2010 Fourth International Conference on Secure Software Integration and Reliability Improvement

Traceability Enhancement Technique through the Integration of Software Configuration Management and Individual Working Environment

Dae-Yeob Kim
Department of Computer Engineering
Chungnam National University
Daejeon, South Korea
kdymn2@cnu.ac.kr

Cheong Youn
Department of Computer Engineering
Chungnam National University
Daejeon, South Korea
cyoun@cnu.ac.kr

Abstract— **Software productivity and quality improvement through software configuration management is based on organized and consistent change management. In change management, artifact identified as configuration item should be able to provide its changed history to the members in group and the members should be able to trace the changes made for the corresponding artifact. General software configuration management systems provide tracing information for artifacts only within the configuration management system, and it does not go further to changes that occur within individual working environment. This paper provides a solution that helps tracing changes that occur not only in configuration management, but also individual artifact's changes through the integration of configuration management system and individual working environment. A revised version in configuration management system is connected to the artifact version of the working environment by the tagging technique, and traceability can be managed more effectively by sharing the tracing information.**

Keywords – traceability; version control; workspace; software configuration management; cooperative management;

I. INTRODUCTION

Various artifacts(request item specifications, design document, source code, etc.) resulting from the software development process continuously change due to the changes in user working environment, development technology, user requirements, error editing and function improvement, etc. In recent times, as the size and developing organization for software expands at an accelerating pace, the importance of managing changes to development artifacts is increased day by day.

The importance of change management has led to an interest in Software Configuration Management (SCM) and its necessity is being increasingly recognized among development enterprises. SCM, by recognizing configurations through the full life cycle of software, allows systematic management of specification changes, and, as an activity with the purpose of maintaining software reliability and uniformity, it guarantees product quality and allows improvement in development productivity. The SCM system has been developed to systematically support these SCM activities.

SCM systems in current use (Synergy/CM, StarTeam, Source Integrity, CCC/Harvest, PVCS Dimensions, ClearCase(UCM), etc.) provide many functions to support the activities described above; however, the most representative and common feature is version management for artifact change. Through version management, a user can search the past history of changes, obtain a specific version, compare and contrast differences between two versions [1-6]. Most importantly, version management should be able to provide information that allows the user to understand the significance of the process of artifact evolution and to trace what has in fact changed. Tracing information, by helping the user understand the significance of changes and by providing a holistic view of the trend in changes, is absolutely necessary for it provides direction for past problems and for work to come in the future.

This paper presents a management technique characterized by expanded traceability functions achieved through the combination of tracing information provided by the SCM system and the changes occurring in individual working environment. In contrast to the SCM system that manages baselined documents, individuals can freely make changes as they please in the individual working environment. If a baselined document is initially created according to the official process in the SCM, subsequent artifacts changes in the individual working environment leads to the creation of additional versions with respect to the earlier version reflected in the SCM system. Changes made in an arbitrary working environment are difficult to understand by those other than the person making the changes, therefore, this leads to negative side effects, such as a decline in consistency due to the overlapping of tasks, additional work necessary to bring about integration in the changes made, difficulty in guaranteeing the accuracy of the resulting changes. In order to prevent such problems, coordination of changes made in the SCM system and individual working environment is necessary. Furthermore, a method of presenting a combined list of tracing information for each change is necessary.

Artifact changes made in the working environment through the integrated environment are connected with the SCM baselined document. In addition, the new artifact version created through changes in the working environment is reflected in the SCM system, along with the process of checking in the baselined document. Users will be able to verify the artifact version, through the baselined document, reflected in the SCM system and trace subsequently created new versions. When there is a need to change the baselined

978-0-7695-4086-3/10 $26.00 © 2010 IEEE
DOI 10.1109/SSIRI.2010.27

document, the person with the authority to make changes can check the work progress with regard to the requested change by tracing the changes made in the working environment and, depending on the result, can decide whether additional work is necessary. By utilizing this process, problems discussed earlier, such as overlapping of tasks and consequent wasting of time, and incurrence of additional work expenses, can be prevented and, ultimately, development productivity and product quality can be improved.

In this paper, the technique of applying tags in the working environment to the checked-in version of the baseline document so that users can trace the flow of artifact changes more easily has been implemented. By utilizing tag information, users can easily understand the current flow of changes made from the version reflected in the SCM system. In addition, by allowing the tag to be automatically created when the baselined document is checked in, management is more convenient and accurate.

This paper is structured as follows. Following from the Introduction in Section 1, Section 2 introduces the research regarding traceability management. Section 3 discusses the change process resulting from the combining of the SCM system and the individual working environment. Section 4 introduces the structure of the integrated environment and its result. Section 5 provides an explanation of the technique of tagging the baselined document on the version graph used to show the tracing information in the integrated environment. Section 6 discusses the difference between the integrated environment and the preexisting SCM systems. Finally, Section 7 presents the conclusion of this paper and future research.

II. Related Research

The definition of traceability is expressed in various ways depending on the field of research or industry. Areas such as measurement standards in machine industry, material science, software engineering, food processing, etc. each apply their own definition of traceability. The most standardized definition of traceability is regarded as the ability to relate uniquely identifiable objects with the concept of time and to verify that relationship [7]. Certification institutions such as the ISO emphasize the importance of traceability as a method of guaranteeing a customer the quality of a product; accordingly, each industry field has developed and applied guidelines and procedures for traceability management.

