The Optimal Team Size for UML Design Inspections

Boodoo, S.; El-Emam, Khaled; Laitenberger, O.; Madhavji, N.

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

The Optimal Team Size for UML Design Inspections *

Boodoo, S., El-Emam, K., Laitenberger, O., Madhavji, N.
September 2000

* published as NRC/ERB-1081. September 2000. 29 pages. NRC 44149.

Canada

National Research Council Canada
Institute for Information Technology
Conseil national de recherches Canada
Institut de Technologie de l'information

ERB-1081

NRC·CNRC

The Optimal Team Size for UML Design Inspections

Shaheen Boodoo, Khaled El Emam,
Oliver Laintenberger, and Nazim Madhavji

September 2000

Canada

NRC 44149

The Optimal Team Size for UML Design Inspections

Shaheen Boodoo[1]
Khaled El Emam[2]
Oliver Laintenberger[3]
Nazim Madhavji[4]

Abstract

Recent evidence indicates that the UML (Unified Modeling Language) is the most preferred and widely used modeling technique for object-oriented analysis and design. With UML becoming so popular, there is a need to have good quality assurance techniques for projects using it. Our focus in this study is on the inspections of UML design documents. The basic premise of software inspections is that they detect and remove defects before they propagate to subsequent development phases, where their detection and correction costs escalate. However, the performance of inspections can vary considerably, making it important to optimize inspections. One approach for optimizing inspections is by controlling the inspection team size. This paper presents an empirical evaluation of the optimal team size for UML design inspections. Our results show that there is no single optimal team size. Optimal team size in fact depends on various conditions such as the cost of defect detection late in the process, and inspection meeting duration. This paper quantifies these factors and proposes optimal team sizes under various conditions. Our results also indicate strongly that contemporary suggestions of only two-person inspection teams are far from optimal.

1Introduction

Since Fagan's initial work presented in 1976 (Fagan, 1976), software inspections have emerged as one of the most effective and efficient methods for software quality improvement. It has been claimed that inspections can lead to the detection and correction of anywhere between 50% and 90% of the defects in a software artifact (Fagan, 1986; Gilb and Graham, 1993). Inspections not only contribute towards software quality improvement, but also can lead to budget and time benefits through early defect detection. Since inspections can be performed at the end of each development phase and since the defects are typically found close to the point where they are introduced, rework[5] costs can be reduced considerably. For example, a Monte Carlo simulation using parameters collected from the literature has indicated that on average, the implementation of code inspections reduces life cycle defect detection costs by 39% on average, and that the

[1] School of Computer Science McGill University, 3480 University Street, McConnell Engineering Building, Montreal, Quebec, Canada H3A 2A7. sboodo@cs.mcgill.ca

[2] National Research Council of Canada, Institute for Information Technology, Building M-50, Montreal Road, Ottawa, Ontario, Canada K1A OR6. Khaled.El-Emam@nrc.ca

[3] Fraunhofer Institute for Experimental Software Engineering Sauerwiesen 6 D-67661 Kaiserslautern Germany. oliver.laitenberger@iese.fhg.de

[4] School of Computer Science McGill University, 3480 University Street, McConnell Engineering Building, Montreal, Quebec, Canada H3A 2A7. madhavji@cs.mcgill.ca

[5] This constitutes the costs associated with correcting defects.

implementation of design inspections reduces life cycle defect detection costs by 44% on average (Briand et al., 1998b) [6].

Thus far, research on software inspections in general has focussed primarily on textual documents resulting from conventional structured development processes, such as functional requirements documents or functional code modules (Basili et al., 1996; Laitenberger et al., 2000b; Porter et al., 1995; Porter and Votta, 1998). The inspection of object-oriented models, particularly of the graphical form, have so far not been investigated thoroughly. This is worrisome given that over the past decade object-oriented development methods are gaining prominence in practice.

By far, the most popular object-oriented notation nowadays is UML (Unified Modeling Language). This is a graphical notation for modeling object-oriented systems. A recent survey eliciting 160 responses found that amongst experienced developers the OMT, Booch, and Jacobson methods are most familiar, most used and most preferred (Johnson and Hardgrave, 1999).[7] These three methods are the basis for the UML notation (Booch et al., 1999). With UML becoming so popular, there is a need to have good quality assurance techniques. Our focus in this paper is on inspections of UML design documents.

