An Evaluation of Inspection Automation Tools

Vesa Tenhunen and Jorma Sajaniemi

University of Joensuu, Department of Computer Science,
P.O. Box 111, FIN-80101 Joensuu, Finland

Abstract. A key element in manufacturing quality software is the early detection of defects which can be fostered by inspection techniques. Inspections may be boosted by automated tools that help in various tasks during the whole inspection process. We present a framework for inspection automation tool evaluation and use it to evaluate four tools. The framework divides tool requirements in two dimensions: phase and viewpoint. The former deals with the successive phases of the process, and the latter considers various issues that must be dealt with during the process.

1 Introduction

A key element in manufacturing quality software is the early detection of defects throughout the whole software production life cycle. If defects are not removed as soon as possible they give rise to other defects which may be hard to trace and are expensive to remove. Early defect detection can be fostered by inspection techniques [3,4] yielding an effective way to quality [1,11]. It is, however, sometimes hard to motivate people to spend enough time doing inspections. Especially, when customers are needed for inspections, e.g., in inspecting a Requirements Document, proper participation is often found to be a problem. Any mechanism that helps to show that inspections are an important premise for project success is therefore helpful for achieving good quality products.

Software process improvement (SPI) is only possible if the state of a process can be measured and articulated in a rigorous way so that the effect of improvement acts can be evaluated [5]. Measurements should be automated as far as possible to get objective data and to relieve developers from extra work [7]. A good solution is to integrate measurement collection into the tools that are used for software production.

Inspection automation tools may provide a solution for both of the above problems: they can give the customer – and other inspectors as well – an increased feeling of the importance of inspections, and help SPI in providing metrics of the inspected product and the inspection process itself.

Existing inspection automation tools [6,12,13,10] vary in their scope for inspection process phases and tasks, and support for various document formats and computer platforms. Therefore, a framework for tool evaluation is needed. The framework should take into account the whole inspection process and provide a possibility to make judgments based on individual needs and emphasis.

J. Kontio and R. Conradi (Eds.): ECSQ 2002, LNCS 2349, pp. 351–361, 2002.

In this paper, we will present such a framework (Chap. 3) and use it to evaluate major inspection automation tools (Chap. 4). To get a basis for the framework we will start with an overview of the inspection process (Chap. 2).

2 Inspection Process

Inspection as a method for finding defects within software engineering process was first described by Fagan [3] in the early 1970's. Since then, inspections have become quite popular and they are used during the entire life cycle. Inspections have been tested empirically and they have been found to be an effective way to ensure quality [1,11].

Despite some variations, most inspection processes still follow closely Fagan's original proposal. Fagan divides the process into five separate stages: (1) Overview, (2) preparation, (3) Inspection, (4) rework, and (5) follow-up.

The people involved are cast into well-defined and distinct roles: moderator, producer (or author), inspectors, reader, and recorder.

Usually the process goes as follows: In the overview, the moderator selects the inspection team members and assigns them their roles. The producer then introduces the product (code, document etc.) that is to be inspected.

During the preparation stage, the inspectors familiarise themselves with the material. In Fagan's original description, the inspectors are supposed to gain an understanding of the work – defect detection is only a by-product. In a variation suggested by Gilb and Graham [4], the inspectors should explicitly focus on finding defects.

In the inspection the moderator oversees the event; the reader (or the producer) paraphrases the document in full. The inspectors can stop the reader at any time and tell their comments which they have found in advance or at the meeting. All comments are written down by the recorder. No attempt to fix the defects is made – the inspection must focus solely on finding defects. The end product of this stage is a document containing all the found defects and problems.

It is also possible to have an inspection as distributed (non-local). The participants "meet" electronically using a suitable conferencing software to send their comments to each other. Another possibility is to replace the meeting by an asynchronous inspection, where the moderator oversees that the inspectors complete their tasks before the inspection moves forward [9].

The next stage is rework, where the producer reviews all defects. In the final stage, the follow-up, the moderator ensures that all comments are addressed and then decides whether there is a need for a full or partial re-inspection.