The definition of traceability in the field of software is somewhat varied depending on the extent of its application and purpose. Therefore an application category should be established before a definition of traceability is presented [8]. In the software field traceability is typically defined as the ability to manage the relationship between a requirement and artifacts derived from that requirement [8-12].

It is difficult to view the definition of traceability descried earlier as comprehensive in covering all aspects of traceability in relation to software development [13]. A definition of traceability that can be applied, not only to traceability among different artifacts, but also to various categories, including traceability within an artifact and the traceability of changes over time, is necessary.

This paper defines traceability as the ability to record and analyze changes within a single artifact. In this case, traceability can be used for the purpose of analyzing changes within an artifact and its influence; therefore, it can be used as important information in presenting the history of changes along with the traceability among different artifacts.

Berczuk has defined the traceability of a single artifact in two ways: traceability of changes within an individual's working environment (local traceability) and traceability of changes that all users can share (global traceability) [14]. Local traceability is a concept necessary for individually managing and maintaining changes in a working environment and global traceability is a concept where multiple users, through the SCM system, share information regarding configuration item changes. The two concepts of traceability presented by Berczuk are not to be separated and exist only to efficiently manage the environment flow in organizing specific releases in individual work during the process of developing software products. In short, a more efficient development environment may be established by combining the management of traceability information of artifacts managed individually and traceability information of configuration items shared by multiple users through the SCM system.

However, the common SCM systems currently in use are partially limited with regard to combining the two concepts of traceability and its management. Although these systems apply concepts in supporting procedures in official changes and allow the reflection in the SCM environment of individual changes, a problem is that not all users are able to share local traceability information. This signifies that, when changes occur in the SCM system's baselined document, changes pertaining to artifacts derived in the individual working environment cannot be traced and, therefore, it is difficult to view these systems as truly integrated environments [16-21].

This research, in order to solve the issue presented above, integrates the individual working environment and the SCM system, thereby, not only providing global traceability for configuration items in the SCM system, but also local traceability for artifacts from the individual working environment; this allows the use of improved traceability information.

III. SCM System and Individual Working Environment Integration and Change Management Scenario

This section describes the overall change process under the integration of SCM system and individual working environment. Changes are made while artifacts from the individual working environment are connected to the SCM's baselined document and the results from the changes are reflected in the SCM system according to the baselined document's check-in process. Through the baselined document, users are able to acquire the artifact version reflected in the SCM and, by connecting the acquired version with the newly produced versions in the individual working

environment, obtain traceability information regarding changes. Fig.1 shows the stages from the establishment of the connection between an artifact made in the individual environment and the SCM baselined document to the stage where change results are reflected in the SCM system.

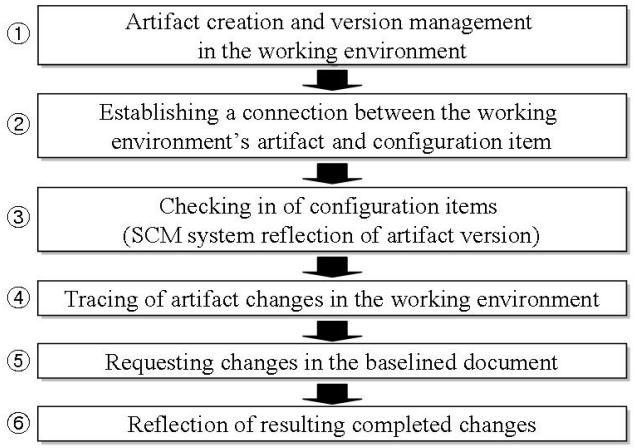

Figure 1. Integrated SCM System and Working Environment Change Process

① Artifact creation and version management in the working environment

Change management in the integrated environment begins with the creation of artifacts in the individual or cooperative working environment. Numerous changes may occur from the creation of an artifact in the working environment to its identification and initial registration as a configuration item in the SCM system or during the process of satisfying change requests after registration in the SCM system. In order to manage these changes a version management method is used. Version management in the working environment allows changes in various trends in artifacts and supports the storing of individual change history. Version management within the working environment is carried out for the purpose of maintaining local traceability of individual work in the unofficial domain.

② Establishing a connection between the working environment artifact and configuration item

When tasks on the artifact in the working environment are completed, the user must make a request for approval in order to reflect the completed version in the SCM system. A request for approval in the SCM system is made through configuration items where the user must connect the artifact from the working environment with the configuration item. The connection between artifacts and configuration items is made through the method of file attachment or referencing the directory structure of the working environment. Artifacts from the working environment derived from such connections are reflected in the SCM system when configuration items are later checked-in as baselined documents.

③ Checking in of configuration items (SCM system reflection of artifact version)

The user makes a request for approval to check-in a configuration item while the artifact from the working environment and the configuration item is connected. When the request for approval is finally authorized, the user executes the check-in function and checks in the configuration item and the configuration item becomes a baselined document in the SCM system. When a configuration item is checked-in as a baselined document it means that the connected artifact version has been reflected and set in the SCM system. In order to display this, the SCM system automatically sends the baselined document version to the working environment and tags the relevant artifact version. As a revised version of the configuration item, the baselined document version can be expressed in the format of 'REV_00' which is a concept of distinguishing it from versions assigned to artifacts coming from the individual working environment. The baselined document version is used as information to show the global traceability in the SCM system, while the use of tags in connecting the baselined document version with the working environment artifact version is to concurrently manage the global traceability in the SCM system and the local traceability in the working environment. Such tag information can be utilized by users to understand the artifact's checked-in version.

