Increasing the Understanding of Effectiveness in Software Inspections Using Published Data Sets

Aybüke Aurum

School of Information Systems, Technology and Management
University of New South Wales, Sydney, NSW 2052, Australia
aybuke@unsw.edu.au

Claes Wohlin

Department of Systems and Software Engineering
School of Engineering, Blekinge Institute of Technology
Box 520, SE-372 25 Ronneby, Sweden
claes.wohlin@bth.se

Håkan Petersson

Software Engineering and Management
IT-University of Göteborg
Box 8718, 402 75 Göteborg Sweden
hakan.petersson@ituniv.se

Since its inception into software engineering, software inspection has been viewed as a cost-effective way of increasing software quality. Despite this, many questions remain unanswered regarding, for example, ideal team size or cost effectiveness. This paper addresses some of these questions by performing an analysis using 30 published data sets from empirical experiments of software inspections. The main question is concerned with determining a suitable team size for software inspections. The effectiveness of different team sizes is also studied. Furthermore, the differences in mean effectiveness between different team sizes are investigated based on the inspection environmental context, document types and reading technique. It is concluded that it is possible to choose a suitable team size based on the effectiveness of inspections. This can be used as a tool to assist in the planning of inspections. A particularly interesting result is that variation in the effectiveness between different teams is considerably higher for certain types of documents than for others. Our findings contain important information for anyone planning, controlling or managing software inspections.

ACM Classification: D.2 (Software Engineering), D 2.5 Testing and Debugging (Inspections and Walkthroughs), D 2.8 Metrics, D 4.8 Performances

1. INTRODUCTION

Software inspection encompasses a set of methods in which the purpose is to identify and locate faults in software. Since Fagan developed the inspection process in the early 70s, there have been many variations of the process put forth by others. Overall, the aim in any review process is to apply inspection to the working product as early as possible so that major faults are caught before the

Manuscript received: 3 September 2004
Communicating Editor: Colin Fidge

product is released. The state-of-the-art in software inspections after 25 years of research and practice is summarised by Aurum *et al* (2002).

In Fagan's inspection method (Fagan, 1976), the focus is on group meetings, where the members of an inspection team meet to find as many faults as possible and discuss the issues found by each reviewer during the preparation stage of the inspection. The aim is to uncover faults in the product, rather than to correct them. The goal of inspection meetings is to collect the faults discovered and bring synergy (process gains) to the software inspection. It is believed that the combination of different viewpoints, skills and knowledge from many reviewers creates this synergy. Synergy can overcome many difficulties encountered in organisational life, including those in the software development process. It has been suggested that reviewers who have inspection meetings are more successful at finding a larger quantity of faults, as opposed to those who work individually (Fagan 1976; Ebenau and Strauss, 1994).

Recently, researchers in Software Engineering have found some empirical findings that contradict this suggestion. Their findings point to the understanding that holding inspection meetings has minimal effect on finding the faults in the software product (Porter *et al*, 1995; Votta, 1993). Inspection meetings can also be quite expensive because they typically involve 3–6 people. The process is labour intensive and it takes many meetings to completely inspect a product since each inspection covers only a small part of the product (Mashayekhi *et al*, 1993). One critical issue here is determining the size of an inspection team.

Fagan suggests that four people are a good inspection team size (Fagan, 1976). This suggestion also ties in with the industrial work done by Weller (1993), who reports that four person teams are more effective and efficient than three person teams. IEEE STD 1028-1997 suggests teams of 3 to 6 people (IEEE, 1998). Teams of 4 or 5 people are common in practice (Wheeler *et al*, 1996). Owens (1997) reports that although it is expensive to have more reviewers, it is more effective to have more points of view for requirements inspection (e.g. 5–6 inspectors) than for design, and more inspectors are needed for design than coding (e.g. 1–2 inspectors for coding). Bisant and Lyle (1989) also support the idea of having two people in coding inspections. Their findings illustrated that programming speed increased significantly in two-person inspections.

In reviewing the arguments identifying team size for software inspection teams, the main objective of this paper is (a) to study the relationship between the size of an inspection team and its effectiveness and, (b) to examine the effectiveness of nominal inspection teams (c) to investigate the effect on the inspection effectiveness when altering some of the inspection attributes, such as environmental context, document type and reading technique. This is done using 30 available data sets that are used to illustrate what could be done to identify suitable team sizes in software inspections. This paper contributes by studying team size and effectiveness based on a number of existing data sets, which allows for the study of both median effectiveness and variation. Moreover, the paper provides novel research in analyzing and illustrating the differences in inspection effectiveness when varying inspection attributes.

The remainder of the paper is organized as follows: Section 2 describes the method of analysis and the data sets used in this paper and it also addresses issues related to nominal teams. Section 3 presents the results from the data analysis. Section 4 provides a discussion and Section 5 summarizes the findings.

2. METHOD OF ANALYSIS

This study aims at collating knowledge from a number of different studies in order to provide a common view of what could be expected from inspection teams when conducting a software inspection. The method of analysis is based on a number of data sets from prior experiments in software inspections. This type of approach is similar to the one used in medical science to combine results from different studies.