In the following, we will use a slight variation of these inspection stages. We rename overview as planning, for even in Fagan's proposal, there are more activities in this stage than just overseeing the start of the process. Preparation of materials, assignment of tasks to roles and assignment of roles to participants cannot be done without proper planning.

Phase	Moderator	Producer	Inspector	Reader	Recorder
Planning	Start process	Provide materials			
Preparation			Search for defects		
Familiarisation		Check findings			
Inspection	Oversee meeting	Clarify findings	Search for defects	Go through the materials	Record findings; write a summary
Rework		Remove defects			
Follow–up	Check rework				

Fig. 1. Roles and tasks in inspection process

The preparation is the same as above, but we add a new phase between that and the inspection: familiarisation, where the producer steps through the inspectors' remarks and makes her own comments about them. The rest of the inspection process follows as stated previously. Figure 1 lists the phases and participants' major tasks in them.

In practice, companies often have their own inspection process, where some phases may be joined and some tasks done in a different phase. This has no effect on the evaluation framework, as we have used the phases as an analysis tool rather than a fixed process description.

3 Framework for Tool Evaluation

There are several goals in automating the inspection process. Among the most important ones are increasing rigour, effectivity, and efficiency [8,9].

MacDonald et al. [8] have presented requirements for inspection support tools. They look for each phase in the inspection process and point out the elements that could or should be automated. We use a similar approach, but create a more detailed view of the process. We examine each phase from four viewpoints:

- Material Handling (all document-related issues such as file format, versions, navigation etc.)
- Comment Handling (all comment-related functions, such as adding, viewing, reviewing, reporting, summarising etc.)
- Support for Process (features that automate or otherwise benefit the inspection process)
- Interfaces (use of external software or other tools)

In each case, we determined whether there's a possibility for automation and if so, what kind of support the tool should provide. With this division, it was possible to get a more detailed view of tool requirements. We included previously reported requirements [9,2], but the viewpoints suggested some new requirements, too. We have left out voting support suggested by MacDonald et al. [9], as we feel that support for voting is not a necessity in a process where defects are supposed to be found but not resolved.

Phase	Material Handling	Comment Handling	Support for Process	Interfaces
Planning	Distribute materials		Defining tasks for roles; casting persons for roles	Different document types; paper documents
Preparation	Navigation; automatic defect detection; support for comprehension	Recording; summary reporting	Monitoring time usage, completeness of review, and completeness of checklist usage	Checklists
Familiarisation	Navigation	Walkthrough; reviewing comments		
Inspection	Navigation	Selecting and modifying final comments; summary reports	Time tracking; summaries for time usage etc.	Checklists
Rework		Listing unfinished comments; bundling corrections		Change control system
Follow–up		Checking for rework		

Fig. 2. Requirements for inspection automation tools

The resulting evaluation framework is presented in Figure 2 and described in more detail below.

Planning. A tool can support material handling by automating the distribution of materials. It can help in support for process, when the moderator defines tasks for roles and casts persons to those roles. The tool may automate interfaces by supporting various kinds of documents, like different file formats (including paper documents).

Preparation. The tool should provide support in every category for this phase. In material handling, it should help navigation, i.e. moving around and searching in the document, even though this may contradict the previous requirement of supporting different document formats. It should also find (at least some) defects automatically and support inspectors' needs to gain understanding of the material.

In comment handling, the tool should enable the recording of comments with information about the comment's location, classification, explanation and references. Another requirement is the ability to create summary reports as needed.

For process support, the tool should monitor inspector's use of time and check that the inspector has completely gone through material and checklists.

The tool should support interfaces by facilitating the use of checklists.

Familiarisation. In this phase, the tool should provide support for the producer: walkthrough and review of comments, and navigation within the material as the producer evaluates the comments.

Inspection. The tool should help comment handling by supporting selection and modification of final comments, and also creation of summary reports (even covering several inspection meetings). It should also support the moderator's tasks in inspection meeting by keeping track of time, so that if too much time is consumed in secondary issues, the moderator can urge to move on. It should also make summaries of time usage. As for interfaces, the tool should support the use of checklists in this phase, too.