④ Tracing of artifact changes in the working environment

Even after a configuration item is checked-in as a baselined document, users can freely make changes to the artifact connected to the baselined document within their own working environments. Frankly, in typical SCM systems, once an artifact is set in the system, temporary changes are not permitted and changes are only permitted after going through strict approval procedures. While the integrated environment presented in this research allows users to carry out temporary tasks in their working environments, this does not influence the checked-in version in the SCM system, thereby supporting heightened development productivity [2, 6, 15].

Flexible changes in the working environment allow the creation of a new trend of versions in addition to the artifact version already reflected in the SCM system. Users sometimes create versions in the same trend as the version reflected in the SCM system and, at other times, create branches to carry out tasks in a different trend. By using branches users can individually work as they please within an independent trend while not influencing the mainline. Branches can be created to revise a bug from a particular version in a separate trend, to develop a

separate release in response to different customer's request, to create a version that fits a particular platform, etc. Tasks carried out through branches, depending on their result, can be merged into the mainline to create a new version.

Various change trends in the working environment can be traced by being connected to the baselined document in the SCM system. The user can confirm the artifact version that has been combined with the baselined document and verify the changes carried out in the working environment by tracing the additional versions that have arisen. The tracing of artifacts arising from the working environment is an activity absolutely necessary in verifying the progress of tasks with respect to change requests on baselined documents in the future.

⑤ Requesting changes in the baselined document

Requesting changes in the baselined document requires an approval process as in the case of the check-in process. When the final authorization is given for the change request in the baselined document, the designated person in charge of changes checks out the baselined document and acquires the authority to make changes. The person with the authority to make changes in the baselined document acquires the artifact connected to the baselined document and, through the integrated environment, can trace changes currently carried out on the artifact in a different working environment.

To allow the sharing of artifact work results of other users, it is necessary to have a cooperative system that can manage the sharing users' information and the artifact version information. All sharing users, in addition to the designated person in charge of changes, are able to share and trace the work results of each other in the cooperative management system. In particular, within the working environment version trend, the designated person in charge of changes can verify the versions that use his checked-out baselined document as a tag and trace additional versions that have been produced, thereby analyzing the changes that have since been made.

⑥ Reflection of resulting completed changes

After tracing artifact changes in the working environment and the work required to satisfy requirements is complete, the designated person in charge of changes for the baselined document should reflect that result in the SCM system. In order to reflect the resulting artifact changes in the SCM system, the same approval procedure in the number ③ of the flowchart needs to be followed; furthermore, after the authorization is given, the check-in function needs to be executed again and the check-in procedure needs to be carried out. As a result, a new version is created in the baselined document and this is connected to the artifact version in the working environment as new tag information.

Thus far was an explanation of the process of how an artifact in the working environment is connected to the baselined document and reflected in the SCM system. Artifact traceability of changes made in the working environment is significant, not so much because of individual need, but for fostering the understanding of tasks among members in an organization, which in turn supports better development productivity. Therefore, the integration of change management tasks carried out through SCM and the individual working environment is absolutely necessary to provide a more mature development environment.

IV. INTEGRATED SYSTEM ENVIRONMENT

This section presents the structure necessary for an integrated operation of the individual working environment and the SCM system. The reason for integrating the management of global traceability in the SCM system and the local traceability of the individual working environment is because the artifact is an object that can be shared by multiple users. Additional versions created on top of versions already reflected in the SCM system need to be made public to other users that desire to make changes to the relevant artifact; traceability information based on this premise can prevent overlapping tasks by multiple users.

In order to handle the problems stemming from flexible changes in the individual working environment and the sharing of artifacts, this research provides a cooperative management system which, based on the integration of the cooperative system and the SCM system, supports holistic change management and traceability management.

Fig.2 presents the functional elements necessary for the cooperative management system and the SCM system, which are the objects of integration. The cooperative management system is designed to support the individual and community working environments while the SCM system is comprised of functions that support typical configuration management tasks.

