Autom Software Eng (2006) 13:373–394
DOI 10.1007/s10851-006-8531-5

A family of experiments to investigate the effects of groupware for software inspection

Stefan Biffl · Paul Grünbacher · Michael Halling

Abstract It is widely accepted that the inspection of software artifacts can find defects early in the development process and gather information on the quality of the evolving product. However, the inspection process is resource-intensive and involves tedious tasks, such as searching, sorting, and checking. Tool support for inspections can help accelerating these tasks and allows inspectors to concentrate on tasks particularly needing human attention. Only few tools are available for inspections. We have thus developed a set of groupware tools for both individual defect detection and inspection meetings to lower the effort of inspections and to increase their efficiency. This paper presents the Groupware-supported Inspection Process (GRIP) and describes tools for inspecting software requirements. As only little empirical work exists that directly compares paper-based and tool-based software inspection, we conducted a family of experiments in an academic environment to empirically investigate the effect of tool support regarding defect detection and inspection meetings. The main results of our family of experiments regarding individual defect detection are promising: The effectiveness of inspectors and teams is comparable to paper-based inspection without tool support; the inspection effort and defect overlap decreases significantly with tool support, while the efficiency of inspection teams increases considerably. Regarding tool support for inspection meetings the main findings of the experiments are that tool support considerably

S. Biffl (✉)
Institute of Software Technology and Interactive Systems, Vienna University of Technology, 1040 Vienna, Austria
Tel: +43 1 58801-18810
e-mail: Stefan.Biffl@tuwien.ac.at

P. Grünbacher
Christian Doppler Laboratory for Automated Software Engineering, Johannes Kepler University Linz, 4040 Linz, Austria
Tel: +43 70 2468 8867
e-mail: paul.gruenbacher@jku.at

M. Halling
Department of Finance, University of Vienna, 1210 Vienna, Austria
Tel: +43 1 4277 38081
e-mail: Michael.Halling@univie.ac.at

lowers the meeting effort, supports inspectors in identifying false positives, and reduces the number of true defects lost during a meeting. The number of unidentified false positives is still quite high.

Keywords Software inspection · Defect detection · Inspection meeting · Tool support · Software quality measurement · Controlled experiment · Empirical software engineering

1. Introduction

It is widely recognized that the inspection of software artifacts such as requirements, plans, designs, or code is effective to assess product quality and to reduce the number of defects (Aurum, Petersson and Wohlin, 2001; Biffl and Halling, 2003; Parnas and Lawford, 2003). Inspections can help project managers in managing risks and engineers in suggesting specific improvements. An inspection consists of several clearly defined activities including inspection planning and preparation, individual defect detection, team meeting, as well as evaluation and rework (Fagan, 1976; Gilb and Graham, 1993). Within the inspection process two activities play a crucial role: in *defect detection* individual inspectors examine the inspection object to identify potential defects, possibly by following a detection technique. During the *inspection meeting* the inspectors discuss all reported defects and agree on true defects. The goals of the meeting are to collect defects, to eliminate false positives, and to find new defects.

Although the benefits of inspections are obvious their high costs often inhibit widespread adoption in industry. The inspection process is resource-intensive and involves rather tedious tasks, such as searching, sorting, and checking. Tool support for inspections (MacDonald and Miller, 1999) has therefore been devised to speed up tedious tasks thus helping the inspectors to concentrate on tasks creating the highest value (MacDonald, Miller and Ferguson, 1999). Tool support can also ease data collection for subsequent inspection performance analysis and inspection planning (Laitenberger and Dreyer, 1998; MacDonald and Miller, 1998; MacDonald, Miller and Ferguson, 2002).

It has been observed that due to inappropriate techniques inspection meetings may even lead to a loss of defects identified during individual defect detection (Johnson and Tjahjono, 1997; Bianchi, Lanubile and Visaggio, 2001; Lanubile and Mallardo, 2003). Researchers and practitioners have therefore been trying to optimize the inspection process through specialized techniques (e.g., reading techniques) and tools aiming at reducing the administrative overhead to increase inspection effectiveness and efficiency (Basili, Green, Laitenberger, Lanubile, Shull, Soerumgaard and Zelkowitz, 1996; Thelin, Andersson and Harrell, 2004).

Existing reports on tools for inspections mostly deal with inspections of code rather than early-life cycle artifacts like requirements. Furthermore, little empirical work exists directly comparing paper-based (i.e., manual) and tool-based software inspections. Also, the number of empirical studies evaluating the impact of tool support on inspection performance is very limited (MacDonald and Miller, 1998).

Inspections are also interesting from the perspective of computer-supported cooperative work (CSCW) since they pose challenges to tool developers. For example, an inspection environment has to support individual work as well as team meetings, it has to enable the collaboration of heterogeneous stakeholders, and it has to support a large variety of inspections objects and work procedures.

Based on our experience with manual inspections (Biffl, 2001; Halling, 2002) and previous work on groupware support for requirements negotiation (Boehm, Grünbacher and Briggs,

2001) we developed GRIP (Groupware-supported Inspection Process) (Halling, Grünbacher and Biffl, 2001a). GRIP provides a framework and collaborative tools for an inspection team and supports individual defect detection, team meetings, as well as inspection management. Encouraged by feedback from early trials our goal was to empirically validate the benefits of GRIP. We have thus carried out experiments to compare GRIP to previous large-scale manual inspection experiments. The anticipated benefits of tools are the improved handling of the reference information presented to inspectors and improved communication in the inspection team. Although the introduction of new tools creates additional training effort, we expected a much higher overall effort reduction by streamlining inspections with GRIP. It is not obvious however whether inspectors using an inspection tool are more effective or more efficient.

In this paper we thus present an empirical evaluation of tool- and paper-based inspections. We conducted a family of experiments (Ciolkowski, Shull, and Biffl, 2002) consisting of three related inspection experiments: the first two experiments were manual, i.e., based on paper and pencil; the third experiment was using the GRIP. We conducted the third experiment in an academic environment with 37 subjects to empirically investigate the effect of tool support regarding effectiveness and effort for defect detection tasks of inspection teams during the individual defect detection step and for inspection meetings. Our previous results in two large-scale experiments indicated that meeting losses are on average higher than meeting gains (Halling and Biffl, 2002). This paper thus explores whether groupware tools can improve inspection meetings such that their effort can be justified.

The remainder of the paper is structured as follows. Section 2 summarizes related work on tool support for inspections. Section 3 summarizes the GRIP framework and tool suite. Section 4 describes the empirical research approach: a family of experiments with a focus on the third tool-supported experiment. Section 5 reports the results. Section 6 compares the results to related work and Section 7 summarizes the paper and suggests further work.

2. Existing tool support for requirements inspection

In this section we describe related work on inspection tools with a focus on available empirical data for comparing tool-based and paper-based inspections. Numerous tools and platforms are available for automating software inspections (MacDonald and Miller, 1999). Most tools have a strong focus on reporting and collecting identified defects and there is only little support available for inspection meetings. Also, most of these tools support only specific inspection processes for code inspection, e.g., InspeQ (Knight and Myers, 1993) for the phased inspection method; or tools for fine-grained static code analysis (Anderson, Reps and Teitelbaum, 2003).

There are only few empirical studies on the benefits of automating inspections. Two studies exist that explicitly compare tool- and paper-based inspections. MacDonald and Miller (1999) conclude that in these studies there is often no comparison to a manual inspection process in the same environment which makes it hard to assess how much tools actually improve the performance of inspections. MacDonald and Miller (1998) present a controlled experiment in an academic environment to compare the performance of their ASSIST platform with paper-based code inspections. For code inspections with exogenously determined efforts they report no significant differences between tool- and paper-based inspection regarding defect detection both on individual and team level. With respect to support for inspection meetings the empirical findings are even more limited. Genuchten et al. for example report on a study on applying a group support system (GSS) in code inspection meetings (Genuchten, Cornelissen and Dijk, 1998; Genuchten, Dijk, Scholten and Vogel, 2001). The authors present empirical

evidence that tool support significantly increases performance and the overall contribution of the inspection meeting.

