Content-based Configuration Management System for Software Research and Development Document Artifacts

Dusan Baek[1], Byungjeong Lee[2] and Jung-Won Lee[1]
[1] Department of Electrical and Computer Engineering, Ajou University
Suwon, South Korea
[e-mail: whitedusan@gmail.com, jungwony@ajou.ac.kr]
[2] Department of Computer Science and Engineering, The University of Seoul
Seoul, South Korea
[e-mail: bjlee@uos.ac.kr]
*Corresponding author: Jung-Won Lee

Received October 16, 2015; accepted February 18, 2016; published March 31, 2016

Abstract

Because of the properties of software such as invisibility, complexity, and changeability, software configuration management (SCM) for software artifacts generated during software life-cycle has been used for guarantee of the quality of the software. However, the existing SCM system has only focused on code artifacts and software development document artifacts such as Software Requirements Specification (SRS), Software Design Description (SDD), and Software Test Description (STD). Moreover, software research-oriented project comes out late the code artifacts and the software development document artifacts. Therefore, there is a need for trace and management of software research document artifacts composed of highly abstracted non-functional requirements like ‘the purpose of the project’, ‘the objectives’, and ‘the progress’ before generation of the code artifacts and the software development document artifacts for a long time. Nevertheless, the existing SCM system cannot trace and manage them. In this paper, we propose content-based configuration management system comprised of the relevance link generation phase and content-based testing phase to trace and manage them. The preliminary application results show applicability and feasibility of the proposed system.

Keywords: Document Artifact, Relevance Analysis, Software Research and Development Document Artifact, Software Configuration Management, Traceability

A preliminary version of this paper was presented at APIC-IST(Asia Pacific International Conference on Information Science and Technology) 2015, July 13-15, 2015, Vietnam, and was slected as an outstanding paper. This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014M3C4A7030504).

http://dx.doi.org/10.3837/tiis.2016.03.027 **ISSN : 1976-7277**

1. Introduction

Recently, the size of a software research and development (R&D) project has expanded, and the importance of its quality has increased. For this reason, the management of artifacts produced during the software R&D life cycle is becoming more significant. Managing the artifacts leads researchers to improve the quality of their software R&D project [1-3].

In order to manage the artifacts, researchers utilize a software configuration management (SCM) system which identifies, traces, and controls configuration items in the artifacts [4-6]. Most of the SCM systems evaluate and manage reflection of the requirements in code artifacts and software development document artifacts such as Software Requirements Specification (SRS), Software Design Description (SDD) and Software Test Description (STD) [7-9]. However, the software R&D project like a project aiming original technology generates late the software development document artifacts and the code artifacts. Therefore, before they are come out, it is necessary to trace and manage highly abstracted non-functional requirements like 'the purpose of the project', 'the objectives', 'the progress', and 'the performance history' in research-related documents for a long time. Even so, it is hard that the existing SCM system traces and manages them properly.

In order to figure the reason out, we classify the widely used artifacts among software R&D artifacts into the software research document artifact (e.g. project proposal, annual report, monthly statement, etc.), the software development document artifact (e.g. SRS, SDD, STD, etc.), and code artifacts. After that, we apply existing SCM methods to them. **Fig. 1** shows the SCM methods by software R&D artifacts.