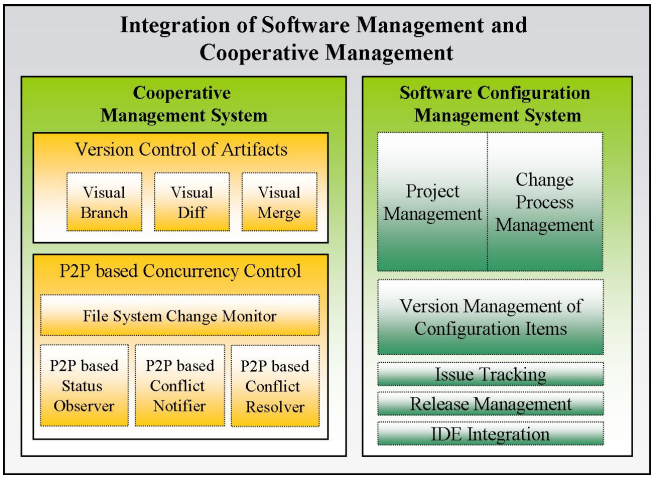

Figure 2. Functions of the cooperative management system and the SCM system

Fig.3 is a schematization of the process of each system's mutual application in the integrated environment. As one method used to connect an artifact from the cooperative management system with a web-based SCM system's configuration item, which supports the referencing of the working environment's directory structure, it is necessary to register library files in the format of ActiveX in each user's registry. This is an automatic process that occurs when the SCM system's configuration item page is executed; through this process, artifact versions from each user's working environment can be reflected in the SCM system. In addition, the JNI (Java Native Interface) programming method has been used to connect the revised version of the registered configuration item, which was sent to the working environment, with the version of the artifact. When a revised version of the configuration item, which originates from the Java-based webpage, is used as a parameter to call the tag-creation function in the cooperative management system, the revised version of the configuration item is attached to the version of the artifact as a tag. These methods have been implemented for the mutual application of the two systems which have materialized in differing development platforms.

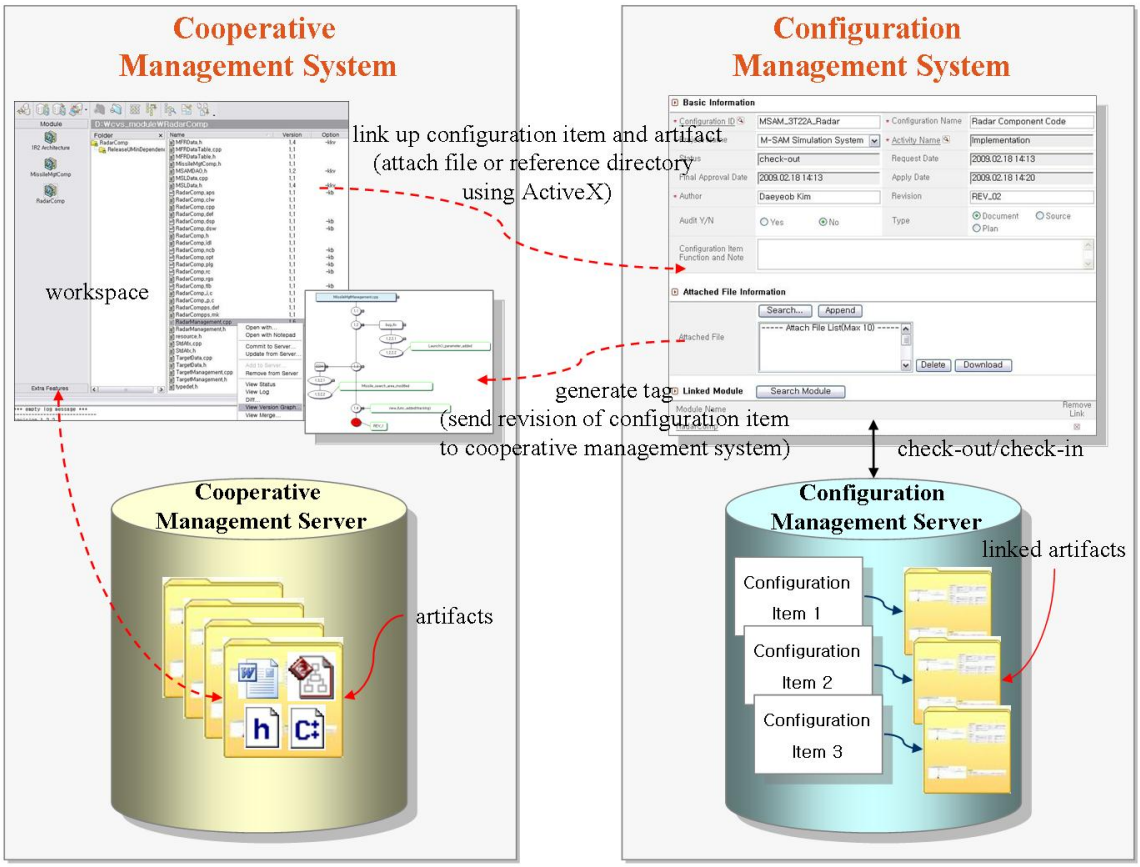

Figure 3. Mutual application of the cooperative management system and the SCM system

The cooperative management system is divided into two functions: the version management function for the management of changes in the individual working environment and the function to handle concurrency control in situations of cooperation between sharing users. Artifact version management provides a visual display making it convenient to execute branching, merging and comparison of differences. For the purpose of version management, the cooperative management system materialized, founded on the CVS version control method and repository management mechanism; furthermore, it provides a collision resolving method to support simultaneous carrying out of tasks.

Fig.4 shows the workspace environment in the cooperative management system. The purpose of the workspace is to provide an area for independently organizing a working environment which can be used in freely making individual changes. The artifacts existing in the workspace each have their own version information, and the version information for each artifact can be confirmed on the version graph. The user can make changes to an artifact by executing the appropriate application for the relevant file format.