Part of the challenge in this paper is to work with heterogeneous data, and the combined analysis of the data (Montgomery, 1997). Firstly, the goals of the individual empirical studies, which form the data sources in this study, are different from the goal of the study in this paper. For example, no results concerning effectiveness of different team sizes are reported in previous studies. The data in this study is combined internally to create nominal teams and to externally investigate the general effect of the team size on the effectiveness of inspections. Secondly, we do not have a view of the full context of the environments in which the source studies were conducted such as the size of the artifacts and experience of the reviewers. Furthermore, there are consistency issues regarding descriptions of contexts, artifacts and different aspects of the other studies.

Factors such as the lack of micro-level documentation of these details in previous studies and other environmental/cognitive barriers impose significant impediments in examination of heterogeneous data. This is clearly a threat to the accuracy of the actual results reported here, and it means that the results should be interpreted with some caution. However, this does not lessen the need or obvious benefits of replicating such empirical studies and integrating their results so that we can build a better understanding of the discipline (Miller, 1999; Pickard *et al*, 1998). In this way, although the analysis of heterogeneous data may display some variations in the context, subjects, artefacts etc., our aim is to draw together the results of the studies to develop a general understanding of the concepts/ideas. The aim of this paper is to examine what is feasible when combining data from previous studies.

2.1 Effectiveness and Efficiency

The measure of interest here is the effectiveness of nominal inspection teams. In the literature, when discussing inspection performance, the two terms *efficiency* and *effectiveness* are occasionally used interchangeably. In this paper, they have distinct meanings. The efficiency of an inspection team can be defined in several ways but must include the amount of effort spent by the team. The focus here is on the effectiveness of the team. The effectiveness of an inspection team denotes what proportion of the existing faults the team found.

The effectiveness E of a team T, is in this study calculated as: $E_T=D_T/N$. D_T is the number of unique faults found by team T and N is the number of existing faults before the inspection took place.

The number of existing faults is known for all data sets in this paper since the data is taken from controlled experiments. If applying this analysis to live data, N can, for example, be estimated. One way of estimating is to use an estimation method called capture-recapture (Petersson *et al*, 2004). Another approach would be to perform experiments within the company to generate data where N is known.

2.2 Analysed Data Sets

The first part of the analysis focuses on effectiveness of a general inspection team with no specific attributes. In the second part, the data sets are partitioned using three attributes of the inspection: a) the environmental context (i.e. profession) in which the study was conducted, b) the type of document, and c) the reading techniques used. Note that no interaction aspects between the above attributes have been studied here as this is outside scope of this paper. These three attributes are by no means optimal from a practical perspective. However, they were the attributes that were identifiable from the available data sets. This raises two issues. First, the analysis presented here should be read as an example of the type of analysis that can be made for software inspections. Second, it highlights the need to have data sets with attributes that practitioners view as important, which may include, for example, the application domain. The data sets and their attributes are shown in Table 1 and the attributes are further explained below. Three of the data sets, no. 6, 11 and 12, have been divided into subsets to lessen the effect of any single data set in the analysis.

No	No of Reviewers	Profession	Doc. Type	Read Tech	Reference
1	8	NASA	Artif. Req.[a]	AdH	Freimut, 1997
2	6	NASA	Artif. Req.	AdH	Freimut, 1997
3	6	NASA	Artif. Req.	AdH	Freimut, 1997
4	6	NASA	Artif. Req.	AdH	Freimut, 1997
5	6	Acad	Artif. Req.	Chkl	Unpublished[b]
6a	8	Acad	Textual	AdH	Wohlin *et al*, 1995
6b	7	Acad	Textual	AdH	Wohlin *et al*, 1995
6c	7	Acad	Textual	AdH	Wohlin *et al*, 1995
7	7	NASA	Req.	AdH	Freimut, 1997
8	6	NASA	Req.	AdH	Freimut, 1997
9	6	NASA	Req.	AdH	Freimut, 1997
10	6	NASA	Req.	AdH	Freimut, 1997
11a	8	Acad	Artif. Req.	PBR	Regnell *et al*, 2000
11b	7	Acad	Artif. Req.	PBR	Regnell *et al*, 2000
12a	8	Acad	Artif. Req.	PBR	Regnell *et al*, 2000
12b	7	Acad	Artif. Req.	PBR	Regnell *et al*, 2000
13	6	NASA	Artif. Req.	PBR	Freimut, 1997
14	6	NASA	Artif. Req.	PBR	Freimut, 1997
15	6	NASA	Req.	PBR	Freimut, 1997
16	6	NASA	Req.	PBR	Freimut, 1997
17	7	NASA	Req.	PBR	Freimut, 1997
18	6	NASA	Req.	PBR	Freimut, 1997
19	8	NASA	Artif. Req.	PBR	Freimut, 1997
20	6	NASA	Artif. Req.	PBR	Freimut, 1997
21	8	Prof.	Code	PBR	Freimut, 1997
22	7	Prof.	Code	PBR	Freimut, 1997
23	8	Prof.	Code	PBR	Freimut, 1997
24	7	Prof.	Code	PBR	Freimut, 1997
25	8	Prof.	Code	PBR	Freimut, 1997
26	7	Prof.	Code	PBR	Freimut, 1997
27	5	Acad	Code	Chkl	Runeson *et al*, 1998
28	5	Acad	Code	Chkl	Runeson *et al*, 1998
29	5	Acad	Code	Chkl	Runeson *et al*, 1998
30	5	Acad	Code	Chkl	Runeson *et al*, 1998

a. Artificial requirements specification

b. Collected in connection to the study in Regnell *et al* (2000) though the data set is not published