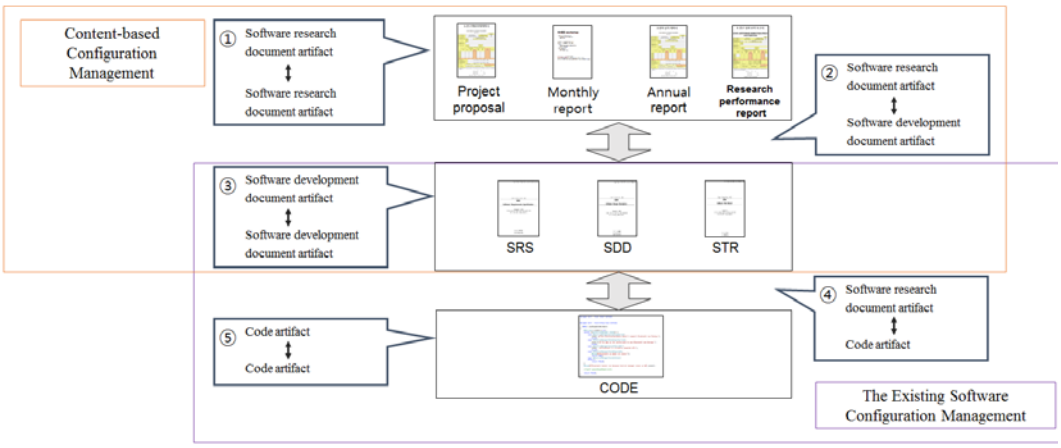


Fig. 1. The SCM methods by software R&D artifacts

The existing SCM system has used three types of comparison methods for SCM. First, it has traced and managed configuration items including the same requirement in different software development document artifacts by comparison between the software development document artifacts (③). Second, it has verified that the requirements in the software development document artifacts are reflected in the code artifact properly by comparison between the software development document artifacts and the code artifacts (④). Third, it has traced modified state, managed software version and dependency, and controlled the changes of

functional or physical characteristics by comparison between the code artifacts(⑤). However, the existing SCM system only considers the specified and well-formed requirements in software development document artifacts or code artifacts. Thus, it cannot manage and trace highly abstracted and non-functional requirement in the software research document artifact through comparisons between the software research document artifacts (①), or the software research document artifact and the software development document artifact (②) .

There were studies for SCM using the comparison between document artifacts [10-15]. The studies focused on identification and trace of configuration items in the software document artifacts. The studies used traceability matrix which was made manually for identifying and tracing configuration items. Moreover, they applied information retrieval techniques to automate them. Nevertheless, in common with the existing SCM system, they only took into account the specified and well-formed requirements. Therefore, they could not cover the SCM for the software research document artifacts structured as highly abstracted non-functional requirements.

The more R&D project aims original technology, the research period takes relatively long time compared to the development period, so the software development document and code artifacts generate late, which makes it difficult to predict the outcome of the R&D project. Due to this characteristics, the SCM for the software research document artifacts is more important in software R&D project. However, the existing SCM system is not able to manage them.

Thus, in this paper, we report the underlying reasons why the existing SCM system cannot manage the software research document artifacts, and propose a content-based configuration management system to manage them. The proposed system comprised relevance rule generation phase and content-based testing phase traces and manages not only the software development document artifact but also the software research document artifact.

The remainder of this paper is organized as follows. Section 2 reviews some related work for SCM using a comparison between document artifacts. Section 3 analyses characteristics of the software R&D document artifacts and the underlying reasons why software research document artifacts cannot be managed by the existing SCM system. Section 4 explains content-based configuration management system for SCM of software R&D document artifacts. Section 5 show the preliminary application results. Finally, Section 6 discusses the conclusion and future works.

2. Related Work

As mentioned previously, there were many studies for SCM by comparing software document artifacts.

[10-12] have managed document artifacts for SCM using a traceability matrix. **Fig. 2** shows an example of the traceability matrix. Each requirement in the traceability matrix is identified by unique ID and located in the first row. Types of document artifacts are located in columns, and index of configuration items mapped with the requirement is placed in content fields in the matrix. The traceability matrix can be used to trace the configuration items that have same requirements in different document artifacts. Furthermore, [13-14] has applied information retrieval technique to automate generation of the traceability matrix. Using the traceability matrix has the advantage of easy to use and easy to read by treating simple link in the software document artifacts. However, it is not appropriate to target software research document

artifacts directly because it is for the well-formed document artifacts created based on the requirements.

Requirement	Project proposal	Project performance plan	Usecase	Component	Test ID	Acceptance criteria ID
R1	PP101	PPe101	UC101	PayRule	TE101	AC101
R2	PP102	PPE102	UC102	PayPolicy	TE101	AC301

Fig. 2. A traceability matrix

[15] has proposed a method for tracing the requirement by using structural characteristics of the software document artifacts as shown in **Fig. 3**.