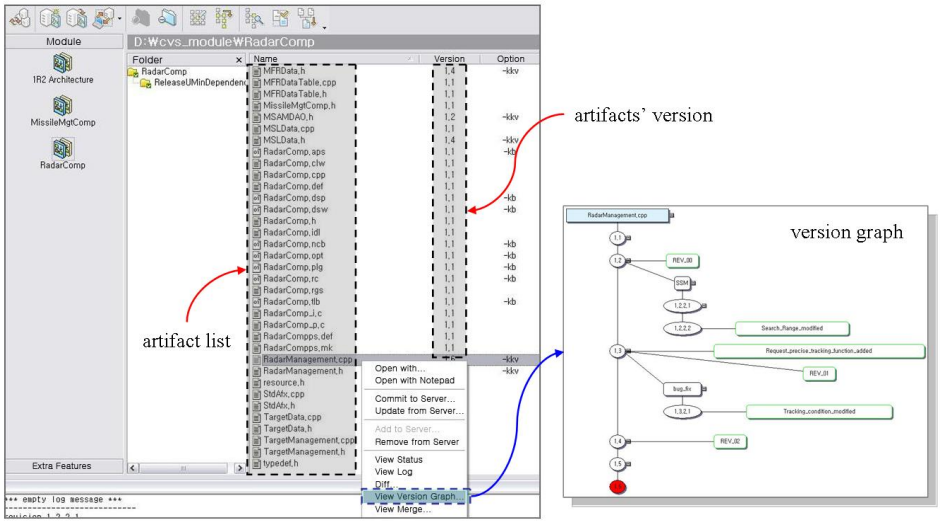

Figure 4. The workspace environment in the cooperative management system

The SCM system was developed as a web-based system to make it easily approachable and has various functions; however, this paper focuses on explaining the baselined document's change control and version management functions. Change control refers to the procedure where a change request regarding a particular baselined document is evaluated and applied. A checked-in baselined document cannot be changed without going through the designated approval process and there must be a prior request to acquire the right to make changes. The version management in the SCM system refers to that of the configuration item that has already been checked-in as a baselined document, which can also be seen as management of the revised version. The revised version of the configuration item is important information in determining the artifact's baseline. The baseline could be defined as the collection of all configuration items that have been fully examined and from a software engineering developmental process point-of-view it signifies input material for tasks in the next stage.

Fig.5 shows the web-based SCM system. The pertinent screen is one of a particular configuration item showing the basic information for the selected configuration item.

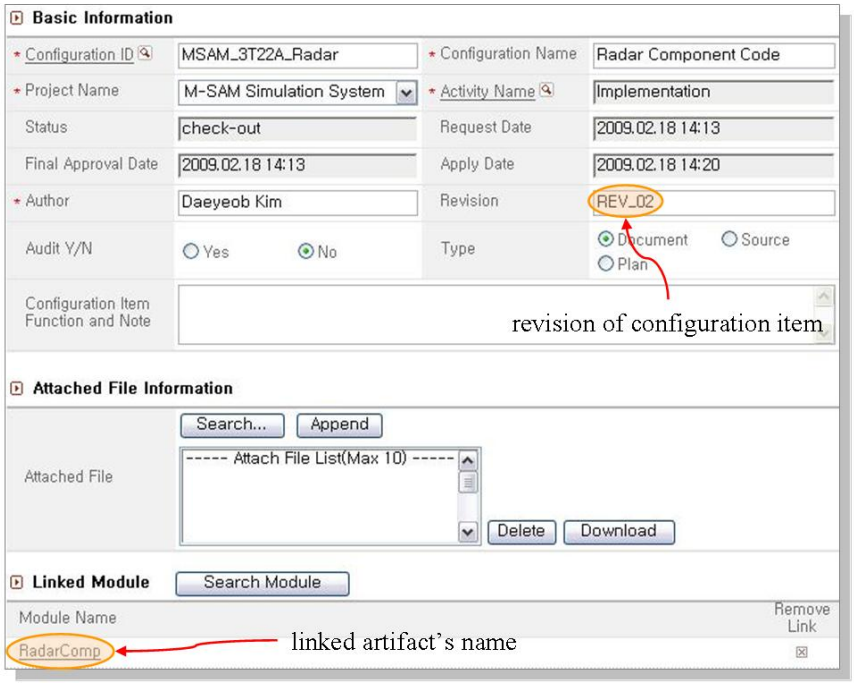

Figure 5. Configuration item screen in the SCM system

In the figure above, the revised version of the current configuration item is represented as 'REV_02'. The revised version of the configuration item is designated whenever the configuration item is checked-in and although it does not have to be represented in the format above, contents should be recorded so that other users can understand it. When a configuration item is checked-in as a baselined document the revised version is connected to the tag which is linked to the artifact's version graph located in the bottom of the screen. The connected artifact is typically managed in the workspace environment of cooperative management system, but in order to maintain the connectivity with the configuration item, the relationship is established as described above. If a new request for change is made in the Fig.5 situation and changes are made to the artifacts therein, the revised version of the configuration item becomes 'REV_03' and the pertinent information will attach to the artifact version graph as a tag once again. As these processes become systematically automated, users involved in making changes will be able to efficiently manage integrated traceability information.

V. The Expression of the Artifact Version Graph and Traceability

This section provides an explanation of the version graph, which is used to show the version trend of an artifact from the working environment, and the method of tracing changes more easily by attaching the revised version of the configuration item to the version graph as a tag.

Various forms of version models are used to express the management of repetitive changes made to artifacts in the working environment and the version trend. This research has used the version graph to express the version trend in the working environment. The version graph resembles the tree structure and the version title can be defined differently for each system. Fig.6 is the version graph expressed through this research.