Table 1: Data Sets

The first context attribute of the experiments is connected to the environmental context in which the experiments took place. Software engineering experiments having students as subjects are often criticized as they are not representatives of the real-life software inspection teams, and hence studies conducted on academics are categorized as a separate group. Several of the studies have been conducted as part of the Software Engineering Laboratory work at NASA (Basili *et al*, 1995). This initiative has been running for more than 20 years and hence the people involved in the studies are likely to have been exposed to more empirical research than other people from industry. There is a substantial number of studies conducted in NASA, and hence NASA is separated as one group. Finally, studies conducted in other industrial settings are viewed as a third group. This results in the following three groups that are related to the environmental context (i.e. profession in Table 1) of the studies:

1. Acad: Mix of students, faculty members and some professionals.
2. NASA: Professional software engineers at NASA.
3. Prof: Professional software engineers other than NASA.

Several different types of artifacts have been used in the studies. A large number of the studies most likely used small size artifacts due to controlled experiments, though the actual size of the artifact being reviewed in many cases was not available in previous publications. Nevertheless, it is here assumed that there are other aspects of the artifacts that we can examine as illustrated in this section. Furthermore, based on the information available on artifact types, it is a management decision to estimate the performance of an inspection when practitioners review real software artifacts. This means that the information here should be viewed as a starting point for taking informed management decisions, and not as any prescribed values. Based on the literature forming the data source for this study, four types of artifacts (i.e. document type in Table 1) are identified in this paper:

1. Req (Requirements specification). This includes studies where a requirements specification from a software development project is inspected.
2. Artif Req (Artificial requirements specification). In several studies, requirements specifications have been developed for the sake of the study. A potential problem with the artificial requirements specifications is that there is a lack of real context, although it resembles a real specification.
3. Code. Several studies have used code in the inspections.
4. Textual. This is text documents written in English. One study used a textual document where the faults where grammatical faults rather than software faults.

Finally, three different types of reading techniques are also identified in Table 1:

1. Adh (Ad Hoc). In this approach the reviewers perform to the best of their ability with no guideline and instructions. There is always some risk in using the ad hoc approach within a control group, since most reviewers tend to apply some sort of formalized process they are familiar with anyway. Hence, it is difficult to understand what their actual behaviour is in comparison with other methods.
2. Chkl (Checklist-based). In this approach, reviewers use a checklist which guides them regarding what kind of faults to look for. The reviewers read the document using the checklist to guide the focus of their review.
3. PBR (Perspective-based). The perspective-based reading technique instructs the reviewer to perform an active review by assigning different perspectives to each reviewer. Common perspectives are user, tester, and designer.

2.3 Individual vs. Nominal Team Effectiveness

A typical inspection includes an inspection meeting. At this meeting, the reviewers, depending on the type of inspection, either focus on identifying as many faults as possible, or discuss the faults the reviewers found during the preparation. The data used in this paper contains no meeting data because of the nature of the experiments conducted; only individual data showing which of the faults a specific individual found or missed.

There have been a number of studies during the last ten years which show little or virtually no meeting gain with respect to the number of found faults in inspections where fault discovery is held in the preparation phase. Votta found in his experiment that the meeting on average found only an additional 4% of faults (Votta, 1993). Johnson and Tjahjono (1998) found that although meeting based reviews are more costly than non-meeting based reviews, there was no significant difference between them in terms of number of faults found. The advantage of having a meeting is that it significantly reduces the number of false positives. Porter *et al* (1995) even found a negative meeting gain averaging around 1%.

One thing the meeting does is to remove *false positives*. False positives are issues, found by individuals that are not actually true faults. However, these false positives do not affect the team's effectiveness of identifying true faults. The data used in this study shows which of the true faults each individual found so the data contain no information of any false positives. If neither the meeting nor the false positives affect a team's effectiveness in terms of finding true faults then a good approximation of how a team would behave can be achieved by counting the number of unique faults found by the team of reviewers in the individual preparation, and then dividing this by the number of total faults in the document. This is the approach employed in this study.

The inspection meeting may have other positive effects such as learning and sharing of knowledge, though this is not in the scope of this paper.

2.4 Generation and Comparison of Nominal Inspection Teams

The terminology on "nominal teams" and "virtual teams" is used interchangeably in software engineering literature (Biffl and Halling, 2003; Briand *et al*, 1997). In some cases the notion of forming teams where the individuals do not communicate with each other is referred as "virtual teams" (Briand *et al*, 1997), in other case this is called "nominal teams" (Biffl and Halling, 2003). Here we follow the management terminology and refer to this notion as "nominal teams".

