Traceability Management Technique for Software Artifacts which Comprise Software Release

Chan Lee
Department of Computer Engineering
Chungnam National University
Daejeon, South Korea

Dae Yeob Kim
Department of Computer Engineering
Chungnam National University
Daejeon, South Korea

Cheong Youn (Corresponding author)
Department of Computer Engineering
Chungnam National University
Daejeon, South Korea

Abstract

The capacity for tracing relationships among various artifacts which are created at each phase of software system development is essential for software quality management. Software release refers to delivering a set of newly created or changed artifacts to customers. The relationships among artifacts which comprise software release must be traced so that the work for customer's requirement of change and functional enhancement is effectively established. And release management can be effectively realized through the integration of configuration management and change management. This paper proposes the technique for supporting change management of artifacts and for tracing relationships of artifacts which comprise software release through the integrated environment of personal workspace and configuration management system. In the proposed environment, the visualized version graph and automated tagging function are used for tracing relationships of artifacts.

Keywords: release, traceability, baselined document, configuration management, version control

1. Introduction

Software system is composed of artifacts which were made in each development phase, and the ability that trace the relationships among these artifacts, that is called traceability, is very important not only in the development phase, but also in the maintenance phase. For example, if a requirements change is requested after the software system has been delivered to the customer, the change task can be established efficiently by tracing the relationship among artifacts which compose previous release.

The concept of system includes hardware system and software system, and this paper proposes traceability management method for elements which compose software system. In general, systems such as automobile or home appliances, etc. are made by combining hardware and software, and the software system can be a configuration item in the view of the whole system. In this paper, we consider not only the software system itself which is released with the whole system, but each element which composes the software system, as a configuration item.

Researches related with traceability of software artifacts have been conducted in various perspectives. When the researches related with traceability were combined, traceability for software artifacts can be divided into two groups: requirement based traceability and change history based traceability.

Requirement based traceability can be defined as the ability to trace from user requirements to other artifacts which are related with requirements. Requirement based traceability can also be divided to: pre-requirement traceability and post-requirement traceability.

Pre-requirements traceability supports activities such as requirements validation by indicating relationship between requirements of stakeholder and identified requirements. Post-requirement traceability supports activities such as verification and testing by indicating the relationship between requirements and other artifacts(logical model, design model, source code, test case, etc.) which were created based on the requirements[1, 2-8]. Requirements based traceability focuses on the semantic relationships among artifacts. This type of traceability has limits because it does not consider change history of the artifacts[9, 10].

Change history based traceability is the ability to trace the evolution of each artifact and relationships of various artifacts based on past change history. When change of particular artifact cause changes in other artifacts, the relationship among artifacts which were changed together is identified through the change history information. Relationships of artifacts are represented by linking versions of artifacts[9-14]. When the relationship among artifacts is identified by using change history, users can conduct change impact analysis or various maintenance tasks for existing software more efficiently. But, existing researches have limits because they just utilize the change history information which is provided from version management tools for identifying relationships of artifacts. In change history based traceability management, if version management tool and software configuration management tools are integrated and used together, the management task may be established more efficiently.

This paper proposes traceability management method based on the change history. The method provides more extended traceability information than the existing researches related to change history based traceability management through integration of personal workspace and software configuration management system. The integrated environment of this research supports three types of traceability as follows.

1) traceability for evolution of a single artifact
2) traceability for relationships among artifacts included in a single baselined document
3) traceability for relationships among baselined documents and total artifacts which compose particular release of a software system.

The first type is the ability to trace evolution of a single artifact. Artifacts change in personal workspace, and the result of change checked-in configuration management system. The integrated environment of this paper supports integrative management for the personal change history and the changes that are made official in the configuration management system[15].

The second type is the ability to trace relationships among artifacts through identification of version links among artifacts which were included in a configuration item[16].

Methods for supporting the two types of traceability mentioned above are already introduced in the preceding research. In this paper, we focus on the method for supporting the third type of traceability. This method supports to trace relationships among baselined documents and among total artifacts which compose particular release of software system.

The integrated environment which is proposed in this paper, provides visualized version graph and automated tagging function to support the three types of traceability mentioned above. The version graph is a representative version model, and it represents the versions of each artifact as a tree form. Users can easily understand the evolution of a particular artifact through the version graph. We developed functions for acquiring past version or creating new version through the version graph[17].

Tagging function links some meaningful information to a particular version of artifact as tag, and the integrated environment of this research automated this function to support efficient traceability management. The tag not only helps to distinguish the voluntary personal changes and the formal changes checked-in the configuration management system, but also provides clue for users to trace relationships among artifacts.

