Proceedings of the Korean Nuclear Society Autumn Meeting
Taejon, Korea, October 2000

SIS-RT: An Integrated Software Inspection Support and Requirement Traceability Tool

Han Seong Son, Seo Ryong Koo, and Poong Hyun Seong

Korea Advanced Institute of Science and Technology
Department of Nuclear Engineering
373-1 Kusong-dong, Yusong-gu, Taejon, Korea 305-701

Dae Seong Son and Seong Soo Choi

Atomic Technology International
KAIST HTC #4408
62-1 Hwam-dong, Yusong-gu, Taejon, Korea 305-348

Abstract

This article introduces a computer-aided software inspection support tool, SIS-RT, which has requirement traceability analysis capability. Inspection and requirement traceability analysis are widely believed to be the most effective software verification and validation methods. These techniques are labor-intensive and thus require to be partially automated. SIS-RT is designed to partially automate the software inspection process and requirement traceability analysis. This tool is on the construction. After further development efforts, SIS-RT will turn out to be a unique and promising software verification and validation tool.

I. Introduction

Inspection is widely believed to be an effective software verification and validation (V&V) method. It can provide a great increase in both productivity and product quality, by reducing development time, by removing more defects than is possible without using inspection, respectively. Inspection applies to the whole lifecycle. By inspecting products as early as possible, major defects will be revealed sooner and will not be propagated through to the final product.

However, software inspection has been found to be difficult to put into practice [1]. This can be attributed to several factors [2]. Software inspection is labor-intensive, and it can be difficult to justify the investment in time and money to introduce it. Although the investment is reasonable when compared with the benefits, there may be a reluctance to devote the resources necessary to them, especially part way through a project when progress has shipped behind schedule. This labor-intensive nature is compounded

by a view that since software inspection uses little technology, they do not fit in well with a more technology-oriented development environment. These problems are mainly due to a lack of understanding. Many software engineers have heard of software inspection, but fewer know enough detail to be able to implement it. They are aware of the often-quoted benefits, but are unwilling to risk implementing inspections. They feel much more secure with traditional methods such as testing. Another obstacle for software inspection is that they are difficult to implement properly. If implemented wrongly, they will produce poor results, in comparison with the effort expended. When this happens, people are discouraged from using inspections again, and apocryphal tales of how "inspection was disaster for us" soon spread.

However, software inspection is gaining in popularity. More people are using it and gaining benefit from it. At the same time, new variations are being created which are tailored for certain types of products or for use under certain circumstances. In order to promote the application of software inspection, the authors are developing a software inspection support tool, SIS-RT. SIS-RT is designed to partially automate the software inspection process that is labor-intensive.

Requirement traceability analysis is to identify requirements that are either missing from, or in addition to, the original requirements. The requirement traceability applied to the software architecture phase can aid in identifying requirements that have not been accounted for in the architecture. Stepwise refinement of the requirements into the architecture produces a natural set of mappings from which to derive the requirement traceability. For large systems, automation is desirable. In this work, requirement traceability analysis is considered as one of the items of software inspection. This is the motivation of integrating requirement traceability analysis capability into the software inspection support tool. SIS-RT is on the construction. After further development efforts, This tool will turn out to be a unique and promising software verification and validation tool.

II. Introduction to Software Inspection

Since M.E. Fagan first defined the software inspection process in 1976 [3], there have been many variations of software inspection [4, 5]. We describe here the original method.

II-1. Inspection Team

An inspection team generally consists of four to six people. Each person has a well-defined role as follows:

Moderator: The moderator is the person in overall charge of the inspection. It is the moderator's task to invite suitable people to join the inspection team, distribute source materials and to organize and moderate the inspection meeting itself.

Author: The inspection requires the presence of the author of the product under inspection. The author can give invaluable help to the inspectors by answering questions pertaining to the intent of the document.

Reader: During the inspection meeting, it is the reader's job to paraphrase out loud the document under

inspection.

Recorder: It is the recorder's duty to note all defects found along with their classification and severity. Although Fagan indicates that this task is accomplished by the moderator, another member of the team is usually chosen, since the workload involved can be quite high, though mainly secretarial. The recorder is often known as the scribe.