Rework. In rework, automation is most useful in comment handling. The tool can help by listing unfinished comments and by enabling the producer to bundle corrections together. For interfaces, the tool can work in cooperation with a change control system.

Follow-Up. The tool should support comment handling by automating the moderator's overview of finished rework.

4 Tool Evaluation

We will now apply the evaluation framework to four inspection automation tools: CSRS, ReviewPro, sfia and Microsoft Word. This selection of tools is based on our search for currently supported inspection automation tools. Selected tools are particularly designed for inspection automation except Microsoft Word, which is included as an example of word processors that are sometimes used for inspection purposes. A summary of the analysis is presented in Figure 3.

4.1 CSRS

CSRS, Collaborative Software Review System [6], is a tool for Formal Technical Asynchronous review method (FTArm), which is a modification of Fagan's inspection. In FTArm, nearly all parts of the inspection process are managed asynchronously: all participants have certain tasks, and when they are completed, the inspection can continue to the next stage. CSRS works on Unix operating system.

Material Handling. CSRS manages material using a database where documents are stored as nodes which are linked together. A node can be a whole file or a portion of it (e.g. a function within code). Different file formats are not supported. Only plain text files can be used, because CSRS is built on XEmacs editor.

The pages of a document can be easily navigated, and standard XEmacs search functions are available. There is no automatic fault detection, neither there is any special support for comprehension. Automatic fault detection can be added by using CSRS's process modelling language.

Comment Handling. There can be three different types of comments, or annotations, as they are called in CSRS: a *comment* is a public note available to all inspectors and it is used for making questions and answering them in a general level. An *issue* is a private inspector's note which is used to address a defect. An *action* is also a private note and used to suggest an action to correct a defect.

All comments are stored as nodes and they are linked to the relevant parts of the document, so that they can be viewed concurrently with the content which they refer to.

Comments can be categorised with pre-made classifications of the types of the defect and its severity. When inspectors have finished their preparation tasks, comments can be reviewed in a public review which corresponds to an inspection meeting. CSRS ensures that all comments are visited.

	CSRS	ReviewPro	sfia	MS Word
Material Handling				
distribution	−	++	−	+
navigation	++	−	+	+++
automatic defect detection	−	−	−	−
comprehension support	−	−	−	−
Comment Handling				
comment attributes	+++	+++	+++	−
walkthrough	++	+	+++	+
summary reports	++	++	++	−
reviewing & modifying	++	++	++	++
listing unfinished	+	+	++	−
Process				
tasks & roles	−	+	−	−
monitoring time usage	+	+[1]	−	−
summaries of time usage	+	+[1]	−	−
monitoring completeness	+	+	−	−
monitoring checklists	+	+	−	−
Interfaces				
different document types	−	++	++	+
checklists	+	+	−	−
change control system	−	−	−	−

− = not supported, + = supported, ++ = good support, +++ = extensive support

[1] partially manual

Fig. 3. Summary of the inspection tool analysis

Support for Process. CSRS keeps logs of each inspector's usage of time. It also sees that every node is reviewed and that checklists are completely gone through. CSRS doesn't support role casting and task assignments unless they are added using the system's programming language.

CSRS is not meant to provide support for inspection meetings. It only helps the moderator to summarise decisions and to produce a report of results.

Interfaces. CSRS does not support different file formats or paper versions. It does support checklists which are included in the node database.

Summary. CSRS is aimed to support FTArm, a method that differs from traditional Fagan inspection. Yet the program is quite versatile because of its built-in process modelling language, so it can be tailored to support many variations of inspection processes. However, it works only with text files.

4.2 ReviewPro

ReviewPro[1] 3.0 [12] is a web-based groupware application that supports both synchronous and asynchronous inspections. All inspection data is kept in a server database and the reviewers need only web browsers. ReviewPro is available for Windows and Unix.

Material Handling. Material can be of any type as ReviewPro is not used to handle it. ReviewPro can store copies of documents in different file formats. These documents are downloadable from the server and other programs are used to manage them.