This paper is structured as follows. Following from the introduction in section 1, section 2 introduces the concept of software release management and some researches regarding traceability management. Section 3 discusses the concept of traceability management related with software release. Section 4 introduces the concept of traceability items and traceability management method proposed in this paper.

Section 5 introduces how the traceability management method introduced in section 4 is implemented. Section 6 discusses the differences between the integrated environment of this research and the preexisting configuration management systems by qualitative and quantitative evaluation. Finally, Section 7 presents the conclusion of this paper and future work.

2. Related works

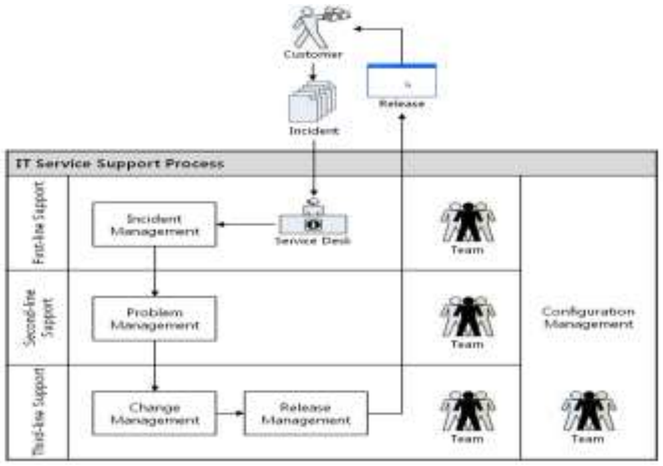
Release is defined as delivering changed software, hardware and related documents to users after formalized change approval process and testing. ITIL(Information Technology Infrastructure Library) defined release as approved change set for IT(Information Technology) service and derived from request for change[18]. ISO/IEC 20000-1 defined release as new or changed set of configuration items which is introduced in using environment after testing[19].

Release management is the process of checking and validating all changes related to the release. The main purpose of release management is providing safe IT operating environment and services through official procedure and validation[20]. ISO/IEC 20000-1 standard emphasizes that release management must be conducted with change management and configuration management activities[19].

Change management is conducted to assure that all changes are evaluated, approved, implemented and reviewed in a controlled way, and configuration management is conducted to define and control configuration items and to maintain the configuration information accurately. Change management and configuration management must be treated with release management in unified approach. Configuration management specially provides mechanism for tracing versions of configuration items. Again, it must record the related information for tracing changes of the configuration items.

ITIL proposes process that integrates release management process, change management process, and configuration management process. Fig. 1 indicates service management process which is proposed in ITIL in the abstract. The service management process of ITIL includes various subordinate processes, and among them, change management process, configuration management, and release management must be related organizationally. In the integrated process, various activities such as identification of changes, evaluation and approval, implementation of change, test and release, status accounting, and audit must be performed systematically and all artifacts which compose system should be traced.

Fig. 1. Service Management Process of ITIL



Mohan pointed out a problem that in general, configuration management and traceability management are conducted independently, and proposed a process framework for integrating the management processes mentioned above. The framework shows how configuration management process and traceability management process are interacted. He asserts that various tasks such as plan, management of integrated environment, request for change and management of delivery, management of baseline and release, monitoring and reporting for status of configuration, and etc. must be connected organizationally for integration of configuration management and traceability management[21].

According to his assertion, in plan phase of the process, it must be determined that what information must be managed how level of granularity, and in work process of configuration management and traceability management, relationships of versions of artifacts and relationships among configuration items must be well managed. Also, relationships among artifacts must be identified properly in management of request for change and delivery. For management of baseline and release, all artifacts which compose a system must be complete and consistent, and also the relationships among all artifacts have to be established. Finally, in configuration status accounting and reporting, it is necessary to record not only background of creation of configuration items, but also relationships of artifacts with their versions.

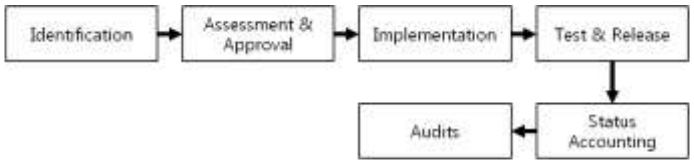
The integration of configuration management and traceability management that Mohan asserted is closely related with integrated management process which is emphasized in ITIL or ISO/IEC 20000-1 standard. But, additional research is needed because management methodology or tools are not introduced yet.

And aside from this, various researches have been conducted to manage traceability of software artifacts. Techniques to identify relationships among various artifacts based on requirements are researched[22-25], and also researches to identify relationships among artifacts based on change history are conducted[26-31]. In technical perspective, it is important to identify and to trace relationships among artifacts, but it is also necessary to do research on how the traceability management of artifacts is conducted in managerial perspective. Namely, the researches for integration of release management, configuration management and change management to construct the new software system from change request and for representation of relationships among artifacts in the integrated process need to be conducted.

