C. Wohlin, H. Petersson and A. Aurum, "Combining Data from Reading Experiments in Software Inspections: A Feasibility Study", in Lecture Notes on Empirical Software Engineering, pp. 85-132, edited by N. Juristo and A. Moreno, World Scientific, 2003.

In Lecture Notes in Empirical Software Engineering", edited by N. Juristo and A. Moreno, World Scientific Publishing, ISBN 981-02-4914-4.

Combining Data from Reading Experiments in Software Inspections
-
A Feasibility Study

Claes Wohlin
Dept. of Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520, SE-372 25 Ronneby
Sweden
Phone: +46-457-385820
Fax: +46-457-27125
E-mail: claes.wohlin@bth.se

Håkan Petersson
Dept. of Communication Syst.
Lund University
Box 118
SE-221 00 Lund
Sweden
Phone: +46-46-222 4910
Fax: +46-46-145823
E-mail: hakanp@telecom.lth.se

Aybüke Aurum
School of Information Systems, Technology and Management
University of New South Wales
Sydney NSW 2052
Australia
Phone: +61-2-9385 4418
Fax: +61-2-9662 4061
E-mail: aybuke@unsw.edu.au

Abstract

Software inspections have been around for 25 years, and most software engineering researchers and professionals know that they are mostly a cost-effective means for removing software defects. However, this does not mean that there is consensus about how they should be conducted in terms of reading techniques, number of reviewers or the effectiveness of reviewers. Still, software inspections are probably the most extensively empirically studied technique in software engineering. Thus, a large body of knowledge is available in literature. This paper uses 30 data sets from software inspections found in the literature to study different aspects of software

inspections. As a feasibility study, the data are amalgamated to increase our understanding and illustrate what could be achieved if we manage to conduct studies where a combination of data can be collected. It is shown how the combinated data may help to evaluate the influence of several different aspects, including reading techniques, team sizes and professionals vs. students. The objective is primarily to illustrate how more general knowledge may be gained by combining data from several studies. It is concluded that combining data is possible, although there are potential validity threats. Research results are examined with reference to software inspections on three levels: organization, project and individual.

Keywords:

Software inspections, reading technique, empirical study, combining data.

1. Introduction

Software inspections have over the years been accepted as a key principle in software engineering. It was first formalized and described by Fagan in 1976 [Fagan76]. Since then inspections have been researched and widely applied. Several variants of inspections have been proposed [Parnas85, Bisant89, Martin92, Knight93]. Software inspections are probably also the most thoroughly empirically studied subject in software engineering [Basili96, Laitenberger97, Porter95, Porter97, Regnell00, Votta93]. Consequentially, several books are now available on this subject [Gilb93, Ebenau94].

The volume of studies in this area implies that it may be possible to combine the various empirically derived information together to build a body of knowledge regarding the effectiveness of software inspections and different aspects of inspections. Examples of such aspects are reading techniques, team size and performance of individual reviewers. Combining empirical information, however, is not a simple task.

To build a body of knowledge in software inspections from published studies requires that the results from these studies are comparable. This imposes significant requirements on the descriptions of the published studies. For example, there are consistency issues regarding descriptions of context, subjects, artifacts and other aspects between the different studies. There have been successful attempts to produce so-called lab packages to encourage replication continuity, such as those based on Basili *et al*'s study

[Basili96]. This is a great starting point, but there is still much to be done. We need ways of documenting empirical studies so that it is possible to combine the results from different studies to allow both meta-analysis [Pickard98, Miller99, Hayes99] and the pooling of data. The latter refers to the combination of data sets, which is the approach used in this paper. The objective is of course to create new or more general results by amalgamating the results from other studies. However, the validity of both meta-analysis and pooling of data may be challenged, since it is always problematic to combine information from different sources. From the published literature it is often hard to understand the exact context of a given study and different studies may have dependencies through, for example, usage of the same artifacts or subjects.

However, the alternative of not combining information or data from different studies is not attractive, since it would mean that studies are primarily interpreted as single events and generalized knowledge is hard to construct. Thus, the challenge is to try to combine information and data in such a way that the results indeed become a general collection of knowledge and experiences. This may be particularly appropriate for some examples in software inspections, especially when the inspections can be viewed as a random sample of inspections in general or when the context is limited to, for example, a specific company. The data, in this paper, does not fulfil these criteria since they are based on convenience sampling [Robson93]. Hence, the main objective is to illustrate what is feasible if combining information or data that is available.

The primary objective of this paper is to illustrate the types of generalized results that can be derived if we were able to combine different studies, whether combining the data or combining the results. In particular, the intention is to show the opportunities for evaluating results at different organizational levels including the organization itself, teams in the preparation phase in software inspections and individual performance. A secondary objective is to present some results from the combination of data from 30 data sets found in the software inspection literature. The actual results should be interpreted with some caution since the data sets are based on availability and hence they are not a true random sample from a population. However, the results may be used as a first indication of what can be expected. In addition, it is of course very important to see whether the results of our combination of data sets is more generally valid even though they are based on convenience sampling.

The primary and secondary objectives are illustrated on three different levels, i.e. organization, project and individual (see Sections 5, 6 and 7), where each level has its own objectives. These are however primarily presented to illustrate how the overall approach can be applied to the different levels of analysis.

We have chosen to perform this feasibility study in an area where quite a number of experiments have been carried out. However, when performing the analysis we realized that we have insufficient knowledge of published studies, thus it is still very hard to perform studies of this type and come to generally accepted conclusions. This points to a very important issue, namely that we must improve the way we document and report experiments. Otherwise, experimental studies will continue to be isolated studies and we will be unable to build a solid body of knowledge based on empiricism. The ability to combine data or results is, in the authors' opinion, a key issue for the success of empiricism in software engineering. With this paper, we hope to illustrate that if the challenges of combining information from different studies could be overcome then there are opportunities to answer some important research questions in the software engineering community.

The paper is structured as follows. Section 2 discusses characterization of software inspections studies. The data sets used in the analysis and some issues related to the data sets are introduced in Section 3. Analyses and discussions are made on three different levels: organization, project and individual. These levels are discussed in Section 4. The following three sections discuss the results for the levels. In Section 5, the organizational benchmarking in software inspections is discussed. Software inspection planning on the project level in terms of team size for software inspections is examined in Section 6. Section 7 presents the results on the individual level and finally the conclusions are presented in Section 8.

2. Characterization of studies

2.1 Introduction

There are many reasons for combining the results and data sets of software inspections. Potential reasons include to make an organizational benchmark study, an internal study to maximize effectiveness within an organization or

to measure the ability of individual reviewers. The different types of studies are further discussed in Section 4, and then elaborated on with specific data in the following sections to illustrate the actual opportunities at different levels. Anyhow, it is important to document these studies to enable greater understanding and comparison among them. To support inspection comparisons, it is necessary to:

- Characterize each inspection process to be compared by application, environment and people factors (i.e. qualitative description),
- Use comparable measures across inspections (i.e. quantitative measurements).

It is possible to only perform a qualitative comparison using the characterization. Fuller characterization also yields the possibility of comparing inspection processes quantitatively. In this case the qualitative description may act both as a means for comparison and as a way of characterizing the process to enable quantitative comparison. The addition of measures means that it is possible to quantitatively compare inspection processes. The main measures to compare include effectiveness (how large a proportion of the defects were found?), efficiency (how many defects were found over a specific period of time? This can be described as defects found per time unit) and the number of reviewers (how many reviewers were needed to achieve a certain level of effectiveness or efficiency?) respectively.

2.2 Characterization

A key aspect of comparison is the characterization, which can either be used as a stand-alone qualitative description or as part of a quantitative evaluation, where the characterization can also be used to support the identification of suitable quantitative comparisons. The characterization includes three perspectives and five aspects that are characterized (see Table 1). The characterization is based on the authors' experience from working with software inspections. The first perspective is the normal working situation, which should capture characteristics related to the working environment and the typical applications developed. The second perspective is related to the resources in the study, i.e. the people participating in the study and the applied inspection process. The third perspective is a characterization of the unique aspects of the study. The latter refers to the fact that, in many studies a document is developed for a specific study or

reused from another study. In many cases, this means that a specific study is conducted in a controlled environment where other artifacts, notation and so forth differ from what the subjects are used to.

TABLE 1. Characterization of software inspections.

Work		**Resources**		**Study**
Environment	**Application**	**People**	**Process**	**Specifics**
Phase	Domain	Native language	Inspection type	Artifact type
Normal notation		Experience in application	Roles	Artifact notation
		Experience in environment	Individual defect detection technique, e.g. reading technique	English or translated
			Meeting	Number of known defects
			Tool support	Experience in study application
			Protocol	Distance from normal artifacts
			Procedure for re-work	

From an environmental point of view, it is important to document the type of inspection (e.g. Fagan [Fagan76] or Phased-Inspections [Knight93]) that is studied as well as the normal notation used in each phase. The characterization in Table 1 may be used both for quantitative and qualitative comparisons. The former is however the main focus in here. The type of application normally developed is important. This should not only include the application domain, but also some additional information, for example, whether it is a soft- or hard real-time system that is normally developed.

Next, the people participating in inspections have to be characterized. This includes their native language and experience, both in terms of the application domain and the environment. The inspection process has to be

documented. It is important to collect as much relevant information as possible about the process. This includes the type of inspection (e.g. Fagan, or walkthrough); the roles used in the inspection; the techniques used for individual defect detection, if any; the data collection procedure (for example comments are sent by e-mail or collected during a meeting); who participates (both as reviewers and in any prospective meeting); and whether any tool support is used in the inspections. It is also essential to document how protocols are written and the procedure for re-work. The processes as applied may be different from the processes as documented, meaning that ethnographic techniques may be appropriate.

Finally, it is important to document aspects that relate to a particular study, i.e. aspects that are specific for the study at hand. This includes the type of artifact, notation used and the number of defects in the artifact used for the study. Preferably, artifacts are developed and made available to other researchers. In these cases, it is advantageous if the artifact can be reused as is. However, in some cases this may be impossible, and it may be necessary to translate it. If this is the case, it needs to be documented that the artifact has been translated from and to particular languages. In many controlled studies, the number of defects is known. This number needs to be identified. Moreover, it is important to document the experience of the people in the application domain of the study (especially if different from their normal application domain). To increase the comparative value of a study, it is important to document the difference in the inspection situation in comparison with how inspections are either described in the literature or conducted at a specific company. In other words, the distance from the normal artifacts and the normal inspection situation to that of the study has to be captured. This is preferably done in a survey after having done the study. The descriptions should include as many aspects as possible, including application type, language, notation, complexity of the review artifact and also general opinions among developers. The developers could, for example, answer questions such as: Was this inspection representative compared to those you normally perform? Was it easier or more difficult? Did you find more, less or an equal number of defects than usual?