In order to simulate a full inspection the individual data is combined to form nominal teams of a certain size and by calculating the nominal teams' effectiveness, the effectiveness of a *nominal inspection team* is created. To investigate the whole span of possible outcomes from the data sets, all possible combinations of teams are formed (within each original data set). For example a team of four reviewers a, b, c and d can form teams of size two, i.e. ((a,b); (a,c);...;(c,d)). Each such team is called a nominal inspection team. The approach of combining the reviewers to form inspection teams of different sizes has been used by Briand *et al* (1997) to analyse estimations of remaining faults after inspections. The main objective of combining the data from the different data sets is to:

1. Generate all combinations of nominal teams of all sizes for all data sets
2. Calculate the effectiveness value for all the nominal teams
3. Generate graphs and tables sorted on the number of reviewers

The nominal teams are generated for all data sets separately, i.e. a nominal team does not consist of reviewers coming from different data sets. In other words, the data sets are not combined on an individual level. The data is combined on a nominal team level when the teams have been generated from each data set.

However, since the data sets contain a different number of reviewers, if data from all possible nominal teams was generated then each individual data set's influence on the data as a whole would be highly dissimilar. With six reviewers in a data set, $\binom{6}{3} = 20$ teams of size three could be created, while for 22 reviewers this number would be 1540. This is partly solved by randomly dividing the three largest data sets (data sets no. 6, 11 and 12) into teams of only seven or eight reviewers. This leaves 34 data sets with between five to eight reviewers in each. The differences in influence are thereby reduced.

When generating the nominal teams from the data sets, team sizes from one up to one less than the number of available reviewers are created. This means that, when investigating team sizes larger than four, some data sets have to be excluded. In Section 3, two graphs showing nominal team effectiveness are presented. One with team sizes up to four and one with up to six. In these two graphs, zero and seventeen data sets respectively are excluded. To further decrease the difference in data set influence, the reviewers were selected randomly. For example, in the graph showing the general effectiveness behaviour of teams with size one to four, all reviewers from data sets number 27–30 were included while five reviewers were randomly selected from each of the other data sets.

2.5 Limitations with Nominal Inspection Teams

The approach of creating virtual teams or nominal teams has been used in other studies as well, (see Biffl and Halling, 2003; Briand *et al*, 1997). An advantage of nominal inspections is that it is possible to generate and investigate the effect of different team sizes. A disadvantage is that no effects of the meeting and possible team synergy are present in the data.

The properties of nominal inspection teams have never actually been explored. The rationale for the investigation of nominal teams is to compare nominal inspections with the real world situation where teams would be formed without any resampling. We believe that there are some limitations using nominal inspection teams. By creating all combinations of teams, each reviewer is included in multiple teams. This introduces dependencies among the nominal teams. This results in two potential problems. First, it is difficult to know how representative they are of real inspection teams. Secondly, it is questionable if statistical methods for independent teams can be applied.

To avoid using the same reviewer several times, the alternative would be to randomly create teams out of the reviewers without repetition. This would mean that if we were interested in teams of size two and we had individual data from four reviewers (a, b c and d), only two teams could be created. One outcome would be {(a,d), (b,c)}. This is not a viable method since it would result in very few data points and hence it is disregarded as a possible method to use. One possible option is to use bootstrapping (Efron and Tibshirani, 1993). Bootstrapping is a statistical term that refers to the drawing of a sample from a sample space, followed by immediately returning the sample so that it possibly can be drawn again. However, bootstrapping on an individual level does not work since it will increase the overlap if the same reviewer is chosen more than once in a team. Another option is to use bootstrapping on a team level. This has been evaluated and the differences between bootstrapping on a team level and the use of all nominal teams are primarily in terms of a slightly smaller variation for nominal teams (Petersson, 2002). We have here chosen to use nominal teams and perform a qualitative analysis of the descriptive statistics, instead of applying statistical tests on dependent data. Thus, the main objective here is to create an increased understanding of effectiveness in software inspections looking at the variables, pointed out in Section 2.2: environment context, document type and reading technique. Moreover, it should be noted that the main intention is to make relative comparison between data rather than trusting the actual values too much given that nominal teams are used.

We have also some concerns with the PBR approach in our data set. The fact that PBR assigns different perspectives to the reviewers, combined with the use of nominal teams, leads to problems when analysing the PBR data. In order to make the best use of the PBR, the review teams should include at least one reviewer from each perspective. This would greatly limit the number of inspection cases that can be generated. To be able to use all possible combinations, instead of discarding the PBR data, it is marked as representing an active form of reading technique. This allows for groups of inspection teams without the optimal set of perspectives. This leads to the impossibility of evaluating the true potential of PBR, and therefore, any conclusions concerning PBR are not drawn in this study. In order to be able to handle the issue discussed in this paragraph, we renamed the PBR data as Active Reading Technique (ART) and conducted the data analysis in Section 3 based on ART.

Finally, there are also some dependencies among the 30 data sets. A couple of the experiments are based on an experiment kit prepared by Basili *et al* (1996) during their PBR experiment. In these data sets, the inspected documents are the same or similar to one another. In other cases, the same person has participated in more than one of the experiments. However, there are no cases where the same person inspected the same document.