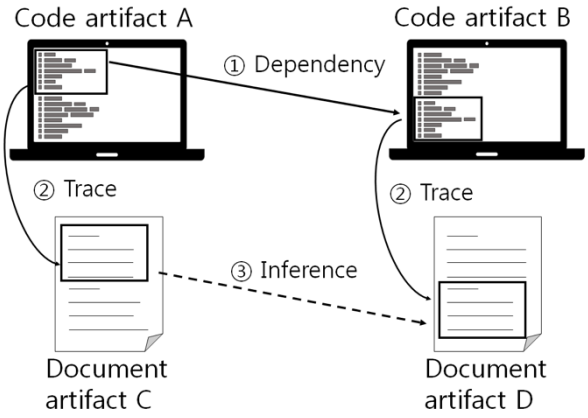


Fig. 3. A traceability link graph between code artifacts and document artifacts

In **Fig. 3**, the code artifact has a dependency on other in the dependency relation (①), and the document artifacts trace the code artifacts in the trace relation named ‘traceability link’ (②). In this situation, [15] has inferred new trace relation (③) between the document artifacts by using the dependency relation (①), the trace relation (②), and the structural information of the document artifacts. These kinds of studies have suggested new approach using the software document artifacts. However, the method is unsuitable for directly applying to the software research document artifacts composed of various formats because it aims the well-formed document artifacts. Moreover, the generation of software code artifacts takes especially long time in the software R&D projects for original technology. Thus, the method which needs software code artifacts for inferring new trace relation has a limitation of SCM of software R&D document artifacts.

In this paper, we propose the content-based configuration management system. The proposed system generates ‘relevance link’, which would manage highly abstracted non-functional requirements in software research document artifacts. The relevance link consists of related two configuration items and the rule which the two configuration items have to follow, and it is used for SCM of the software R&D document artifacts. Before explain about what is the relevance link in detail and how can they manage the software research document artifacts, we analyze characteristics of the software R&D Document artifacts to find out the underlying reasons why the existing SCM system cannot manage them.

3. Analysis of Characteristics of the Software R&D Document Artifacts

In this section, we analyze characteristics of the software R&D document artifacts and the underlying reasons why the existing SCM cannot manage the software research document artifacts.

3.1 Classification of artifacts

We classify the artifacts in order to figure the reason out why the existing SCM cannot manage. **Table 1** is a classification table for the artifact produced during the software R&D life cycle.

Table 1. The classification table for artifacts

Artifact Type	Document Type
Software Development Document Artifact	Software Requirement Specification (SRS)
	Software Design Description (SDD)
	Software Product Specification (SPS)
	Software Test Description (STD)
Software Research Document Artifact	Project Proposal
	Annual Report
	Monthly statement
Software Code Artifact	Source Code

Software artifacts are classified into the code artifacts and the document artifacts. The document artifacts are composed of the software development document artifacts and the software research document artifacts. The software development document artifacts include SRS, SDD, STD, SPS, etc. The software research document artifacts contain project proposal, monthly statement, annual report, etc.

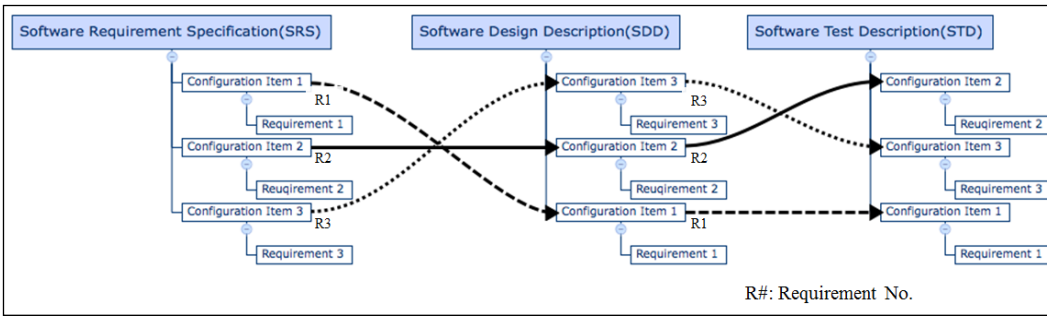
The existing SCM system conducts SCM for the software code artifacts and the software development document artifacts. However, it cannot carry out SCM for the software research document artifacts which are comprised of abstracted items and various formats. The next section explains the reason.

