An Empirical Study on Groupware Support for Software Inspection Meetings

Paul Grünbacher Michael Halling
Johannes Kepler University Linz
Systems Engineering & Automation
Altenbergerstr. 69
4040 Linz, Austria
pg@sea.uni-linz.ac.at mh@sea.uni-linz.ac.at

Stefan Biffl
Vienna University of Technology
Inst. of Software Technology
Karlsplatz 13
1040 Vienna, Austria
Stefan.Biffl@tuwien.ac.at

Abstract

Software inspection is an effective way to assess product quality and to reduce the number of defects. In a software inspection the inspection meeting is a key activity to agree on collated defects, to eliminate false positives, and to disseminate knowledge among the team members. However, inspection meetings often require high effort and may lose defects found in earlier inspection steps due to ineffective meeting techniques. Only few tools are available for this task. We have thus been developing a set of groupware tools to lower the effort of inspection meetings and to increase their efficiency. We conducted an experiment in an academic environment with 37 subjects to empirically investigate the effect of groupware tool support for inspection meetings. The main findings of the experiment are that tool support considerably lowered the meeting effort, supported inspectors in identifying false positives, and reduced the number of true defects lost.

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 reduce the number of defects [4]. An inspection consists of several clearly defined activities including inspection planning and preparation, individual defect detection, team meeting, as well as evaluation and rework.

Within the inspection process defect detection and inspection meetings 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 step are to collect defects, to eliminate false positives, and to find new defects.

Although the benefits of inspections are obvious the high costs associated with inspections often inhibit widespread adoption in industry. It has been observed that due to inappropriate techniques meetings may even lead to a loss of defects identified during individual defect detection. 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.

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.

We have been addressing these issues in our previous research on GRIP (*GR*oupware Supported *I*nspection *P*rocess) [15]. GRIP provides a framework and collaborative tools for an inspection team. Encouraged by feedback from early trials our goal was to empirically validate the benefits of GRIP. We have thus carried out an experiment to compare GRIP to previous large-scale manual inspection experiments. In the experiment we learned that for *defect detection* (a) the effectiveness is similar for manual and GRIP-based inspections; (b) inspection effort and defect overlap decrease significantly with tool support, while (c) efficiency increases considerably [14].

In this paper we describe an experiment on tool support for inspection meetings and discuss how GRIP supports inspection meetings. Our previous results in two large-scale experiments indicated that meeting losses are on average higher than meeting gains [11]. This paper thus explores whether groupware tools can improve inspection meetings such that their effort can be justified.

The paper is organized as follows: Section 2 discusses tools for inspection meetings and briefly introduces GRIP. Section 3 presents our family of experiments for evaluating tool support for inspection meetings. Section 4 discusses the results of the experiment on inspection meetings and compares it to our previous manual experiments. Conclusions are given in Section 5.

Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE'03)
1527-1366/03 $ 17.00 © 2003 IEEE

2. Tool Support for Inspection Meetings

Numerous tools and platforms are available for automating software inspections [21]. However, most tools have a strong focus on reporting and collection of identified defects. There is only little support available for inspection meetings. Furthermore, there are only few empirical studies on the benefits of automating the inspection process. In [21] the authors 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 whether and how much tools actually improve the performance of inspections.

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 [6, 7]. The authors present empirical evidence that tool support significantly increases performance and the overall contribution of an inspection meeting. Defect detection is, however, not addressed in this work.

2.1 Groupware-Supported Inspection Process

In our research on software inspections to date we have been pursuing the following objectives: (1) Development of collaborative tools and techniques covering the entire inspection process [15]; (2) Integration of technical inspection aspects with management activities such as planning, monitoring, or process analysis [12]; (3) Empirical studies to validate the usefulness, effectiveness, and efficiency of our approach [14]; and (4) the improvement of inspection support for informal requirements [13].