In addition to the problem of weak tool validation, we identified the following shortcomings:

- *Inflexibility with respect to artifact:* Most tools are optimized for code inspections. Therefore they usually provide support for plain text documents and limit inspectors to make annotations to single words or specific lines of text. When inspecting more complex artifacts such as requirements specifications this approach is too simple as navigating/reporting on different level of abstraction is required (e.g., use cases, actors, domain class, document hierarchy, etc.).
- *Process inflexibility:* Most of the tools support only one specific inspection process and therefore cannot be easily applied in different contexts. Furthermore, they are usually specialized on either asynchronous or synchronous inspection processes.
- *Minor support for inspection management:* Existing inspection tools usually do not support inspection management, i.e., the ability to monitor, control, and adjust a process based on a set of metrics. We regard support for inspection management as a key factor for increasing inspection efficiency (Halling, Biffl and Grünbacher, 2002).

3. GRIP framework and tools

In our research on software inspections we have been pursuing the following objectives: (1) Development of collaborative tools and techniques covering the entire inspection process (Halling, Grünbacher and Biffl, 2001b); (2) Integration of technical inspection aspects with management activities such as planning, monitoring, or process analysis (Halling, Biffl and Grünbacher, 2002); (3) Empirical studies to validate the usefulness, effectiveness, and efficiency of our approach (Halling, Biffl and Grünbacher, 2003b); and (4) the improvement of inspection support for informal requirements (Halling, Biffl and Grünbacher, 2003a). The GRIP framework (Fig. 1) has been guiding our research. Figure 1 shows that GRIP covers the complete inspection life-cycle (Laitenberger and DeBaud, 2000) including a planning phase, individual defect detection, a team meeting, the collation of defects, and the final analysis of inspection performance. In this process two roles require support: (a) the inspection manager for planning and monitoring the inspection and for analyzing the inspection success or failure; and (b) individual inspectors who are expected to detect defects:

- The *inspection manager* is in charge of planning and tailoring the inspection process. This includes selecting the inspection guidelines, preparing inspection object(s), and configuring communication channels (among inspectors as well as between inspectors and the inspection manager). During defect detection and during the inspection meeting the inspection manager continuously monitors the ongoing process and makes adjustments if necessary. Finally, measures are analyzed during inspection evaluation.
- *Individual inspectors* are responsible to electronically annotate defects during defect detection. In the inspection meeting they rely on decision support tools to agree on collated annotations.

GRIP is designed to overcome the shortcomings listed in section 2 and is targeted at both inspection managers and inspectors. It provides automatic data collection throughout the inspection process and is tailorable to different inspection process designs (i.e., synchronous/asynchronous, with/without an inspection meeting) (Halling, Grünbacher and Biffl, 2001b). The groupware tools in GRIP are supposed to improve team productivity by supporting people

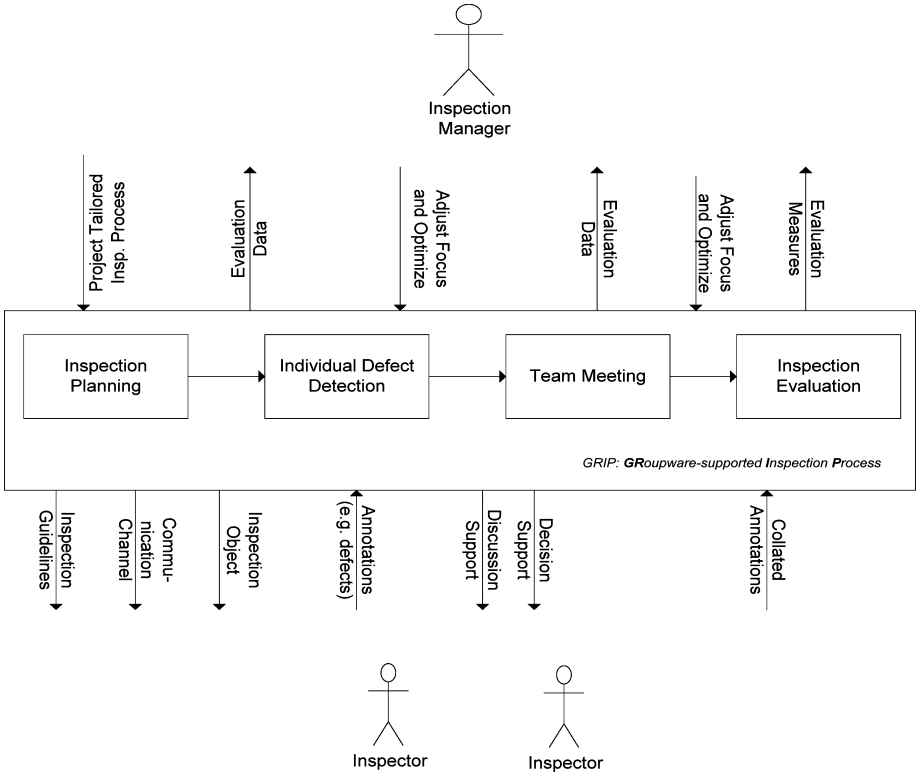

Fig. 1 The GRIP framework (Halling, Grünbacher and Biffl, 2001b).

engaged in the common inspection task. Within the vast number of groupware tools available Group Support Systems (GSSs) focus on supporting group decision-making (Nunamaker, Briggs, Mittleman, Vogel and Balthazard, 1997). We found the GSS GroupSystems® as an ideal infrastructure for GRIP as it is fairly flexible, easy to tailor, easy to use, and robust. In the following we briefly describe the aspects of GRIP we empirically evaluated in our experiments.

3.1. Groupware-support for defect detection

Tool support promises to increase efficiency by reducing overhead costs caused by tedious tasks and by making data collection more accurate. Especially with documents in natural language tools cannot replace inspectors in detecting defects but they can make their tasks easier. More specifically, groupware tools for defect detection are expected to offer the following benefits:

- *Improved information sharing* increases awareness about the defects that are reported by the inspection team members and raises process visibility for the inspection manager. Unlike in a paper-based inspection inspectors are informed about defects and comments contributed by other inspectors (Perpich, Perry, Porter, Votta and Wad, 1997).
- *Clear defect reporting* through one-step entry of defects, no loss or distortion of defect information caused by inadequate handling or consolidation of hardcopy documents.

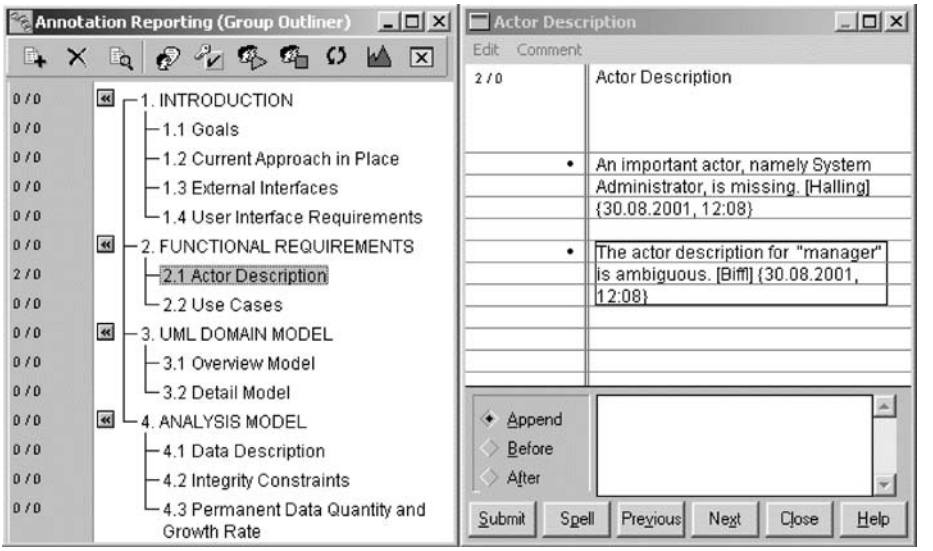

Fig. 2 Reporting Defects in GRIP.

Furthermore tools provide an automatic well-structured electronic documentation of the entire communication.