3.2 Differences between Software Research Document Artifacts and Software Development Document Artifacts

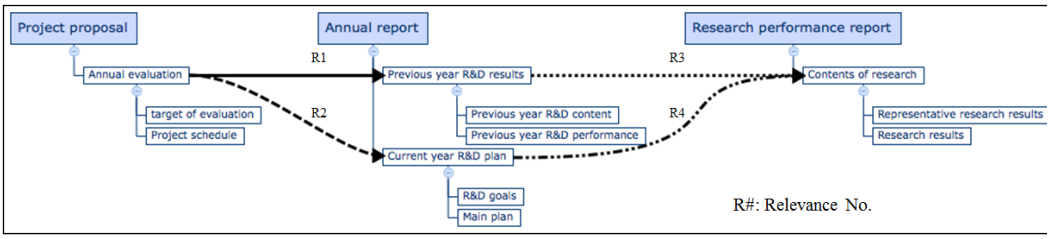
In the case of the software development document artifacts, each item(or section) has been composed based on the requirements, and identified by the identifier based on the requirement as shown **Fig. 4 (a)**. The existing SCM system distinguishes and traces the configuration items having the same requirement in different documents by using the identifier. In **Fig. 4 (a)**, each arrow means the traceability link between the configuration items having the same requirement. And the traceability link has the number to differentiate itself from the others that have different requirements.

On the other hand, in the case of the software research document artifacts, each item(or section) has been comprised based on the understanding of the researcher as shown **Fig. 4 (b)**. Therefore, even if there are configuration items having the same relevance in the different software research document artifacts, it is more important what the relevance means than itself. Because the traceability has just one meaning as itself but the relevance can have various

meanings such as sufficient, necessary and equal. For this reason, traceability link is transitive



(a) The tracability link graph of the software development document artifacts



(b) The relevance graph of the software research document artifacts

Fig. 4. The difference between the traceability link graph and the relevance graph

relation, but relevance is not transitive relation.

The existing SCM system cannot manage the software research document artifacts. To solve the problem, we design the content-based configuration management system.

In this paper, we coin the term as follows.

- Relevance: semantic homogeneity between configuration items in different document artifacts
- Corresponding items: two configuration items having same relevance.
- Relevance rule: the rule which corresponding items have to follow.
- Relevance link: mapping between corresponding items and relevance rule.

4. Content-based Configuration Management System

Content-based configuration management system consists of relevance rule generation phase and content-based testing phase. In relevance link generation phase, a researcher generates relevance links. In content-based testing phase, the researcher conducts content-based testing for SCM of the software R&D document artifact using the relevance links generated from previous phase. In this section, we explain each phase in detail.

4.1 Relevance Link Generation Phase

The relevance link generation phase is composed of four modules, which are template-based document artifact parsing module, configuration item identification module, corresponding items extraction module and relevance link generation module. In this phase, a researcher can make the relevance link of each document artifact type using the document artifacts and

pre-defined relevance rules. **Fig. 5** shows the system architecture in the relevance link generation phase.