Inspector: Any remaining team members are cast as inspectors. Their only duty is to look for defects in the document.

II-2. Inspection Process

Fagan describes five stages in the inspection process:

Overview: The entire team is present during the overview. The author describes the general area of work then gives a detailed presentation on the specific document he has produced. This is followed by distribution of the document itself and any necessary related work to all members.

Preparation: Each team member carries out individual preparation, consisting of studying the document to gain an understanding of it. Errors in the document will be found during this stage, but in general not as many as will be found at the next stage. Checklists of common defect types can help the inspectors concentrate on the most beneficial areas of inspection. Each inspector produces a list of comments about the document, indicating defects, omissions and ambiguities.

Inspection: The inspection meeting involves all team members. The reader paraphrases the document, covering all areas. During this process inspectors can stop the reader and raise any issue until a consensus is reached. If an issue is agreed to be a defect, it is classified as missing, wrong or extra. Its severity is also classified (major or minor). At this point the meeting moves on. No attempt is made to find a solution to the defect; this is carried out later. After the meeting, the moderator writes a report detailing the inspection and all defects found. This report is then passed to the author for the next stage.

Rework: During rework, the author carries out modifications to correct all defects found in the document and detailed in the moderator's report.

Follow-Up: After the document has been corrected, the moderator ensures that all required alterations have been made. The moderator then decides whether the document should be re-inspected, either partially or fully.

III. Features for Tool Support

Manual software inspection is labor intensive. By automating some pats of the process and providing

computer support for others, the inspection process has the capability of being made more effective and efficient, thus providing even greater benefits than are normally achieved.

A desirable attribute of inspections is rigor. Using computers to support the process helps provide this rigor, and improves the repeatability of the inspection process. Repeatability is essential if feedback from the process is to be used to improve it. In this section we describe some features of inspection that are suitable for the application of tool support that was suggested in [6].

Document Handling: The most obvious area for tool support is document handling. Traditional software inspection requires the distribution of multiple copies of each document required. Apart from the cost and environmental factors associated with such large amounts of paper, cross-referencing from one document to another can be very difficult. Since most inspection documents are produced on computer, it is natural to allow browsing of documents online. Everyone has access to the latest version of each document, and can cross-reference documents using, for example, hypertext. Furthermore, documents should not be restricted to text only. The ability to inspection diagrams, as well as use them as supporting documentation, is invaluable. These features demonstrate that computerizing documents is not simply a medium change, but provides an opportunity to enhance the presentation and usability of those documents.

The comments produced by inspectors (also known as annotations) are a major part of the inspection process, as they indicate when an inspector takes issue with a part of the document. In the traditional inspection, they are recorded on paper. Computer support allows them to be stored on-line, linked to the part of the document to which they refer. They can then be available for all inspectors to study both before and, more importantly, during the inspection meeting.

Individual Preparation: There are several ways in which tool support can assist in individual preparation, in addition to the document handling and annotation facilities described above. Automated defect detection can be used to find simple defects such as layout violations. This type of defect, while not being as important as such items as logic defects, must still be found to produce a correct document. If finding them can be automated, inspectors can concentrate on the more difficult defects that cannot be automatically found and that have a greater impact if not found. This may be achieved by the introduction of new tools, or the integration of the inspection environment with existing tools. The latter is obviously preferable. There are various levels of integration, from simply reporting defects to actually producing relating to the defect for the reviewer to examine.

Computer support can provide further help during individual preparation. Generally, inspectors make use of checklists and other supporting documentation during this stage. By keeping these on-line, the inspector can easily cross-reference between them. On-line checklists can also be used by the tool to ensure that each check has been applied to the document, thereby enforcing a more rigorous inspection, while on-line standards, such as those pertaining to the layout of documents, assist the inspector in checking a document feature for compliance.