1.1 Inspections of UML Documents

A software inspection usually consists of several activities including planning, defect detection, defect collection, and defect correction (Laitenberger and DeBaud, 2000). Inspection planning is performed by an organizer who schedules all subsequent inspection activities. The defect detection activity can be performed either by inspectors individually or in a group meeting. Recent empirical findings reveal that the so-called synergy effect of inspection meetings is rather low in terms of impact on defects detected (Johnson and Tjahjono, 1998; Land et al., 1997; Votta, 1993). Therefore, we are looking at defect detection as an individual rather than a group activity. Defect collection, on the other hand, is often performed in a team meeting (i.e., an inspection meeting) led by an inspection moderator.

Although each of these activities is important for a successful inspection, the key part of an inspection is the defect detection activity. Throughout this activity, inspectors read UML documents and check whether they satisfy quality requirements, such as correctness, consistency, testability, or maintainability. Each deviation is considered a defect.

The software engineering community has identified a number of factors that have an impact on software inspections performance. For example, the particular reading technique used can influence the number of defects found (Basili et al., 1996; Laitenberger et al., 2000b; Porter et al., 1995; Porter and Votta, 1998), the process of inspections (for example, defect detection meetings versus no meetings and team size) (Gilb and Graham, 1993; Johnson and Tjahjono, 1998; Madachy et al., 1993; Porter et al., 1997), and explicit criteria for deciding when to stop inspections (Adams, 1999). Our focus in this paper is on inspection team size.

1.2 Optimal UML Team Size

Our goal in this paper is to find an optimal team size, if one exists. It is clear that the larger the inspection team size, the more costly an inspection will be. However, it is also the case that the more inspectors who read a given artifact, the more likely it is that more defects will be detected. Therefore, there is a tradeoff between cost and effectiveness. It then becomes important to identify the minimal team size that will maximize the number of faults found. Votta (Votta, 1993)

[6] These numbers are in comparison with the testing only defect detection life cycle. So, for example, if you introduce design inspections to a testing only life cycle, you will save 44% of the testing cost.

[7] The percentage of developers who used these methods were as follows : OMT(Analysis: 36%, Design: 39%), Booch (Analysis: 23 %, Design: 30%), Jacobson (Analysis: 8%, Design: 2%), and all the others (Analysis : 30%, Design: 25%).

notes that determining the optimal team size for inspections is an important contemporary research issue.

Previous work in this area, for non-object-oriented inspections, produced inconsistent results with recommended team sizes ranging from 2 to 12 (Bisant and Lyle, 1989; Bottger and Yetton, 1988; Buck, 1981; Eick et al., 1992; Fagan, 1976; Gilb and Graham, 1993; Grady, 1992; Lau et al., 1996; Madachy et al., 1993; Porter et al., 1996; Porter et al., 1997; Strauss and Ebeneau, 1994; Weller, 1993). Furthermore, none of these studies focused on UML design inspections.

We performed a simulation using data collected from an empirical study with 16 professional designers whereby they inspected a design document in the UML notation. All used the same reading technique and followed the same process. Using the data collected from this study, we then performed an extensive simulation of 'virtual teams[8]' of sizes ranging from 2 inspectors to 15 inspectors. We simulated inspections without team meetings, and with team meetings ranging in duration from 1 hour to 2 hours (the range generally recommended in the literature). Our criterion measure was the efficiency of the inspections. This is defined as the percentage savings in defect detection costs from performing the inspections. The advantage of this measure is that it takes into account the costs of inspections, and the savings that accrue from finding defects earlier (during design) as opposed to during code inspections and testing. We also factored in the the effort to fix a defect after the inspection.

Our results indicate that there is no single optimal team size. In fact, the optimal team size depends first on the relative costs of finding (design) defects later in the life cycle. Therefore, say, if the cost of testing is prohibitive, then a very large team size is likely to be optimal as it is worth expending the extra effort to find that additional defect. On the other hand, if the cost of testing is minor, then a smaller optimal team size is recommended. The optimal team size also depends on whether a team meeting is performed, and the duration of that meeting. If meetings are held and the longer the duration of the meetings, then the smaller the optimal team size. This is because meetings increase the overall inspection effort.

At an abstract level, these results confirm what one would expect intuitively. However, we have also quantified these effects and make initial recommendations for optimal team size under the different conditions. Furthermore, we found that the sometimes suggested inspection team size of two (Bisant and Lyle, 1989; Porter et al., 1997) is far from optimal in the case of UML inspections. In fact, even under modest assumptions about the relative cost of design inspections and post-design defect detection activities, the optimal team size is considerably large.