Figure 1 shows the GRIP framework which has been guiding our research and covers the complete inspection life-cycle. Figure 1 also outlines the roles in GRIP:

- The *inspection manager* is in charge of planning and tailoring the inspection process. This includes the selection of inspection guidelines, the preparation of the inspection object(s), and the configuration of 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 where necessary. Finally, measures are analysed 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.

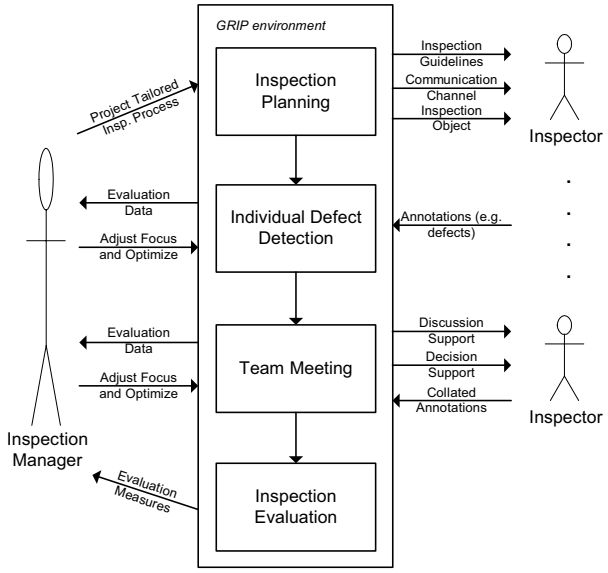

Figure 1: The GRIP framework [15].

In a recent study [14] we have documented the positive effects of GRIP on individual inspectors' performance with respect to defect detection effectiveness and efficiency. In this work we focus on an empirical evaluation of the support offered by GRIP for inspection meetings.

2.2 Inspection Meeting Support in GRIP

When discussing inspection meetings it is important to briefly outline the evolution of the inspection process over time. While Fagan [5] viewed the inspection meeting as the key activity for defect detection, Parnas and Weiss [22] 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 different views on the role of the inspection meeting help to understand why empirical results on meeting effectiveness differ considerably. While Fagan reports that inspection meetings were very effective [5], more recent reports show that this is not necessarily the case in an inspection process with a different focus [2, 11, 22, 23, 26].

However, shifting defect detection from meeting to individual preparation does not necessarily mean abolishing meetings. Land *et al.* [20] as well as Sauer *et al.* [24] emphasize the importance of inspection meetings for defect collection and defect discrimination (i.e., 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 *et al.* [20] false positives can become a problem if occurring frequently because they incur further costs such as the time spent on repairing

them. Johnson and Tjahjono [17, 18] as well as Porter and Johnson [23] report similar results and argue that inspection meetings are beneficial mainly due to the considerable reduction of false positives.

Consequently, the inspection team meeting plays an important role in GRIP. The starting point for a meeting is the inspection object (e.g., a set of requirements) together with all annotations collected during individual defect detection.