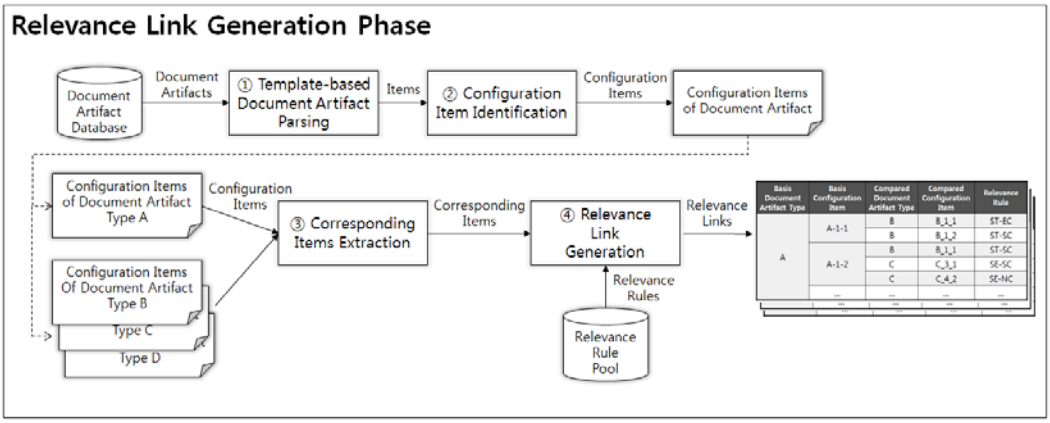


Fig. 5. The system architecture in relevancelink generation phase

Template-based Document Artifact Parsing Module (①): This module has the role of parsing the items used in the configuration item identification module. For this, the module parses document artifacts provided from a document artifact database. Because the items will be used for identification of configuration items, all titles of section and subsection levels are needed, not contents (i.e. template-based parsing). Therefore, the document artifacts having a same structure are handled as an equivalent type, and they don't need to be redundantly parsed.

Configuration Item Identification Module (②): For efficient SCM, configuration items which have to be managed should be extracted and identified from the indiscriminately parsed items (e.g. 'the purpose of the project', 'the objectives', 'the progress', and 'the performance history', etc.). **Fig. 6** is the example of identifying the configuration items. In **Fig. 6**, the items in the left tree chart are extracted and identified to the configuration items in the right tree chart.

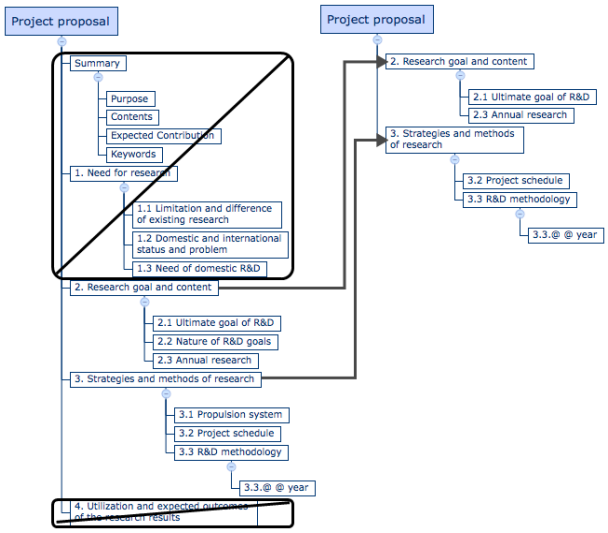


Fig. 6. The example of identifying the configuration items

Corresponding Items Extraction Module (③): Corresponding items are two configuration items having same relevance, and they are extracted for the relevance link. For extracting the corresponding items, in addition to the basis document artifact having the configuration item which a researcher wants to extract, all the compared document artifacts having corresponding items of it are required. Because the relevance link is not transitive relation, one-to-one comparison is essential. Furthermore, since the relevance link is not commutative relation, the researcher has to consider not only the forward comparison, but also the backward comparison. **Fig. 7** is the example of corresponding items extraction by considering the forward comparison of the project proposal as the basis document artifact.

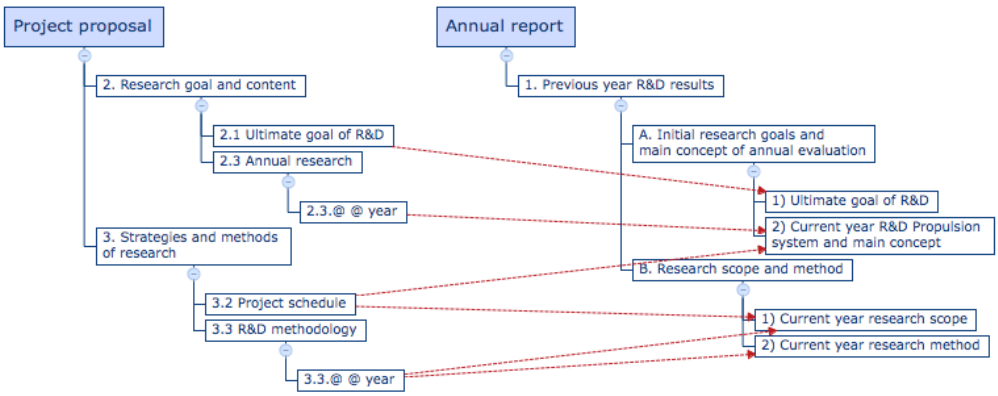


Fig. 7. The example of corresponding items extraction

Relevance Link Geneartion Module (④): As mentioned above, a relevance link is composed of corresponding items and the relevance rule which they have to follow. Thus, after extracting the corresponding items, the researcher assigns the relevance rule to the corresponding items. The relevance rules defined by the results of the previous study, and are shown in **Table 2** [16]. The pre-defined relevance rules can be modified, added or removed during the software R&D life cycle. After the mapping between the corresponding items and relevance rules, generating relevance link is completed, as shown in **Fig. 8**.