The paper is structured as follows. In the following section we review the literature on inspection team size. We also describe why only considering the number (or proportion) of defects found when evaluating inspection team size can give misleading results. Section 3 presents our research method in detail, and our results and a their interpretation are given in Section 3.3.6. We conclude the paper in Section 5 with a summary and directions for future work.

2 Background

This section presents a review of the literature on inspection team size, and discusses the problems with one obvious approach for empirically evaluating optimal inspection team. The identified problems motivate the approach that we followed in our study.

2.1 Current Recommendations on Optimal Team Size

There exists a plethora of recommendations about the optimal team size for software inspections. Some of these recommendations are based on systematic empirical study, while others are

[8] As noted in the text, we consider that defect detection occurs in the preperation phase. A "virtual team" is created by considering all combinations of inspectors of a certain size. For example, there are 120 combinations of 2-inspector teams from a pool of 16 potential inspectors ($^{16}C_2$).

based on personal experiences of their authors. This literature, however, gives conflicting accounts of what is indeed the optimal team size.

Bisant & Lyle (Bisant and Lyle, 1989) found an improvement in individual productivity as a result of 2-person inspections. They also mentioned that this 2-person method can be applied to those environments where larger team resources are not available. Porter et al. (Porter et al., 1997) reported that inspections are usually carried out by a team of 4 to 6 inspectors. They also conducted an experiment and they found out that a decrease in the number of inspectors from 4 to 2 would result in a decrease in effort without any reduction in effectiveness or interval, but for 1-inspector teams the effectiveness was poorer. Bottger et al. (Bottger and Yetton, 1988) stated that behavioral research found expert pairs perform as well as larger groups.

Lau et al. (Lau et al., 1996) suggested three person inspections. Grady (Grady, 1992) stated an optimum size of 4 to 5 inspectors. Laitenberger et al. (Laitenberger and DeBaud, 2000) recommended 3 to 4 inspectors. They stated that there would be a ceiling effect after which an additional inspector would not necessarily pay off in more defects. Fagan (Fagan, 1976) recommended to keep inspection teams small, that is, 4 people. However, if there are a number of (code) interfaces, then the programmers of the code related to those interfaces can be involved in the inspection. In a study at IBM, Buck (Buck, 1981) indicated that there was no difference in effectiveness between 3, 4, or 5-person inspection teams. Weller (Weller, 1993) reported that 4-person teams were twice as effective[9], and more than twice as efficient[10] as 3-person teams. However, a 3-person team with a low preparation rate (lines per hour) seemed to do as well as a 4-person inspection team with a high preparation rate. He suggested that the preparation rate and inspectors' familiarity with the product, not the team size, determined inspection effectiveness and efficiency. Yetton et al. (Yetton and Bottger, 1983) noted that behavioral theory showed inspection performance for team sizes of three to four. Beyond four members, there is no performance improvements. Madachy et al. (Madachy et al., 1993) suggested an optimal size of 4 to 5 people for inspections. Gilb et al. (Gilb and Graham, 1993) mentioned a team size of 2 to 3 people for maximum efficiency[11], and 4 to 5 people for maximum effectiveness[12]. Strauss et al. (Strauss and Ebeneau, 1994) suggested a minimum inspection team size of 3, and a maximum of 7. They also mentioned that any more persons would tend to reduce the efficiency and effectiveness of the process.

In an experiment at the Jet Propulsion Laboratory, Kelly et al. (Kelly et al., 1992) stated that inspections are usually carried out by 6 people. They also recommended larger teams for requirements and high-level artifacts and smaller teams for code. The recommended size for software inspections defined by IEEE STD 1028-1988 is 3-6 persons (STD, 1989). Ackerman et al. (Ackerman et al., 1989) reported that inspections are conducted by at least 3 people, one of whom is the moderator who is responsible for the effectiveness of the examination. Briand et al. (Briand et al., 1999a) performed a study of meetings[13] in software development. They found that when the number of participants exceeds 7, the perceived quality of the meeting was low. Johnson (Johnson, 1998) notes that there is widespread consensus that the inspection team size should never exceed 6-9 members. Industrial practice varies even within a single enterprise, for example, sizes between 4 and 12 are used at AT&T (Eick et al., 1992). Sauer et al. (Sauer et al., 2000) stated that behavioral theory has found that the relationship between defect detection performance and review group size is a function of expertise.