- *Easier handling of documents* for easier and quicker searching and navigation ultimately causing less distraction from the defect detection task.

In order to support information sharing and clear defect reporting, we arrange the inspection object as a hierarchy illustrating the document's semantic structure (see Fig. 2). In our experiment we used a requirements document in natural language that adopts a small subset of the Unified Modeling Language for diagrams. Each inspector then attaches defect reports to elements of the document structure. For each defect reported the tool automatically records a time stamp and enables inspectors to enter additional defect information, such as severity or type. As soon as a new defect report is entered, the defect information is available to all participating inspectors. Finally, defects are stored in a central database and can either be processed directly in the tool or by additional tools (e.g., for creating statistics).

Defects can be grouped and inspection reports are available on different levels of detail. This makes it easier to determine similar or equal defects and to generate a collated defect list as the cognitive load is considerably reduced.

3.2. Groupware support for the inspection meeting

When discussing inspection meetings it is important to briefly outline the evolution of the inspection process. While (Fagan, 1976) viewed the inspection meeting as the key activity for defect detection, (Parnas and Weiss, 1987) argued to perform defect detection during individual preparation. They proposed to minimize the number of inspectors participating in a meeting as only a limited number of inspectors (usually two) can interact at the same time while others are just listening. These two fundamentally different views on the role of the inspection meeting have to be kept in mind when interpreting reports on empirical results of meeting effectiveness. While Fagan reports that inspection meetings were very effective

(Fagan, 1976), more recent reports show that this is not necessarily the case in inspections with a different focus (Parnas and Weiss, 1985; Votta, 1993; Porter and Johnson, 1997; Bianchi, Lanubile and Visaggio, 2001; Halling and Biffl, 2002; Lanubile and Mallardo, 2003).

However, shifting defect detection from meeting to individual preparation does not necessarily mean abolishing meetings. Land, Jeffery and Sauer (1997) as well as Sauer, Jeffery, Land and Yetton (2000) emphasize the importance of inspection meetings for defect collection and defect discrimination, i.e., the identification of false positives. They report that meetings have a clear advantage over individual defect detection in discriminating between true defects and false positives. False positives are defect reports, which actually are not true defects. According to Land, Jeffery and Sauer (1997) false positives can become a problem if occurring frequently because they incur further costs such as the time needed for fixing them. Johnson and Tjahjono (1997); Johnson and Tjahjono (1998) as well as Porter and Johnson (1997) report similar results and argue that inspection meetings are beneficial mainly due to the considerable reduction of false positives. Vitharana and Ramarmurthy (2003) investigate the issue of anonymity and openness in a GSS inspection meeting in industrial environments which was not a major factor in our investigation. As a result, the inspection team meeting plays an important role in GRIP.

The starting point for a meeting is the inspection object (e.g., a requirements document as described above) together with all annotations (or defect candidates) collected during individual defect detection. During the inspection meeting the inspectors are first asked to assess these defect annotations for severity. All defect candidates above a previously defined threshold are then further classified by using a customizable taxonomy of defects. All defect candidates below a previously defined threshold are considered false positives, i.e., not true defects.

In order to support this group decision task we have been customizing electronic voting tools. *GroupSystems.com's* Alternative Analysis tool allows assessing a set of voting items using a set of criteria with customizable voting methods (e.g., 1–10 scale, ordinal scale, etc.). The tool aggregates all individual ballots thus allowing the quick elimination of defects not worth further consideration. The tool also allows visualizing situations where the team of inspectors had diverging opinions on defect severity. In such situations the inspector manager can use the tool to initiate a discussion about diverging opinions. The tool can also be used to automatically discriminate between true defects and false positives by analyzing the ballots.

Of course, potential benefits of tools have to be compared to the meeting costs. In general, inspection meetings are very costly due to the number of participants and limited opportunities for parallel work in traditional meeting settings. However, as pointed out, tool support can considerably reduce costs by offering means for parallel contributions during a meeting. For example, inspectors can assess the severity of defects in parallel and team discussions can then focus on critical issues only, e.g., defect candidates with controversial ratings. Another important aspect of inspection meeting costs are true defects found during individual defect detection that are classified as false positives during the meeting. The meeting process should ensure to reduce the probability of true defects being classified as false positives. GRIP tries to meet this requirement by focusing inspectors' attention on defects leading in diverging votes.

After experimenting with these different decision support capabilities our goal was to empirically validate their benefits and costs. In particular we were interested to investigate whether tool support can increase the efficiency of team meetings so that meetings would be easier to justify from an economic point of view. Note that the current implementation of GRIP focuses on pure defect discrimination as proposed in Sauer, Jeffery, Land and Yetton (2000). We did not explicitly support other potential meeting goals like defect detection *synergy* or other *soft benefits*. In the case of synergy existing empirical evidence suggests that if the individual preparation phase already focuses on defect detection, inspection meetings

show very little defect detection synergy benefits (see (Bianchi, Lanubile and Visaggio, 2001; Halling, Biffl and Grünbacher, 2003b) for a detailed review of existing material). As far as soft benefits are concerned, it is difficult to measure them empirically.

4. Research approach-empirical studies: Hypotheses

Our research goal was to quantify the effects of GRIP in the context of requirements inspection. We had earlier conducted two paper-based experiments (called *ExpA* and *ExpB* in this paper) (Biffl, 2001; Halling, 2002) and decided to replicate the experimental design using groupware tools in a third experiment, *ExpC*, (Halling, Grünbacher and Biffl, 2001a) to compare the results and to derive empirical implications of the impact of GRIP on inspection performance. In this section we describe the research hypotheses and introduce our family of experiments, focusing on the third experiment.

4.1. Research approach and hypotheses regarding defect detection

In this paper we focus on an empirical evaluation of the influence of different inspection processes on the entire inspection process, i.e., we compare the performance of inspection teams at the end of the inspection. Furthermore, we summarize the performance during individual inspection activities, namely individual defect detection and inspection meetings. Another dimension considered in the analysis involves a defect classification that distinguishes different defect severities. We define major defects to be defects whose removal in later development phases would potentially cause a considerable additional effort if not detected during the inspection (e.g., a forgotten Use Case or a wrong association between two objects).

We use the following evaluation criteria to compare manual and GRIP-based inspections:

- *Effectiveness:* The ratio between the number of reference defects detected (i.e., defects that have been re-seeded by the experiment team) and the total number of reference defects. The defects had been found in several review cycles during the development of the requirements and then re-seeded into the document, i.e. the defects occurred during creation and were not created artificially.
- *Efficiency:* The number of reference defects detected per hour.
- *Effort:* The total effort in staff hours invested by inspectors for individual defect detection. Team effort is simply the sum of efforts of individual team members.
- *Defect overlap:* The ratio between the number of redundant reference defect reports (i.e., the total number of defect reports for a specific reference defect reduced by one) and the total number of detected reference defects.

Based on these evaluation criteria we investigate the following research hypotheses that are derived from the theoretical motivation for GRIP. Basically these hypotheses are evaluated for: the entire inspection process, the individual defect detection activity and the inspection meeting. Of course, some aspects of individual hypotheses may not make sense for all inspection activities. We discuss such situations in more detail as they occur.

ID1. Similar Effectiveness Regarding All Defects: *Teams using GRIP find a similar number of reference defects as teams following a manual inspection process.* The foundation for this hypothesis is that GRIP does not include any concepts that should, from a theoretical point of view, support the inspectors in detecting more reference defects. The newly introduced information sharing capability of GRIP allows inspectors to continuously view all defects reported in a team. There are two possible effects: (a) a positive effect:

due to the availability of information provided by team members inspectors are able to focus on the detection of new and previously uncovered defects rather than re-detecting defects; (b) a negative effect: inspectors may focus on reading defects reported by other inspectors and their individual motivation to detect new defects may suffer, if other team members are very active. However, we cannot identify a theoretical reason why one of the two effects should outperform the other and therefore we expect no significant difference in effectiveness.