Navigation or searching of the documents cannot be done with ReviewPro. It is meant to be used simultaneously with a word processor that shows the document, or with paper versions. ReviewPro is used to record inspectors' comments.

Comment Handling. Comments are linked to documents manually by a separate log page describing their place within the document (page, paragraph etc.). Attributes of a comment include type, severity, status, and description. All comments are saved into the database.

ReviewPro can create several types of reports from comment information. Comments can be reviewed and modified both in synchronous and asynchronous inspections.

Support for Process. ReviewPro includes some support for defining roles and tasks. The moderator can create task lists for every participant.

The use of checklists and the control of their usage is automated. Tracking and summarisation of time usage requires that inspectors enter the data manually.

Interfaces. ReviewPro requires that the inspected material is available to inspectors in some form. There are no restrictions to document types or their file formats.

As many as six checklists can be created to be used in preparation and inspection. ReviewPro checks that inspectors have completely gone through them before the inspection moves to the next phase. There is no support for a control change system.

Summary. ReviewPro serves as a versatile note pad that inspectors can use to save and review their comments. It leaves the handling of inspected materials to other tools. As a web-based application, ReviewPro provides support for distributed, asynchronous and traditional inspection processes.

[1] This evaluation is based on the demo version of ReviewPro. Consequently, some functionality could not be assessed.

4.3 sfia

sfia 1.0 (software for inspection automation) [13] is a tool for inspection of code and documents in every stage of software development. It uses pictures to represent documents and XML-type plain text files for inspection data. By utilising GIF files as document pages and Tcl/Tk as the implementation language, sfia can be used with basically any document type and in various operating systems (including Unix, Linux, Windows and Macintosh).

Material Handling. sfia does not use the material per se; instead it manages images of the material, a graphic representation in GIF format of each page of the original document. Thus it gets around the problem of different file formats and works also with scanned paper documents.

sfia does not convert files to pictures, but there's an abundance of tools which can be used to do that. The producer can create the pictures and then write a definition file, which tells sfia what files comprise the document and what classifications of comments are used.

There is no automatic fault detection nor support for document comprehension. The pages of the document can be easily navigated, but there's no search function for document content.

Comment Handling. The main feature of sfia is its ability to handle comments. Inspectors can add comments and at the same time attribute classification, severity and other criteria to them. These classifications are freely definable by the moderator (or company). Comments are shown as semitransparent marks in the insertion point and they can be viewed simultaneously with their content texts. Inspectors can also view at any time detailed or summary reports of the comments.

For inspection meeting, all comments made by several inspectors can be combined into a single file. Comments can be deleted as necessary and also marked according to their status: finished, unfinished and notified.

Using summary reports, the producer can view unfinished comments, make modifications to original material and edit comments and change their status.

Support for Process. sfia does not support definition of roles or tasks. It does not record logs of time usage and neither does it track completeness of preparation or inspection.

Interfaces. Based on the use of standard graphics format it is possible to use sfia with any kind of file formats, including paper. However, sfia does not support any third party software like change control systems. Also, there is no support for checklists: they can be used only separately and manually.

Summary. sfia works in virtually all systems and it provides all necessary support for comment handling. However, it leaves process support and content-related issues like search and comprehension to be managed with other content-aware tools.

4.4 Microsoft Word

Microsoft Word 97 [10] is a word processor, but it is widely used in many organisations also for document handling. The program has some characteristics that can be used to automate inspections, so it is appropriate to review it. Word can be used on Windows and Macintosh.

Material Handling. Word can keep track of different versions of documents. The producer can prepare the material using Word's protection functions: the inspectors can be granted rights to add only comments, add comments and modify the text, or add comments and tracked changes. Documents can be distributed directly from Word via e-mail.

Navigation through document is supported as well as search of content, but there is no fault detection or comprehension support.

Comment Handling. Inspectors can add comments to documents. Comments can be either text or recorded sound files. Depending on the granted rights, inspectors may also modify documents.

By using track changes feature, Word keeps track of any modifications inspectors make and it is later possible to merge all copies into a single document with all the comments and modifications.