3. DATA ANALYSIS AND RESULTS

The data is presented in boxplots and analysed qualitatively based on a visual inspection of the boxplots. In a boxplot, the box extends from the 25th percentile, lower quartile, to the 75th percentile, upper quartile, of the estimates. The whiskers (lines extending from the boxes) show the limit for non-outlier values. Outlier values have the following characteristics:

$$\left.\begin{array}{c} Outlier > UQ + 1.5\,(UQ - LQ) \\ or \\ Outlier < LQ - 1.5\,(UQ - LQ) \end{array}\right\} \quad \begin{array}{l} UQ \text{ – Upper quartile} \\ LQ \text{ – Lower quartile} \end{array}$$

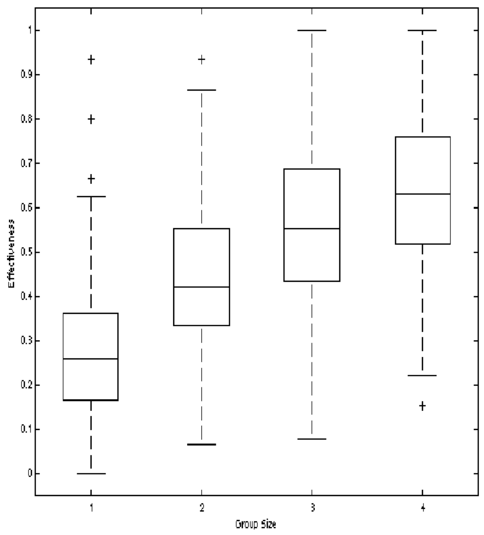
Outliers are marked with plus signs in the graphs below.

3.1 Effectiveness in General

The first two boxplots, Figure 1 and Figure 2, show the effectiveness of teams from data sets where no attribute filtering has been done. In Figure 1, all data sets are represented; however, the data sets with six reviewers or more had five reviewers randomly chosen to generate the nominal teams. Figure 1 shows that the median value for effectiveness is 0.26 for team sizes of one whereas this rises to 0.64 for teams of four reviewers.

In Figure 2, 17 of the data sets have been removed since they only had five or six reviewers and provide no or too few data points for team sizes of six reviewers, see Section 2.4. The median value for six reviewers is 0.71. As expected, the largest gain in adding a reviewer is achieved when using two reviewers instead of one.

Figure 1 and Figure 2 demonstrate how inspection effectiveness increases as team size increases, and how the added value of an additional reviewer decreases as the team size grows. Moreover, it is



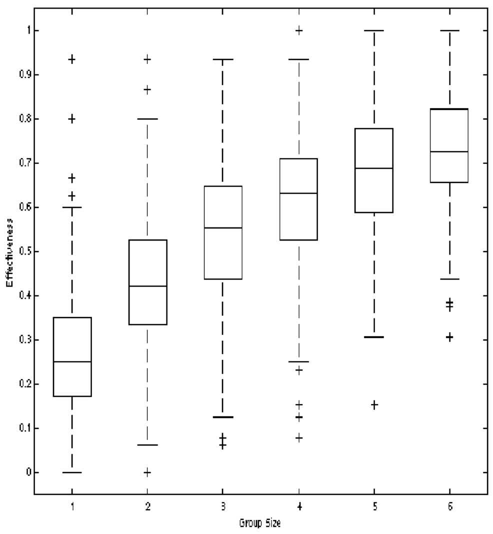


Figure 1: Team effectiveness for all data sets

Figure 2: Team effectiveness with the data sets having less than 7 reviewers excluded

possible to see the variation between different combinations of reviewers. For example, the figures show that for a team size of four reviewers, the median effectiveness is 0.64, and that in 75% of the cases the effectiveness is above 0.5 in both figures. On the other hand, it is also possible to note that in some cases the effectiveness for teams of four is only around 0.2. This type of information would be important for anyone planning, controlling and managing software inspections.

3.2 Effectiveness based on Filtered Data

The next set of boxplots, Figure 3 to Figure 5, show the effectiveness when data sets have been filtered based on the three attributes discussed in Section 2.2: Environmental context, Document type and Reading Technique. The shading legend is shown in Table 2. Moreover, Figures 3–5 should be read from left to right. For example, in Figure 4, the leftmost bar is for textual documents and the rightmost is for code.

From a visual inspection of the figures, some interesting observations can be made. In Figure 4, the large dispersion of the requirements specifications inspections is striking. The good performance of the checklist-based inspections in Figure 5 is also noteworthy. The use of nominal teams does, as stated earlier, not cause any differences in terms of the means. However, the actual variances are

	Figure 3	Figure 4	Figure 5
▓	—	Textual	—
▒	NASA	Artif Req.	AdH
░	Prof.	Req.	Chkl
□	Acad	Code	ART