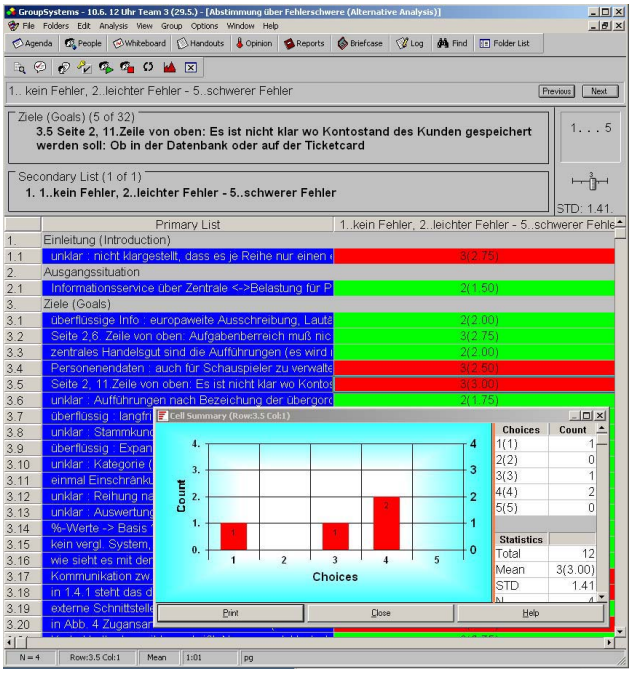

Figure 2: Assessment of defect severity and visualization of diverging inspectors' opinions in the GRIP tool.

During the inspection meeting the inspectors are first asked to assess these defect annotations for severity. All defects above a previously defined threshold are then further classified by using a customizable taxonomy of defects. All defects below a previously defined threshold are considered false positives, i.e., no true defects.

In order to support this group decision problem we have been customizing electronic voting tools for our purpose. GroupSystems.com's Alternative Analysis tool allows assessing a set of voting items using a set of criteria. Voting methods are customizable (e.g., 1-10 scale, ordinal scale, etc.). Figure 2 shows a snapshot of the moderator screen after the team of inspectors has finished assessing all defects. The tool aggregates all individual ballots thus allowing quick elimination of defects not worth further consideration. The tool also allows visualizing situations where the team of inspections had diverging opinions on defect severity. The inspector manager can use the tool to initiate a discussion about defects needing special attention. The tool can also be used to automatically discriminate between true defects and false positives by analyzing the ballots.

After initial trials 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 they would be easier to justify from an economic point of view. Please note that the current implementation of GRIP focuses on pure defect discrimination as proposed in [24]. We did not explicitly support other potential meeting goals like s*ynergy* and *soft benefits*. In the case of synergy existing empirical evidence (see [2] and [14] for a detailed review of existing material) suggests that if the individual preparation phase already focuses on defect detection, inspection meetings show very little synergy benefits. As far as soft benefits are concerned, it is difficult to measure them empirically.

Of course, potential benefits 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, tool support can considerably improve this situation by offering means for parallel contributions during meeting. For example, inspectors can assess the severity of defects in parallel and team discussions focus on critical issues only.

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 a true defect being classified as a false positive. GRIP tries to meet this requirement by focusing inspectors' attention on defects leading in diverging votes.

3. An Empirical Study to Evaluate Tool Support for Inspection Meetings

In the following two sections we describe our study on tool support for defect discrimination in inspection meetings and relate the derived performance measures to similar measures from conventional paper-based inspection meetings.

3.1 Research Approach and Hypotheses

Defect Discrimination: The main goal of inspection meetings in our study is to identify false positives that were reported as defects during individual preparation. As false positives increase the rework effort it is reasonable to evaluate techniques for their reduction. However, 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 one true defect are considerably higher than the costs of keeping one false positive. However, these costs depend on the specific context of the inspec-

tion and have to be assessed individually. In this paper, we do not assume any specific distribution for these costs but simply compare absolute number 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.

Inspection Meeting Effort: Due to the tool support we expect a reduction in meeting effort, as the inspectors can work concurrently and are more focused on important issues. We analyze an efficiency criterion indicating the number of false positives removed per time unit. Furthermore we compare the effort of tool-based meetings to the effort of paper-based meetings and expect tool-based meetings to be more cost-efficient.

Fully Automated Defect Discrimination: We further evaluate a so-called "Automated Defect Discrimination" method based on simple statistical measures for diverging voting results. We discuss how different threshold parameters influence the relationship between false positives removed and true defects lost and investigate whether there is a threshold with superior meeting output compared to actual discussion in a meeting regarding (a) eliminating less true defects and (b) eliminating more false positives.

3.2 The Requirements Inspection Experiments

In our previous work we have performed a family of experiments for the empirical evaluation of various research issues in software inspection. Details on these experiments can be found in [14] and [9].