ID2. Increased Effectiveness Regarding Major Defects: *Teams using GRIP find a higher number of major reference defects than teams following a manual inspection process.* Regarding a specific defect class, which can be harder to find, the information-sharing capability is expected to increase detection effectiveness. As a result the majority of minor defects will be found and communicated early among team members. This allows inspectors to focus on identifying major defects, which are often hidden by more obvious defects in a paper-based inspection. Therefore, we believe that the positive effect of tool support as outlined in above will persist in this case and will positively influence the effectiveness to detect major defects.

ID3. Reduced Effort: *Inspection teams using GRIP spend less time on individual defect detection, inspection meetings, and thus on the overall inspection than inspectors following a paper-based inspection process.* The key goal of GRIP is to reduce inspection effort by accelerating searching and defect reporting activities. Furthermore, the information-sharing capability potentially reduces inspection effort because less time is spent on redundant defects. Due to tool support we further expect a reduction in meeting effort, as the inspectors can work concurrently and can focus on more important issues.

ID4. Decreased Defect Overlap: *The defect overlap of teams using GRIP is lower than of paper-based inspection teams.* Again this research hypothesis is based on the information-sharing capability, which is supposed to focus inspectors on finding new defects rather than reporting already listed defects.

ID5. Increased Efficiency: *Teams using GRIP are more efficient than teams following a manual inspection process.* Combining expectations regarding similar effectiveness, lower effort and lower overlap yield this research hypothesis for inspection efficiency.

TM1. Efficient Defect Discrimination: *Teams using GRIP for defect discrimination lose fewer reference defects and at the same time eliminate more false positives compared to teams without tool support.* We regard the identification and removal of false positives as an important goal of inspections, as false positives can increase the rework effort dramatically. Although this issue has not received considerable attention in academia so far, we regard it as important for increasing the acceptance of inspections in practice. We expect that GRIP succeeds in reducing false positives.

Defect discrimination suffers from the risk of labeling true defects as false positives and removing them from the final, collated defect list. Usually, the costs associated with the loss of a true defect are considerably higher than the costs of keeping a false positive. However, these costs depend on the specific context of the inspection and have to be assessed individually. We limit ourselves to simply comparing absolute numbers of removed false positives and lost true defects. We calculate these values for paper-based and tool-based inspection meetings and expect that tool-supported meetings are able to remove more false positives while losing less true defects since the tool allows more objective and transparent discussion and voting.

We further briefly analyze a so-called "Automated Defect Discrimination" method based on simple statistical measures for diverging voting results. We expect that this automated procedure further reduces the effort required for defect discrimination while showing a similar performance to the discrimination based on inspectors' discussions.

Table 1 Overview on family of experiments

Experiment year	*ExpA* 1999–2000	*ExpB* 2000–2001	*ExpC* 2002
Number of CBR Inspectors	86	47	37
Number of CBR Teams	16	9	7
Average Team Size	5.4	5.9	5.3
Number of reference defects	86	97	93
Inspection Meeting Conducted?	Yes	No	Yes
Inspection Process Support	Manual	Manual	GRIP

4.2. Study description: Family of experiments

Table 1 summarizes key characteristics about the three experiments. The first two experiments *ExpA* and *ExpB* studied the influence of different reading techniques on inspection performance. For details on *ExpA* and *ExpB* (see Biffl and Halling, 2002; Biffl and Halling, 2003) for a comparison of the two experiments (see Halling, Biffl, et. al., 2001). *ExpA* and *ExpB* used checklist-based reading (CBR) and scenario-based reading techniques. For tool evaluation in *ExpC* we decided to select the CBR technique in order to have a sufficient sample size.

We regard the three experiments as a family of experiments due to the following similarities:

- The experimental design (i.e., controlled experiment in an academic environment), their operation (including planning and tutorial activities), and the experiment administration team were similar in all experiments.
- The subjects participating in the experiment were all selected from computer science students at Vienna University of Technology.
- Checklists, requirements documents, and seeded defects were very similar for all experiments. There have only been minor modifications between the experiments.
- Although the experimental inspection process was changed (e.g., inspection meetings, number of inspection cycles, tool support) individual defect detection was a focus in all three experiments.

We are therefore confident that the overall results of these three experiments can be compared. As far as inspection meeting performance is concerned, we can only compare the results of *ExpA* and *ExpC*. However, the comparison has to be done with caution as inspectors in *ExpA* were asked to identify new defects during the meeting. In *ExpC* the meeting purpose was only to discriminate between false positives and true defects.

4.3. Description of *ExpC*

ExpC involved 37 undergraduate computer science students and almost the same requirements document as in *ExpA* and *ExpB*. The inspection object was a 47-page requirements specification, containing about 13,000 words, 16 UML diagrams, and 97 seeded defects. All seeded defects were found before the experiment during the development of the requirements document in numerous quality assurance iterations. In the experiment, all seeded defects could be found by the inspectors without referring to additional documents. Please

refer to Biffl and Halling (2002) and Halling, Biffl, et al. (2001) for details on these aspects of the experiment.

In the following, however, we focus on the specifics of tool support for inspection meetings. *ExpC* consisted of an individual defect detection activity and an inspection meeting. Details on the individual defect detection step can be found in (Halling, Biffl and Grünbacher, 2003b). The planning step for the experiment included tailoring the tool by preparing the requirements document for defect reporting and customizing the individual inspection process in the groupware tool. A detailed description of the tailoring process is given in (Halling, Grünbacher and Biffl, 2001b). Three tutorials were prepared to teach the inspection process and the tool to participants: 1) a UML tutorial to ensure the proper understanding of the notations used in the requirements document; 2) a tutorial to teach the checklist-based reading technique; and 3) a tutorial explaining the use of the groupware tool.

Although GRIP supports both synchronous as well as asynchronous inspections, the entire inspection process was carried out synchronously to ensure control over the experiment environment. During all process steps an inspection manager supervised each team and was responsible for the accuracy and feasibility of data collected from the inspectors. The inspection manager was also using the groupware tool for his tasks.

In the beginning of individual defect detection, each inspector logged on to the groupware tool and was uniquely identified by the tool, which allowed us to link all defect reports and time stamps to specific inspectors. Each inspector had access to an electronic version of the requirements document via the tool (for quick searching and navigation) and to a printed version (for easier reading and informal note taking). Furthermore, the groupware tool provided information about the checklist-based reading techniques. The groupware tool also presented an outline of the requirements document for defect reporting (see Fig. 1). Down to the third level in the hierarchy, each node of this requirements document hierarchy represented a chapter or subsection of the document. Nodes on deeper levels contained specific instances such as use cases or actors. All inspectors could report new defects and instantly view all defects reported.

Between individual defect detection and the inspection meeting the inspection manager, a member of the experiment staff, generated a collated defect list from all individually reported defects. As this task is supported by GRIP, this (minor) effort is ignored in the further analysis.

Figure 3 summarizes the individual activities performed during the inspection meeting experiment. In a first step participants individually assign a severity level between 1 (i.e., no defect) and 5 (i.e., major defect) to each reported defect. The tool then aggregates all the individual votes for each defect and presents intuitive and illustrative (e.g., traffic light) measures for the homogeneity of submitted ballots. The inspection manager can thus easily identify reported defects where all inspectors have similar opinion regarding severity (in this case no further discussion is necessary) or, more importantly, spot defects where opinions diverge. The definition of homogeneous and heterogeneous voting can be adjusted with thresholds. For the defects with diverging votes the inspection manager then initiates an oral discussion in which the team has to agree upon a severity level for this defect. Finally, inspectors assess the defect type by voting. The results of this second voting procedure are not analyzed in detail in this paper.