Table 2: Legend to Figure 3 – Figure 5

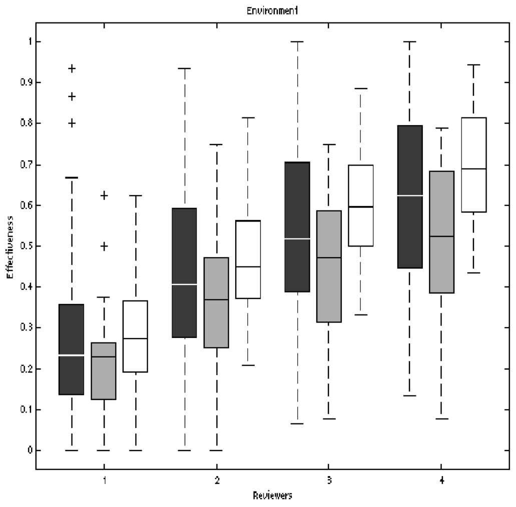


Figure 3: Environment Context

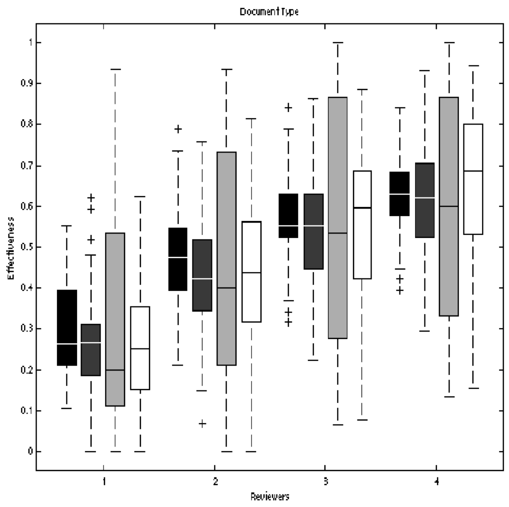


Figure 4: Document Type

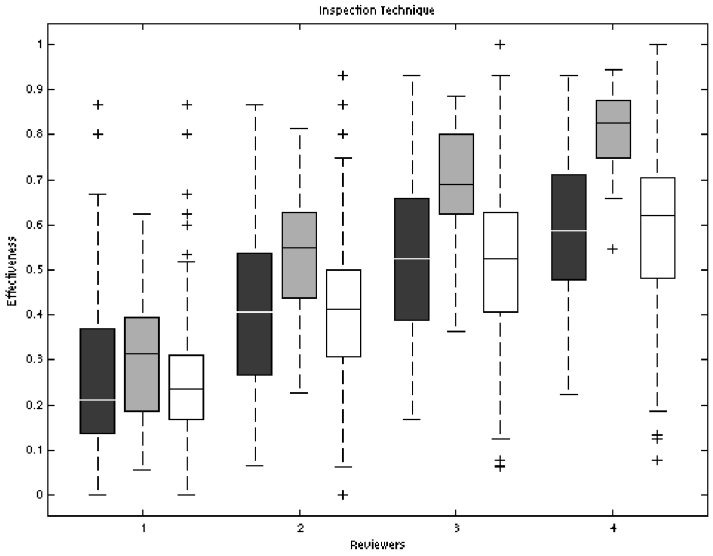


Figure 5: Reading Technique

smaller, although this is the case for all analyses of variation and hence some informal relative observations are made. To further investigate the differences, the following research questions (RQ) have been studied:

RQ1: Are there any visible differences in terms of the mean effectiveness for different: (a) environmental context, (b) types of documents, (c) reading (inspection) techniques.

RQ2: Are there any visible differences in terms of variance in effectiveness for different: (a) environmental context, (b) types of documents, (c) reading (inspection) techniques

Some qualitative findings can be observed from the above plots.

RQ1: Mean effectiveness

(a) Environmental Context: A general trend may be observed, i.e. a pattern is discernible from the plot in Figure 3. It is interesting to note that the order (descending) of the groups in terms of mean effectiveness is: Academia, NASA and other professionals. One possible reason for this result is that experiments at universities are often based on an isolated artifact which has no system context. Another factor that may contribute is that people in industry may be less motivated when performing an experiment on artifacts which are not part of their daily work. If using an artifact from their daily work, then that artifact would most likely have a complex system context which would make it harder to inspect than a stand-alone artifact.

(b) Document type: There are some patterns when it comes to the mean value. The most remarkable one is that the effectiveness for real requirements specifications is the lowest for all group sizes. However, the mean effectiveness is about the same for all types of documents, including reviews aiming at finding grammatical faults in a text document. The main observation from Figure 4 is the large range of variances, which are analysed below.

(c) Reading technique: Checklist-based reading seems to be better for 2–4 reviewers. There are no discernible differences between ad hoc and active reading techniques. This result may be due to the fact that several of the faults are fairly trivial (although this is not known since the severity of faults are not reported in most studies) and they are easily spotted using checklists. Ad hoc reading poses a definition problem in itself, which was also pointed out above. The active reading techniques are not used to their full potential, as different perspectives are combined and analysed here, and hence it may not be so surprising that checklist-based reading comes out more favourably.

RQ2: Variances in effectiveness

(a) Environmental context: NASA has a higher variation than the two other groups. This may be due to that NASA studies include a mixture of artifacts (i.e. a real and artificially created artifacts) compared to the other groups, who largely used only one type of artifact.

(b) Document type: Here, several interesting results are observable. Both requirements specifications and code have a higher variance than the textual documents and the artificial requirements specifications. This indicates that the real artifacts are more challenging and result in greater dispersion between different teams. In addition, it is also noteworthy that the requirements specifications have a higher variance than the code. This implies that requirements specifications are harder to review than code, since variations between different teams are high.

c) Reading technique: Checklist-based reading has a lower variation than ad hoc and active reading techniques. Once again, this outcome could be explained by several of the faults being relatively easy to uncover and furthermore the checklists support finding such faults.