2.3 Quantitative measures

The measure of primary interest here is the *effectiveness* of an inspection team. The effectiveness E of a team T, is in this study calculated as:

$$E_T = \frac{D_T}{N}$$

D_T is the number of unique defects found by team T and N is the total number of defects in an artifact.

In the long run it would be very important to also address cost-effectiveness. However given the availability of data and that effectiveness is a starting point also for judging cost-effectiveness, the focus in this paper is on effectiveness. The effectiveness of an inspection team denotes what proportion of the existing defects the team found. The efficiency of an inspection team can be defined in several ways [Briand98] but must include the amount of effort spent by the team.

To obtain comparable measures regarding, for example, the effectiveness of software inspections, it is necessary to list both the defects that were and were not discovered, as this is needed in order to determine the true effectiveness. The most common way of doing this is to conduct a controlled experiment, where the number of defects is known (either through seeding or through previously identified real defects) in a document such as an information or software artifact. The document may be from either a generic or a company-specific domain. The advantage of having a document from a generic domain is that it makes comparison easier. The disadvantage is that the document may not reflect the usual nature of such documents in a specific organization. The company specific documents may on the other hand make comparison more difficult across different environments.

The documents with seeded defects may be from any application domain. In case of a standardized (or generic) artifact (for example in lab packages), it is preferable to find an area, which is familiar to people in general. However, it is also preferable, if few developers have actually developed systems in the chosen application domain, to minimize the risk of affecting the results due to knowledge in that specific domain. Examples of specific domains include an elevator system or a reservation system for taxis. Most people have an intuitive feeling for how these types of systems should work, although most developers have not developed systems in these application domains. Systems, for example, in the telecommunication domain are probably not suited since some of the software is hard to understand unless you have worked in the area. Subjects who have worked with the chosen type of system have major advantages in domain experience to those who have not. This makes comparison of subject’s inspection results difficult.

Another aspect of the artifacts is the phase they represent. It is important to consider different development phases when studying software inspections. One of the main strengths of inspections in general is the possibility of applying software inspections to any type of artifact, but for comparative purposes it is important to document exactly what was inspected. As a first step, inspections of requirements specifications and code could be studied, since several experiments have been conducted which review these types of documents (see Table 1), and hence baseline data already exists. The approaches used in requirements inspections may be extended to other artifacts in the future. The requirements review is especially useful when the specification is written in natural language and hence is readable by most developers, i.e. they need not have any knowledge of any specific high-level language. Code is also readable for developers, even if they are not experts in that specific programming language. However, the use of more common programming languages, such as Java, C or C++, is preferred as more developers are familiar with these languages.

3. Data sets

3.1 General information

This study is based on publicly available data, and the main objective herein is to illustrate how software inspection data may be used to evaluate several important questions regarding software inspections. It must be noted that full characterizations of the different contexts for some of the individual data sets used here are not available and hence the derived empirical results are presented with a degree of caution. The objective is to describe how, if an appropriate characterization is conducted, this type of information can be used for comparison purposes and for drawing more general conclusions about software inspections.

The data has primarily been collected as part of formal experiments [Wohlin00]. For the sake of this illustration, let us imagine that the data is collected from different companies. This is done for symbolic purposes to show the feasibility and opportunity of combining data from software inspections.

The data used is a collection of 30 data sets from a number of controlled experiments performed by different researchers in the field. In some of the analyses, the data sets are analyzed as one entity and for other analyses the data sets are classified based on three attributes: a) the environment in which the study was conducted (Profession), b) the type of document, and c) the reading technique used (Inspection Technique). The data sets and their attributes are shown in Table 2, and the attributes are further explained below. Three of the data sets, i.e. no. 6, 11 and 12, have been divided into sub sets to lessen the effect of any large single data set in the analysis, see Section 3.2.

The data sets are used without having complete information about the full context of the environments, and it should once again be pointed out that the data sets are based on availability. This is means that results should be interpreted with some caution, and primarily viewed as an illustration of what can be achieved if able to combine information from different studies. It also illustrates some of the problems associated with combining data sets. Several factors that may influence the results are unknown about in the data sets. For example, information is not available regarding time spent in inspections, motivation of the inspectors, severity of defects and several other factors describing the actual context of each study in Table 2. It was only possible to use three attributes to describe the context, as pointed out above, and shown in the table.

3.2 Virtual inspection teams

A typical inspection includes an inspection meeting. At this meeting, the reviewers gather the defects, and depending on the type of inspection, they either focus on identifying as many defects as possible, or discussing the defects they have found during the preparation. The data given from the 30 experiments contain no meeting data; only individual data showing which of the defects a specific individual found or missed.

It should also be noted that a number of studies during the last ten years which, when performing inspections where the fault discovery is focused to the preparation phase, show small or virtually no faults are found during meetings. For instance, Votta found in his experiment that during the meeting on average only an additional 4% of faults were found [Votta93]. When including true faults that were reported by an individual but did not get logged at the meeting, Johnson *et al*. found no significant difference

between having a meeting or not [Johnson98]. Porter *et al*. even found a negative meeting gain of on average around minus 1% [Porter95].

In the experiments generating the data for this study, the focus of reviewers was to find defects during the preparation, not in the inspection meeting itself. In order to study the effect of teams, the individual data are combined to form nominal teams of a certain size and by calculating the team's effectiveness, a virtual inspection is created. This virtual inspection does not take meeting effects into account. To investigate the whole span of possible outcomes from the data sets all possible combinations of groups are formed.

One approach for 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

However, since the data sets contain different numbers of reviewers, each data set's influence on the graphs would be dissimilar. With six reviewers, 20 teams of size three could be created, while for 22 reviewers this number would be 1540. This is partly solved by dividing the three largest data sets (data set no. 6, 11 and 12) between groups of only seven or eight reviewers. This leaves 34 data sets with five to eight reviewers in each. The differences in influence are thereby reduced. It should be noted that in the long run, the aim should be to base the comparison only on real groups to ensure that the conclusions are based on groups that are similar to the ones found in industry.

When generating the nominal teams from the data sets, team sizes from 1 up to one less than the number of available reviewers is created. This means that, when investigating larger team sizes than four, some data sets have to be excluded. In Section 6.3, two graphs showing general behavior are presented. One with team sizes up to four and one with up to six. In these two graphs, the data sets numbered 0 and 15 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 1 to 4 (Figure 3), all reviewers from data sets number 26-30 were included while 5 reviewers were randomly selected in the other data sets.

The disadvantage of virtual groups is that there is a high dependency between the groups. On the other hand, all data sets are treated the same, and

since the main concern is comparison, this should not be critical to the outcome. A random selection of all combinations is used so that all data sets get a similar weight. Otherwise data sets with the most reviewers would dominate over the others.

3.3 Dependency concerns

Since each reviewer was included in many of the teams, there exist an obvious dependency between the data points. To evaluate some of the dependency, a simulation of virtual teams versus an approach that randomly creates teams without redraw and an approach having only real teams has been conducted. The simulation approach seems better than the random-no-redraw approach. Compared to the real-teams-only, the virtual team approach generates results with the same mean value but reports less variance in the results. This should be remembered when looking at the graphs. However, the approach of using virtual teams shows the full scope of what the effect could be of having these people as reviewers in a company and picking some of them to be included in each inspection team.

There are also some dependencies among the 30 data sets. A couple of the experiments are based on an experiment kit or lab package developed during Basili *et al*'s PBR experiment [Basili96]. 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.

3.4 Classification of the data sets

The characterization of the data sets is shown in Table 2. The data is characterized based on type of subjects (NASA representatives**,** Academics and professionals other than NASA), document type (requirements specification, artificial requirements specification[2], text and code) and reading technique (ad hoc, checklist, and active-based reading; an example of the latter is perspective-based reading [Basili96]). The data provides

[2] The term artificial requirements specification is used when the specification is developed for the sake of the experiment.

opportunities to make controlled comparisons to evaluate if, for example, inspection rates vary by the profession, document type or reading technique.

TABLE 2. Data Sets.

No.	No. of review.	Profession	Doc. Type	Insp. Tech.	Reference	No.	No. of review.	Profession	Doc. Type	Insp. Tech.	Reference
1	8	NASA	Artif. Req[3]	AdH.	Freimut97	14	6	NASA	Artif. Req	ART	Freimut97
2	6	NASA	Artif. Req	AdH.	Freimut97	15	6	NASA	Req	ART	Freimut97
3	6	NASA	Artif. Req	AdH.	Freimut97	16	6	NASA	Req	ART	Freimut97
4	6	NASA	Artif. Req	AdH.	Freimut97	17	7	NASA	Req	ART	Freimut97
5	6	Acad	Artif. Req	Chkl	Unpubl.[4]	18	6	NASA	Req	ART	Freimut97
6a	8	Acad	Textual	AdH.	Wohlin95	19	8	NASA	Artif. Req	ART	Freimut97
6b	7	Acad	Textual	AdH.	Wohlin95	20	6	NASA	Artif. Req	ART	Freimut97
6c	7	Acad	Textual	AdH.	Wohlin95	21	8	Pro.	Code	ART	Freimut97
7	7	NASA	Req	AdH.	Freimut97	22	7	Pro.	Code	ART	Freimut97
8	6	NASA	Req	AdH.	Freimut97	23	8	Pro.	Code	ART	Freimut97
9	6	NASA	Req	AdH.	Freimut97	24	7	Pro.	Code	ART	Freimut97
10	6	NASA	Req	AdH.	Freimut97	25	8	Pro.	Code	ART	Freimut97
11a	8	Acad.	Artif. Req	ART	Regnell00	26	7	Pro.	Code	ART	Freimut97
11b	7	Acad	Artif. Req	ART	Regnell00	27	5	Acad	Code	Chkl	Runeson98
12a	8	Acad.	Artif. Req	ART	Regnell00	28	5	Acad	Code	Chkl	Runeson98
12b	7	Acad	Artif. Req	ART	Regnell00	29	5	Acad	Code	Chkl	Runeson98
13	6	NASA	Artif. Req	ART	Freimut97	30	5	Acad	Code	Chkl	Runeson98

The first context attribute of the experiments is connected to the environment in which the experiments took place. Software engineering experiments, having students as subjects, are often criticised as they are not representative of the real life software inspection teams. Hence studies conducted on academics are categorized as a separate group. Several of the

[3] Artificial requirement specification.

