A Path to Virtual Software Inspection

Lasse Harjumaa, Henrik Hedberg and Ilkka Tervonen
Department of Information Processing Science,
University of Oulu, P.O.Box 3000, FIN-90014 OULUN YLIOPISTO
Lasse.Harjumaa@oulu.fi, Henrik.Hedberg@oulu.fi, Ilkka.Tervonen@oulu.fi

Abstract

A number of computer tools have been introduced to overcome the difficulties in traditional software inspection. Actual inspection tools are rarely used, however, as the current tool implementations have some critical inadequacies.

This paper briefly summarizes the evolution of computer-supported software inspection and outlines the concept of virtual software inspection, which introduces two important aspects: flexibility and integration of the tools. Paying attention to these dimensions could result in more attractive and comprehensive tools for inspection collaboration.

1. Introduction

Traditional software inspections [3, 5] are often considered too laborious and time consuming to arrange, as development teams are geographically distributed and team members are involved in many projects simultaneously. Finding a time and place for inspection meetings can be tricky. Furthermore, there are a number of error-prone routine tasks, such as issue logging and material handling. To overcome these difficulties and automate the process, a variety of software tools for inspection collaboration have been developed.

With computer support, the management of inspection material and the data gathered – actions such as the recording metrics and distribution of documents - should be a straightforward matter. Furthermore, this enforces a proper level of rigor in the form of deadlines and fairly strict workflow control. The most important aspect of computerization, however, is that it enables the most laborious and costly parts of the process to be reorganized into a more effective form.

The World Wide Web further enhances the feasibility of computerized inspections since the web is global and work in a virtual environment is not limited to normal working hours. The HTML document format is supported by all of the most popular word processors, so that virtually all written material becomes effortlessly available for inspection. Web browsers are already widely installed and used, and as practically all developers are already familiar with the web platform, the need for training is minimal. The World Wide Web provides a powerful infrastructure for collaborative inspection tools, featuring prevalence, platform independence and familiarity [1, 7, 14].

As Porter and Johnson state, face-to-face meetings do not make the defect detection process significantly more effective (or less effective, for that matter) [17], and several other sources [11, 12] show that there are no substantial differences in efficiency between traditional and computer-supported inspections, and that inspections can even gain from computer support. [4]

Three web-based inspection tools and prototypes have been developed and tested in the University of Oulu, and the results of several experiments provide very similar evidence to that gained from other research performed in this area: Computerized inspection is feasible, and the concept of a distributed inspection tool with the ability to handle documents, checklists and annotations electronically is much appreciated.

The Web has removed many of the problems which may have hindered the employment of inspection tools, especially the need for specific client-server software. However, even with web-based tools present, inspections are often carried out manually or not at all. It seems that existing tools still have some shortcomings. The latest experiments and research have identified further reasons

This work was supported by Academy of Finland under grant 74780.
which may impede the full utilization of inspection tools. To ensure successful adoption of inspection tools, these challenges must be confronted.

2. A brief history of inspection tools

Computer tools are an important part of an efficient software inspection process. As development teams are geographically scattered and time limits are strict, the best solution is to arrange inspections virtually through a network. There are a number of research reports providing evidence that computer-supported inspections really work, e.g. [7] and [16].

The first inspection tools were aimed at supporting manual inspections, mainly providing help in error-prone logging tasks. The fundamental idea was the enabling of on-line issue recording. One of the first inspection tools providing that functionality was ICICLE [2]. The next crucial step forward was taken in Scrutiny, which made distributed inspections possible [6]. The CAIS tool then brought in asynchronous inspection meetings [15]. Experiments with these tools immediately established the concept of time and place-independent inspection.

AISA [19] pioneered reviews of more expressive document types than plain text. As the HTML format was now supported, the next logical phase was to transfer the inspection process to the World Wide Web. WWW-based inspections were demonstrated with AISA, WiP [7] and WiT [8], for example.

Johnson [11] and Macdonald [13] suggested that the underlying inspection process should be customized for the needs of a particular project. The CSRS and ASSIST tools enable such customizing. If this idea is taken further, by increasing flexibility and integrating the inspection tool into other development software, we can reach true and efficient virtual software inspection.

Each inspection tool has its strengths. It seems that developers have had their own aspirations and ideas regarding computer-supported inspection, so that each tool implementation stresses some specific aspect of the software inspection method. In consequence, a list of requirements for an ideal tool has matured, although none of the implementations includes all the desired features.