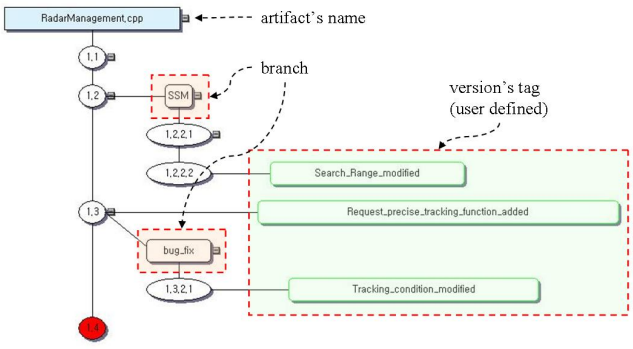

Figure 6. Version graph of an artifact

The version graph includes, in addition to the artifact version information, branch and tag information. A branch may be created to fix a bug in a particular version or to create a new developmental trend that may implemented in a different application field. As can be seen in the figure above, each branch can be distinguished by its label showing its purpose or intent and the branch created is carried out in a separate trend apart from the mainline. When necessary, the resulting changes carried out by the branch is merged into the mainline and reflected. A branch in the SCM pattern, which was presented by Berczuk, as an individual codeline that does not affect the mainline, is an element that can maximize development efficiency [14].

A tag is used when additional information needs to be added to each version. A tag allows users to summarily understand the types of changes that have occurred in each version and it can also be seen as a milestone in the artifact's change trend.

This research, in order to more clearly represent traceability information related to the baselined document on the artifact's version graph, has used the method of attaching as a tag the revised version of the configuration item. The tag information in Fig.6 has been arbitrarily created by an individual, while coinciding with the time it is checked-in, the revised version of the configuration item is automatically created. In order to distinguish the arbitrarily created tag, the SCM system standardizes the revised version of the configuration item under the agreed upon rule (similar to the form of 'REV_01' defined in Section 3) and combines it with the version graph. Automated tag creation and the rule-based tag naming method have the advantage of preventing confusion among users and improving work productivity. Fig.7 shows the revised version of the configuration item attached as a tag to the version graph of the artifact changed by an individual.

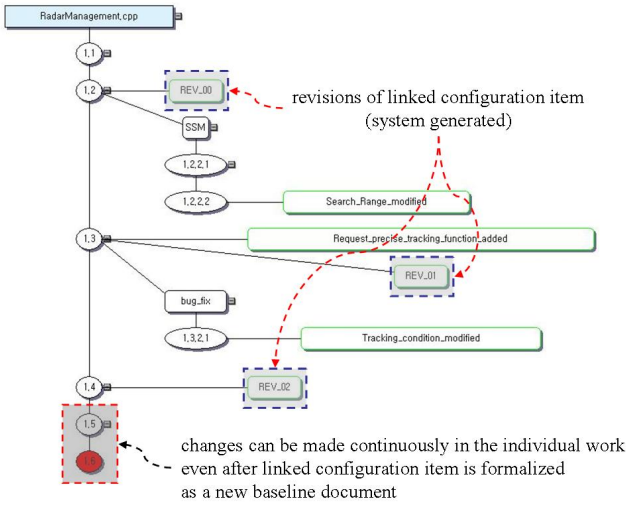

Figure 7. Version graph of an artifact (attached revision tags)

Fig.7 depicts 'REV_00', 'REV_01', 'REV_02,' which represents the revised versions of the configuration items, attached as tags to the version graph. As can be seen in the figure, even after version '1.4' of the artifact is reflected on the configuration item's revised version 'REV_02,' changes can continuously be made in the individual working environment in creating new versions (in the case of version

'1.5' and '1.6'). In this case, users can directly understand the progression of change in the artifact after it has finally been reflected in the SCM through the configuration item's revised version tag called 'REV_02'.

VI. Comparison Evaluation with Preexisting SCM Systems

Although there are a wide variety of SCM tools today they can be roughly divided into two categories; productivity centric tools used to improve productivity in the cooperative management dimension and process centric tools used to support management processes in the SCM dimension.

Functions pertaining to productivity centric tools focus on development productivity which typically includes version management and parallel development functions. Process centric tools, in addition to the functions mentioned in the previous paragraph, contain the management processes pertaining to the SCM dimension, which includes areas such as defining task relevant to each procedural stage, controlling work authorization, and controlling the entire project through an authorization process.

In the case of productivity centric tools, although the focus is on improving productivity through cooperation-related functions in the working environment, it does not provide change management functions such as the configuration control seen in SCM. Typical productivity centric SCM tools include Source Integrity (SI) and StarTeam. In the case of process centric tools, although stable configuration control is made possible by integrating the working environment and SCM, the possibility of efficient cooperation in the working environment is restricted, thus posing limitations to productivity [22]. Process centric tools supporting SCM include Synergy/CM, CCC/Harvest, PVCS Dimensions, ClearCase(UCM), etc.

This section seeks to evaluate the issues regarding the integration of traceability and the utility of the system suggested through the typical functions supported in the productivity centric and process centric SCM.

Source Integrity establishes a logical unit called a "change package" which may contain a single change from a working environment or a set of many changes and connects it to a software repository. The working environment in Source Integrity is called the "sandbox." When a file in the sandbox is checked out to be modified the aforementioned file enters into a locked state within the repository. This signifies that users, other than the user who has checked out the file, may not carry out changes on the relevant file. Although anyone may carry out changes on a desired file in the working environment, since the SCM configuration control flow is not supported, such locking algorithms limit the possibility of multiple users sharing and modifying artifacts [16].