Table 1: Description of Experiment Family.

Experiment	A	B	C
Year	**1999-2000**	**2000-2001**	**2002**
# CBR-Inspectors	86	47	37
# CBR-Teams	16	9	7
Average Team Size	5.4	5.9	5.3
# Reference defects	86	97	93
Inspection Meeting	Yes	No	Yes
Inspection Process	Manual	Manual	GRIP

Table 1 summarizes key information about the three related experiments. The first two experiments A and B studied the influence of different reading techniques on inspection performance. For details on Experiments A and B see [3, 4] for a comparison of the two experiments see [10]. Experiments A and B used checklist-based reading (CBR) and scenario-based reading techniques. For tool evaluation in Experiment C we decided to select the CBR technique in order to have a sufficient sample size.

We regard the three experiments as an experiment family due to a number of similarities: (a) 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, (b) the subjects participating in the experiment were all selected from computer science students at Vienna University of Technology, and (c) checklists, requirements documents, and seeded defects were very similar for all experiments.

As far as inspection meeting performance is concerned, we can only compare the results of Experiment A and C. However, the comparison has to be done with caution as inspectors in Experiment A were asked to identify new defects during the meeting. In Experiment C the meeting purpose was only to discriminate between false positives and true defects.

3.3 Experimental Tool-Supported Meeting Process in Experiment C

Experiment C involved 37 undergraduate computer science students and almost identical requirements documents as Experiments A and B. 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. All seeded defects could be found by the inspectors without referring to additional documents. Please refer to [3] and [10] for details on these aspects of the experiment.

In the following, however, we focus on the specifics of tool support for inspection meetings. Experiment C consisted of an individual defect detection activity and an inspection meeting. Details on the individual defect detection step can be found in [14].

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 [15]. 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 increase 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.

Between individual defect detection and the inspection meeting the inspection manager had to generate a collated

defect list from all individually reported defects. As this task is supported by GRIP, this (minor) effort is ignored in the analysis.

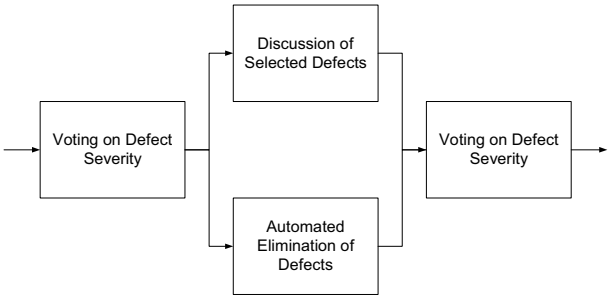

Figure 3. Inspection Meeting Process in Experiment.

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 votes. 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 are diverging. The definition of homogeneous and heterogeneous voting can be adjusted via 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 the discussed defect. Finally, inspectors assess the defect type. The results of this second voting procedure are not analyzed in detail in this paper.

3.4 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 validity deals with generalization [27].

Internal Validity. The primary threat to internal validity is *selection*. This comes from the selection of subjects and their assignments to particular treatments. In Experiments A and B, we used randomization to avoid systematic bias from selection. In Experiment C selection was not an issue as we only had 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*. However, the guidance provided by the groupware tool and the supervision by the inspection manager enabled us to easily enforce process conformance in Experiment C.

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 effort they want to spend on the inspection meeting is reasonable in a controlled, synchronous inspection. However, effort values of Experiment A and C 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 Experiment C due to tool support. In Experiments A and B, 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 representing a realistic situation. The document size and defect density were somewhat above the levels from other reported experiments [1], but not particularly high compared to documents in industrial settings [8]. Moreover, we used *inspection activities* that had been implemented in a number of professional development environments [19].

The subjects were students participating in a university class. As pointed out in the literature [24] students may not be representative of real developers. However, Höst *et al*. [16] observe no significant differences between students and professionals for small tasks of judgment. According to Tichy [25] using students as subjects is acceptable if students are appropriately trained and the data is used to establish a trend. These conditions are both met in our case.