3. The concept of traceability management of software release

The integration of release management, change management and configuration management includes several processes like Fig. 2[32]. The integrated process starts with identification of change. Assessment and approval for identified change, implementation of the change, test and release are carried out in turn, and then status accounting for system is needed. At this moment, all information related with the change of artifacts is recorded in database. The records are used in tracing and validating contents related with release in the future.

Fig. 2. Integrated Process of Configuration Management, Change Management, and Release Management



Traceability for release, the nub of this paper, means the ability to trace the relationship among baselined documents and among total artifacts which compose particular release. Each baselined document has a new version when it is checked in the configuration management system. The version of baselined document is called as revision in the integrated environment of this research. Revision is different from version which is created by general version management tools because it is created when baselined document is formally approved through configuration management system.

In this research, the revision of each baselined document is linked to others for indicating relationships among baselined documents which compose particular release. For this mechanism, we connected release number of system to revision of each baselined document. If the revisions which were linked with same release number are identified, the relationship among baselined documents can be traced.

In the integrated environment, link of revision of baselined document and release number of system is made through status accounting process of Fig. 2. The release number of system is recorded for each baselined document which composes the release when changes for baselined documents are tested and the system is released. If the release number is recorded in all baselined documents, revisions of baselined documents are linked by the release number, and then users can trace the relationship among baselined documents through the release number.

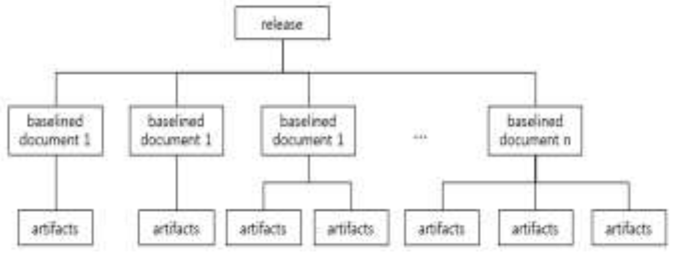
Release number of the system can also be used for indicating relationships among total artifacts. The relationships among artifacts can be traced when version of artifacts are linked with release number of system. Further details of this mechanism were explained in chapter 4 and 5.

4. The configuration of the traceability managed and related method

In general, requirement specification, design specification, source code and executive files are identified as baselined document, and these artifacts exist as physical files. Among these artifacts, the source codes are assembled as a unit component. The artifacts must be linked to configuration item in configuration management system. The configuration item can be a baselined document through official approval process, and the baselined document is an element of particular release. These elements can be represented as a hierarchical structure like Fig. 3.

In the preceding research, we introduced the traceability management methods for artifacts and baselined documents existing in the bottom level and middle level of Fig. 3. respectively [15,16].

Fig. 3. Hierarchy of Release Component



In this paper, we propose the traceability management method to support tracing relationships among baselined documents and among all artifacts which compose particular release. The relationships not only among the baselined documents, but also among the artifacts can be represented by using release number which is created when the system is released.

The relationships among baselined documents which compose a release can be indicated through recording the release number of system to each baselined document. More precisely, each baselined document has a new revision each time it is checked in the configuration management system, and recording the release number to baselined documents means representing the relationship between release number and revisions of baselined document.

For representing the relationships among all artifacts by using release number, the visualized version graph and the automatic tagging method are used. All artifacts are substantive object of changes, and its changes are made in personal workspaces. Each artifact has it's own change history. The change history of artifacts can be recorded by using version number, and this research provides the visualized version graph to allow the users to understand the change history of each artifact clearly.

The automatic tagging method is used to allow users to trace the version numbers which were related with baselined document from all versions of each artifact[15, 16]. If users completed the changes of artifacts which were included in baselined document, they check in the baselined document to store the result in configuration management system. At this time, new revision of baselined document is created, and it is connected to versions of each artifact as a tag. The connection of version number of artifacts and revision of baselined document can be confirmed in the version graph easily. In case various artifacts are included in a baselined document, users can acquire the versions of each artifact which were connected with revision of baselined document by the same tag.

In case of using the automatic tagging method, the relationships of all artifacts which were related with particular release number of the software system can be represented easily. The relationships among all artifacts can be traced by linking the release number to version of artifacts which were included in a baselined document as a tag when the release number is recorded in all baselined documents.

5. Implementation result of traceability management method

In this chapter, we introduce how the concept of traceability management mentioned in section 4 is realized.