Meeting Support: Intentionally, or otherwise, some members of the team may not spend sufficient time on individual preparation, but will still attend the group meeting and try to cover up their lack of preparation. Inevitably, this means that the inspector in question will have little to contribute to the group meeting, thus wasting both the group's time and the inspector's time. Computer support can help avoid this situation by monitoring the amount of time spent by each inspector in preparation. The moderator can use this information to exclude anyone who has not prepared sufficiently for the group meeting, or to encourage him or her to invest more effort.

Since the guidelines state that a meeting should last for a maximum of only two hours [3], it may take many meetings to complete an inspection. There is a large overhead involved in setting up each meeting, including finding a mutually agreeable time, a room to hold the meeting and so forth, and there is also an overhead involved for each participant travelling to the meeting. By allowing a distributed meeting to be held using conference technology, it may be easier for team members to 'attend' the meeting using any suitably equipped workstation. Furthermore, use can be made of existing electronic meeting support, such as that described by Nunamaker et al [7]. When a distributed meeting is taking place, it can sometimes be useful to conduct polls to quickly resolve the status of an issue. This is especially important if the meeting is being held in a distributed environment. Automatic support for this can greatly increase the productivity of a meeting.

Data Collection: An important part of inspection is the collection of metrics which can be used to provide feedback to improve the inspection process. The metrics will include such data as time spent in meeting, defects found, overall time spent in inspection and so forth. Collecting these metrics is time consuming and error-prone when carried out manually, so much so that Weller states [8]:

"…you may have to sacrifice some data accuracy to make data collection easier…"

This is obviously undesirable. Computer support allows metrics from the inspection to be automatically gathered for analysis. This removes the burden of these dull but necessary tasks from the inspectors themselves, allowing them to concentrate on the real work of finding defects. Furthermore, the computer can often be used for analyzing these metrics with little further work. This is unlike manual data collection, where the data has to be entered before it can be analyzed. Automated data collection has the advantage of being less defect-prone than manual counterpart, with the added bonus of being capable of providing more finely grained data.

IV. SIS-RT

In this section we describe SIS-RT, which is a computer-aided software inspection support tool developed in this work. SIS-RT stands for Software Inspection Support and Requirement Traceability. As mentioned before, we have integrated requirement traceability analysis capability into the software inspection support tool because requirement traceability analysis is considered as one of the items of software inspection. SIS-RT is designed to support inspection of all software development products. In

addition, SIS-RT is a PC-based application designed for use by anyone who needs to manage requirements. It supports an extraction function that reads a text file and copies paragraph numbers and requirement text to a SIS-RT file. It can read any text data that is convertible to '.txt' format. It also supports manual addition of individual requirements and import from various formats.

SIS-RT permits users to associate database items by defining attributes; attributes attached to individual database items provide a powerful means to identify subcategories or database items and manage requirements. SIS-RT supports normal parent/child links to manage requirements. Furthermore, it supports peer links between items in the database and general documents to provide an audit trail showing compliance to quality standards or contractual conditions.

Figure 1 shows a screen shot of the requirement extraction function of SIS-RT. SIS-RT reads source document, identifies requirement, and extracts them for import into the database. SIS-RT automatically finds and extract requirements based on a set of keywords defined by the user. As requirements are found, they are highlighted as shown in Figure 1. The user may also manually select and identify requirements. SIS-RT enables us to produce a user-defined report that shows various types of inspection results. Users build up the architecture of the reports that they want to produce on the right-hand side window shown in Figure 1. If a user write down checklists in the window, SIS-RT can directly support the software inspection with this functional window.

As mentioned before, SIS-RT supports normal parent/child links and peer links between items in the database and general documents. This is a function related to requirement traceability analysis. Figure 2 shows a screen shot representing the requirement traceability function of SIS-RT. Figure 2 shows that SIS-RT provides mechanisms to easily establish and analyze traceability through the real-time visual notification of change. This capability allows users to pinpoint its impact across the project and assess coverage for verification and validation.

Another function of SIS-RT is supporting software inspection meeting. In order to allow a distributed meeting to be held, SIS-RT introduces a conference technology based on the web technique. Through the web site shown in Figure 3, the moderator can prepare an inspection meeting and the inspectors presents their inspection results.

Now we describe SIS-RT in view of the features of tool support enumerated in Section III.