4.4. Threats to validity

As any empirical study, this experiment exhibits a number of threats to internal and external validity. While internal validity investigates if the treatment causes the outcome, external

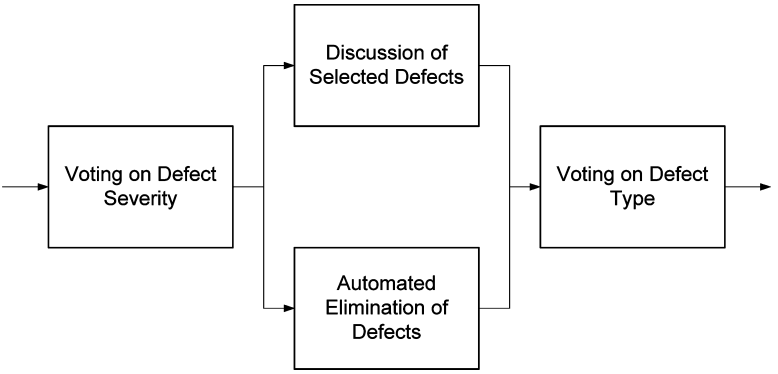

Fig. 3 Inspection Meeting Process in Experiment.

validity deals with generalization (Wohlin, Runeson, Höst, Ohlsson, Regnell and Wesslén, 2000).

Internal Validity. The primary threat to internal validity is *selection*. This comes from the selection of subjects and their assignments to particular treatments. In *ExpA* and *ExpB*, we used randomization to avoid systematic bias from selection. In *ExpC* selection was not an issue as we had only one treatment. However, to ensure comparability of inspection team performance we randomly selected students to form teams.

A second threat to internal validity is *process conformance*. While checking for this was tedious work during *ExpA* and *ExpB*, the groupware tool and the supervision by the inspection manager enabled us to easily enforce process conformance in *ExpC*.

A third threat to internal validity arises from the fact that we did not control the *inspection effort.* However, effort also represents an important dependent variable we want to analyze. We thus think that letting the inspectors decide how much time they wanted to spend on the inspection meeting is reasonable in a controlled, synchronous inspection. However, effort values of *ExpA* and *ExpC* cannot be directly compared as the emphases of the meeting steps where somewhat different.

A fourth threat to internal validity is *data consistency*. As with process conformance, data consistency was much easier to ensure during *ExpC* due to tool support. In *ExpA* and *ExpB*, inspection supervisors checked the completeness and validity of the collected defect and effort data immediately after each step.

External Validity. With respect to external validity, we took *specifications* from a real-world application context to develop an inspection object to represent a realistic situation. The document size and defect density were somewhat above the levels from other reported experiments (Basili, Green, Laitenberger, Lanubile, Shull, Soerumgaard and Zelkowitz, 1996), but not particularly high compared to documents in industrial settings (Gilb and Graham, 1993). Moreover, we used *inspection activities* that had been implemented in a number of professional development environments (Laitenberger and DeBaud, 2000).

The subjects were students participating in a university class. As pointed out in the literature (Sauer, Jeffery, Land and Yetton, 2000) students may not be representative of real developers. However, (Höst, Regnell and Wohlin, 2000) observe no significant differences between students and professionals for small tasks of judgment. According to (Tichy, 2001) using students as subjects is acceptable if students are appropriately trained and the data is used to establish a trend. Both conditions are met in our case. The present study is a controlled

experiment in an academic environment with low external validity for general practice. However, positive results encourage empirical studies with a representative sample of practitioners.

5. Results

In this section we present a comprehensive summary of results regarding the empirical comparison of paper-based inspection and GRIP. Note that these results can be used to empirically evaluate two combined effects: (a) tool-based vs. paper-based inspection and (b) information-sharing vs. no communication among inspectors.

Partial results in this section have been published in two conference papers (Grünbacher, Halling and Biffl, 2003; Halling, Biffl and Grünbacher, 2003b). We considerably extended the analysis presented therein by including (a) a holistic inspection perspective combining individual defect detection and inspection meeting; and (b) a defect classification focusing on major defects (i.e., defects that cause a high effort in later life-cycle phases if not detected during requirements inspection). We now present descriptive summary statistics (i.e., means and standard deviations) for our family of experiments with respect to the research hypotheses proposed in section 4. Furthermore we point out statistically significant differences between means using standard t-tests. For this purpose we use simple two-sample t-tests with homogeneous variances. We evaluate statistical significance against the 5% level and indicate explicitly in the text if different thresholds are used in individual situations. We use these tests to evaluate our research hypotheses stated before. Additional information like the defect detection effectiveness and efficiency of individual inspectors is not tested due to a lack of comparability.

5.1. Individual defect detection

All evaluations presented in this section take into account the performance of inspectors during individual defect detection (i.e., without any potential inspection meeting effects). Furthermore, we concentrate on a team-based perspective to ensure comparability of results. Note that there are no statistically significant differences between the team sizes of the three experiments. The average team size is 5.4 (5.9) [5.3] for *ExpA* (*ExpB*) [*ExpC*], i.e., the variation of team size should not drive the reported empirical results.

The individual effectiveness of inspectors during *ExpC* is not comparable to the individual effectiveness of inspectors from *ExpA* and *ExpB* as GRIP supports information-sharing. This could lead individual inspectors to report very few defects and to find few reference defects because they invest a considerable amount of time in analyzing other inspectors' defect reports and concentrate on identifying additional defects. Therefore, we expect that the information-sharing approach decreases the number of re-detected defects (i.e., reduces the defect overlap). This expectation is confirmed as we observe that inspectors in *ExpC* (mean: 11.5%; standard deviation: 10.9%) detect considerably fewer reference defects than inspectors in *ExpA* (mean: 20.1%; standard deviation: 9.7%) and *ExpB* (mean: 16.2%; standard deviation: 9.3%).

Table 2 summarizes the mean and standard deviation of team effectiveness after individual defect detection for all three experiments for all and major defects. The data shows little differences in mean and standard deviation. For all defects the GRIP inspection exhibits slightly lower team effectiveness. For major defects the performance of GRIP inspection is in between the performance of paper-based inspection in *ExpA* and *ExpB*. Neither teams in *ExpC* are significantly more effective than teams in experiments *ExpA* and *ExpB* nor the other way around.

Table 2 Comparison of team effectiveness (%) during individual defect detection

	All defects		Major defects	
	Mean	Std.dev.	Mean	Std.dev.
ExpA	54.1	10.1	46.5	11.0
ExpB	56.1	9.8	57.1	11.0
ExpC	53.3	10.7	53.5	10.4

The only exception is the performance of *ExpA* regarding major defects, which is significantly lower than the performance in the two other experiments. Overall our empirical data thus suggests that there is little impact of the GRIP process on defect detection effectiveness.

In addition to defect detection effectiveness we want to evaluate the impact of tool-support on the inspection cost factors on defect detection effort and defect overlap (for details refer to (Halling, Biffl and Grünbacher, 2003b)). While the average team effort amounts to 30.2 (29.6) hours in *ExpA* (*ExpB*), it reduces to 21.2 in *ExpC*. A similar but less dramatic reduction can be observed in the case of individual inspection effort. In both cases the reductions are statistically significant (on the 90%-level). GRIP process does not necessarily reduce effort variability. Effort reduction represents an important advantage for tool support. Another important cost factor for requirements inspections is defect overlap, i.e., the redundant detection of defects. One argument for using information sharing is to reduce the defect overlap in an inspection team. Our empirical data shows that differences in overlap between experiments *ExpA* (overlap of 96.7) and *ExpC* (overlap of 16.6) as well as *ExpB* (overlap of 68.1) are statistically significant on the 99%-level. In experiments *ExpA* and *ExpB* defects have been reported up to six times, while during *ExpC* no defects were reported more than three times. Combining our results regarding effort and overlap we come to the conclusion that GRIP successfully reduces important cost drivers of inspection.

Finally, we want to explore efficiency, which puts the previously used measures of effectiveness, effort, and overlap in relation to each other. Similar to the situation regarding effectiveness, the individual efficiency of inspectors during *ExpC* is not comparable to the individual efficiency of inspectors in experiments *ExpA* and *ExpB* due to the information-sharing concept of GRIP, which shifts effort away from defect detection. Inspectors spend more time reading existing defect reported.