As mentioned earlier, linking the revision of baselined document and the release number of the system is achieved in the status accounting process. To trace the relationships among baselined documents, users have to confirm what revision of each baselined document is linked to the release number.

Configuration management officer records the release number to all baselined documents which compose the release, and then the release number is linked to version of each artifact which composes the baselined document as a tag.

The tag can be confirmed in version graph of each artifact, and the version of artifacts related with particular release can be acquired by using the release number as a query.

Fig. 4 shows a version graph of particular artifact. The version graph is composed of name of artifact, version number, tag and branch. The version number indicates change history of the artifact, and users can confirm the overall change history of the artifact through the version number of version graph. The tag is used to record subsidiary information to each version of the artifact. According to Conradi‘s paper, task or significant symbol(in general, revision of configuration item or variant information) which were related with particular version can be a tag[17].

The tag has meaning like milestone in change history of an artifact. The tag can be created not only manually by the users(in case of 'payment_way_add' in Fig. 4), but also automatically by the system(in case of 'REV_00' and 'REV_01' in Fig. 4). The tag is created automatically by the system when the baselined document is officially checked in the configuration management system(the revision of baselined document is attached as a tag) or the configuration management officer records the release number to the baselined document(the release number is attached as a tag).

The branch is used to develop new variant independently for existing versions. For example, the branch can be created when the users want to fix the bugs for a particular version or want to progress new development flow for another application. If the branch is used, users can conduct tasks efficiently without affecting the existing development flow. In case of Fig. 4, the branch 'SEShop' was created to make a new requirement specification from for the previous one.

Fig. 4. Version Graph of Artifact

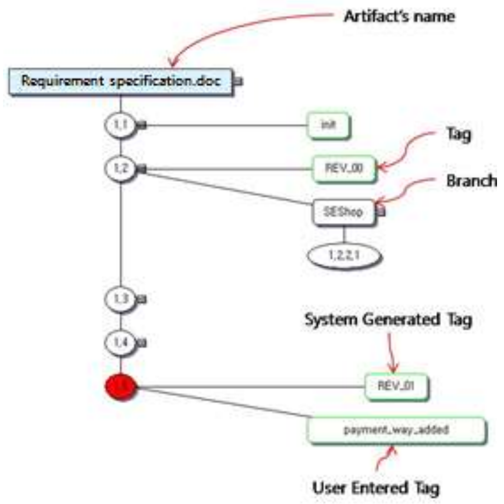


Fig. 5 shows that the relationships among various artifacts included in a baselined document. The artifacts which were included in a baselined document have different change history to each other, but the relationships among the artifacts can be formed by linking the revision of the baselined document to the latest version of the artifacts automatically.

Fig. 5 shows that the four artifacts were included in the same baselined document, and that the relationships among artifacts can be identified by revision of the baselined document through the version graph. In case various artifacts are included in a baselined document, the management tool which was developed in this research supports to acquire the versions of artifacts related with particular revision of the baselined document easily. This function provides efficiency for tracing and analyzing the previous change history of baselined document. As a result, the function also helps to analyze the impact of changes that will happen in the future. Table 1 shows how the versions of artifacts are linked by revision of baselined document. User can identify the version links of various artifacts included in a baselined document, and as a result, trace the relationships among them can be done easily.