The working environment in StarTeam is called StarTeam view and it is composed of working folders and working files. All work activities are carried out in an area called StarTeam view. StarTeam defines the logical units of change as changes and tasks. Developers, according to process rules, connect the newly created file versions to a change or task. A feature of StarTeam that distinguishes it from other systems is that it has the ability to collectively express the definition of files and connection, changes, requirements, tasks, and topics. Files are connected to related changes, tasks, and topics and, through these relationships developers are able to trace changes with regard to these files. However, tracing of changes through collection is also only possible after the resulting changes are reflected on the pertinent change or task; therefore there is a limit to tracing changes occurring in the individual working environment before they are reflected in the SCM system [17]. This can also be seen as an example of disunity between global traceability in SCM and local traceability in the working environment.

In the case of Synergy/CM, the logical unit for changes is defined as task. Developers set up their own working environments called work areas or work projects to carry out their assigned tasks. When a change with respect to a task is completed the complete task operation is executed and the results are reflected on the project at which point other developers may verify the results of the change. It is difficult to say that integrated traceability information is being provided since other developers are unable to trace the progression of change in the individual working environment until the resulting change with respect to the task is reflected. In other words, only after the completed changes are reflected in the SCM can the results be traced; therefore, global traceability is provided, however, there is a limit to providing local traceability [18].

In the case of CCC/Harvest, all changes carried out in the working environment are controlled at the center. Thus, in order to prevent arbitrary changes made at the request and need of an individual, such possibilities are sealed off at the source. Change requests and the reflection of change results undergo an authorization process, and this control process assures the stability of a project. Change results that are reflected after having undergone the authorization process are recorded in the SCM server along with information on who was responsible for the change, change content, change time, edited version, etc. In the case of CCC/Harvest, because work authorization is controlled at the center and all change history managed in the SCM dimension, the change history of the working environment is not separately managed [19].

PVCS Dimensions, similar to other typical process centric SCM tools, allocates change targets, organized in logical units called "work packages," to whoever is responsible for changes. When modified work packages are reflected in the SCM server, it takes on the significance of a modified document (checked-in document) and a label is attached signifying the baseline of the artifact version. This also signifies global traceability. Basically, the control flow regarding work authorization is identical to other process centric tools and, because change in the working environment is limited depending on work authorization, local traceability is not separately managed [20].

ClearCase is composed of a non-UCM system that provides basic functions for version management in the working environment and a UCM (Unified Change Management) system which supports change tasks in the

SCM environment. From the viewpoint of integrating the working environment and SCM system, this system provides the most similar environment and the method in expressing tracing information is not very different. What I would like to point out are the functional differences in the individual working environment, not so much the issue of integration of environment and traceability. However, whereas the working environment presented in this research basically provides the possibilities of multiple users sharing an artifact and participating in all change tasks, in the case of ClearCase, the fact that it does not provide such possibilities can be pointed out as a problem. Although ClearCase allows the sharing of artifacts, actual changes can only be carried out by one user. Therefore, users that cannot acquire the authority to make changes first only have a read-only right and cannot participate in making changes. This is a control method to guarantee the consistency of changes made to an artifact; however, when compared to the working environment presented in this research, which allows the participation of multiple users in making changes without compromising consistency, ClearCase presents some limitations in developmental productivity [21, 23].

As reviewed above, typical SCM systems have yet to provide traceability information that integrates changes made in the working environment. The integration of the working environment and the SCM system with traceability as discussed in this research, can be assessed as an element that can support a further enhanced development environment. Table 1 organizes the evaluative results.

The 'branch creation and parallel development support' item in Table 1 is an evaluation criteria that assesses how freely multiple users can make changes to a single artifact in their individual working environments. The 'visualized version model' item assesses whether a particular model is provided such that users can directly view and understand the change trend in an artifact. The 'SCM change management' item assesses whether an official approval procedure is provided in the SCM system. The 'integration of traceability' item assesses whether each system distinguishes between local traceability and global traceability, and if it supports management based on integrating the two concepts of traceability. Providing 'development process management' allows the systematic management of activities and tasks at each stage of a project and is a core function that process centric tools should be equipped with. 'Artifact sharing and simultaneous work support' is an evaluative criterion examining whether multiple users are able to share a specific artifact and simultaneously carry out tasks in their individual working environments. While in the case of typical SCM tools, regardless of the configuration control process, if one user is modifying the relevant artifact other users are not allowed to make changes, the system proposed in the paper allows tasks to be carried out simultaneously and provides a method of adequately resolving conflicts, thereby supporting enhanced development productivity.

Considering the comparison criteria discussed so far, the proposed system can be judged to overcome all of the partial limitations of the existing SCM tools. What is most significant is the enhancement of development productivity through the use of expanded traceability information. Although this paper does not focus on the shortcomings of the proposed system in detail, they include the build automation function, the relationship with integrated development environment (IDE), and support of heterogeneous platform environment. Such areas will be improved and efforts will be made to support a more mature development environment.

