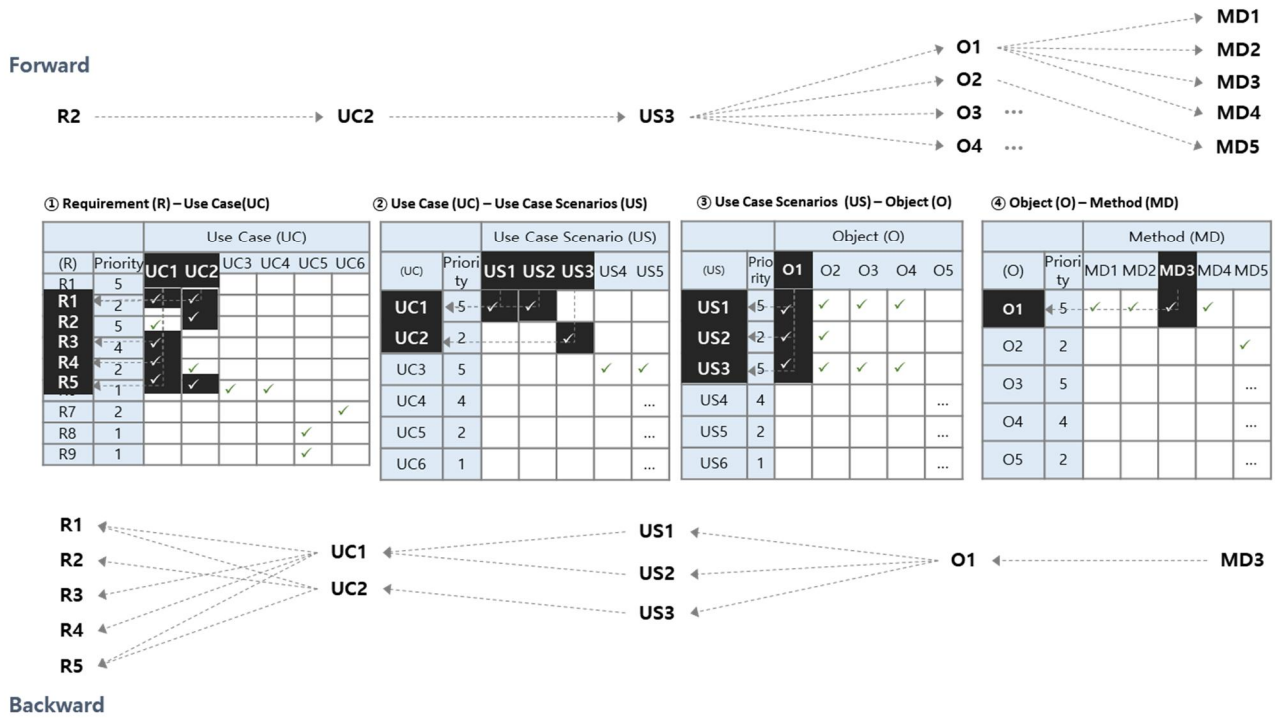


Figure 4. Both Forward and Backward Traceability

With this matrix mechanism, we guarantee the satisfaction of requirements for customer requirements and frequent changes. We can give stakeholders confirm completely to develop a system based on requirement specifications. We are still developing our automatic requirement traceability. Figure 4 shows both forward and backward ways of traceability mechanism.

5. Conclusion

In the 4th industrial revolution, there are many projects for diverse software applications of smart city environments. Most stakeholders focus on considering software quality for their developed software. Nobody doesn't guarantee requirement satisfaction after complete development. At this time, we propose a way of completely satisfy customer requirements on frequently changing. Therefore, we keep the requirement traceability to identify risks related to requirements, to assure correct software development based on customer requirements. We are researching how to implement requirement traceability across each artifact's relationship on each activity of a whole development lifecycle. Shortly, we will develop an automatic requirement

traceability mechanism for the software process.

Acknowledgement

The research was supported by the Ministry of Trade, Industry and Energy (MOTIE), KOREA, through Education Program for Creative and Industrial Convergence (Grant Number N0000717) and supported by the BK21 plus program through the National Research Foundation (NRF) funded by the Ministry of Education of Korea.

References

[1] B. Park, G. Yi and R. Kim. “Effective Requirement Analysis Method based on Linguistic and Semantic Textual Analysis.” The Journal of The Institute of Internet, Broadcasting and Communication (IIBC) Vol. 17, No. 6, pp. 99-103, Dec. 31, 2017. DOI: https//doi.org/10.7236/JIIBC.2017.17.6.97

[2] O. Gotel, J. Cleland-Juang et al. "Traceability Fundamentals”, *Software and system Traceability*, Springer-Verlag London, pp. 4-22, 2012

[3] Orlena C. Z. Gotel, and Anthony C. W. Finkelstein, “An Analysis of the Requirement Traceability Problem,” IEEE, pp. 94-101, 1994. DOI: https//doi.org/10.1109/ICRE.1994.292398

[4] B. Ramesh, and M. Jarke, “Toward Reference Models for requirements Traceability”, IEEE Transactions on software engineering, Vol. 27, Vol. 27. No.1, pp. 58-93. DOI: https//doi.org/10.1109/32.895989

[5] A. Wibowo, and Joseph Davis, “Requirements Traceability ontology to support requirements management,” Proceeding of the Australasian Computer science week Multi-conference, Melbourne, Australia, February 2020. DOI: https://doi.org/10.1145/3373017.3373038