Fig. 5. Relationships among Artifacts included Baselined Document

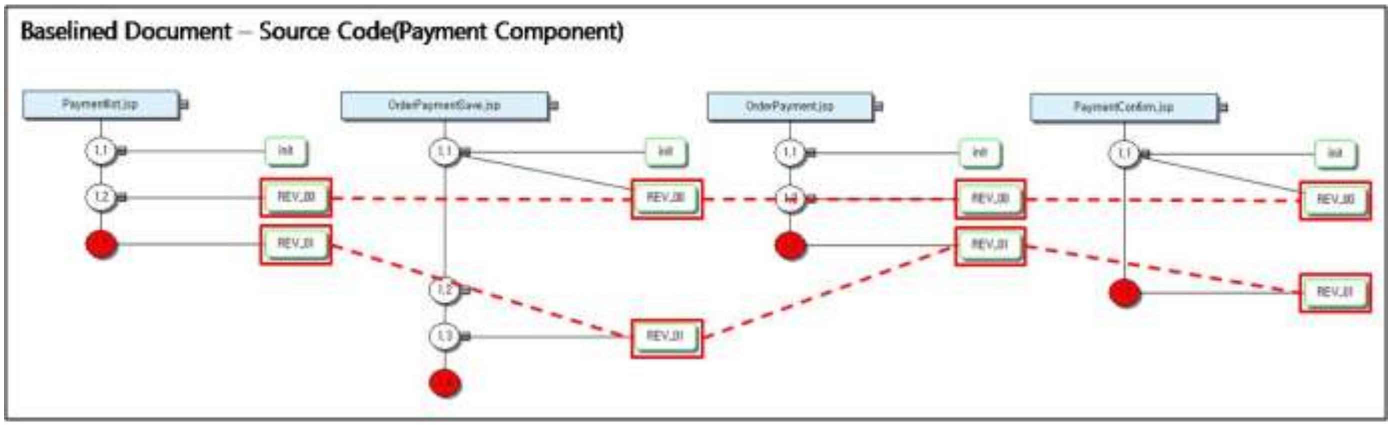


Table 1. Relationships among Artifacts through Revision

Name of Artifacts \ Revision	REV_00	REV_01
PaymentList.jsp	1.2	1.3
OrderPaymentSave.jsp	1.1	1.3
OrderPayment.jsp	1.2	1.3
PaymentConfirm.jsp	1.1	1.2

Finally, Fig. 6 shows that artifacts are linked by the release number of system. Through the figure, we can know that the all artifacts can be related to each other by the release number of the system. In the status accounting, the release number is recorded to baselined documents to present that the baselined document is included in particular release. At this time, the release number is linked baselined document linked to version of all artifacts which compose the baselined document as a tag. Finally, the all artifacts which compose the release are related through the release number when configuration management officer conducts the status accounting for all baselined documents.

Table 2 shows how the baselined documents and the artifacts of Fig. 6 can be related by the release number. Users can efficiently conduct the change from for previous release or the enhancement of functions through precise tracing for relationships between the baselined documents and all related artifacts.

The three types of traceability management methods have been explained to represent the hierarchical traceability information of Fig. 3. The process of linking the revision of baselined document or the release number of system to version of artifacts can be established automatically through the system which was developed in this research.

Fig. 6. Relationships among Baselined Document and Artifacts through Release Number

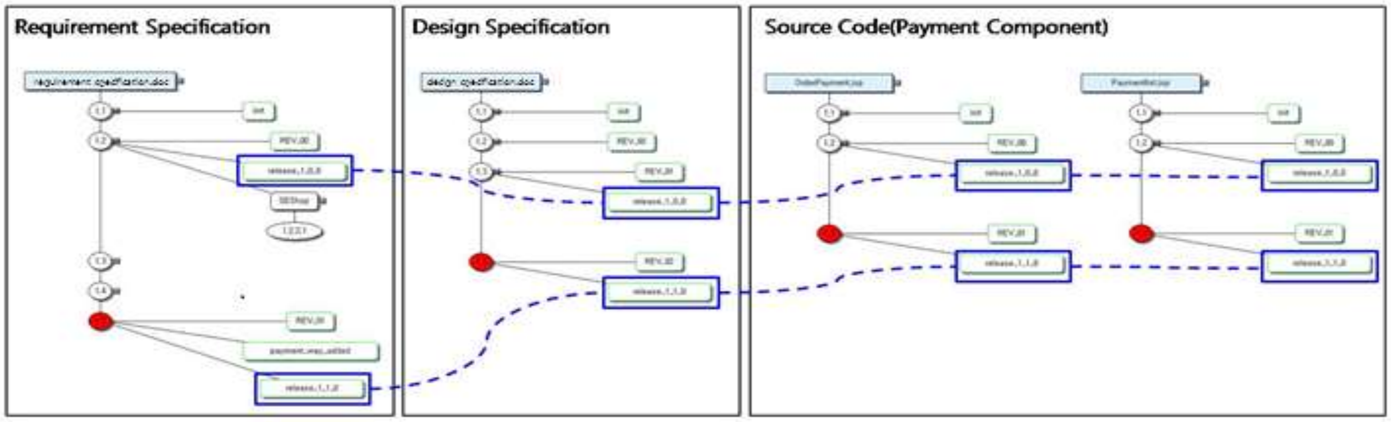


Table 2. Relationships among Baselined Document and Artifacts through Release Number

	Baselined Document	Requirement Specification	Design Specification	Source Code(Payment Component)	
	Artifact	Requirement Specification.doc	Design Specification.doc	OrderPayment.jsp	PaymentList.jsp
Release No.	release_1_0_0	REV_00	REV_01	REV_00	
		1.2	1.3	1.2	1.2
	release_1_1_0	REV_01	REV_02	REV_01	
		1.5	1.4	1.3	1.3

6. Research evaluation

In this chapter, we evaluate the system based on the functional requirements configuration management and traceability management. First, the qualitative assessment is carried out based on the requirements for traceability management support tools which were proposed by Marcus and the other functional requirement for configuration management tools. In the second place, the quantitative assessment is accomplished by comparing the general configuration management tools and proposed system in functional perspective[33, 34].