TABLE I. COMPARISON OF FUNCTIONALITIES BETWEEN PROPOSED SYSTEM AND OTHER SYSTEMS

	Source Integrity	StarTeam	Synergy/CM	CCC/Harvest	PVCS Dimensions	ClearCase (UCM)	Proposed system
Branch creation and parallel development support	O	O	X	X	X	X	O
Visualized version model	O	O	O	X	O	O	O
SCM change management	X	X	O	O	O	O	O
Integration of traceability	X	X	X	X	X	O	O
Development process management	X	X	O	O	O	O	O
Artifact sharing and simultaneous work support	X	X	X	X	X	limited	O

VII. CONCLUSION AND FUTURE RESEARCH

This paper has proposed a method of enhancing artifact traceability for the purpose of providing developmental productivity on a higher dimension while maintaining the goal aspired by typical SCMs. In contrast to SCM systems that manage changes under strict planning and control, in the individual working environment, since changes are freely made and an integrated management of such changes becomes a method of enhancing development efficiency.

The SCM system provides global traceability to all members of an organization with respect to checked-in configuration items and local traceability is maintained by individuals managing artifact changes in their own working

environment. Change management based on integration of global and local traceability supports consistency in change management and can provide higher productivity in software development.

Connecting the revised version of a configuration item, which is a basic element in showing global traceability in the SCM system, to an artifact version in the working environment is one method of integrating global and local traceability. Configuration items that have been checked-in as baselined documents in the SCM system provide a baseline with regard to artifact changes connected to them. By displaying which checked-in revised version of a configuration item has connected with which version of an artifact may provide a logical foundation for future changes and allows accurate tracing of the change trend.

Using this research as a foundation, further research should be undertaken to provide a broader meaning of traceability that can illuminate the relationship between artifacts. If in this research traceability was a result of the relationship between a single artifact and a configuration item, in the future, research regarding the traceability between multiple artifacts and, furthermore, a plan of managing the traceability of artifacts on a project level is necessary. From the point of software development to the completion of a product, a variety of artifacts are created. Although this makes the process of examining the relationship between all artifacts and the management of traceability very complicated and tedious, it is a task that absolutely needs to be undertaken in order to guarantee the quality and productivity of software, which is continuously growing in size and complexity.

ACKNOWLEDGMENT

This research was supported by Chungnam University Industry Collaboration Foundation and Human Resources Development Consortium for Next Generation Software in Information Technology.

REFERENCES

[1] M. Kögel, "Towards Software Configuration Management for Unified Models", CVSM'08, May 17, 2008.

[2] R. Conradi and B. Westfechtel, "Version Models for Software Configuration Management", ACM Computing Surveys, vol. 30, issue 2, June 1998, pp. 232-282.

[3] J. Estublier, G. Clemm, et al., "Impact of Software Engineering Research on the Practice of Software Configuration Management", ACM Trans on Software Engineering and Methodology, vol. 14, no. 4, Oct. 2005, pp. 383-430.

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

[5] A. Sarma, D. Redmiles, et al., "Empirical evidence of the benefits of workspace awareness in software configuration management", Proceedings of the 16th ACM, 2008, pp. 113-123.

[6] D. Junqueira, T. Bittar and R. Fortes, "A fine-grained and flexible version control for software artifacts", Proceedings of the 26th annual ACM international conference on Design of communication, Lisbon, Portugal, Sept. 2008, pp. 185-192.

[7] "Glossary", ASME Boiler and Pressure Vessel Code, Section Ⅲ, Article NCA-9000.

[8] O. Gotel and A. Finkelstein, "An Analysis of the Requirement Traceability Problem", Proceedings of the 1st IEEE International Conference on Requirement Engineering(ICRE), April 1994, pp. 94-101.

[9] G. Spanoudakis, "Plausible and Adaptive Requirement Traceability Structures", Proceedings of the 14th international conference on Software engineering and knowledge engineering, Ischia, Italy, July 2002, pp. 135-142.

[10] B. Ramesh, M. Edwards, "Issues in the Development of a Requirement Traceability Model", IEEE In Proc. of the 1st Intl. Symposium on Requirements Engineering, San Diego, CA, Jan. 1993.

[11] L. Murray, A. Griffiths, P. Lindsay and P. Strooper, "Requirements Traceability for Embedded Software – an Industry Experience Report", www.itee.uq.edu.au/~pal/SVRC/tr00-41.pdf, 2000.

[12] "IEEE Guide to Software Requirements Specification", IEEE 830-1998.

[13] M. F. Bashir, M. A. Qadir, "Traceability Techniques: A Critical Study", INMIC'06, IEEE Multitopic Conference, Dec. 2006, pp. 265-268.

[14] S. P. Berczuk and B. Appleton, "Software Configuration Management Patterns, Effective Teamwork, Practical Integration", Addison Wesley, 2002.

[15] J. Estublier, S. Garcia, "Process Model and Awareness in SCM", SCM 2005, Sept. 2005.

[16] http://www.mks.com/products/sie

[17] http://www.borland.com/us/products/starteam/index.html

[18] http://www-01.ibm.com/software/awdtools/synergy/

[19] http://www.ca.com/us/products/product.aspx?ID=255

[20] http://www.serena.com/products/pvcs/index.html

[21] http://ww-01.ibm.com/software/awdtools/clearcase

[22] C. Schwaber, "The Forrester Wave™: Process-Centric Software Configuration Management, Q4 2005", Forrester Research, Inc., Nov. 2005.

[23] http://www.bitkeeper.com/Comparisons.ClearCase.html