4. DISCUSSION

The results indicate differences in the mean effectiveness for some of the studied attributes. This includes differences between different environments, and differences showing that checklist-based reading outperformed ad hoc and active reading techniques. The results are certainly worth noting, but further studies are required to better understand these issues.

The variations observed are fascinating. They show that the variation in inspection effectiveness when using real software documents, including both requirements specifications and code, is higher than when using other types of documents. It is also noteworthy that the variation in effectiveness for requirements specifications is higher than for code inspections. This indicates some of the problems when performing inspections, and it forms an important input for anyone who plans, controls or manages software inspections.

Another issue to consider is how useful an analysis can be when data is taken from inspections with such a wide range of conditions. It is clear that several factors that may influence the results are not documented, which of course is a threat to the accuracy of the analysis. On the other hand, it is important to start doing these types of analyses to build a body of knowledge, rather than only generating new relatively freestanding studies. It may also be argued that the variety in conditions increases the generality, and this is the best way to start before collecting metrics from within a specific department or company. It should also be noted that if all different conditions are taken into account, then every single inspection becomes unique which compromises estimation efforts.

The analysis in this study uses effectiveness of the inspection to evaluate the results. The effort put in by the reviewers is assumed to be approximately the same in all experiments. The cost in terms of effort is important too, since the effort put into an inspection by the reviewers could instead be used to develop the document or product further (see Petersson, 2001) where the added value of individual reviewers is discussed in more detail). If this had not been a concern, the best option would be to use as many reviewers in the inspection as available. However, even with the effort in mind when deciding on how to perform the inspection it comes down to: *How important is the document?* When this is known, the effectiveness of the inspection team is the next thing to consider. This study provides some initial input for supporting such decisions.

The important part when selecting a team is the overall performance. Selecting a team of people who only find overlapping faults will not add to the effectiveness of the inspection. This implies that further work should be done in investigating and developing methods and tools with aims similar to those of perspective-based reading, namely to help teams detect more faults.

5. SUMMARY

The question *'How many reviewers should be included in the inspection?'* is important to take into consideration when planning an inspection. This study has provided a preliminary understanding, in terms of a qualitative analysis. The analysis is based on 30 published data sets, and it provides a starting point, but to get a better picture of the effectiveness, it is necessary for organisations to build their own localized analyses.

Apart from providing support for project managers or whoever plans and schedules inspections, there are some other issues of interest in the results. The main objective of this paper is to show how inspection teams perform, depending on team size, to support people planning inspections. Another objective of this paper was to investigate the inspection effectiveness when altering some of the inspection attributes, such as the environmental context, document type and reading technique. These attributes have been used for illustrative purposes in this paper. Further studies are needed to analyze the differences based on other attributes such as, the application domain.

This study highlights the need for more investigations into the differences in terms of mean effectiveness between different environmental contexts, document types and reading techniques. The most important result from these comparisons is probably the difference in variance between inspecting different types of documents. It is clear that the differences between different teams when inspecting requirements specifications is significantly higher than for code, which, in turn, results in significantly higher variations between teams than when inspecting artificial requirements specifications and textual documents for grammatical faults. The difference in variance is important to keep in mind when continuing with experimentation in software inspections.

Finally, there is of course a great need for more studies in the field to generate more individual studies that in turn can be used in analysis or even meta-analysis of a series of experiments. In this way, a body of knowledge can be built that increases our general understanding of how to conduct cost-effective software inspections. Furthermore, when investigating issues such as optimizing the number of reviewers, it is important to consider the effort/effectiveness trade-off. Further studies are needed to investigate this issue.

REFERENCES

AURUM, A., PETERSSON, H. and WOHLIN, C (2002): State-of-the-art: Software inspections turning 25 years. *Journal on Software Testing, Verification and Reliability,* 12(3):133–154.

BASILI, V.R., ZELKOWITZ, M., McGARRY, F., PAGE, J., WALIGORA, S. and PAJERSKI, R. (1995): SEL's software process improvement program. *IEEE Software*, 12(6):83–87.

BASILI, V.R., GREEN, S., LAITENBERGER, O., LANUBILE, F., SHULL, F., SØRUMGÅRD, S. and ZELKOWIT, M.V. (1996): The empirical investigation of perspective-based reading. *Empirical Software Engineering: An International Journal*, 1(2):133–164.

BIFFL, S. and HALLING, M. (2003): Investigating the defect detection effectiveness and cost benefit of nominal inspection teams. *IEEE Transactions on Software Engineering*, 29(5):385–397.

BISANT, D.B. and LYLE, J.R. (1989): Two-person inspection method to improve programming productivity. *IEEE Transactions on Software Engineering*, 15(10):1294–1304.

BRIAND, L.C., EL EMAM, K., FREIMUT, B.G. and LAITENBERGER, O. (1997): Quantitative evaluation of capture-recapture models to control software inspections. *Proceedings of the 8th International Symposium on Software Reliability Engineering*, 234–244.

IEEE (1998): *IEEE Standard for Software Reviews*, The Institute of Electrical and Electronics Engineering, Inc. ISBN 1-55937-987-1.