The comments can not be classified automatically: the inspectors must add manually the appropriate class, severity and other information into each comment's text. Word also lacks all summarising or reporting features; it can only show a listing of comments made by every inspector or by one selected inspector.

In rework, the producer can step through the comments, make necessary modifications and edit comments to show that they are dealt with. If there are any tracked changes, the producer can accept or discard them.

Support for Process. Word does not offer any support for the inspection process. There is no functionality regarding roles and tasks; it does not keep track of time usage (although it records the time when a modification is made); and there is no way to create reports about comments, although it is possible to print a list of comments and modifications.

Interfaces. Word provides some support for different file formats, but added comments can only be saved in Word's own format. This can cause a variety of problems, if the members of the inspection team have different versions of the program.

By using Word's macro language, Word can be made to support the use of checklists (which must be also in Word document format). There is also a possibility to ease distribution of the material by using e-mail directly from Word. The e-mail connection provides also some sort of support for asynchronous meetings, even though the program has no direct facilities for them.

Summary. Word is a familiar tool for many users. It does have some functionality usable in an inspection process, but it does not recognise any special features of inspection, so it requires disciplined working from users. It can be utilised with Word documents only.

5 Conclusions

Inspection automation tools motivate inspectors, especially customers, to do their best by providing an increased feeling of the importance of inspections. Furthermore, tools can automate metrics collection and help in SPI.

We have presented a framework for inspection automation tool evaluation. The framework divides tool requirements in two dimensions: phase of the inspection process and viewpoint of the contemplation. Our viewpoints are: material handling, comment handling, support for process, and interfaces. Using this framework we evaluated four tools: CSRS, ReviewPro, sfia and Microsoft Word.

CSRS is based on a special inspection method and it supports plain ASCII files only. ReviewPro serves as a versatile note pad with a manual connection to the inspected material. sfia supports any document type and gives good support for comment handling but has no process support. Finally, Microsoft Word is a word processor that provides no special support for inspections, and can be used with caution when Word documents are inspected by disciplined users.

The number of inspection automation tools is quite limited and a need for new tools applicable in different environments is apparent. Our evaluation framework can be used in tool design, too. It lists the major tasks that inspection participants encounter and thus poses requirements for future inspection automation tools.

References

1. Doolan, E. P.: Experience with Fagan's inspection method. Software – Practice and Experience, Vol 22, No. 2 (1990) 173–182
2. Dunsmore, A. P.: Comprehension and visualisation of object-oriented code for inspections. Technical Report, EFoCS-33-98, Computer Science Department, University of Strathclyde (1998)
3. Fagan, M. E.: Design and code inspections to reduce errors in program development. IBM System Journal, Vol. 15, No. 3 (1976) 182–211
4. Gilb, T., Graham, D.: Software inspection. Addison-Wesley (1993)
5. Humphrey, W. S.: Managing the software process. Addison-Wesley (1989)
6. Johnson, P. M., Tjahjono, D.: Improving software quality through computer supported collaborative review. Proceedings of the Third European Conference on Computer Supported Cooperative Work (1993)
7. Kitchenham, B.: Measuring software development. In Software Reliability Handbook, Ed. by P. Rook, Elsevier Applied Science (1990) 303–331
8. MacDonald, F., Miller, J., Brooks, A., Roper, M., Wood, M.: A review of tool support for software inspection. Proceedings of the Seventh International Workshop on Computer Aided Software Engineering (1995) 340–349
9. MacDonald, F., Miller, J., Brooks, A., Roper, M., Wood, M.: Automating the software inspection process. Automated Software Engineering Vol. 3, No. 3–4 (1996) 193–218
10. Microsoft Corporation. http://www.microsoft.com/office/word/
11. Russell, G. W.: Experience with inspections in ultralarge-scale developments. IEEE Software, Vol. 8, No. 1 (1991) 25–31

12. SoftWare Development Technologies (SDT). http://www.sdtcorp.com
13. Tenhunen, V.: sfia – software for inspection automation. http://www.cs.joensuu.fi/pages/saja/se/sfia