Regarding individual defect detection efficiency we can observe, as expected, that inspectors of *ExpC* (mean: 2.7 reference defects/hour of effort; standard deviation: 2.6) have a considerable lower efficiency than inspectors of experiments *ExpA* (mean: 4.1 reference defects/hour of effort; standard deviation: 2.2) and *ExpB* (mean: 3.3 reference defects/hour of effort; standard deviation: 2.0). Table 3 summarizes the mean and standard deviation of team defect detection efficiency after individual defect detection. The positive effects from the reduction of cost factors effort and overlap more than offset the slightly reduced effectiveness for all defects and major defects. The mean efficiency of teams applying GRIP may look considerably higher than the mean efficiency of teams applying paper-based inspection. However, the observed differences are not statistically significant and suffer from comparatively large standard deviation especially for *ExpC*.

5.2. Inspection meeting: Defect discrimination results

In this section we describe key empirical results regarding tool support for inspection meetings. We analyze defect discrimination performance including (a) team discussions and

Table 3 Comparison of team efficiency (number of reference defects detected per hour inspected)

	All defects		Major defects	
	Mean	Std.dev	Mean	Std.dev.
ExpA	1.7	0.5	0.6	0.2
ExpB	1.9	0.6	0.9	0.3
ExpC	2.5	1.1	1.2	0.6

(b) fully automated discrimination. Furthermore, we present information on meeting effort. We also compare the results from the tool-based meeting to data from paper-based inspections where possible.

Based on our experience with inspection meetings and the prevailing opinion in recent research, inspection meetings promise little benefits with respect to the detection of new defects (Halling and Biffl, 2002). Inspection meetings can however reduce rework effort by removing false positives reported during individual defect detection. The focus of this empirical study is on defect discrimination performance. So far, the identification and reduction of false positives have hardly been discussed in literature, although practitioners frequently point out that a large number of false positives can dramatically reduce the benefits of inspections. With respect to the identification of false positives we focus exclusively on the data from *ExpC*. In the case of *ExpA* we performed a meeting but lack detailed information regarding the discrimination of false positives. *ExpB* was intentionally designed without an inspection meeting due to resource and timing constraints.

When summarizing our empirical results (Grünbacher, Halling and Biffl, 2003) we observed that in *ExpC* the number of defects reported is on average reduced by 11% (standard deviation 8%), the number of false positives by 16% (standard deviation 10%) and the number of reference defects by 3% (standard deviation 6%). Although some true reference defects are lost during GRIP inspection meetings (on average 2.7 defects), the overall performance of tool-supported inspection meetings documents that such meetings are effective for discriminating between true defects and false positives. It is quite difficult to compare the results from *ExpA* and *ExpC* because the inspection meetings had different goals. While the only purpose of the inspection meeting in *ExpC* was to identify false positives, the meeting in *ExpA* also had the goal to find new reference defects. Summarizing the results of team meetings in *ExpA* we observe on average 32.2 lost and 23.2 newly detected reference defects (Halling and Biffl, 2002).

When comparing the overall performance of inspection meetings including lost and newly found reference defects we can observe that tool-supported inspection meetings outperform paper-based inspection meetings. Although there were new defects identified during inspection meetings in *ExpA*, these gains could by far not outweigh the significant meeting losses. In the case of *ExpC* there were no meeting gains, but overall meeting losses were on average reduced by a factor of three. Note that the standard deviation of meeting performance is very high in *ExpA*, indicating that meeting performance varied a lot. In *ExpC* the clearer and simpler focus of meetings also reduced standard deviations considerably. These results confirm our expectation that tool usage supports the meeting process well.

In the previous section we described meeting performance with respect to the meeting process including discussion among participants on defects with diverging votes. However, as a comparable benchmark to the performance of these meetings we developed and evaluated a simple automated defect discrimination technique in GRIP. The advantage of this technique is a further reduction of meeting effort, as no discussions need to take place. The automated

defect discrimination is based on analyzing the inspectors' votes for defect severity. The inspection manager can define a threshold for the average team's vote on a defect's severity. If this vote is below the threshold the defect is classified as false positive and removed from the defect list; if the vote is above the threshold the defect is classified as a true defect and included in the final defect list.

To determine the optimal threshold one has to consider the trade-off between the removal of false positives and the loss of reference defects because both values increase with a more stringent threshold. In fact, one would require a cost-benefit model that compares the benefits from removing another false positive and the costs from losing another reference defect. Such a cost-benefit model represents an important goal for future research in this area.

Finally, we analyze the meeting effort invested in experiments *ExpA* and *ExpC*. As we applied somewhat different meeting processes and defined different meeting goals the effort values are not directly comparable. Nevertheless our data shows that the meeting in *ExpC* (on average 100 min per inspector) took considerably less time than the meeting in *ExpA* (on average 293 minutes). Furthermore the standard deviation is reduced to a reasonable level, while it was very high for *ExpA*. Note that we could even further decrease meeting effort by using the automated defect discrimination approach (see Table 5).

5.3. Overall inspection performance

The results presented in the previous sections address individual steps in the inspection process. In this section we present empirical results that analyze the overall inspection process. In this comparison one has to keep in mind that there was no inspection meeting in Experiment B. Throughout this section we compare the performance of *ExpA*, *ExpB* and *ExpC*. In the case of *ExpC* we distinguish between the implementation with discussion during inspection meetings and the one with automated defect discrimination (based on a threshold of 1.75, which represents a reasonable compromise between high false positive reduction and low reference defect loss as illustrated).

Table 4 summarizes the overall inspection effectiveness, i.e., the percentage of reference defects detected at the end of the entire inspection process. As expected, we cannot observe any significant differences. The highest inspection performance could be observed in *ExpB*, both for all and major defects. However, inspection performance of GRIP is slightly above the performance of paper-based inspection in *ExpA* and only slightly below the one in *ExpB*.

Following our general strategy Table 5 summarizes the total inspection effort. It is obvious that *ExpB* without an inspection meeting should have the lowest effort values. This is actually the case but with surprisingly little difference to *ExpC*. Both experiment show significantly lower effort than the inspection process of *ExpA*.

Table 4 Overall comparison of inspection effectiveness (%)

	All defects		Major defects	
	Mean	Std.dev.	Mean	Std.dev.
ExpA	48.2	16.6	44.1	15.9
ExpB	56.1	9.8	57.1	11.0
ExpC, Disc.	52.2	12.1	52.5	14.6
ExpC, T = 1.75	49.8	11.4	50.8	15.0

Table 5 Overall team effort in staff hours

	Mean	Std.dev.
ExpA	48.7	10.9
ExpB	29.6	4.3
ExpC, Disc	30.0	8.1
ExpC, Auto	28.2	7.6

Team effort values cannot be directly compared because they also depend on the average team size. However, as pointed out before there are only slight differences in average team size, and therefore the influence of team size on these results is almost negligible.

Another important dimension of inspection performance we evaluated is the number of false positives. We think that inspection processes that result in a high number of false positives are of little practical use. Table 6 shows how the number of reported defects and false positives distributes among the different experiments. Here, we observe a significant advantage of GRIP and tool support because during *ExpC* both the number of defect reports and false positives have been reduced dramatically (to approximately 50% of the values observed in paper-based inspection without information sharing). Although the analysis of false positives is less usual in academia, we believe that this result is an important argument for tool support in practice.

Combining the different measures from above we end up with the efficiency measure, i.e., the number of defects reported per time unit. Table 7 summarizes our results with respect to this perspective. We observe that both *ExpB* and *ExpC* significantly outperform *ExpA* in this dimension. Note that, however, there was no inspection meeting in *ExpB* and therefore it is quite impressive that the GRIP inspection process resulted in similar efficiency measures as the paper-based inspection process in *ExpB*. Again, we cannot observe notable differences with respect to defect severity. It seems in general that the observed empirical results are quite independent of these defect classes.

6. Discussion

In this section we summarize the most important results of the presented empirical study with respect to our research hypotheses and present possible explanations.