4. Results

In this section we describe key empirical results regarding tool support for inspection meetings. We analyze defect discrimination performance (a) including team discussions and (b) fully automated and present information on meeting effort (see hypotheses in Section 3.1). We also compare the results from the tool-based meeting to data from paper-based inspections where possible.

4.1 Defect Discrimination

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. Inspection meetings can however reduce rework effort by removing false positives reported during individual defect detection. As argued before the focus of this empirical study is on defect discrimination performance. With respect to the identification of false positives

we focus exclusively on the data from Experiment C as detailed information regarding false positives is not available for Experiment A.

Table 2 summarizes the number of total defect reports removed, the number of false positives eliminated and the number of true defects lost during meeting for each team participating in Experiment C. Values given in Table 2 show the percentage of the appropriate base value (e.g., team 1 reduced the number of total defect reports by 25.7%, the number of false positives by 30.1% and the detected reference defects by 15.6%).

Table 2: Reduction of total defect reports, elimination of false positives, and loss of reference defects in percent for Exp C (%).

	Total Reports	False Positives	Reference Defects
Team 1	25.7	30.1	15.6
Team 2	6.1	8.2	1.9
Team 3	15.7	24.0	1.6
Team 4	9.0	16.2	0.0
Team 5	3.7	6.3	0.0
Team 6	1.9	4.3	0.0
Team 7	13.4	19.6	0.0
Mean	**10.8**	**15.5**	**2.7**
Std.dev.	**8.2**	**9.7**	**5.7**

Table 2 illustrates that between 4% and 30% of the reported false positives were removed during the inspection meeting. Furthermore only few reference defects were lost (with the exception of team 1). Overall tool-supported inspection meetings proved to be a viable means for discriminating between true defects and false positives.

Table 3 documents the impact of inspection meetings on the number of true defects detected in more detail. As we do have appropriate data for Experiment A, we can compare paper-based and tool-based inspection meetings with respect to this criterion.

Table 3: Net gain of reference defects (%) for ExpA and ExpC.

	Reference defects lost		Reference defects gained		Net gain of reference defects	
	Mean	**Std.dev.**	**Mean**	**Std.dev.**	**Mean**	**Std.dev.**
ExpA	32.2	19.2	23.2	17.5	-8.9	33.9
ExpC	2.7	5.7	n/a	n/a	-2.7	5.7

Please note however an important difference between the meeting goals in Experiments A and C. While we aimed at also identifying new reference defects in Experiment A, we focused exclusively on defect discrimination in Experiment C and consequently did not detect any new defects. Details on the meeting performance of Experiment A can be found in [11]. When comparing the net gain criterion we can observe that tool-supported inspection meetings outperform paper-based inspection meetings. Although there were new defects identified during inspection meetings in Experiment A, these gains could by far not outweigh the significant meeting losses. In the case of Experiment C there were no meeting gains but overall meeting losses were on average reduced by a factor three. It is further important to observe that the standard deviation of meeting performance is very high in Experiment A, indicating that meeting performance varied a lot. In Experiment C the clearer and simpler focus of meetings also reduced standard deviations considerably.

4.2 Automated Defect Discrimination

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 apply a simple automated defect discrimination technique that is supported by GRIP. The advantage of this technique is a reduction of meeting effort as no discussions need to take place.

This automated defect discrimination is based on 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.

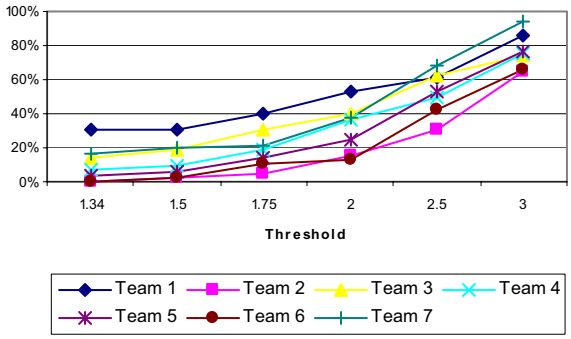

Figure 4: Automated Defect Discrimination in percent of total false positives for Exp C.