6.1 Qualitative evaluation of traceability management and configuration management requirements.

Table 3 is the result of listing the requirements for traceability management which were proposed by Marcus and the fundamental functional requirements for configuration management tools. Table 4 presents requirements satisfaction level of the existing commercial tools and the system of this research.

CVS and Subversion provide fundamental version management functions as open software, but do not provide the functions of configuration management system such as change control function, project management function and etc. Other configuration management tools provide improved functions and convenience than CVS or Subversion, but there are somewhat inconvenience to trace the change history of an artifact or the relationships among artifacts. Especially the changes which occurred in personal workspace, that is the changes which were not checked in the configuration management system, can not be traced. Also, we found the hindrance for efficient parallel development in some tools because the tools strictly separate the configuration management system and the personal workspace. The tracing functions(tagging or labeling) for release which is the main contribution of this paper are not provided in some configuration management systems. The basis for result of the evaluation can be confirmed in reference 16 and 34.

Table 3. Requirements of Traceability Management and Configuration Management and Evaluation of Proposed System

Req.no	Requirements of Traceability and Configuration Management	Evaluation of Proposed System
1	Visualize and store traceability information among various artifacts, regardless of the extraction methodology used.	Provide traceability information of artifacts, regardless of the extraction tools used. Traceability information of each artifact is provided by visualized version graph.
2	Interface or integrate with traceability link recovery tools.	Proposed system support itself identification of traceability link based on change history. But, this requirement is partially satisfied because the system can't provide traceability information based on requirements.
3	Allow the user to browse the traceability links.	Users can confirm the change history of single artifact and the relationships of various artifacts through workspace system.
4	Interoperate with other software engineering tools (e.g., analysis tools, document management tools, etc.)	Interoperability is satisfied by integration of version management system and configuration management system. But, interoperation with other CASE tools such as design tools or development tools is limited.
5	Provide comprehensive configuration management and change tracking facilities.	Provide the tracking facility for configuration management process and change through integration of configuration management system and personal workspace.
6	Support user querying and filtering of the traceability links.	Support version graph viewing function of single artifact and version link identifying function for tracing relationships of various artifacts.
7	Analyze and summarize the data on the traceability process and links.	Proposed system can extract and visualize the traceability information. But the analyze and summarize function are not provided in the system.
8	Integrate local traceability and global traceability.	Support tracing not the change in personal workspace but also the change in configuration management system.
9	Support branching function for manage multiple versions for various customer.	Provide branching function to develop same artifact with multiple flow.
10	Support tagging or labeling for release.	Support tagging function to indicate what version of artifact is linked with release number of system.
11	Support traceability for release.	Support tracing what version of artifact is linked with release through attach the release number as tag. Also tracing relationships of artifacts related some release number is supported.
12	Provide the environment in which parallel development is available.	Developers can work independently in own workspace by using branching function.

Table 4. Comparison of Configuration Management Tools for the Requirements of Traceability Management and Configuration Management
(○ - satisfied / △ - partially satisfied / × - not satisfied)

SCM tools / Req. No.	CVS	SubVersion	Perforce SCM	Rational ClearCase	AccuRev	Borland StarTeam	Proposed System
1	△	△	○	○	○	○	○
2	×	×	△	△	△	△	△
3	×	×	△	△	△	△	○
4	△	△	△	○	△	△	△
5	×	×	×	○	×	×	○
6	×	×	×	×	×	×	○
7	×	×	×	×	×	×	×
8	×	×	△	○	△	×	○
9	○	○	○	○	○	×	○
10	○	○	○	○	×	×	○
11	×	×	○	○	×	×	○
12	○	○	△	△	△	×	○

6.2 Quantitative evaluation of traceability management and configuration management requirements.

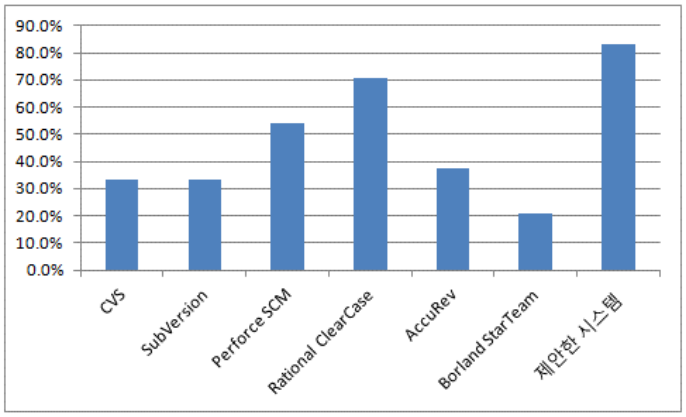
In this chapter, we quantitatively evaluate each configuration tools on the basis of the result of qualitative evaluation for requirements of traceability management and configuration management. The quantitative evaluation model based on the 12 requirements of Table 3 is the following.