[4] Collected in connection to the study in [Regnell00] though the data set is not published.

studies have been conducted as part of the Software Engineering Laboratory work at NASA [Basili95]. 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. As a result, 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 environment of the studies:

1. Mix of college students, faculty members and some professionals. (Acad)
2. Professional software engineers at NASA. (NASA)
3. Professional software engineers from outside NASA. (Prof.)

Several different types of artifacts have been used in the studies. The following four types were identified:

1. Requirements specification (Req.): This includes studies where a requirements specification from a software development project is inspected.
2. Artificial requirements specification (Artif. Req.): In several studies, requirements specifications have been developed for the sake of the study. The objective is that these should resemble real requirements specifications. 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 (Code): Several studies have used code in the inspections.
4. Plain text document written in English. (Textual): One study used a textual document where the defects where grammatical defects rather than software defects. This study is included to observe whether the effectiveness is significantly different when reviewing with a different purpose compared to normal software development.

Finally, three different types of reading techniques are identified:

1. Ad Hoc (AdH.): This simply means that the reviewers were neither taught nor instructed to use any special kind of formal inspection or reading technique. The reviewers all performed to the best of their ability. It should be noted that there is always a risk with using ad hoc as a control group, since most reviewers apply some method and hence it is difficult to understand what their actual behavior is in comparison with other methods.

2. Checklist-based (Chkl): When checklist-based inspections are performed there is a checklist introduced to the reviewers beforehand. This list is used to guide the reviewers regarding what kind of defects to look for. The reviewers read the document using the checklist to guide their review.
3. Active Reading Technique (ART): Most of these studies use a perspective-based reading (PBR) technique. However since we would like to use virtual groups it is not possible to guarantee that all groups include all of the different perspectives and hence we would like to refer to this new type of inspection as being active-based reviews. Thus, the results should not be interpreted as representative of PBR. It has also been discussed elsewhere, [Laitenberger01], that some of the benefits of PBR comes from team effect. In short PBR instructs the reviewer to use an active form or review by assigning different perspectives to each reviewer. The common perspectives are *user*, *tester*, and *designer*. With the perspective follows a detailed description of how to perform the inspection. The instructions involve active steps such as 'Construct test cases for...' or 'Make a small design of...'. Some concerns regarding PBR and the analysis here are discussed in the following paragraph.

The fact that PBR (perspective-based reading) assigns different perspectives to the reviewers, combined with the use of nominal groups, leads to problems when analyzing the PBR data. The use of different roles was proposed by Fagan [Fagan76], although the emphasis on active reading is more recent. In order to make the best use of the PBR, the review teams should include at least one reviewer from each perspective. This greatly limits the number of PBR compliant inspection teams that can be generated. Our virtual team generating approach 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 cannot be drawn in this study. The PBR data are renamed to ART (Active Reading Technique).

4. Levels of study

Three levels of comparisons can be identified: organization, project and individual. At the organizational level, of potential interest could perhaps be benchmarking a particular inspection process with respect to industry standards or other specific partners. Alternatively, an objective could be to select a reading technique or better understand the effectiveness of inspections in different development phases. Organisational benchmarking is discussed in Section 5.

At the project level, it is often important to learn more about the effectiveness of different team sizes. Typically a project manager would also like to plan the inspections within projects so as to maximize effectiveness. In these cases, a manager would like to know how many reviewers to assign in different phases of the development in order to obtain a certain degree of inspection effectiveness. The effectiveness of different team sizes is studied in Section 6.

Finally, it is important to know more about the individual performance. Of particular interest is the gaining of an understanding of the differences between individuals to be able to select a suitable inspection team. It is well known that there are individual differences, but it is important to ascertain how large they are. This is investigated in Section 7.

5. Organisation: Benchmarking

This chapter investigates the inspection data from an organizational perspective. The intention is to examine what we can learn from the data with a view to benchmarking the software inspection process of an organization.

5.1 Benchmarking in general

Benchmarking is a widely used business practice and has been accepted as a key component in an organization's search for improvement in quality, competitive position or market share. According to a survey in 1992, 31% of US companies were regularly benchmarking their products and services. Another survey in UK (1996) revealed that 85% of the business was using benchmarking practices [Ahmed98]. In Japan, benchmarking is called

"dantotsu", which means "striving to be the best of the best" [Corbett98]. Here, we would like to define benchmarking of processes as an activity that allows people to strive to be the best of the best. Thus, both qualitative and quantitative comparisons with this objective are viewed here as being benchmarking.

The literature describes several types of benchmarks [Sole95, Ahmed98, Longbottom00]. Sole and Bist point out that the level of benchmarking sets the degree of the challenge from a slight improvement in the development process to a radical change in the process [Sole95]. Benchmarking may be divided into different types, depending on with whom the comparison is made and what the objective of the comparison is. Some common types of benchmarking include:

- comparison within the same organizations (internal benchmarking),
- comparison with external organizations (external benchmarking),
- comparison with competitors (industry benchmarking),
- identification of best practices (generic benchmarking),
- comparison of discrete work processes and systems (process benchmarking),
- comparison of performance attributes e.g. price, time to market (performance benchmarking), and
- addressing strategic issues (strategic benchmarking).

5.2 Benchmarking goal

The goal here is to be able to compare different software inspection processes. Given the characterization and standardized artifacts, it is possible to identify, for example, whether a specific inspection process is better or worse than another.

Some key concerns regarding benchmarking are scalability, thresholds, simplicity and atypical situations. Scalability should not be a major problem, as long as the inspections scheduled on real projects are of limited size. Since normal recommendations on the length of the preparation phase and an inspection meeting are in the order of hours, it is feasible to benchmark a realistic approximation of the process. Scalability must also be addressed by using documents representative of what is normally seen at an organization, with respect to size and defect density. In this case, the objective is not to set

quality thresholds on the documents, but rather to provide feedback on effectiveness and efficiency and expectations on these two factors in terms of group size. Simplicity is also very important because in order to make a benchmark useful, it should be possible to replicate it without having to have a number of experts present. For instance, the characterization scheme from Table 2 supports simplicity. Finally, it is often reported that software projects are so different from each other that it is not possible to compare them. It may be true that projects are very different, but it should still be possible to compare certain aspects of software projects, for example, software inspections. The differences and similarities should be captured by the characterization used and hence atypical inspections should be accounted for in subsequent analysis. Atypical inspections may be important to learn from, but they should not be part of the normal benchmarking data, since, by definition, it is not anticipated that their individual situations will reoccur**.**

5.3 Benchmarking in software development

Benchmarking provides many opportunities for comparisons in software development, for example, compilers may be compared by compiling the same program on several different compilers and by logging compilation time and errors.

Benchmarking in software development is perceived as an assessment method, which is concerned with the collection of quantitative data on topics such as effectiveness, schedules and costs [Jones95, Beitz00]. It allows the comparison between an organizational process and industry best practice. It also helps managers to determine whether significant improvements are required to maintain a particular business [Beitz00]. Here, the term benchmarking is used for both qualitative and quantitative comparisons as long as the main objective of benchmarking is fulfilled. Thus, a characterization of several processes in qualitative terms would qualify as benchmarking if the objective is to improve these processes. Informally, the following definition of benchmarking is used in this paper. Process benchmarking is the comparison of similar processes in different contexts, it implies multiple points of comparison (e.g. two data points is not a benchmarking), and it requires a representative sample in terms of, for example, organizations and applications.

Several assessment tools for software benchmarking have been developed. Maxwell and Forselius report on the development of an experience database which consists of 206 business software projects from

26 companies in Finland [Maxwell00]. This database allows managers to compare their projects with the existing projects from the database.

5.4 Benchmarking in software inspection

To our knowledge, limited work has been done in the area of benchmarking software inspection. One of the few examples is described by Jones [Jones95] who argues that function points provide useful metrics on two components of software quality: (a) potential defects, which is the total number of defects found in a work product, and (b) defect removal effectiveness level, which is the percentage of software defects removed prior to delivery. Jones reports that in the US the average for potential defects is about five per function point, and overall defect removal effectiveness is about 82%. According to one recent study, code inspection reduces life cycle defect detection costs by 39%, and design inspection reduces life cycle defect detection costs by 44% [Briand98].

5.5 Research questions

Using the data described in Section 3, it should be possible to answer, the following benchmark questions:

1. Are there any differences in terms of effectiveness between requirements specification inspections and code inspections? Assuming that the primary interest is to benchmark an organization, the artificial requirements specifications may be treated together with the requirements specifications. However, the textual documents cannot be included when answering this benchmark question.
2. Are there any differences in terms of effectiveness between the different reading techniques? This question is important to the organization as an answer to it allows for the selection of a suitable reading technique.
3. Hypothetically, we could also remove data sets 19 and 29 from the database, and assume that the organizations represented by these data sets would like to compare their inspections with the ones remaining in the database. Thus, it is possible to use the data from the other organizations and compare this with these to two fictitious companies for requirements and code inspections respectively.

These types of questions and studies can be conducted as more and more data becomes available. The intention here is to illustrate how a benchmarking database for software inspections can be created.

To answer the above questions the data will be presented in box plots and discussed qualitatively. The reason being that there is dependence between the data points and hence the data do not fulfil the requirements for using statistical tests.

5.6 Analysis

The three questions above are addressed to illustrate how software inspection benchmarking can be used to study a number of issues of interest to software management in their effort to improve the software development process. It is here assumed that all data sets used for comparisons come from "comparable" companies, i.e. the characterization shows that it is reasonable to compare the companies. The benchmarking is illustrated with effectiveness and the intention is that a manager can transfer this information to cost-effectiveness within his or her own environment .

1. Effectiveness in requirements specification and code inspections

   In the left box plot group in Figure 1, the effectiveness in software inspections are shown for different types of documents. These box plots are shown for 1 to 4 reviewers with requirements specifications inspections to the left and code inspections to the right for the different cases. The focus is on 1 to 4 reviewers since the number of combinations is very few for higher number of reviewers, and the results would depend too much on single data points rather than representing a more general outcome.