Table 6 Number of defect reports and false positives after inspection

	Number of defect reports		Number of false positives	
	Mean	Std.dev.	Mean	Std.dev.
ExpA	212.9	63.7	171.5	58.5
ExpB	245.8	37.6	191.4	38.1
ExpC, Disc	121.0	26.4	67.3	19.6
ExpC, T = 1.75	110.4	26.1	64.1	22.1

Table 7 Overall team inspection efficiency (number of reference defects found per staff hour)

	All defects per hour		Major defects per hour	
	Mean	Std.dev.	Mean	Std.dev.
ExpA	0.9	0.3	0.4	0.1
ExpB	1.9	0.6	0.9	0.3
ExpC, Disc	1.7	0.8	0.8	0.4
ExpC, T = 1.75	1.8	0.8	0.9	0.5

6.1. Individual defect detection

ID1. Teams using GRIP find a similar number of reference defects as teams following a manual inspection process. This hypothesis was confirmed: We could not observe a significant difference between tool- and paper-based inspections with respect to the number of defects identified. The average effectiveness of *ExpC* is slightly below (53.3%) *ExpA* (54.1%) and *ExpB* (56.1%). It seems that information sharing does not specifically increase overall defect detection effectiveness. These results are consistent with earlier empirical comparisons of tool- and paper-based inspections (MacDonald and Miller, 1998).

ID2. Teams using GRIP find a higher number of major reference defects than teams following a manual inspection process. This hypothesis was not confirmed. In our family of experiments major defects were similarly hard to find as all defects for paper-based and tool-based processes. Our explanation is that either the effect of uncovering defects with tool support is not noticeable since only few defects actually were covered, or the uncovering is not very effective.

As far as inspection effort, defect overlap, and inspection efficiency are concerned, our research hypotheses are all confirmed on statistically significant levels.

ID3. Inspection teams using GRIP spend less time on individual defect detection, inspection meetings, and thus on the overall inspection than inspectors following a paper-based inspection process. This hypothesis was confirmed: The total effort for the individual defect detection phase in *ExpC* is significantly lower than the effort observed in the other two experiments. As average team sizes do not differ significantly between the three experiments, the difference in inspection effort can be attributed to the use of information-sharing tools.

ID4. The defect overlap of teams using GRIP is lower than of paper-based inspection teams. This hypothesis was confirmed. One important goal of GRIP was to reduce defect overlap through information sharing. While overall inspection team effectiveness was not increased, we observe that the defect overlap was dramatically reduced in *ExpC*. During *ExpA*, on average, every reference defect was reported redundantly (overlap of nearly 100%). The overlap was similarly high in *ExpB* (2 out of 3 reference defects were reported redundantly). In contrast, in *ExpC* the overlap was reduced to 16%, meaning that only a little bit more than every 6th reported reference defect was reported redundantly.

ID5. Teams using GRIP are more efficient than teams following a manual inspection process. This hypothesis was confirmed. We observed that GRIP-based inspection teams inspected more efficiently than paper-based inspection teams. In detail, teams of *ExpC* managed to detect 50% more reference defects per hour of effort than teams of *ExpA* and one third more defects than teams in *ExpB*.

We can therefore summarize that in this empirical study the overall picture regarding individual defect detection strongly supports the use of GSS-based tools to improve inspection performance. Although there was no significant increase in defect detection effectiveness, we were able to increase inspection efficiency considerably. As inspections are very cost-intensive the reduction of cost factors represents an important result and a strong argument for using tools in practice. However, further empirical analysis has to evaluate the impact of tool support on the entire inspection process including the efficiency of inspection meetings and the effort required for inspection planning and inspector training. In *ExpC*, for example, the additional effort required for inspector training was up to two hours per inspector.

6.2. Inspection meeting

A further hypothesis is dealing with support for software inspection meetings:

TM1. Teams using GRIP for defect discrimination lose fewer reference defects and at the same time eliminate more false positives compared to teams without tool support. This hypothesis was confirmed. We focused on the performance of tool-supported meetings where the main purpose was to discriminate between false positives and true defects to reduce the rework effort after inspection. Our empirical data illustrates that tool support results in good discrimination performance and reduces the number of false positives by 15% on average.

However, it is even more important to evaluate the number of lost true defects, as these meeting losses usually incur higher costs than any false positive not removed during meeting. With respect to this criterion the tool-supported meeting process shows, in fact, very good performance and considerably outperforms comparable results from paper-based inspection meetings (2.7% vs. 32.2% true defects lost on average).

Another advantage of tool-support is that it offers additional, statistically-oriented techniques based on individual inspectors' votes to discriminate automatically between false positives and true defects. We have experimented with different thresholds and illustrated the trade-off between removal of false positives and loss of true defects. Our data shows that simple techniques for automated defect discrimination show discrimination performance comparable to discussion-based meetings but further reduce meeting effort.

However, especially for automated defect discrimination techniques but also for the general planning step of traditional inspection meetings, more explicit decision support is required in order to enable inspection managers to determine thresholds for discrimination and to optimize the effort invested into inspection meetings. An economic model including the costs of false positives and of lost true defects is necessary to appropriately optimize the trade-offs between false positives not removed and true defects lost.

Although the performance of tool-supported inspection meetings reported in this paper is promising the number of unidentified false positives is rather high. A possible explanation is the fact that we used a requirements specification as the inspection object. Due to the nature of requirements the definition and exact identification of defects is much harder than with design or implementation artifacts. This is confirmed by further analysis of our empirical data showing that for defect reports in the introductory chapter of the requirements document, it turned out to be more difficult to discriminate between false positives and true defects than for defect reports in object diagrams or data models. An efficient discrimination of true defects and false positives thus depends highly on the precise and understandable definition of a defect. Consequently, defect definitions represent a challenge in

the case of rather informal, early-life-cycle software development products (like requirements specifications).

The clear and simple meeting definition (defect discrimination instead of discrimination plus synergy) used in the experiment helped inspectors to focus on the important tasks and reduced performance variation considerably. We, therefore, think that these empirical results further support the argument to focus inspection meetings on defect discrimination rather than on detecting new defects. For this purpose tools like GRIP can efficiently and effectively support inspection meetings.

7. Summary and further work

In this paper we have described an empirical evaluation of automating software inspections. Our research integrates concepts from the areas of computer-supported cooperative work, requirements engineering, as well as verification and validation. This paper reported on a family of experiments: two experiments on paper-based inspection and a third experiment on a groupware-supported inspection to empirically investigate the effect of tool support on defect detection effectiveness and inspection effort in an academic environment with 37 subjects. The main results of the family of experiments regarding individual defect detection are:

(a) The effectiveness of inspectors and teams is comparable to paper-based inspection without tool support; (b) the effort of inspectors decreased while (c) the efficiency of inspectors increased significantly with tool support.

The main findings of the experiment regarding tool support for inspection meetings are (a) tool support considerably lowered the meeting effort; (b) supported inspectors in identifying false positives, and at the same time (c) reduced the number of true defects lost.

Our empirical analysis confirms that the key advantage of GRIP is the reduction of cost drivers. In combination with the fact that the GRIP environment is based on commercially available groupware technology, which can be comparatively easily tailored to company-specific environments and processes (Halling, Grünbacher and Biffl, 2001b), tool support represents a competitive alternative to paper-based inspection. Furthermore, GRIP supports inspection management thus making inspections easier to plan and track, as a well-structured electronic report is immediately available at the end of inspection.

Further work. These results strongly encourage using appropriate tool support for inspections in practice. Furthermore our research emphasizes the need to study the use of tool support for inspection in different contexts with other types of inspected artifacts, e.g., artifacts from design or implementation stages; (b) with other reading techniques for inspection such as scenario-based reading, and (c) also in selected business environments.

From a management perspective more explicit decision support is required in order to enable inspection managers to determine thresholds for discrimination and to optimize the effort invested into inspection meetings. Further an economic model including the costs of false positives and of lost true defects is necessary to appropriately optimize the trade-offs between false positives not removed and true defects lost.

Acknowledgements This work was in part supported by the Austrian Science Fund (Research Grant P14128). We want to thank the participants of all three experiments and in particular Martin Weninger for their contributions.