Schneider et.al (Schneider et al., 1992) stated that N inspection teams are more effective than any single team. An experiment was carried out by 27 graduate students grouped into 9 teams of 3. They did individual reviewing and the defects were collected for each team through a meeting. Then, all the inspection results for all the teams were collated by a single moderator who also

---

[9] Effectiveness is defined as the defects found by inspection /total number of defects.
[10] It is not clear how Weller defined effciency.
[11] Efficiency is the major issues per work-hour.
[12] Effectiveness is the percentage of total majors found in inspections.
[13] This study covered informal technical discussions and project planning meetings, as well as meetings to inspect work packages.

removed duplicate reports. It was found that the 9 teams of 3 persons combined found twice as many defects as the average found by any single team (78% compared to 35%).

One reason for such a diversity in the recommendations is that different authors define "optimal" in a different way. Differing definitions of "efficiency" and "effectiveness" of inspections are used to highlight the benefits of the various team sizes. For example, Porter et al. (Porter et al., 1997) defined effectiveness as defect density, which is defects/KNCSL where NCSL is the number of noncommentary lines of code. Madachy et al. (Madachy et al., 1993) described effectiveness as defects found per unit of inspection effort. According to Gilb et al. (Gilb and Graham, 1993), maximum effectiveness is the percentage of total major faults or issues found in inspections. He defined efficiency as major issues per work-hour. Fagan (Fagan, 1976) and Buck (Buck, 1981) defined efficiency as (errors found by an inspection / total number of errors before inspection) x 100.

Another reason for such differences is that some authors like Buck (Buck, 1981) and Fagan (Fagan, 1976) have used meetings for defect detection, whereas others consider the meeting as a means for defect collection only. Furthermore, some studies evaluate the team size for the meeting only rather than for the whole inspection, such as (Bottger and Yetton, 1988; Briand et al., 1999a; Yetton and Bottger, 1983).

Such a state of affairs is not conducive to making strong recommendations about the optimal team size in practice. Furthermore, none of the above articles tackle optimal team size for the inspection of UML designs.

2.2 Inspection Effectiveness and Team Size

An obvious way to identify the optimal team size is to perform an empirical study comparing teams of different sizes. This comparison can then determine which team size results in more defects detected, greater defect density (for defects found), or a greater proportion of defects found (out of the total seeded in the inspected artifact).

Below we demonstrate that all of the above measures of "inspection quality" are uninformative as a means of evaluating team size. By definition, the expected number of defects found during an inspection (Chao et al., 1992) is given by:

$$E(D) = N\left(1 - \prod_{i=1}^{t}(1 - p_i)\right) \qquad \textbf{Eqn. 1}$$

where N is the number of defects in the artifact, p_i is the probability that inspector i will find a defect, and t is the total number of inspectors on the team. This formulation assumes independence among inspectors. The p_i probability denotes the long-run probability that an inspector will find a defect, and its expected value is the average effectiveness of an inspector.

It is clear from the above equation that for any $p_i > 0$ the $E(D)$ value will increase as you add inspectors, and $\lim_{t\to\infty} E(D) = N$ (the expected number of defects found will equal the actual number of defects in the artifact as the team size approaches infinity). This means that by definition, irrespective of the additional inspectors' capability at finding defects (as long as their probability of defect detection is greater than zero), the addition of inspectors will result in a greater *expected* number of defects. *More inspectors means more defects will be found, on average*.[14] Furthermore, the above equation makes clear that the rate at which the $E(D)$ value increases will decrease as more inspectors are added.

[14] It should be noted that this is a statistical argument. For instance, if a a team of two inspectors finds all the defects in an artifact, then the addition of more inspectors will ot increase the number of defects found. However, over many inspections, keeping the total number of defects constant, a larger team will find more defects on average.

If we plot the values from our data set, as in Figure 1, this pattern can be seen.[15] In addition, if we consider the optimal team size as that which maximizes the (expected) number of defects found, there is no optimal team size. The maximum is reached with a team of size plus infinity.