$$SCM_{tool} = \frac{10\sum_{i=1}^{n} w_i \times p_i}{n} \%$$

- SCMtool : total evaluation index of configuration management system for the requirements of traceability management
- n : total number of requirements
- i : number of requirement
- w : $0 < w \leq 1$, weight for each requirement
  (In the evaluation of this research, the weight was set as 1 for all requirements)
- p : evaluation score for each requirement. assign the score among 0, 5 and 10 according to satisfaction level for each requirement

The result of quantitative evaluation on the basis of the model is indicated in the Fig. 7.

Fig. 7. Evaluation Result of Each System for 12 Requirements



7. Conclusions

Traceability for software artifacts is very important to develop high quality software. Developers have confidence that they developed the right software through tracing, and customers have high trust for the software. It is especially necessary to trace the relationships among artifacts for the release of the system to increase efficiency of change tasks when requirements change or improvement of function is requested.

Traceability for software has been researched in various ways. Various researches have been conducted according to the unit and representation method of tracing, method for traceability supporting, automation level of tracing and etc. Traceability can be defined as the ability to trace the change history of an artifact and the relationships among artifacts, and not only in this paper, but also the existing researches focus on how they fully support the traceability.

Proposed method is to support tracing the evolution of an artifact and the relationships among artifacts based on the change history management. This paper proposed the method for tracing relationships among baselined documents and all artifacts which compose the release of system by expanding the previous researches.

This paper proposed the method for recording the release number to the baselined document and attaching it to the version of artifacts to support tracing the release. The release number can be used to confirm what revision of the baselined document or what version of the artifact is included in the release, and through this, users can trace the relationships of the baselined documents and the all artifacts. The process for attaching the release number to the version of artifacts is automatically established through integrated environment of the personal workspace and the configuration management system.

We explained that software release management can be achieved efficiently by integrating the change management process and the configuration management process. This research integrates the three processes emphasized in ITIL and ISO/IEC 20000-1, and supports tracing the relationships of elements(baselined documents and all artifacts) which compose the release. In the integrated environment, the visualized version graph and the automatic tagging method are used for tracing the relationships among all artifacts included in a release. In the future research, we will conduct additional study to improve the functions of the configuration management system that support tracing the relationships among baselined documents.

References

[1] K. Pohl, "PRO-ART: Enabling Requirements Pre-Traceability", Proceedings of the 2nd IEEE International Conference on Requirements Engineering, 1996.

[2] J. Cleland-Huang, D. Schmelzer, "Dynamic Tracing Non-Functional Requirements through Design Pattern Invariants", Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering, Canada, Oct. 2003.

[3] J. Cleland-Huang, C. K. Chang, G. Sethi, K. Javvaji , H. Hu, J. Xia, "Automatic Speculative Queries through Event-based Requirement Traceability", Proceedings of the IEEE Joint International Requirements Engineering Conference, Germany, Sept. 2002.

[4] A. Egyed, P. Gruenbacher, "Automatic Requirements Traceability: Beyond the Record and Replay Paradigm", Proceedings of the 17th IEEE International Conference on Automated Software Engineering, Edinburgh, UK, Sept. 2002.

[5] X. Song, B. Hasling, G. Mangla, B. Sherman, "Lessons Learned from Building a Web-Based Requirements Tracing System", Proceedings of 3rd International Conference on Requirements Engineering, pp. 41-50, 1998.

[6] G. A. Stout, "Requirements Traceability and the Effect on the System Development Lifecycle(SDLC)", http://www.reveregroup.com/articles/137_005-RevereThoughtLeadership.pdf

[7] B. Ramesh, M. Jarke, "Toward Reference Models for Requirements Traceability", IEEE Transactions on Software Engineering, 27(1), Jan. 2001.

[8] O. Gotel, A. Finkelstein, "An Analysis of the Requirements Traceability Problem", Proceedings of the 1st International Conference in Requirements Engineering, pp. 94-101, 1994.

[9] H. Kagdi, I. M. Jonathan, S. Bonita, "Mining Software Repositories for Traceability Links", 15th IEEE International Conference on Program Comprehension, 2007.