   From the box plots, it seems obvious that the differences in terms of effectiveness between requirements specifications inspections and code inspections are minor. From a benchmarking perspective, this means that we may conclude that we can expect that our effectiveness for different types (or at least for requirements specification inspections and code inspections) of inspections should be approximately the same. Given that the faults in code inspections ought to be easier to find, we could have expected a higher effectiveness in code inspections. This was however not the case, which is an interesting observation. This information is valuable when planning different types of inspections. In particular, it implies, for example, that any experience regarding effectiveness for

code inspections could probably be transferred to inspections of requirements specifications due to effectiveness in the different types of inspections (in our case requirements and code) being fairly similar.

2. Effectiveness in inspections using different reading techniques
   In the right box plot group in Figure 1, the box plots for the effectiveness for different reading techniques are shown. Once again, the plots provide information for 1 to 4 reviewers with the plots in the following order: ad hoc, checklists and active reading technique. From the box plots, it seems as though checklists may be more effective than the other two techniques. This may result from us looking at individual inspection preparations only and not the potential team effect from having different perspectives. From a benchmarking perspective, this tells us that if we have good checklists they ought to outperform ad hoc inspections and active reading techniques on an individual preparation level. However, our study has not taken any team effects similar to the ones introduced by different perspectives in PBR into account. The benefits of a team approach, in particular when using active-reading techniques such as PBR, is further discussed by Laitenberger *et al.* [Laitenberger01]. The challenge is to develop inspection techniques that are strong both in terms that the individual will find many faults during the preparation and also that from a team perspective, the individuals complement each other well.

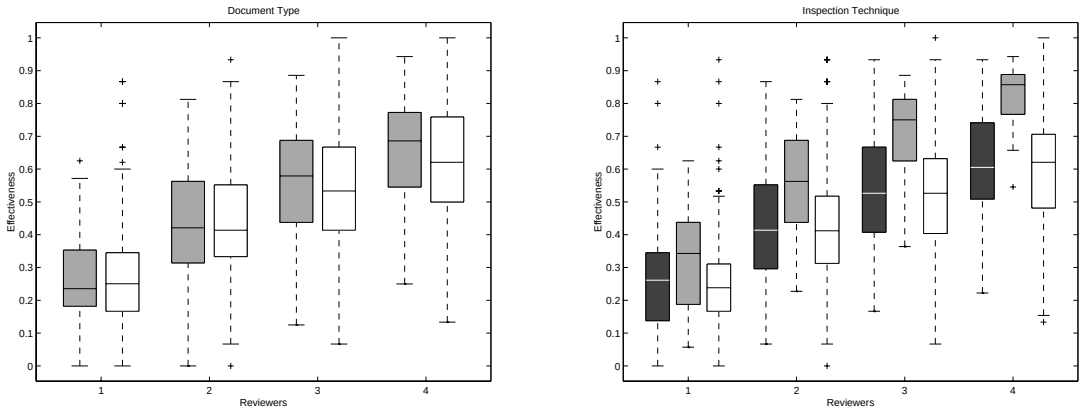

FIGURE 1. Box Plots[5] showing effectiveness in inspections for different types of documents (left) and for different reading techniques (right). The documents are from the left: requirements specification and code. The reading techniques are from the left ad hoc, checklists and active reading technique.

3. New company scenario
   Here it is assumed that data sets 19 and 29 are not part of the experience base, and the box plots in Figure 2 are created without these two data sets. In particular, it is assumed that the data sets represent two companies which we will compare with a subset of the companies in the experience base selected on the basis of their

[5] In box plots, a line in the box indicates the median value. Moreover, each 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:

$$\boldsymbol{Outlier} > \boldsymbol{UQ} + 1.5(\boldsymbol{UQ} - \boldsymbol{LQ}) \quad \textit{or} \quad \boldsymbol{Outlier} < \boldsymbol{LQ} - 1.5(\boldsymbol{UQ} - \boldsymbol{LQ})$$

$$\begin{cases} \boldsymbol{UQ} - \text{Upper quartile} \\ \boldsymbol{LQ} - \text{Lower quartile} \end{cases}$$

Box plot outliers are marked with plus signs.

similar characterizations. Thus, the box plots may be seen as the benchmarking experience base that new companies may use for planning and comparing their own software inspection process. The circles in the box plots represent the two new data sets (or companies from a scenario perspective) with data set 19 on the left diagram and 29 on the right.

Company No. 19 could use the left diagram in Figure 2 to ascertain what to anticipate when performing inspections. It is possible to see the median value for requirements specification inspections as well as the different quartiles and whiskers. Thus, the company has a good picture of the industry standard for effectiveness of requirements specification inspections. After having conducted controlled inspections at the company, it is possible to plot how this company is performing. This is illustrated with the circles in the box plot. It can be seen that company No. 19 in this case performed below median for all group sizes. The difference between how close the circle is to the median depends on the individuals that are added as we increase the group size. It may be particularly interesting to actually study the performance of individuals and hence use this information to put together inspection teams. This issue is discussed in Section 7.

To the right in Figure 2, a similar diagram is shown for code inspections. In this case, it is worth noting that company No. 29 is performing better than the industry standard.

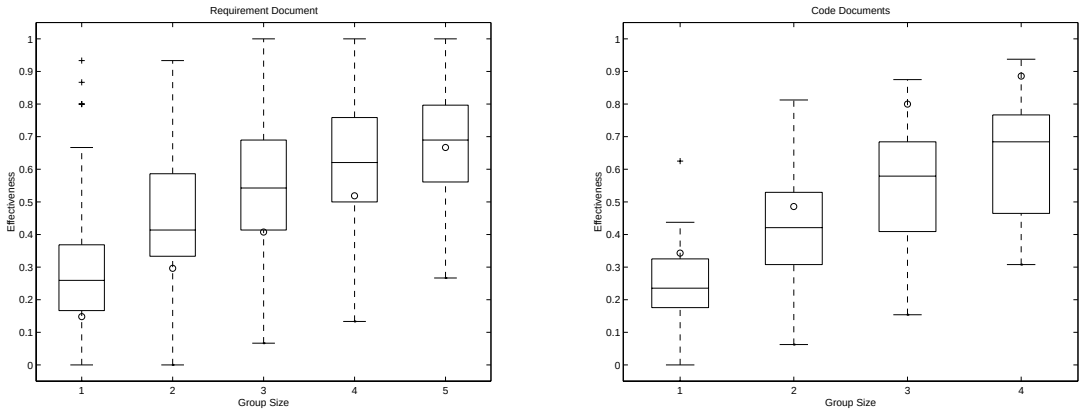

FIGURE 2. New company scenario for requirements specifications inspections (left) and code inspections (right).

In summary, the illustration above shows examples of questions that can be addressed from using benchmarking in software inspections. The actual figures are not the main issue. The illustration is based on the assumption that the data sets are indeed comparable, i.e. the characterizations of the data sets are similar. These figures indicate the level of effectiveness that we can anticipate for different reading techniques, different types of documents and different group sizes.

6. Project: Team size

In Section 4, three different levels of study were introduced. In the previous section, the opportunity of benchmarking software inspection processes was investigated. The objective in this section is to evaluate the use of the data from a project perspective. The intention is to show how a project manager may use the data to help in planning and managing software inspections.

6.1 Teamwork in small groups versus large groups

Teamwork is essential for software developers. There are some major assumptions behind teamwork. Firstly, the software products in today's

market are too complex to be designed by individuals. The complexity of the product has become the driving force behind the creation of teams. It has been found that the quality of the product improves as it is inspected from multiple viewpoints [Laitenberger97]. Secondly, there is a belief that people are more committed to their work if they have a voice in the design of the product or the work in general [Smart98]. According to researchers, effective teams work best when the nature of their work requires a high level of interdependence [Smart98]. Valacich and Dennis emphasize that while many factors affect group performance, group size is a key ingredient since it places a limit on the knowledge available to the meeting group [Valacich94].

Several researchers in management science have focused on determining the optimal group size for teamwork where the members of the team are assigned to a set of tasks *e.g.* new product development or strategic decision-making. The findings from these studies are contradicting. Some researchers indicate a dozen is an optimal number and recommend an odd number to assure a majority in the case of conflicting ideas [Osborn57]. Some earlier studies report five people as an optimum number for a typical group size [Hackman70, Chidambaram93, Tan94]. In [Jessup90], some experiments conducted using electronic supporting systems suggested a small number -- four persons per group as an ideal size. On the other hand, Nagasundaram and Dennis remark that it is better to have small groups in face-to-face meetings, and large groups for electronic meetings [Nagasundaram93] . In summary, team size is a key issue when trying to optimize available resources. Given the need to use software development resources effectively, there is also a need to understand how to choose an appropriate team size for software inspections.

6.2 Team size in software inspection

Software inspection involves teamwork, where a group of individuals work together to analyze the product in order to identify and remove defects. Inspection teams are formed from small groups of developers. Fagan [Fagan76] suggests that four people are a good inspection team size. This suggestion also ties in with Weller's industrial work, where it is reported that four person teams are more effective and efficient than three person teams [Weller93]. IEEE STD 1028-1997 suggests teams of 3 to 6 people [IEEE98]. Teams of 4 or 5 people are common in practice [Wheeler96]. Owens 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) [Owens97]. This idea of having two people in a coding inspection is supported by other researchers [Bisant89]. Bisant and lyle's findings illustrate that programming speed increases significantly in two-person inspections. Thus, it is clear, from literature, that there is no real consensus on the number of reviewers to use and hence this illustrates the need to combine data from different studies to obtain a more general understanding.

Given the different arguments and results when identifying a good size for teams in both management and software inspections, an analysis is required to investigate the relationship between performance and team size. In particular, team size refers to the combination of individuals and not real teams. The focus here is directed at performance comparisons for various team sizes, which provides valuable information to people planning inspections. In the work presented here, we have used virtual teams, see Section 3.2. The team size in virtual inspections has also been studied by others [Biffl01].

6.3 Effectiveness in general

The first two box plots, Figure 3 and Figure 4, show the effectiveness of teams from the analyzed data sets where no attribute filtering has been done. In Figure 3, all data sets are represented, however, the data sets with six reviewers or more had *five* reviewers randomly chosen to generate the virtual teams. The median value is 0.26 for a team size of one and 0.64 for a team size of four. In Figure 4, 17 of the data sets have been removed since they only had five or six reviewers and do not provide any data for team sizes of six reviewers. The median value for 6 reviewers is 0.71. As expected the largest gain in adding a reviewer is achieved when using two reviewers instead of one.

In Figure 3 and Figure 4, it can be seen how the inspection effectiveness increases as the team size increases, and how the added value for an additional reviewer decreases as the team size grows. Moreover, it is 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. On the other hand, it is also possible to note that in some cases the

effectiveness is only around 0.2. This type of information is important for anyone planning, controlling and managing software inspections.