Document Handling: SIS-RT supports document handling very well. It supports cross-referencing from one document to another. As mentioned before, since most inspection documents are produced on computer, it is natural to allow browsing of documents online. Everyone has access to the latest version of each document, and can cross-reference documents using, for example, hypertext. SIS-RT has all these features. SIS-RT can deal with the comments produced by inspectors. They are a major part of the inspection process, as they indicate when an inspector takes issue with a part of the document. SIS-RT allows the comments to be stored on-line, linked to the part of the document to which they refer. They can then be available for all inspectors to study both before and, more importantly, during the inspection meeting.

Individual Preparation: SIS-RT does not have the ability of automated defect detection yet. However, finding them automatically enables inspectors to concentrate on the more difficult defects that cannot be automatically found and that have a greater impact if not found. Thus we are planning to include this capability into SIS-RT.

As mentioned before, computer support for software inspection can provide further help during individual preparation in that, by keeping the checklists on-line, the inspector can easily cross-reference between them. On-line checklists can be used by SIS-RT to ensure that each check has been applied to the document. In addition, on-line standards in SIS-RT can assist the inspector in checking a document feature for compliance.

Meeting Support: SIS-RT can help avoid taking many meetings to complete an inspection. By allowing a distributed meeting to be held using web meeting technology, it becomes easier for team members to 'attend' the inspection meeting.

Data Collection: Computer support allows metrics from the inspection to be automatically gathered for analysis. This is a very important aspect. SIS-RT, however, does not have the data collection capability. Further development effort for SIS-RT will bring the ability of data collection to it.

V. Conclusions

In this study we developed SIS-RT, which is a computer-aided software inspection support tool. SIS-RT also has requirement traceability analysis capability. Inspection and requirement traceability analysis are widely believed to be the most effective software verification and validation methods. These techniques, however, are labor-intensive. Therefore, it is required to automate the activities even partially. SIS-RT is designed to partially automate the software inspection process and requirement traceability analysis. This tool is on the construction. After further development efforts, SIS-RT will turn out to be a unique and promising software verification and validation tool.

[References]

[1] A. F. Ackerman, L.S. Buchwald, F.H. Lewski, "Software Inspections: An Effective Verification Process," IEEE Software, Vol. 6, No. 3, pp. 31-36, May 1989.

[2] G.W. Russell, "Experience with Inspections in Ultralarge-Scale Developments," IEEE Software, Vol. 8, No.1, pp. 25-31, January 1991.

[3] M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," IBM system Journal, Vol. 15, No. 3, pp. 182-211, 1976.

[4] Tom Gilb and D. Graham, Software Inspection, Addison-Wesley, 1993.

[5] W.S. Humphrey, Managing the Software Process, Addison-Wesley, 1989.

[6] F. Macdonald, J. Miller, A. Brooks, M. Roper, M. Wood, "A Review of Tool Support for Software Inspection," EfoCS-6-95, January 1995.

[7] J.F. Nunamaker, A.R. Dennis, J.S. Valaich, D.R. Vogel, J.F. George, “Electronic Meeting systems to Support Group Work,” Communications of the ACM, Vol. 33, No. 2, July 1991.

[8] E.F. Weller, “Lessons from Three Years of Inspection Data,” IEEE Software, Vol. 10, No. 5, pp. 38-45, September 1993.