[10] H. Kagdi, S. Yusuf, J. I. Maletic, "Mining Sequences of Changed-files from Version Histories", in Proceedings of 3rd International Workshop on Mining Software Repositories, pp. 47-53, Shanghai, China, May, 2006

[11] T. Zimmermann, A. Zeller, Weibgerber P., Diehl S., "Mining Version Histories to Guide Software Changes", IEEE Transactions on Software Engineering, vol. 31, no. 6, pp. 429-445, 2005.

[12] S. Sundaram, J. H. Hayes, A. Dekhtyar, "Baselines in Requirements Tracing", in Proceedings of Workshop on Predictive Models of Software Engineering, pp. 12-17, St. Louis, May 2005.

[13] S. Harvey, C. Parvathi, J. R. Daniel, S. Mahadevan, "Discovering Dynamic Developer Relationships from Software Version Histories by Time Series Segmentation", 23rd IEEE International Conference on Software Maintenance, pp. 415-424, Paris, Oct 2007.

[14] G. Harald, J. Mehdi, K. Jacek, "CVS Release History Data for Detecting Logical Coupling", in Proceedings of the 6th IEEE International Workshop on Principles of Software Evolution", 2002.

[15] D. Y. Kim, C. Youn, "Traceability Enhancement Technique through the Integration of Software Configuration Management and Individual Working Environment", Proceedings of IEEE International Conference on Secure Software Integration and Reliability Improvement, pp. 163-172, Jun. 2010.

[16] D. Y. Kim, C. Youn, “Traceability Enhancement Technique for Dependency Relations of Software Artifacts based on the Integration of Software Configuration Management System and Personal Workspace”, The KIPS Transactions, Vol. 18-D, No. 6, 2011.

[17] R. Conradi, B. Westfechtel, "Version Models for Software Configuration Management", ACM Computing Surveys, Vol. 30, No. 2, June 1998.

[18] Office of Government Commerce, ITIL Service Operation, The Stationary Office, UK, 2007.

[19] ISO/IEC 20000-1, Information technology - Service management-Part 1: Specification

[20] TTA.KO-10.0256, "Guideline for Configuration and Change Management of Information Systems“, 2007.

[21] K. Mohan., P. Xu and B. Ramesh, "Improving the Change Management Process", Communications of the ACM, Vol. 51, No. 5, pp. 59-64, May 2008.

[22] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, E. Merlo, "Recovering Traceability Links between Code and Documentation", IEEE Transactions on Software Engineering, 2003.

[23] J. H. Hayes, A. Dekhtyar, J. Osborne, "Improving Requirements Tracing via Information Retrieval", Proceedings of the 11th IEEE International Requirements Engineering Conference, Monterey Bay, 2003.

[24] J. Cleland-Huang, R. Settimi, C. Duan, X. Zou, "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability", 13th IEEE International Conference on Requirements Engineering, Paris, pp. 135-144, 29 Aug.-2 Sept. 2005.

[25] Y. J. Yoo, "Development of a Traceability Analysis Method based on Case Grammar for NPP Requirement Documents written in Korean Language", M.S. Thesis, Department of Nuclear and Quantum Engineering, KAIST, 2003.

[26] G. Canfora, L. Cerulo, "Impact Analysis by Mining Software and Change Request Repositories", Proceedings of 11th International Symposium on Software Metrics, pp. 20-29, 2005.

[27] H. Gall, K. Hajek, M. Jazayeri, "Detection of Logical Coupling based on Product Release History", Proceedings of 14th ICSM, pp. 190-198, 1998.

[28] H. Gall, M. Jazayeri, J. Krajewski, "CVS Release History Data for Detecting Logical Coupling", Proceedings of 6th International Workshop on Principles of Software Evolution, pp. 13-23, 2003.

[29] A. T. Ying, G. C. Murphy, R. Ng, M. C. Chu-Carroll , "Predicting Source Code Change by Mining Change History", IEEE TSE, 31(6), pp. 429-445, 2005.

[30] T. Zimmermann, P. Weisserber, S. Diehl, A. Zeller, "Mining Version Histories to Guide Software Changes", IEEE TSE, 31(6), pp. 429-445, 2005.

[31] H. Kagdi, J. I. Maletic., B. Sharif., "Mining Software Repositories for Traceability Links", 15th IEEE International Conference on Program Comprehension (ICPC'07), pp. 145-154, 2007.

[32] http://www.processdox.com/, "Configuration, Change and Release Management Policies and Procedures Guide"

[33] A. Marcus, X. Xie, D. Poshyvanyk, "When and How to Visualize Traceability Links?", Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 56-61, 2005.

[34] T. Fatma, "Evaluating Software Configuration Management Tools for Opticon Sensors Europe B.V.", Masters Thesis Software Engineering, 25 June 2004.