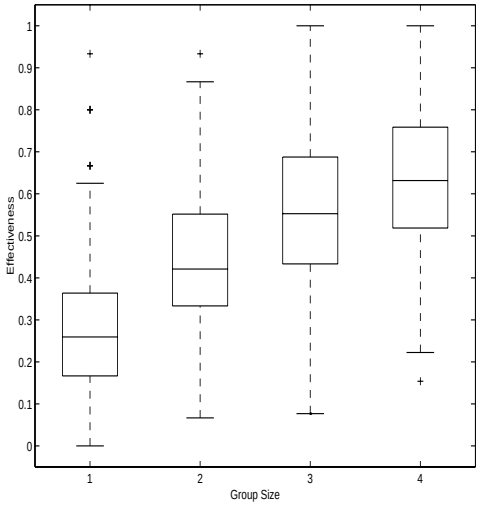

FIGURE 3. Team effectiveness for all data sets.

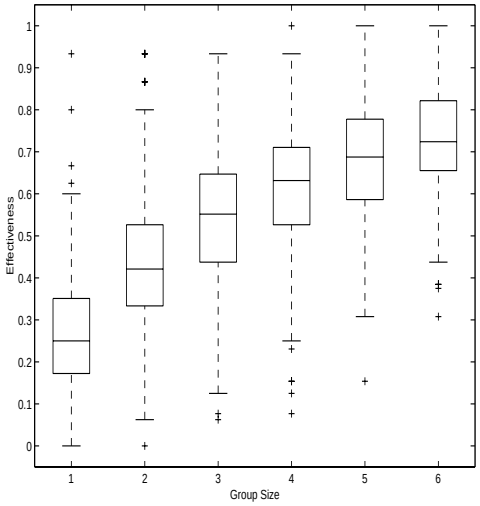

FIGURE 4. Team effectiveness for data sets with less than 7 reviewers not included.

6.4 Effectiveness based on filtered data

The next set of box plots, Figure 5 to Figure 7, shows the effectiveness of different team sizes when data sets have been filtered based on the three attributes discussed in Section 3.4: Environment, Document type and Inspection type. The legend is shown in Table 3.

TABLE 3. Legend to Figure 5—Figure 7

	Figure 5	**Figure 6**	**Figure 7**
	-	Textual	-
	NASA	Arti. Req.	AdH
	Prof.	Req.	CheckL
	Acad.	Code	ART

From a visual inspection of the figures, some interesting observations can be made. In Figure 6, the large dispersion of the requirements specifications inspections is striking. The good performance of the checklist-based inspections in Figure 7 is also noteworthy. To further investigate the differences, the following research questions have been studied qualitatively:

1. Are there any differences in terms of the mean effectiveness for different:
   a) environments,
   b) types of documents,
   c) reading techniques.
2. Are there any differences in terms of variance in effectiveness for different:
   a) environments,
   b) types of documents,
   c) reading techniques.

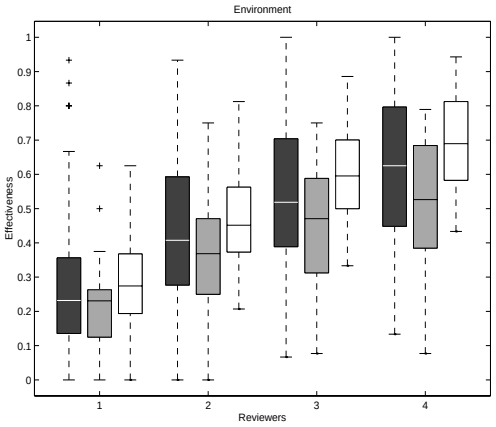

FIGURE 5. Environment

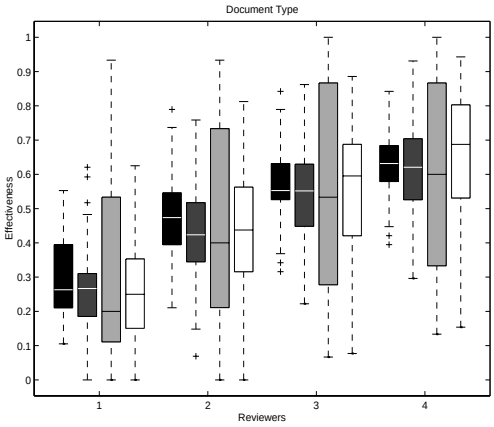

FIGURE 6. Document Type

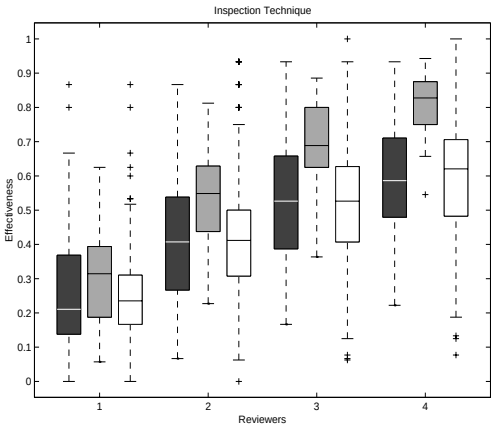

FIGURE 7. Reading Technique

The result were as follows:

1. Mean effectiveness

   a) **Environment**: It seems that the academic environment better supports inspections compared to the other two environments. This is particularly visible for 3 and 4 reviewers. It also seems that the

professionals at NASA perform better than professionals in general. 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 often are 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 they use an artifact from their daily work, then that artifact most likely has a complex system context which may make it harder to inspect than a stand alone artifact.

b) **Document type**: There are no visible differences in mean effectiveness for different types of documents. It is interesting to note that independently of the type of artifact, the mean effectiveness is about the same, including reviews aimed at finding grammatical defects in a text document. The main observation, from Figure 6, is the large variations, which are discussed below. Future work includes investigating the combined effect of, for example, a document type and a specific reading technique.

c) **Reading technique**: Checklist-based reading seems to be better, at least this is the case when only looking at the combination of individual data. However, the result may be a result of not having team meetings and hence the full potential of active-reading techniques is not explored. Effectiveness changes through having meetings or not is discussed further in Section 3.2. This result may be due to the fact that several of the defects are fairly trivial (although this cannot be fully ascertained since the severity of defects is not reported in most studies) and these are easily spotted using checklists. As pointed out above, ad hoc reading poses a problem in itself. Further studies are needed to explore whether, for example, certain individuals perform better using one technique rather than another.

2. Variations of effectiveness
   a) **Environment**: NASA seems to have a higher variation than the two other groups. This may be due to the fact that the studies at NASA include a mixture of real software artifacts and artificially created artifacts to a larger extent than the other groups.
   b) **Document type**: Several interesting results were observed. 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 worth mentioning that the requirements specifications seems to have a higher variance than the code. This implies that requirements specifications are harder to review than code, since the variation between different teams is high.
   c) **Reading technique**: Checklist-based reading seems to have a lower variation than ad hoc and active reading techniques respectively. Once again, this outcome could be explained by the fact that several of the defects were relatively easy to uncover. Furthermore, the checklists support finding such defects.

6.5 Planning table

The box plots provide an insight into how the effectiveness of inspection teams varies, depending on the number of reviewers in each team**.** To support the decision-making process regarding which team size to choose in a specific situation, information is extracted and presented in Table 4. This table shows the percentage of all the virtual inspections that had a certain minimum level of effectiveness. For example, if we would like to be at least 75% certain of finding a minimum of 50% of the defects, then 4 reviewers are needed, (see the grey cell in Table 4). A table of this type could be used as a rule of thumb in different ways, depending on whether we would like to determine the team size, or if we would like to know the expected effectiveness resulting from a specified team size.

TABLE 4. Certainty of having a specific effectiveness for a specific team size.

Effectiveness	1	2	3	4	5	6	7
0.10	90	99	100	100	100	100	100
0.20	68	93	99	100	100	100	100
0.30	36	80	93	96	97	100	100
0.40	17	59	80	88	92	94	100
0.50	9	36	62	78	86	92	95
0.60	6	20	41	58	66	77	75[6]
0.70	2	9	22	37	43	55	56
0.80	2	6	13	22	22	30	34
0.90	1	2	4	7	11	13	5
1.00	0	0	1	2	3	3	0

6.6 Discussion of managing inspection effectiveness

When planning an inspection, several issues have to be considered. For example: How many reviewers should be included? Who should participate? What roles should they have? When should the inspection meeting be held? and What inspection technique and reading technique should be used? It is common when dealing with these more practical matters of the planning, to forget one very important question; What level of quality do we aim for in the inspected document? If we know the approximate answer to this question, we automatically have a general outline of how to answer many of the other questions.

The level of quality aimed for in a specific document depends on many factors, which can be summarized by answering the one question; *How important is the document?* This is where the project manager or whoever plans the inspection should begin. By processing and analyzing the different aspects of a document's importance, the manager gets an understanding of what level of quality is needed before releasing the document to the next

[6] This is an example of a lower value although we have more reviewers; this is a statistical artifact based on the limited number of data sets to create the columns for six and seven reveiwers.

development phase. Thus, he or she also may identify what amount of effort should be spent on that specific document.

The number of defects within a document strongly affects the document´s quality. In order to improve the quality, the number of defects must be reduced. As can be seen in Section 6.3, the number of reviewers involved greatly affects the proportion of defects detected by an inspection team. Therefore, this is one of the more important decisions to be made by the person planning an inspection.

This study shows the opportunities in terms of providing a basis for supporting such decisions. The person planning the inspection can use Table 4 to get an indication of the level of risk that exists when choosing a certain number of reviewers. This risk can then be weighted against the importance of the document to obtain an estimate of the number of reviewers to use.

Table 4 shows general results from a number of inspection experiments. By letting each company collect data to build their own tables, more relevant data can be attained. This would increase the accuracy of the data by narrowing down the variety of variables including document types, people involved and reading techniques used. Most of the measures needed to construct similar tables can be easily collected from inspections. The most difficult part would be in getting accurate numbers of the total number of defects. Different approaches could be taken to acquire this measure. Two examples include using Capture-Recapture estimations [Wohlin95] or controlled experiment, within the company in which the defects are known.

7. Individual: Reviewer effectiveness

This is the third level of investigation as pointed out in Section 4. Here we would like to study the data to better understand the effect of individual reviewers. This is important when planning inspections and also helps reviewers understand their own abilities.