A great deal of inspection tool research has focused on comparing computer-supported methods with traditional ones in terms of defect finding efficiency. Some experiments have aimed at automatic defect detection or other technical details. In our opinion, however, the inspection process should first be adjusted to be as effortless and efficient as possible.

3. Virtual software inspection

Although the purpose of inspection tools is to provide assistance, they have reshaped the whole process to some extent. Inspector roles, tasks and phases of the process have been adapted to match up to distributed working. Thus, computer-supported software inspection is counted as a method of its own, separate from traditional software inspection. This computerized process could be called virtual software inspection.

Virtual software inspection conforms to a defined workflow (or process) and is performed in a distributed manner with the aid of an inspection tool. The process can be lightweight or all-inclusive, including asynchronous and synchronous phases. It is usually carried out through a network, but traditional meetings can be included if necessary. Geographical distribution is typically achieved through the World Wide Web. An inspection tool is a software package particularly designed for inspection collaboration, and it should be capable of at least managing and delivering the inspection documentation online, enabling the effortless recording of defects and automatic gathering of metrics.

Experiences with several tests in software companies show that there are three important aspects to virtual software inspection: tools, flexibility and integration. All of these must be taken into consideration in order to implement the inspection process effectively [8]. These characteristics are described in more detail below.

3.1 Tools

Computer tools are the essence of the virtual inspection process. There are a number of tools and prototypes available for inspection support. An example of an inspection tool is presented in figure 2, which depicts SATI (Site Annotation Tool for Inspection). This facilitates reviewing and commenting on any HTML-based documentation, including linked, structured documents.

The document under inspection - which in this example contains hyperlinks to related material – is presented in the main browser window. Findings are reported using a simple dialogue, and after submission they are stored in a centralized database, thus allowing their distribution to all participants. Each annotation is connected to an element in the HTML document, or to an individual word. The scope can be decided by the inspector while making the comment [10].
SATI is an example of a document-oriented inspection tool, placing emphasis on the ability to review content-rich material. Tools of this class are mostly web-based, and the document format that it supports is HTML. On the other hand, there are tools that focus on categorization and analysis of the defects. ICICLE, for example, provides the means for automatic defect detection to some extent, and ReviewPro [18] allows very exact and helpful defect classification. Yet another approach is constituted by the defect reporting tools. These do not necessarily enable distribution of the material or commenting on it on-line, but instead they have comprehensive set of forms and templates for reporting the findings.

An ideal inspection tool would include characteristics of all these tool classes, of course, but choosing between the objectives usually has some influence on the tool capabilities. Automatic defect detection is impossible for document formats other than plain text (including code), so that if the tool must be able to manage more complex documents, the actual reviewing has to remain manual.

The fundamental elements of virtual inspection are included in the majority of available tools: distribution of the inspection material electronically, facilities for recording and attaching comments (suspected defects) directly on the document, classification of defects according to their severity and category, checklist management and reporting of inspection efficiency.

In addition to SATI, two other inspection tool prototypes have been developed and experimented with. Each tool implementation and experiment has focused on a specific aspect of inspection: The web-based software package WiT (Web inspection Tool) places emphasis on the virtual inspection process, while another web tool, Inspection Window, demonstrates the flexibility aspect.

Experiments with these tools [9] have proved the usefulness of the web-based approach. The tools are convenient to use, as they are independent of time and place and enable on-line commenting. It was also discovered, however, that these prototypes are not adequate when inspecting large document bundles, especially if the documents were not originally written in HTML, or if they have to be fetched from separate data repositories.

3.2 Flexibility

Flexibility of the inspection process and inspection tools means independence of time and place, and most importantly, tailorability. Some organizations wish to carry out inspections exhaustively, while others – usually those that are just learning the process – desire simple, yet efficient inspections, such as pair inspection [20].

Most inspection tools that are available support only one, fixed, inspection process model, which is not necessarily the most suitable for the organization. If significant modifications to existing working procedures are required, initiation of an inspection tool (and process) will probably fail. Why carry out inspections if it only means more work?

Inspection tools should provide sufficient tailorability to enable effortless utilization of inspections. Instead of a fixed process model, they should provide capabilities for customizing the process for an individual organization or project. There are only a couple of examples of such tools [2, 13].

In addition to the process, flexibility should encompass the documents reviewed. Code is not the only type of document that has to be inspected. Artefacts from every phase of development should be reviewed. Thus, inspection tools must deal with requirement documents, designs and other content-rich documentation including text, images and possibly hyperlinks. Even code, if it is assembled in a visual development tool, may contain more information than pure text lines.