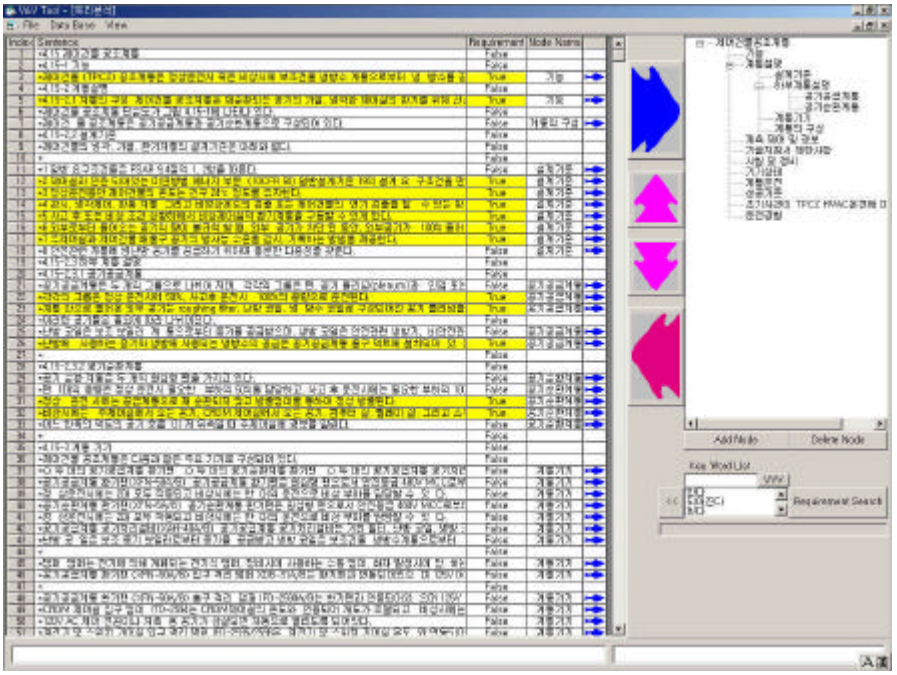

Figure 1. Requirement Extraction in SIS-RT

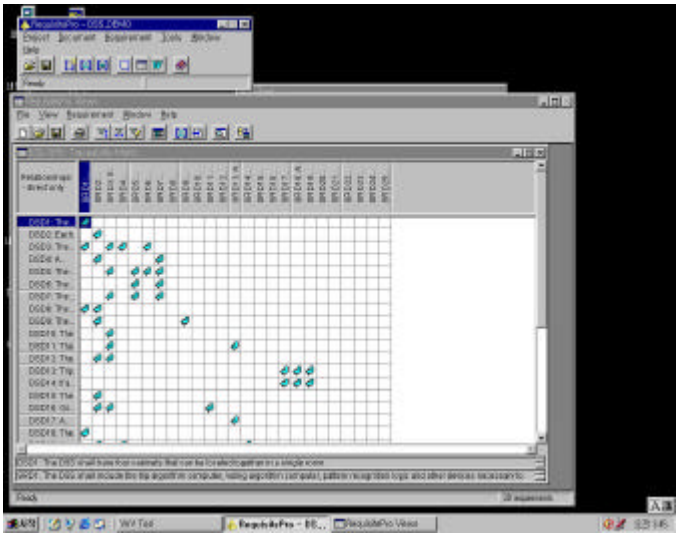

Figure 2. Requirement Traceability Analysis in SIS-RT

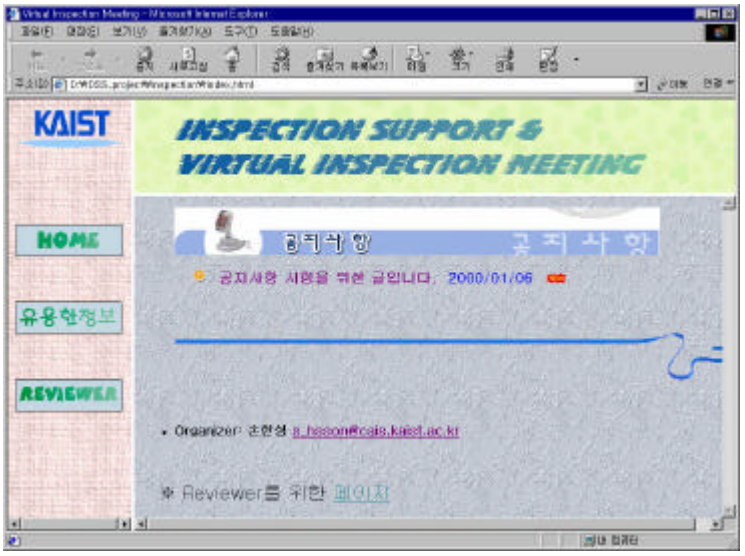

Figure 3. Inspection Meeting Support in SIS-RT