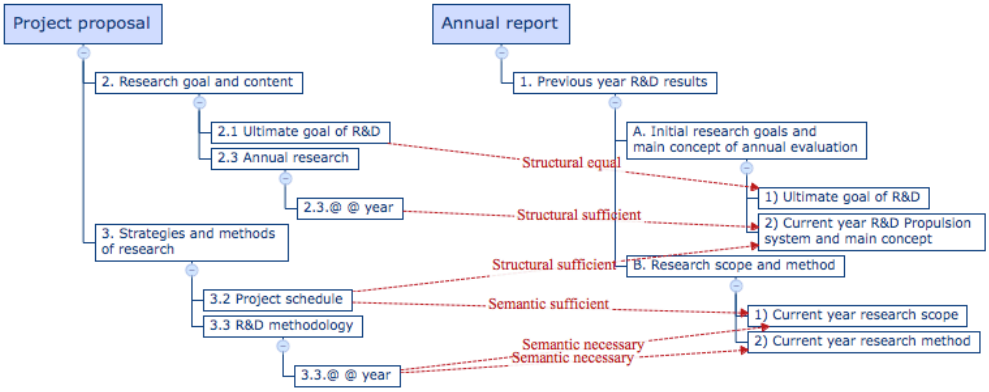


Fig. 8. The example of the relevance link

Table 2. The relevance rules

Relevance Rule	Explanation
Structural ~ (prefix)	Structure of contents in configuration item is very simple, it can be inferred by using simple comparison.
Structural sufficient condition (ST-SC)	All contents in configuration item in the basis document are included in the configuration item in the compared document
Structural necessary condition (ST-NC)	The configuration item in the basis document includes all contents in the configuration item in the compared document
Structural equal condition (ST-EC)	All contents in the configuration item in the basis document are the same as all contents in the configuration item in the compared document
Structural partial equal condition (ST-PEC)	Some contents in the configuration item in the basis document are included in the configuration item in the compared document
Semantic ~ (prefix)	Structure of contents in configuration item is complex, it can be inferred by human or by using an inference algorithm.
Semantic sufficient condition (SE-SC)	Semantically sufficient condition
Semantic necessary condition (SE-NC)	Semantically necessary condition
Semantic equal condition (SE-EC)	Semantically equal condition
Semantic partial equal condition (SE-PEC)	Semantically partial equal condition

4.2 Content-based testing phase

Content-based testing phase is comprised of one module, which is the content-based test module. In this phase, a researcher can test software R&D document artifacts for SCM of them using the relevance link. **Fig. 9** shows the system architecture in the content-based testing phase.