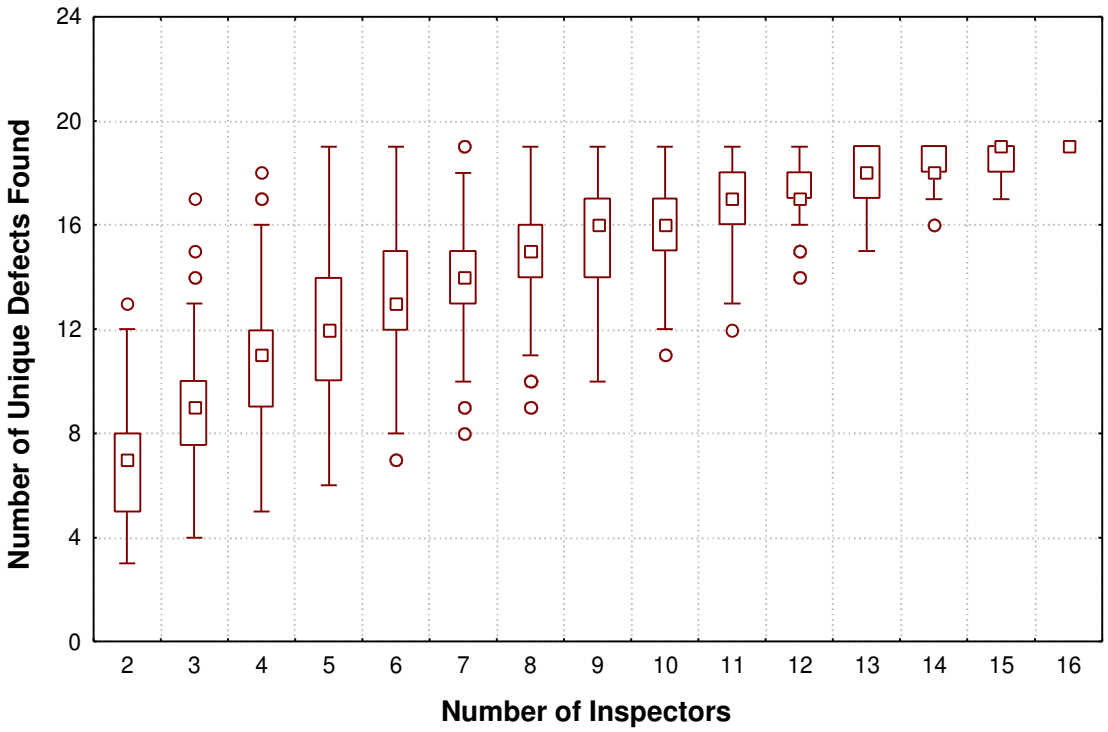

Figure 1: Box and whisker plot of unique number of defect against team size from our study. An explanation of box and whisker plots is given in Appendix C.

If one were to conduct an experimental study comparing different team sizes on their effectiveness or number of defects found, then the result will depend on the statistical power[16] of the test used. One typically draws the conclusion that a given team size results in higher effectiveness if the difference is statistically significant. We have demonstrated above that for any two different team sizes, there will *always be a difference*. Therefore, an experimental study will find a statistically significant difference if the statistical power is large enough.

To illustrate the point, we performed a Monte Carlo simulation for an experiment comparing team sizes. We used a two group design, with one group having a team size of two, and the other group having a team size of four. We assumed that all the inspectors taking part in the experiment come from the same population. The population of effectiveness is defined as a Beta distribution. The Beta distribution's parameters were obtained from the observed effectiveness in our data set (this data set is described later in this paper). These parameters were Maximum Likelihood estimates. The Beta distribution specified the probability of an inspector finding a defect, i.e., p_i. An example of a Beta distribution is shown in Figure 2.

[15] As will be explained later, inspection teams of the various sizes were generated by simulating all combinations of inspection teams of that size.

[16] The statistical power of a test is the probability that the test will correctly lead to the rejection of the null hypothesis.

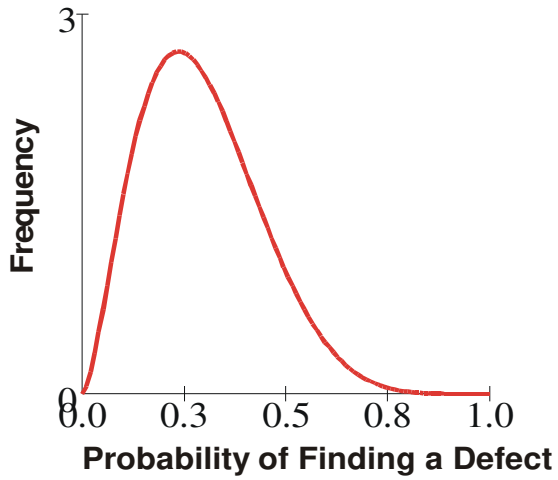

Figure 2: A Beta distribution.