Inspection tools should be aware of the type of document under inspection. In a design phase, for example, if an UML diagram is under inspection, the tool should automatically associate design and UML-specific checklists and gather metrics, which are defined especially for design documents. Clearly, more emphasis must be given to complex sets of documentation in future inspection tool research.

As well as numbers and metrics, inspections always include implicit information to some extent. Although
discussions about technical details and possible solutions to problems are discouraged in the original software inspection method, it must be acknowledged that such improvement ideas often emerge in the course of inspection meetings and that they are also often useful. Current inspection tools are strictly limited by their communication channels and the methods used for exchanging and storing experience data. A straightforward way is to establish a general support system for virtual meetings in conjunction with the inspection tool.

### 3.3 Integration

An easy-to-use inspection tool is not a synonym for an easy inspection process. In fact, making the reviewing procedure as easy as possible for the individual inspection team members may result in an extraneously troublesome form of tool and process administration, thus shifting the difficulties from the inspectors to the moderator or system administrator [8].

There are two reasons why the full utilization of inspection software is extremely challenging: the variety of the inspection material qualities, and interfaces with other development tools and procedures.

As inspections should be carried out on artefacts in every phase of development, there is a need for inspections of content-rich documentation, including text, images and possibly hyperlinks. As checklists, metrics and even annotating methods may vary depending on the characteristics of the documents, managing them and the attached annotations is not straightforward from the point of view of the tool.

Not only are the document formats and their dependences complex, but also there usually exist different versions of them. Ensuring that the inspectors are accessing the correct revision at all times and preventing changes to the document under inspection when the reviewing process is still in progress should be one of the main responsibilities of an inspection tool.

The inspection tool must exchange information with other tools through a number of connection points. Document management is an obvious example of the interoperability need. The inspection tool must be aware of how the documents are produced, where their revisions are located and how they can be accessed. In many cases, the documents to be inspected and related material have to be retrieved from several separate systems.

There must be a mechanism for workflow control. Inspection tasks and resources (people, time) are initiated in the project plan, and so they should be imported from a project management system. Accordingly, the project plan should be updated as the inspection tasks are completed. Inspections also usually produce new action points (rework), which it would be convenient to read off directly from the inspection tool database and insert into the task list in the project management tool. Some of the relations of an inspection tool to other software are illustrated in Figure 3.

Most distributed inspection tools are based on the web. As web services and servers are usually very restricted and kept isolated from production systems for security reasons, administration of an inspection tool may require a great deal of manual work: copying files, granting user access across servers and systems and cleaning them up after the inspection is completed. The manual handling of data exchange between several repositories is error-prone and strenuous [8, 9].

As a branch of the software inspection tool research at the University of Oulu, an XML-based interoperability framework was implemented and briefly tested. This consists of three legacy applications (a version control system, a project management tool and a change management tool) and an inspection tool. This prototype framework allowed the inspection tool to fetch the necessary data from other applications and minimized administrative and out-of-core tasks. Despite some problems with the semantics between applications, the solution was felt to provide a good start to the larger integration of inspection and other software development tools, and it will be further developed to overcome the integration challenge [9].

### 4. Conclusions

In the changing world of software engineering, even the most rigorous and traditional processes – like software
inspection - are bound to change. The conventional method is not necessarily adequate when development is fast and distributed. Although a number of computer tools enabling time and place-independent inspections exist, they are rarely used. A new path must be taken in inspection tool research and implementation to ensure their successful employment.

An inspection tool should accommodate to the underlying architecture of the methods and tools used in the organization. This requires that the tool should be tailorable, easily administered and connected to the other tools that are already in use. To support every phase of the software development cycle, the tool should be capable of managing a variety of document types – such as design diagrams and hyperlinked documents – without strenuous conversion procedures.

Flexibility and integration issues should be adequately addressed in inspection tool implementations to make them more attractive and widely used. Successful tool employment involves tool and process demonstrations, a precisely defined set of metrics and management sponsorship. Furthermore, the maturity of the whole software process in an organization must be adequate before inspection tools are installed.

Perhaps the most important enhancement that inspection tools need is interoperability. Making use of existing data repositories, authoring tools, version management systems, collaborative systems and project management tools could significantly encourage the employment of inspection tools. Integration would ensure easy initiation and running of virtual software inspections.

The role of virtual software inspection will be even more important in the future than it is today. To meet the challenges inherent in modern software development environments, arranging reviews virtually seems the only competent solution. A great deal of work will be required, however, to refine inspection tools to exploit their full potential.

5. References