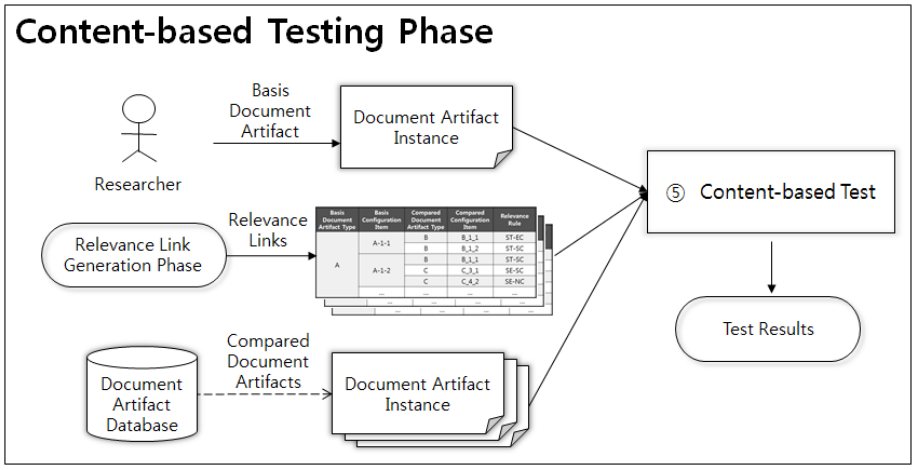


Fig. 9. The system architecture in content-based testing phase

Content-based Test Module (⑤): A researcher inputs the software R&D document artifact which the researcher wants to trace and manage and the relevance rule of it. Then, the module retrieves the compared document artifacts in the document artifact DB storing all the software R&D document artifacts generated during software R&D life cycle. The module tests whether the corresponding items in the basis document artifact and the compared document artifacts keep the relevance rules. In this paper, this process is manually conducted by a researcher. We will expand the module to be automatically tested using the textual inference.

The result of the content-based test provides grounds for determining consistency and completeness of the tested software R&D document artifact. Moreover, the content-based test assists objective evaluation of the software R&D document artifacts.

5. Application

For a preliminary application of the proposed content-based configuration management system, we conduct SCM for 'Project proposal' and '1st annual report' made from a previous real project. We use 'Project proposal' as the compared document, and '1st annual report' as the basis document. Each software research document artifacts are parsed and marshaled by the system. In the preliminary application, we manually fills the corresponding items and relevance rules to make relevance links. After generation of the relevance link, the content-based test module tests contents of the software research document artifacts using the relevance links. For this, the system displays the contents of the corresponding items and the relevance rule, and is fed back whether the corresponding items follow the relevance rule or not by us. After the test is completed for all the corresponding items, it informs the test results, as shown **Table 3**.

Table 3. The result of preliminary application

Basis Configuration Item	Compared Configuration Item	Relevance Rule	Result
AR-1-A-1	PP-2-1	ST-EC	TRUE
AR-1-A-2	PP-2-3	ST-SC	TRUE
	PP-3-2	ST-SC	TRUE
AR-1-B-1	PP-3-2	SE-SC	TRUE
	PP-3-3	SE-NC	TRUE
AR-1-B-2	PP-3-3-1	SE-NC	FALSE

Through the preliminary application, we verify that 5 items in the 1st annual report follow the relevance rules, but 1 item (AR-1-B-2) doesn't follow the relevance rule (SE-NC). AR-1-B-2 item which is about current year research method in the 1st annual report has to include all contents of PP-3-3-1 item which is about 1st year's methodology in the project proposal. However, AR-1-B-2 item omits one plan among the plans in the project proposal.

In this way, the content-based configuration management system can conduct SCM of the software R&D document artifacts which the existing SCM system cannot manage, and it can help the researchers who particularly focus on the original technology.

6. Conclusion

The existing SCM system has carried out the SCM of the software artifacts based on the configuration items which only have one meaning as the requirement. Therefore, it cannot have managed the software research document artifacts which have various meanings such as sufficient, necessary, equal and partial equal. To solve this problem, we propose the content-based configuration management system. It identifies configuration items from software R&D document artifacts, and extracts corresponding items from the configuration items in a basis document artifact and compared document artifacts. Then, it maps between the corresponding items and the relevance rule to generate the relevance link. Finally, it performs the content-based test of the software R&D document artifacts using the relevance link. The results of the content-based test provide grounds for determining consistency and completeness of the tested software R&D document artifact. Moreover, it can assist objective evaluation of the software R&D document artifacts. For the future work, we plan to study how to determine consistency and completeness. In addition, we will expand the content-based test to be automatically tested by using the textual inference.