References

Anderson, P., Reps, T., et al.: Design and Implementation of a Fine-Grained Software Inspection Tool. IEEE Trans. Softw. Eng. **29**(8), 721–733 (2003)

Aurum, A., Petersson, H., et al.: State-of-the-Art: Software Inspections after 25 Years. Softw. Test. Verif. Rel. **12**(3), 134–154 (2001)

Basili, V., Green, S., et al.: The Empirical Investigation of Perspective-Based Reading. Empirical Softw. Eng.: Int. J. **1**(2), 133–164 (1996)

Bianchi, A., Lanubile, F., et al.: A Controlled Experiment to Assess the Effectiveness of Inspection Meetings. Metrics 01, London. pp. 42–50 (2001)

Biffl, S.: Software Inspection Techniques to support Project and Quality Management. Shaker Publishing, Aachen (2001)

Biffl, S., & Halling, M.: Investigating the influence of inspector capability factors with four inspection techniques on inspection performance. 8th IEEE Int. Software Metrics Symposium, IEEE Comp. Soc. Press, Ottawa, pp. 107–117 (2002)

Biffl, S., & Halling, M.: Investigating the Defect Detection Effectiveness and Cost-Benefit of Nominal Inspection Teams. IEEE Trans. Softw. Eng. **29**(5), 385–397 (2003)

Boehm, B.W., Grünbacher, P., et al.: Developing Groupware for Requirements Negotiation: Lessons Learned. IEEE Softw. **18**(3), 46–55 (2001)

Ciolkowski M., Shull F., & Biffl St.: A Family of Inspection Experiments, Proc. Int. Conf. on Empirical Assessment of Software Engineering (EASE), Keele, April 2002 (2002)

Fagan, M.: Design and Code Inspections To Reduce Errors In Program Development. IBM Syst. J. **15**(3), 182–211 (1976)

Genuchten, M., Cornelissen, W., et al.: Supporting Inspections With an Electronic Meeting System. JMIS **14**(3), 165–178. (1998)

Genuchten, M., Dijk, C., et al.: Industrial Experience in Using Group Support Systems for Software Inspections. IEEE Softw. **18**(3), 60–65 (2001)

Gilb, T., & Graham, D.: Software Inspection. Boston, MA, USA, Addison Wesley Professional. (1993)

Grünbacher, P., Halling, M., et al.: An Empirical Study on Groupware Support for Software Inspection Meetings. 18th IEEE International Conference on Automated Software Engineering, Montreal, IEEE Computer Society. pp. 4–11 (2003)

Halling, M.: Supporting Management Decisions in the Software Inspection Process, Vienna University of Technology. (2002)

Halling, M., & Biffl S.: Investigating the Influence of Software Inspection Process Parameters on Inspection Meeting Performance. IEE Proc.-Softw. **149**(5), (2002)

Halling, M., Biffl, S., et al.: *Using Reading Techniques to Focus Inspection Performance*. 27th Euromicro Conference 2001: A Net Odyssey (Euromicro'01), IEEE Computer Society Press, Warsaw, pp. 248–257 (2001)

Halling, M., Biffl, S., et al.: A Groupware-Supported Inspection Process for Active Inspection Management. Proceedings of the 28th Euromicro Conference (EUROMICRO'02), Dortmund Germany, IEEE CS, pp. 251–258 (2002)

Halling, M., Biffl, S., et al.: An Economic Approach for Improving Requirements Negotiation Models with Inspection. Requirements Eng. J., Springer (8), 236–247 (2003a)

Halling, M., Biffl, S., et al.: An Experiment Family to Investigate the Defect Detection Effect of Tool-Support for Requirements Inspection. Proceedings of the Ninth International Software Metrics Symposium (METRICS'03), IEEE Comp. Soc. Press, Sydney, pp. 278–285 (2003b)

Halling, M., Grünbacher, P., et al.: Groupware Support for Software Requirements Inspection. WISE'01: Proceedings of the 1st Workshop on Inspection in Software Engineering, Paris, France, Software Quality Research Lab, McMaster University, Hamilton, Canada, pp. 20–29 (2001a)

Halling, M., Grünbacher, P., et al.: Tailoring a COTS Group Support System for Software Requirements Inspection. 16th IEEE International Conference on Automated Software Engineering, IEEE Computer Society, San Diego, pp. 201–210 (2001b)

Höst, M., Regnell, B., et al.: Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. Empirical Softw. Eng. **5**, 201–214 (2000)

Johnson, P.M., & Tjahjono, D.: Assessing software review meetings: A controlled experimental study using CSRS. International Conference on Software Engineering, Boston, pp. 118-127 (1997)

Johnson, P.M., & Tjahjono, D.: Does Every Inspection Really Need a Meeting? Empirical Softw. Eng. (1998)

Knight, J.C., & Myers, E.A.: An improved inspection technique. C. ACM **36**(11), 51–61 (1993)

Laitenberger, O., & DeBaud, J.-M.: An encompassing life cycle centric survey of software inspection. J. Sys. Softw. **50**(1), 5–31 (2000)

Laitenberger, O., & Dreyer, H.M.: Evaluating the usefulness and the ease of use of a Web-based inspection data collection tool. Proceedings Fifth Intl. Software Metrics Symposium, pp. 122–132 (1998)
Land, L.P.W., Jeffery, R., et al.: Validating the Defect Detection Performance Advantage of Group Designs for Software Reviews. ESEC/FSE, pp. 17–26 (1997)
Lanubile, F., & Mallardo T.: An Empirical Study of Web-Based Inspection Meetings. ACM/IEEE Int. Sym. on Empirical Software Engineering, Rome, pp. 244–251 (2003)
MacDonald, F., & Miller J.: A Comparison of Tool-Based and Paper-Based Software Inspection. Empirical Softw. Eng. **3**, 233–253 (1998)
MacDonald, F., & Miller, J.: A Comparison of Computer Support Systems for Software Inspection. Automated Softw. Eng. **6**, 291–313 (1999)
MacDonald, F., Miller, J., et al.: Automatic Collation of Software Inspection Defect Lists. Proceedings of the 8th International Conference on Information Systems Development, Boise, Idaho, pp. 1–11 (1999)
MacDonald, F., Miller, J., et al.: ASSISTing Management Decisions in the Software Inspection Process. J. Inf. Technol. Manage. **3**, 67–83 (2002)
Nunamaker, J.F., Briggs, R.O., et al.: Lessons from a Dozen Years of Group Support Systems Research: A Discussion of Lab and Field Findings. J. Manage. Inf. Sys. **13**(3), 163–207 (1997)
Parnas, D.L., & Lawford M.: The Role of Inspection in Software Quality Assurance. IEEE Trans. Softw. Eng. **29**(8), 674–676 (2003)
Parnas, D.L., & Weiss, D.M.: Active design review: principles and practices. 8th Int. Conf. on Software Engineering, pp. 259–265 (1985)
Parnas, D.L., & Weiss, D.M. Active Design Reviews: Principles and Practice. J. Sys. Softw. **7,** 259–265 (1987)
Perpich, J.M., Perry, D.E., et al.: Anywhere, anytime code inspections: using the Web to remove inspection bottlenecks in large-scale software development. Proc. of the 19th ICSE, Boston, USA, pp. 14–21 (1997)
Porter, A.A., & Johnson, P.M.: Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies. IEEE Trans. Softw. Eng. **23**(3), (1997)
Sauer, C., & Jeffery, D., et al.: The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research. IEEE Trans. Softw. Eng. **26**(1), 11–14 (2000)
Thelin, T., Andersson, P., et al.: Tool Support for Usage-based reading. Proceedings IASTED Software Engineering Conference, Innsbruck, Austria, pp. (2004)
Tichy, W.: Hints for Reviewing Empirical Work in Software Engineering. Empirical Softw. Eng.: An Int. J. **5**, 309–312 (2001)
Vitharana, P., & Ramarmurthy, K.: Computer-Mediated Group Support, Anonymity, and the Software Inspection Process: An Empirical Investigation. IEEE Trans. Softw. Eng. **29**(2), 167–180 (2003)
Votta, L.: Does every Inspection need a Meeting? ACM Softw. Eng. Notes **18**(5), 107–114 (1993)
Wohlin, C., Runeson, P., et al.: Experimentation in Software Engineering: An Introduction, The Kluwer International Series in Software Engineering. Kluwer Academic Publishers. (2000)