EBENAU, R.G. and STRAUSS, S.H. (1994): *Software Inspection Process*, McGraw Hill.

EFRON, B. and TIBSHIRANI, R.J. (1993): *An Introduction to the Bootstrap*, Monographs on statistics and applied probability, Vol. 57, Chapman & Hall.

FAGAN, M.E. (1976): Design and code inspections to reduce errors in program development. *IBM System Journal*, 15(3): 182–211.

FREIMUT, B. (1997): Capture-recapture models to estimate software fault content. Diploma Thesis, University of Kaiserslautern, Germany.

JOHNSON, P.M. and TJAHJONO, D. (1998): Does every inspection really need a meeting? *Empirical Software Engineering: An International Journal*, 3(1):9–35.

MASHAYEKHI, V., DRAKE, J.M., TSAI, W.T. and RIEDL, J. (1993): Distributed collaborative software inspection. *IEEE Software*, 10(5):66–75.

MILLER, J. (1999): Can results from software engineering experiments be safely combined? *Proceedings of the 6th International Software Metrics Symposium*, 152–158

MONTGOMERY, D. (1997): *Design and Analysis of Experiments*, John Wiley and Sons.

OWENS, K. (1997): Software detailed technical reviews: Findings and using defects, Wescon'97, Conference Proceedings, 128–133.

PETERSSON, H. (2001): Individual reviewer contribution to the effectiveness of software inspection teams. *Proceedings of the 13th Australian Software Engineering Conference (ASWEC'01)*, Canberra, Australia, 160–168.

PETERSSON, H. (2002): Supporting software inspections through fault content estimation and effectiveness analysis, PhD thesis, Dept. of Communication Systems, Lund University, Sweden.

PETERSSON, H., THELIN, T. and WOHLIN, C. (2004): Capture-recapture in software inspections after 10 years research – Theory, evaluation and application'. In *Journal of Software and Systems*, 69(1), 2004.

PICKARD, L.M., KITCHENHAM, B.A. and JONES, P.W. (1998): Combining empirical results in software engineering. *Information and Software Technology*, 40(14):811–821.

PORTER, A.A., VOTTA, L.G. and BASILI, V. (1995): Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on Software Engineering*, 21(6):563–575.

REGNELL, B., RUNESON, P. and THELIN, T. (2000): Are the perspectives really different? – Further experimentation on scenario-based reading of requirements. *Empirical Software Engineering: An International Journal*, 5(4):331–356.

RUNESON, P. and WOHLIN, C. (1998): An experimental evaluation of an experience-based capture-recapture method in software code inspections. *Empirical Software Engineering: An International Journal*, 3(4):381–406.

VOTTA, L.G. Jr. (1993): Does every inspection need a meeting? *Proceedings of 1st ACM SIGSOFT Symposium on Software Development Engineering*, 107–114.

WELLER, E.F. (1993): Lessons from three years of inspection data. *EEE Software*, 10(5):38–45.

WHEELER, D.A., BRYKCZNSKI, B. and MEESON, R.N. Jr. (1996): *Software Inspection: An Industry Best Practice*, IEEE Computer Society Press, USA, ISBN 0-8186-7340-0.

WOHLIN, C., RUNESON, P. and BRANTESTAM, J. (1995): An experimental evaluation of capture-recapture in Software Inspections. *Journal of Software Testing, Verification and Reliability*, 5(4):213–232.

BIOGRAPHICAL NOTES

Aybüke Aurum

Aybüke Aurum is a senior lecturer at the School of Information Systems, Technology and Management, University of New South Wales. She received her BSc and MSc in geological engineering, and MEngSc and PhD in computer science. She is the founder and group leader of the Requirements Engineering Research Group (ReqEng) at the University of New South Wales. She also works as a visiting researcher in National ICT, Australia (NICTA). Dr Aurum is one of the editors of "Managing Software Engineering Knowledge", "Engineering and Managing Software Requirements" and "Value-Based Software Engineering" books. Her research interests include management of software development process, software inspection, requirements engineering, decision making and knowledge management in software development. She is on the editorial boards of Requirements Engineering Journal and Asian Academy Journal of Management.

Claes Wohlin

Claes Wohlin is a professor in software engineering at the School of Engineering at Blekinge Institute of Technology in Sweden. He is also pro vice chancellor of the Institute. Prior to this, he has held professor chairs in software engineering at Lund University and Linköping University. He has an MSc in Electrical Engineering and a PhD in Communication Systems both from Lund University, and he has five years of industrial experience. Dr Wohlin is co-editor-in-chief of the journal of Information and Software Technology published by Elsevier. He is on the editorial boards of Empirical Software Engineering: An International Journal, and Software Quality Journal. Dr Wohlin received the Telenor Nordic Research Prize in 2004 for his achievements in software engineering and improvement of software reliability in telecommunications.

Håkan Petersson

Håkan Petersson is an assistant university lecturer at the IT-University of Göteborg, Sweden. He received his PhD in 2002 in Software Engineering from Lund University, Sweden, and has an MSc in Computer Science and Engineering. His research focus is on software quality with a special interest in Software Inspections and the estimation of remaining faults.