References

[1] Institute of Electrical and Electronics Engineers, “IEEE Guide to Software Configuration Management,” *An American Standard. IEEE*, 1988. Article (CrossRef Link)
[2] Standard, I. "Systems and software engineering–system life cycle processes." ISO Standard 15288, 2008. Article (CrossRefLink)
[3] Wang, Juite, and Yung-I. Lin, "A fuzzy multicriteria group decision making approach to select configuration items for software development," *Fuzzy Sets and Systems* 134.3, 343-363, 2003. Article (CrossRefLink)
[4] Dart, Suan. “Concepts in configuration management systems,” in *Proc. of the 3rd international workshop on Software configuration management*, pp. 1-18, May, 1991. Article (CrossRefLink)
[5] Gotel, Orlena CZ and Anthony CW Finkelstein, "An analysis of the requirements traceability problem,” *Requirements Engineering, 1994, Proc. of the 1st Int. Conference*, pp. 94-101, April, 1994. Article (CrossRefLink)
[6] Aiello, Robert, and Leslie Sachs. *Configuration Management Best Practices: Practical Methods that Work in the Real World*. Pearson Education, 2010. Article (CrossRefLink)
[7] “IEEE Recommended Practice for Software Requirements Specifications," *IEEE Std 830-1998*, vol., no., pp.1-40, Oct. 20 1998. Article (CrossRefLink)
[8] “IEEE Standard for Information Technology--Systems Design--Software Design Descriptions," *IEEE STD 1016-2009* , vol., no., pp.1-35, July 20 2009. Article (CrossRefLink)
[9] “IEEE Standard for Software and System Test Documentation," *IEEE Std 829-2008* , vol., no., pp.1-150, July 18 2008. Article (CrossRefLink)
[10] Kyunghwan Kim, Neunghoe Kim and Donghyun Lee, Hoh Peter In, “Requirements Trace Table based on Pain Chain for improving Maintainability,” in *Proc. of the Korean Information Science Society Conference*, vol. 38, no. 1(A), pp. 206-209, 2011. Article (CrossRefLink)
[11] Ju Young Kim and Sung Yul Rhew, “An Empirical Study on Tracking Table for Consistency and Completeness Validation in the Outputs,” *Journal of KIISE: Software and Applications*, vol. 34, no.5, pp. 419-430, 2007. Article (CrossRefLink)
[12] Lormans, Marco and Arie van Deursen, "Reconstructing requirements coverage views from design and test using traceability recovery via LSI," in *Proc. of the 3rd international workshop on Traceability in emerging forms of software engineering*. ACM, 2005. Article(CrossRefLink)
[13] Hayes, Jane Huffman, Alex Dekhtyar, and Senthil Karthikeyan Sundaram, “Advancing candidate link generation for requirements tracing: The study of methods,” *Software Engineering*, IEEE Transactions on 32.1, 4-19, 2006. Article (CrossRefLink)

[14] De Lucia, Andrea, et al, "Information retrieval methods for automated traceability recovery," *Software and systems traceability*, Springer London, 71-98, 2012. Article (CrossRefLink)
[15] McMillan, Collin, Denys Poshyvanyk, and Meghan Revelle, "Combining textual and structural analysis of software artifacts for traceability link recovery," in *Proc. of Traceability in Emerging Forms of Software Engineering, 2009. TEFSE'09. ICSE Workshop*, pp. 41-48, May, 2009. Article (CrossRefLink)
[16] Dusan Baek, Jung-Won Lee, "An analysis of relevance for traceability of corresponding items between software research and development document artifacts", in *Proc. of the 17th Korea Conference on Software Engineering*, vol. 17, no.1, pp. 13-20, 2015. Article (CrossRefLink)

Dusan Baek received the B.S. degree in Electrical and Computer Engineering at Ajou University, Korea in 2012. He is currently studying toward the Ph.D. degree in Electrical and Computer Engineering at Ajou University. His research areas include embedded software, mobile computing and software engineering.

Byungjeong Lee. He received the B.S., M.S., and Ph.D. degrees in Computer Science from Seoul National University in 1990, 1998, and 2002, respectively. He was a researcher of Hyundai Electronics, Corp. from 1990 to 1998. Currently, he is a professor of the Department of Computer Science and Engineering at the University of Seoul, Korea. His research areas include software engineering and web science.

Jung-Won Lee is an associate professor of the Department of Electrical and Computer Engineering at Ajou University, Korea. She received her PhD. Degree in Computer Science and Engineering from Ewha Womans University, Korea, in 2003. She was a researcher of LG Electronics and did an internship in the IBM Almaden Research Center, USA. Her areas of research include context-aware, embedded software and software engineering.