We evaluated different thresholds in order to illustrate the trade-off between the removal of false positives and the loss of reference defects. Figure 4 shows that the number of eliminated false positives increases with a more stringent threshold.

However, for the determination of the optimal threshold we have to consider the tradeoff between eliminated false positives and lost true defects. If we use a low threshold we ensure that very few reference defects are

lost but also reduce the number of false positives identified. If we use a high threshold, we observe the opposite effect. Figure 5 summarizes the dependency of most important meeting performance criteria on the definition of this threshold.

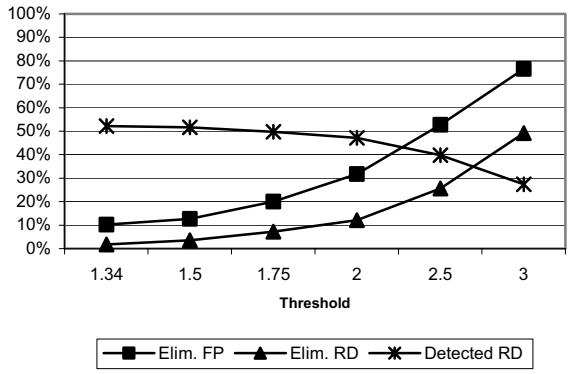

Figure 5: Dependence of eliminated false positives (Elim. FP), lost reference defects (Elim. RD) and detected reference defects (Detected RD) for different thresholds.

4.3 Meeting Effort

Finally we analyze the meeting effort invested in Experiments A and C. As we applied somewhat different meeting processes and defined different meeting goals the effort values are not directly comparable. Nevertheless Table 4 shows that the meeting in Experiment C took considerably less time than the meeting in Experiment A. Furthermore the standard deviation is reduced to a reasonable level, while it was very high for Experiment A. In the last row Table 4 further shows that we could even further decrease meeting effort by using the automated defect discrimination approach.

Table 4: Average effort of meetings in minutes for all teams in ExpA and ExpC.

	Average Effort	Std.dev.
ExpA	293	129
ExpC	100	5
ExpC (automated)	80	5

5. Conclusions and Further Work

In this paper we have described an empirical evaluation of automating software inspection meetings. The research integrates concepts from the areas of computer-supported cooperative work, requirements engineering, as well as verification and validation. 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 incur usually 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 such 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 for inspection. Due to the nature of requirements the definition and exact identification of defects is much harder. Further analysis of our empirical data also shows 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 exact and understandable definition of a defect. However, defect definitions represent a challenge in the case of rather informal, early lifecycle 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.

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

[1] Basili, V., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Soerumgaard, S., and Zelkowitz, M., The Empirical Investigation of Perspective-Based Reading. *Empirical Software Engineering: An International J.*, 1996. 1(2):133-164.

[2] Bianchi, A., Lanubile, F., and Visaggio, G. A Controlled Experiment to Assess the Effectiveness of Inspection Meetings. In: *Metrics 01*. 2001. London.

[3] Biffl, S. and Halling, M. Investigating the influence of inspector capability factors with four inspection techniques on inspection performance. In: *8th IEEE Int. Software Metrics Symposium,*. 2002. Ottawa: IEEE Comp. Soc. Press.

[4] Biffl, S. and Halling, M., Investigating the Defect Detection Effectiveness and Cost-Benefit of Nominal Inspection Teams. *IEEE Transactions on Software Engineering*, 2003.

[5] Fagan, M., Design and Code Inspections To Reduce Errors In Program Development,. *IBM Systems Journal*, 1976. 15(3):182-211.