7.1 Research questions

There are many aspects of individual contributions when considering inspection effectiveness. The analysis in this study as stated previously, is nominal inspection teams, i.e. no meetings are held. One important role of the meeting is to identify false positives, i.e. issues incorrectly reported as

defects. Information on false positives is typically unavailable from our data sets, particularly as they come from controlled experiments where the number of existing defects is known. Including meeting data would also lead to an increased scope of possible impacts an individual may have, including aspects such as how a single person affects the group psychology.

The questions investigated and illustrated in this study are:

1. What impact does a single reviewer have on average in terms of inspection effectiveness?
2. What impact does the reviewer with the best individual effectiveness have in terms of the team's total inspection effectiveness?
3. What impact does the reviewer with the worst individual effectiveness have in terms of the team's total inspection effectiveness?
4. Does the combination of the two best individuals in a team always find most unique defects?

The aim of the first question is to provide a general view of what can be expected when adding or removing a person from a team. Question two and three investigate how the best or worst individual effectiveness affects the team's effectiveness. The fourth question investigates one aspect of a very interesting area: What makes a good team? It is not only individual contributions that are important when being part of a team.

7.2 Theoretical analysis

To gain some insight into what could be expected when investigating the effect of a single reviewer, the process of inspections can be modelled statistically and simulated by Monte Carlo simulations. These two approaches are shown below.

The simplest approach is to assume that all defects are equally difficult to find and all reviewers have equal ability. If the probability for a defect to be found by a reviewer is *p*, then the probability that at least someone in a team of size *R* finds a specific defect is:

$$p_T = 1 - (1 - p)^R$$

If a document contains N defects then the expected value of the number of defects found in the inspection, E(D), is:

$$E(D) = \sum_{D=0}^{N} \binom{N}{D} \cdot p_T^{\,D} \cdot (1-p_T)^{N-D} \cdot D = \ldots = p_T \cdot N$$

The average difference in effectiveness a single reviewer makes on a team of R reviewers is then:

$$\frac{N \cdot (1-(1-p)^R)}{N} - \frac{N \cdot (1-(1-p)^{R-1})}{N} =$$

$$= (1-p)^{R-1} - (1-p)^R = p \cdot (1-p)^{R-1}$$

Table 5 shows the expected differences for different values of p. For example, when finding defects with a probability of 0.4 the average difference in effectiveness when removing one reviewer from a three-reviewer team is 0.14.

TABLE 5. Expected differences when removing one reviewer from a team.

	Group Size				
p	**2**	**3**	**4**	**5**	**6**
0.1	0.09	0.08	0.07	0.07	0.05
0.2	0.16	0.13	0.10	0.08	0.07
0.3	0.21	0.15	0.10	0.07	0.05
0.4	0.24	0.14	0.09	0.05	0.03
0.5	0.25	0.13	0.06	0.03	0.02
0.6	0.24	0.10	0.04	0.02	0.01
0.7	0.21	0.06	0.02	0.01	0.00
0.8	0.16	0.03	0.01	0.00	0.00
0.9	0.09	0.00	0.00	0.00	0.00

The previous model's assumptions are very restrictive. The assumptions can be relaxed by allowing the probability to vary among different reviewers. Then each reviewer i, has the probability p_i to find a defect. Thus:

$$p_T = 1 - \prod_{i=1}^{R} (1 - p_i)$$

$$E(D) = \ldots = N \cdot \left(1 - \prod_{i=1}^{R} (1 - p_i) \right)$$

The different probabilities (p_i, i = 0...R) have then to be estimated or represented in some way. This is done as in [Boodoo00], i.e. using a Beta-distribution to represent the p_i's and using the available data sets to estimate the parameters of the Beta distribution.

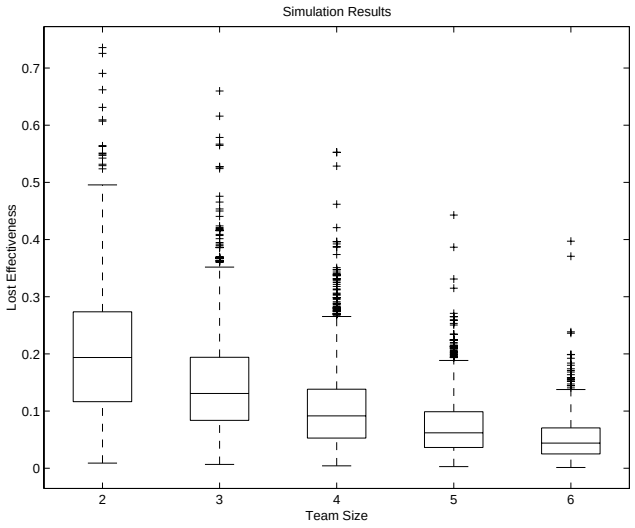

Figure 8. Monte Carlo simulated effect of removing one reviewer.

A simulation of the difference a single reviewer makes was run with team sizes of 2 to 6 people and the p_i's taken from the Beta-distribution. 30 simulated experiments for each team size were run. The approach with virtual team combinations is used in the simulation to facilitate comparison. The result is shown in Figure 8. For example, the median of the lost

effectiveness when removing one reviewer from a three-reviewer team is close to 0.13.

Some caution must be taken when drawing conclusions from this Monte Carlo simulation. The Beta distribution does not handle reviewers with an effectiveness equal to zero. There are 8 cases out of 255 in the data sets where a reviewer did not find any defects. In order to estimate the parameters of the Beta distribution, these sets were removed.

7.3 Analysis procedure

The following measures are used to investigate the four questions.

1. What impact does a single reviewer have on average in terms of the inspection effectiveness?
   Measure I: Eff(*Full Team*)-Eff(*Team with one person removed*)
   All possible combinations is considered. A virtual team, with reviewers A, B and C, generates 3 combinations. Eff(A, B, C) minus either Eff(A, B), Eff(A, C) or Eff(B, C)
2. What impact does the reviewer with the best individual effectiveness have in terms of the team's total inspection effectiveness?
   Measure II a: Eff(*Full Team*)-Eff(*Team with the reviewer with the best individual effectiveness removed*)
   Example: Inspection with reviewers A, B and C, where they have individual effectiveness 0.43, 0.27 and 0.32, respectively. Measure IIa = Eff(A, B, C)—Eff(B, C)
3. What impact does the reviewer with the worst individual effectiveness have in terms of the team's total inspection effectiveness?
   Measure II b: Eff(*Full Team*)-Eff(*Team with the reviewer with the worst individual effectiveness removed*)
   To investigate the difference between the best and the worst another measure is added.
   Measure II c: (Measure II a)—(Measure II b)
4. Does the combination of the two best individuals in a team always find most unique defects?
   Measure III: (Number of times a combination of two reviewers within a team is found to have better effectiveness than the two reviewers with the best effectiveness) / (Number of virtual inspections)

The result of these measures is presented in Section 7.4.

7.4 Results

7.4.1 Measure I

Measure I is calculated for all of the possible virtual teams and all possible removals of one reviewer. The mean and variance of measure I are calculated and presented in Table 6. To investigate whether the *profession*, *document type* or *inspection technique* has any impact on the result, the data sets have been filtered based on these criteria. The average mean and variance, when all data sets are treated together, are shown at the bottom of the table.

TABLE 6. Single reviewer impact on inspection effectiveness

	Team Size									
	2		**3**		**4**		**5**		**6**	
	Mean	**Var**	**Mean**	**Var**	**Mean**	**Var**	**Mean**	**Var**	**Mean**	**Var**
NASA	0.168	0.020	0.107	0.010	0.074	0.005	0.055	0.004	0.042	0.002
Prof.	0.144	0.010	0.103	0.007	0.076	0.005	0.057	0.004	0.045	0.003
Acad.	0.173	0.010	0.112	0.006	0.075	0.004	0.053	0.002	0.038	0.002
Text	0.172	0.011	0.109	0.007	0.073	0.004	0.052	0.003	0.038	0.002
Req.	0.170	0.032	0.098	0.014	0.063	0.007	0.043	0.004	0.029	0.002
Artif. Req.	0.163	0.008	0.110	0.005	0.077	0.004	0.056	0.003	0.042	0.002
Code	0.159	0.012	0.110	0.008	0.079	0.005	0.058	0.004	0.045	0.003
Ad Hoc	0.161	0.016	0.104	0.008	0.074	0.005	0.055	0.003	0.043	0.002
Chkl	0.218	0.015	0.151	0.009	0.105	0.006	0.070	0.004	0.029	0.001
ART	0.159	0.013	0.106	0.007	0.074	0.004	0.054	0.003	0.041	0.002
All	0.165	0.014	0.108	0.008	0.075	0.005	0.055	0.003	0.042	0.002

The table shows that the difference a single person makes on the effectiveness varies on average from about 0.16 in the two-reviewer case to 0.04 in the six-reviewer case. The largest effect is when checklists are

compared to Ad Hoc and the Active Reading Technique. There is a sharp drop in the checklist case between five and six reviewers. This is probably because there is only one data set using checklists for six reviewers (data set number 5), see Table 2. The largest variance is found in the data sets from inspecting requirement specifications.

To further illustrate the outcome, a box plot for four reviewers is shown in Figure 9.

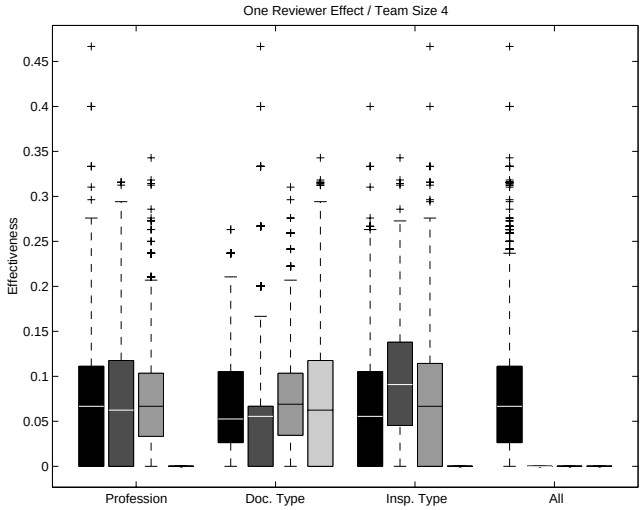

FIGURE 9. One reviewer effect. From left to right (NASA, Prof., Acad.; Text, Req., Artif. Req., Code; Ad Hoc, Chkl, ART and All)

7.4.2 Measure IIa, b and c

The aim of measure II is to capture the effect extreme reviewers, i.e. the best and worst, have on a team's effectiveness. Table 7 shows the mean and variance of all data sets. On average, the effect of the best reviewers varies from 0.24 for two reviewers down to 0.10 for six reviewers. The effect of the reviewer with the worst personal effectiveness drops from 0.10 down to only 0.01.