[6] Genuchten, M., Cornelissen, W., and Dijk, C., Supporting Inspections With an Electronic Meeting System. *JMIS*, 1998. 14(3):165-178.

[7] Genuchten, M., Dijk, C., Scholten, H., and D., V., Industrial Experience in Using Group Support Systems for Software Inspections. *IEEE Software*, 2001. 18(3):60-65.

[8] Gilb, T. and Graham, D., *Software Inspection*. 1993: Addison-Wesley.

[9] Halling, M., Supporting Management Decisions in the Software Inspection Process. 2002, Vienna University of Technology.

[10] Halling, M. and Biffl, S. Using Reading Techniques to Focus Inspection Performance. In: *Euromicro 2001 Conference - Software Product and Process Improvement Track*. 2001. Warsaw: IEEE Comp. Soc. Press.

[11] Halling, M. and Biffl, S., Investigating the Influence of Software Inspection Process Parameters on Inspection Meeting Performance. *IEE Proceedings-Software*, 2002. 149(5).

[12] Halling, M., Biffl, S., and Grünbacher, P. A Groupware-Supported Inspection Process for Active Inspection Management. In: *Euromicro 2002 Conference*. 2002. Dortmund Germany: IEEE CS.

[13] Halling, M., Biffl, S., and Grünbacher, P., An Economic Approach for Improving Requirements Negotiation Models with Inspection. *Requirements Engineering Journal, Springer*, 2003.

[14] Halling, M., Biffl, S., and Grünbacher, P. An Experiment Family to Investigate the Defect Detection Effect of Tool-Support for Requirements Inspection. In: *9th IEEE Int. Software Metrics Symposium*. 2003. Sydney: IEEE Comp. Soc. Press.

[15] Halling, M., Grünbacher, P., and Biffl, S. Tailoring a COTS Group Support System for Software Requirements Inspection. In: *16th IEEE International Conference on Automated Software Engineering*. 2001. San Diego.

[16] Höst, M., Regnell, B., and Wohlin, C., Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment,. *Empirical Software Engineering*, 2000. 5:201-214.

[17] Johnson, P.M. and Tjahjono, D. Assessing software review meetings: A controlled experimental study using CSRS. In: *ICSE*. 1997. Boston.

[18] Johnson, P.M. and Tjahjono, D., Does Every Inspection Really Need a Meeting? *Empirical Software Engineering*, 1998.

[19] Laitenberger, O. and DeBaud, J.-M., An encompassing life cycle centric survey of software inspection. *Journal of Systems and Software*, 2000. 50(1):5-31.

[20] Land, L.P.W., Jeffery, R., and Sauer, C. Validating the Defect Detection Performance Advantage of Group Designs for Software Reviews. In: *ESEC/FSE*. 1997.

[21] MacDonald, F. and Miller, J., A Comparison of Computer Support Systems for Software Inspection. *Automated Software Engineering*, 1999. 6:291-313.

[22] Parnas, D.L. and Weiss, D.M. Active design review: principles and practices. In: *8th Int. Conf. on Software Engineering*. 1985.

[23] Porter, A.A. and Johnson, P.M., Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies. *IEEE Transactions on Software Engineering,*, 1997. 23(3).

[24] Sauer, C., Jeffery, D., Land, L., and Yetton, P., The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research. *IEEE Transactions on Software Engineering*, 2000. 26(1):11-14.

[25] Tichy, W., Hints for Reviewing Empirical Work in Software Engineering. *Empirical Software Engineering: An International Journal*, 2001. 5:309-312.

[26] Votta, L., Does every Inspection need a Meeting? *ACM Software Eng. Notes*, 1993. 18(5):107-114.

[27] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A., *Experimentation in Software Engineering: An Introduction, The Kluwer International Series in Software Engineering*. 2000: Kluwer Academic Publishers.