TABLE 7. Mean and variance of measure II taken of all data sets for different team sizes.

	Team Size									
	2		3		4		5		6	
	Mean	**Var**	**Mean**	**Var**	**Mean**	**Var**	**Mean**	**Var**	**Mean**	**Var**
IIa, best	0.237	0.014	0.186	0.007	0.147	0.005	0.118	0.003	0.096	0.002
IIb, worst	0.092	0.004	0.047	0.002	0.027	0.001	0.016	0.001	0.010	0.000
IIc, difference	0.144	0.017	0.139	0.010	0.121	0.006	0.103	0.004	0.086	0.003

Although the detailed statistics are not presented here, when Table 7's data is filtered on the different context attributes, the results are similar to measure I (see Table 6). Checklists show slightly larger mean values up to five reviewers while requirements specifications show the largest variance.

Figure 10 shows a box plot for measure IIa-c for all data sets with different team sizes. A feature not captured in the table is the occurrence of negative differences in the effect between the best and the worst reviewers. This represents cases where the reviewer with the worst effectiveness has found more unique defects than the best reviewer (Note: This condition can not occur in the two reviewer case).

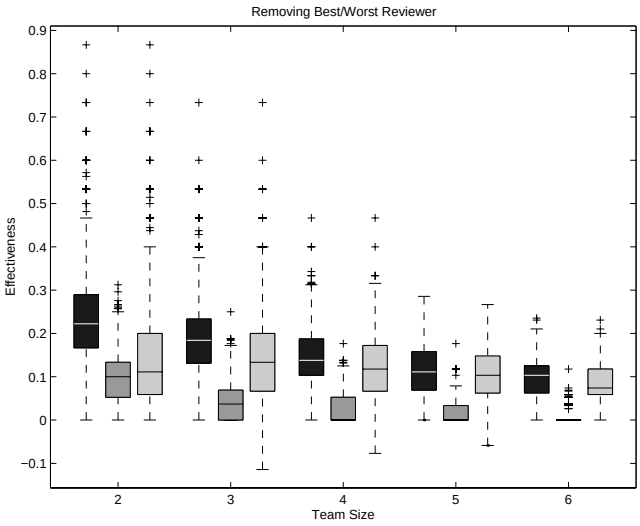

FIGURE 10. Effect of removing best or worst reviewer from the team. From left to right for each team size: Best, Worst, Difference.

7.5 Measure III

Measure III investigates whether teams with the best individual reviewers necessarily form the best team. In Table 8, measure III is presented. This shows how often, within a team, the best combination of two reviewers is not the two reviewers with best effectiveness individually. For example, in 62 percent of the four-reviewer teams it is possible to find a better combination of two reviewers than the effectiveness of the individuals involved would suggest. The two-reviewer case is not investigated, since there is only one combination, and it is automatically the best.

TABLE 8. Percentages of how often a better combination of two reviewers than the two with the best individual effectiveness could be found within a team.

	Team Size			
	3	**4**	**5**	**6**
Measure III	53%	62%	75%	74%

8. Conclusions

Combining data or results from different studies of software inspections provide many interesting opportunities for comparison and evaluation. This includes benchmarking at the organizational level, inspection planning at the project level and an improved understanding of individual performance.

8.1 Organizational benchmarking

The results in Section 5 show variations in the mean effectiveness across some of the studied attributes. This includes differences between environments, and differences showing that checklist-based reading outperformed ad hoc and active reading techniques at least when only comparing the individual results and nominal teams and ignoring the team effects for real teams. The results are certainly interesting, but further studies are required to better understand these issues.

The variations observed are very interesting. They show that differences in inspection effectiveness when using real software documents, including both requirements specifications and code, is greater than when using other types of documents. It is also noteworthy that the variation in effectiveness for requirements specifications is higher than for code. This illustrates some of the problems when performing inspections and shows the potential of performing a study of this type. Such results, if confirmed by later studies, would form an important input for anyone who plans, controls or manages software inspections.

Benchmarking opens a number of interesting opportunities. The need for empirical studies and experimentation in software engineering is well known. Software inspection is an area well suited to experimentation, both in industry and in universities. By agreeing on a number of standardized

artifacts or use of company specific artifacts and a characterization schema, it should be possible to make a large number of experiments world-wide and hence it should be possible to rapidly develop greater understanding in this area.

Software inspection benchmarking is interesting for universities as well as industry. Industry may perform benchmarking as discussed above. Universities may perform experiments allowing for more data to be collected regarding inspections, which would enable researchers to better understand different aspects of software inspections more fully. This becomes particularly valuable if the universities use the same documents as are used for benchmarking in industry. For example, universities should be able to run experiments with students in courses where different reading techniques can be compared.

8.2 Project management

The analysis in Section 6 uses effectiveness of the inspection to evaluate the results. The effort is assumed to be approximately the same in all experiments. The cost of effort is important too, since the effort put into inspection could instead be used to further develop the documents or product itself. If no effort considerations apply, the best option would be to use as many reviewers as available. However, even with effort considerations when deciding on how to perform an inspection, it still comes down to the question stated in Section 6.6, *How important is the document?* When this is known, the effectiveness of the inspection team is the next thing to be considered.

Main objectives of this study include exploring how inspection teams behave, depending on team size, and identifying how inspection planning can be better supported. This was done by combining data from different controlled inspection experiments. Another aspect that was illustrated was the effect on the inspection effectiveness when altering some of the inspection attributes, such as the environment, document type and reading technique.

The question of *How many reviewers should be included in the review?* is an important consideration when planning an inspection. This study has provided an initial answer, in terms of a table from which it is possible to estimate the number of individuals needed in a team to reach a certain effectiveness (in preparation) or vice versa (to estimate the effectiveness for a given team size). The table is based on 30 published data sets, and it

provides a starting point. However, to get a better picture of the effectiveness of inspection, it is necessary for organizations to build their own experience bases.

8.3 Individual performance in inspections

The study in Section 7 used data sets from several controlled inspection experiments to illustrate the impact an individual reviewer has on an inspection team's effectiveness. The results provide some rules of thumb to consider when planning inspections. The averages presented show rather a small individual reviewer contribution to the inspection effectiveness. However, good individual performance is still important and, in reality, may have a larger effect than the averages suggest. For example, having good reviewers can decrease the needed size of the inspection team and thereby reduce the effort cost of the inspection. Therefore it is important for companies to analyze their own inspection process to guide their decisions and to perhaps pinpoint reviewers with inspection 'talents'. Their ability can be utilized on important projects and their knowledge may be able to be captured and taught to others.

When addressing individual performances, four questions were posed. The aim of the first question was to study how much impact in general, an individual reviewer has on the effectiveness of an inspection team. The average effect is of course dependent on the team's size and the mean was found to be 0.16 in the two-reviewer case down to 0.04 for six reviewers. Compared to the theoretical models in Section 7.2, the investigated data sets behaved similar to the second model although the theoretical model had larger median values, especially in the two-reviewer case.

The second and third questions focused on the degree to which the best and worst reviewers effect the team's performance. The difference between their impact on the team's effectiveness (Measure IIc) is on average 0.14 in the two-reviewer case, down to about 0.09 with teams of size six. This can be seen as an approximation of the risk that exists when choosing members of an inspection team. These values show a limited risk but there is still an impact. For example, in a project with 10 design documents each having about 30 major defects, picking a worst reviewer instead of the best reviewer to a team of four (difference of 0.11) would on average lead to 33 defects not being found because of the choice of reviewer.

There are of course, as shown in Figure 10, cases where the difference is much larger but in general most cases show a difference below 0.3. In the

example above, the most extreme outlier for four reviewers would lead to that about 140 defects of the total 300 being missed because of the staffing.

The answer to question four indicates that the individual effectiveness of a typical reviewer only views focuses on a single dimension of the inspection task. Selecting the people with the best individual effectiveness provides no guarantee of finding the most unique defects. As soon as a team is created, the individual effectiveness is not that important. Individual expertise increases the chances of the team finding many defects, but in order to make the team effective, the reviewers dimension focuses should complement each other. This is an important issue to remember when choosing an inspection team.

It was interesting to discover that the inspections that used checklists generally had a larger difference between the best and the worst reviewers. The expected impact of using a checklist would be to increase the number of defects a reviewer finds, but also making the reviewers more homogenous in what defects they find. However, the data shows that on average larger single reviewer impact as well as larger differences between the best and the worst reviewers.

Independent of which method is used, education of the reviewers would be beneficial. Sauer *et al* argue that the dominant factor of a team's effectiveness is the amount of *expertise* within the group [Sauer00]. Consequently, it is important to train reviewers to increase their knowledge and ability of reviewing and create tools and methods that utilize the expertise within the team.

8.4 Conclusions from combining data

An important issue to consider is how useful the combination of data from different studies can be when the data is taken from inspections with such a wide range of conditions. This study and its results are primarily to be viewed as a feasibility study, although the findings themselves also provide some interesting results in terms of summarizing some of the studies that are available in the literature. 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 having a variety of conditions increases the generality, and this is the best way to start before collecting metrics from within a specific department or company.

A study of this type may be criticized for its debatable validity, due to the lack of control and knowledge about the context of many of the studies. This also includes the general problems of performing different studies where either data or results are later combined. However, the solution is certainly not to avoid doing these types of studies. On the contrary, it is necessary to start undertaking such combined studies and attempt to meet the challenges of the resulting analyses in order to gain deeper understanding of this area. Despite the potential threats to the validity, the following key findings are of major interest:

- There are no visible differences in effectiveness for different document types.
- There are clearly differences in the variation of effectiveness for different document types. The variation is larger for real documents and is higher for requirements specification than for code.
- It is possible to determine the effectiveness for different team sizes. This can be used for decision-making when planning and managing software inspections.
- There are differences between different types of subjects. However, the results indicate that people in academia are more effective. This is probably due to the fact that the documents used in academia are more stand-alone than documents investigated in an industrial setting. Thus, the result may be due to a confounding factor. This is an area for further investigation.
- Checklists turned out to be more effective than other types of reading techniques. This may also be a result of a confounding factor, namely experience. It is believed that less experienced reviewers would benefit more from checklists than others. Due to the fact that the experience of the subjects is unknown, we are unable to evaluate this. This is also an area for further studies.
- It is clear from the data that individual differences exist and hence it is important to put together an inspection team cautiously so as to make the best possible use of the available resources.
- A combination of the individually best reviewers is not necessarily the most effective team. This means that there are more effective combinations of reviewers than simply putting together the individuals that perform the best in the individual preparation. This stresses the need to further develop approaches using the expertise of different reviewers in an effective way.

Finally, there is of course a great need for more studies in this field to generate more individual studies that, in turn, can be utilized in meta-analysis, pooling of data or 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. It should also be noted that the combining of data and results in itself is an important area of research in software engineering. In particular, we must increase our understanding of when it is reasonable to combine data or results. The main objective of this paper has been to look at what can be achieved in this type of approach.

Acknowledgment

The authors would like to thank Dr. Forrest Shull, Fraunhofer USA Center for Experimental Software Engineering in Maryland, USA and Marcus Ciolkowski, University of Kaiserslautern in Germany for many valuable discussions regarding benchmarking in software inspections. We would also like to thank Thomas Thelin and Dr. Per Runeson at Lund University in Sweden for many valuable discussions on software inspection research and Peter Parkin and Irem Sevinc from University of New South Wales, Australia.

References

[Ahmed98] Ahmed, P. K. and Rafiq, M. (1998): “Integrated Benchmarking: A Holistic Examination of Selected Techniques for Benchmarking Analysis”. Benchmarking for Quality & Technology, 5(3), pp. 225-242.

[Basili95] Basili, V. R., Zelkowitz, M., McGarry, F., Page, J., Waligora, S. and Pajerski, R. (1995): “SEL's Software Process Improvement Program”, IEEE Software, Vol. 12, No. 6, pp. 83 -87

[Basili96] Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S. and Zelkowitz, M. V. (1996): “The Empirical Investigation of Perspective-Based Reading”. Empirical Software Engineering: An International Journal, 1(2), pp. 133-164.

[Beitz00] Beitz, A. and Wieczorek, I. (2000): “Applying Benchmarking to Learn from Best Practices”. Proceedings 2nd International Conference on Product Focused Software Process Improvement, Oulu, Finland.

[Biffl01] Biffl, S. and Gutjahr, W. (2001): “Analyzing the Influence of Team Size and Defect Detection Technique On the Inspection Effectiveness

of a Nominal Team". Proceedings International Software Metrics Symposium, pp. 63-73, London, UK.

[Bisant89] Bisant, D. B. and Lyle, J. R. (1989): "Two-Person Inspection Method to Improve Programming Productivity". IEEE Transactions on Software Engineering, 15(10), pp. 1294-1304.

[Boodoo00] Boodoo, S., El Emam, K., Laitenberger, O. and Madhavji, N.: "The Optimal Team Size for UML Design Inspections", National Research Council Canada, ERB-1081, NRC 44149. 2000.

[Briand98] Briand, L; El Emam, K., Laitenberger, O. and Fussbroich, T. (1998): "Using Simulation to Build Inspection Efficiency Benchmarks for Development Process". Proc. of the IEEE International Conference on Software Engineering, pp. 340-349.

[Chidambaram93] Chidambaram, L. and Bostrom, R. P. (1993): "Evolution of Group Performance Over Time". Journal of Management Information Systems, 7, pp. 7-25.

[Corbett98] Corbett, L. M. (1998): "Benchmarking Manufacturing Performance in Australia and New Zealand". Benchmarking for Quality Management & Technology, 5(4), pp. 271-282.

[Ebenau94] Ebenau, R. G. and Strauss, S. H. (1994): "Software Inspection Process". McGraw Hill (System Design and Implementation Series), ISBN 0-07-062166-7.

[Fagan76] Fagan, M. E. (1976): 'Design and code inspections to reduce errors in program development'. IBM System Journal. 15(3), pp. 182-211.

[Freimut97] Freimut, B. (1997): "Capture-Recapture Models to Estimate Software Fault Content". Diploma Thesis, University of Kaiserslautern, Germany.

[Gilb93] Gilb, T. and Graham, D. (1993): "Software Inspection". Addison Wesley Publishing Company. ISBN 0-201-63181-4.

[Hackman70] Hackman, J. R and Vidmar, N (1970): "Effects of Size and Task Type on Group Performance and Member Reactions". Sociometrics, 33, pp. 37-54.

[Hayes99] Hayes, W. (1999): "Research Synthesis in Software Engineering: A Case for Meta-Analysis". Proc. of the IEEE International Software Metrics Symposium, pp. 143-151.

[IEEE98] IEEE (1998): "IEEE Standard for Software Reviews". The Institute of Electrical and Electronics Engineering, Inc. ISBN 1-55937-987-1.

]Jessup90] Jessup, L. M., Connolly, T. and Galegher, J. (1990): “The Effects of Anonymity on GDSS Process with an Idea Generation Task”. Management Information Systems, Quarterly, 14(3), pp. 313-412.

[Johnson98] Johnson, P. and Tjahjono, D. (1998): “Does Every Inspection Meeting Really Need a Meeting”, Empirical Software Engineering: An International Journal, Vol. 3, No. 1, pp. 9-35.

[Jones95] Jones, C. (1995): “Software Challenges”. IEEE Computer, 28(10), pp. 102-103.

[Knight93] Knight, J. C. and Myers, E. A. (1993): “An Improved Inspection Technique”, Communications of the ACM, Vol. 36, No. 11, pp. 51-61.

[Laitenberger97] Laitenberger, O. and DeBaud, J. (1997): “Perspective-based Reading of Code Documents at Robert Bosch Gmbh”. Information and Software Technology, 39(11), pp. 781-791.

[Laitenberger01] Laitenberger, O., El Emam, K., Harbich, T. G. (2001): “An Internally Replicated Quasi-Experiment Comparison of Checklist and Perspective-Based Reading of Code Documents”. IEEE Transactions on Software Engineering, Vol. 27, No. 5, pp. 387-421.

[Longbottom00] Longbottom, D. (2000): “Benchmarking in the UK: An Empirical Study of Practitioners and Academics”. Benchmarking: An International Journal. 7(2), pp. 98-117.

[Martin92] Martin, J. and Tsai, W. T. (1992): ‘*N*-Fold Inspection: A Requirements Analysis Technique’. Communications of ACM, 33(2), 225-232, February.

[Maxwell00] Maxwell, K. D. and Forselius, P. (2000): “Benchmarking Software Development Productivity”. IEEE Software, January/February 2000, pp. 80-88.

[Miller99] Miller, J. (1999): “Can Results from Software Engineering Experiments be Safely Combined?”. Proc. of the IEEE International Software Metrics Symposium, pp. 152-158.

[Nagasundaram93] Nagasundaram, M. and Dennis, A. R. (1993): “When a Group is not a Group: The Cognitive Foundation of Group Idea Generation”. Small Group Research, 24(4), pp. 463-489.

[Osborn57] Osborn, A. F. (1957): “Applied Imagination: Principles and Procedures of Creative Thinking”. Charles Scribner’s Son, New York.

[Owens97] Owens, K. (1997): “Software Detailed Technical Reviews: Findings and Using Defects”. Wescon’97, Conference Proceedings, pp. 128-133.

[Parnas85] Parnas, D. L. and Weiss, D. M. (1985): “Active Design Reviews: Principles and Practices”, Proc. of the IEEE International Conference on Software Engineering, pp. 132-136.

[Pickard98] Pickard, L. M., Kitchenham, B. A. and Jones, P. W. (1998): “Combining Empirical Results in Software Engineering”. Information and Software Technology. Vol. 40, pp. 811-821.

[Porter95] Porter, A. A., Votta, L. and Basili, V. R (1995).: “Comparing Detection Methods for Software Requirements Inspection: A Replicated Experiment”, IEEE Transactions on Software Engineering, Vol 21, No 6, pp. 563-575.

[Porter97] Porter, A. A., Siy, H. P., Toman, C. A. and Votta L. G. (1997): “An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development”, IEEE Transactions on Software Engineering, Vol. 23, No. 6.

[Regnell00] 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, Vol. 5, No. 4, pp. 331-356.

[Robson93] Robson, C. (1993): “Real World Research”, Blackwell Publishers, UK.

[Runeson98] 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, Vol. 3, No. 4, pp. 381-406.

[Sauer00] Sauer, C., Jeffery, D. R., Land, L. and Yetton, P. (2000): “The Effectiveness of Software Development Technical Reviews: A Behaviourally Motivated Program of Research”. IEEE Transactions on Software Engineering, Vol. 26, No. 1, pp. 1-14.

[Smart98] Smart, K. L. and Thompson, M. (1998): “Changing the Way We Work: Fundamentals of Effective Teams’”. Proceedings of IEEE International Communication Conference, Vol. 2, pp. 383-390.

[Sole95] Sole, T. D. and Bist, G. (1995): “Benchmarking in Technical Information”. IEEE Transactions on Professional Communication, 38(2), pp. 77-82.

[Tan94] Tan, B. C. Y., Raman, K. S. and Wei, K. (1994): “An Empirical Study of the Task Dimension of Group Support System”. IEEE Transaction on Systems, Man and Cybernetics, 24, pp. 1054-1060.

[Valacich94] Valacich, J. B. and Dennis, A. R: (1994): “A Mathematical Model of Performance of Computer-Mediated Groups During Idea Generation”. Journal of Management Information Systems, 11(1), pp. 59-72.

[Votta93] Votta, L. G. Jr. (1993): “Does Every Inspection Need a Meeting?”. Proc. of 1st ACM SIGSOFT Symposium on Software Development Engineering’, ACM Press New York, N. Y., pp. 107-114.

[Weller93] Weller, E. F. (1993): “Lessons from Three Years of Inspection Data”. IEEE Software, 10(5), pp. 38-45.

[Wheeler96] Wheeler, D. A., Brykczynski, B. and Meeson, R. N. Jr. (1996): “Software Inspection: An Industry Best Practice”. IEEE Computer Society Press, USA. ISBN 0-8186-7340-0.

[Wohlin95] Wohlin, C., Runeson, P. and Brantestam, J. (1995): “An Experimental Evaluation of Capture-Recapture in Software Inspections”. Journal of Software Testing, Verification and Reliability, Vol. 5, No. 4, pp. 213-232.

[Wohlin00] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. and Wesslén, A. (2000): “Experimentation in Software Engineering—An Introduction”. Kluwer Academic Publishers, Boston, USA.