Each group in the simulation had k teams of the requisite (two or four) number of inspectors. Inspectors were selected from the Beta distribution randomly, and randomly assigned to the two groups. Within each group, each inspector was randomly assigned to one of the k teams. This is a simulation of an actual experiment that could be performed. For each run of the experiment we used an independent sample t-test[17] to test the difference in effectiveness between the two groups. We chose sizes of k (that is, number of teams) to be 5, 10, and 15.

We repeated the simulated experiment 1000 times, and determined the proportion of times the two-tailed t-test rejected the null hypothesis of there being no difference (in means) between the two groups. We know that there is a difference between the two groups because that is true by definition. Therefore, the proportion of times that the null hypothesis is rejected gives us an estimate of the power of the test. The results are summarized in Table 1.

No. of Teams in a group (k)	Empirical Power	Number of Subjects
5	58%	(5x4)+(5x2)=30
10	96%	(10x4)+(10x2)=60
15	99%	(15x4)+(15x2)=90

Table 1: Results of the power analysis.

We can see that if we have five teams in each group, the power of the test is only 58%. This can be considered rather low. Cohen (Cohen, 1988) recommends a statistical power of 80%. Thus, if an experiment with 5 teams in each group was to be performed with professional subjects similar to ours the likelihood of rejecting the null hypothesis that there is no difference in effectiveness between a two and a four person inspection team is low. This would then likely lead to the conclusion that there is no difference between 2-person and 4-person teams. However, a larger experiment with 10 or 15 teams in each group has a very high probability of rejecting the null hypothesis. A larger experiment would therefore conclude that indeed there is a difference in effectiveness between 2-person and 4-person teams.

[17] A t-test is a test to compare the means of two independent groups.

It can be seen from the table that an experiment with 5 teams already requires 30 subjects in total. An experiment with 10 or 15 teams would require 60 and 90 subjects respectively. To be able to perform such a large (by software engineering standards) experiment with professional subjects is quite unlikely in practice. Indeed, if we were to use another measure, such as the detected defect density of defects, and assuming that the two groups were inspecting the same artifact, then the conclusion of our simulation would be exactly the same (see the proof of this in the appendix).

The above exposition continued that there is no optimal team size when one considers effectiveness as the evaluative criterion. Furthermore, experimental studies that investigate this issue will draw a conclusion based directly on the size of the experiment, with modest changes in the number of teams resulting in a high probability of drawing opposite conclusions.

To alleviate this, we propose to use the *efficiency* of the inspection as the basis for deciding the optimal team size. Efficiency takes into account the costs of the inspection compared to an alternative defect detection technique. As shown in Figure 1, while the addition of inspectors increases effectiveness, the marginal increase in effectiveness decreases. However, each additional inspector adds to the cost of the inspection. There comes a point where the additional costs of inspectors outweigh the benefits of detecting more defects. These defects could be more cost-effectively detected using another defect detection technique.

3 Research Method

This section explains how the study was carried out, and the analysis of the data collected during the study.

3.1 Design of Study

3.1.1 Context of Study

The study was performed in the context of a course on object-oriented development. The main objective of this course was to teach participants the principles of object-oriented analysis, design, and development of information systems. The course consisted of several modules, each of which addressed a specific topic in object-oriented development, e.g., object-oriented analysis, object-oriented design, UML, software inspections, testing, etc. Each module included practical exercises in the form of case studies to familiarize the participants with the concepts being taught.

Our data was collected during the module on software inspections. This module was scheduled towards the end of the course. This ensured that the participants already had theoretical and practical training in the analysis and design of information systems as well as in the UML. As a result it is assumed that the subjects were familiar with the analysis and design phases as well as the notation used in the study materials.

The subjects in this study were 16 practitioners with various backgrounds. Prior to taking the course they primarily worked as programmers in industry and had various levels of experience in object-oriented programming. The subjects' experience was captured with a questionnaire before the study. The subjects' experience was considered in UML, design, programming, and inspections as the most relevant types of experience that would have an impact on defect detection capabilities. It was found that most of the subjects had already written UML documents (median of 3 on a 5 point scale). Moreover, they had some experience in developing design documents and between 1 and 16 years of programming experience with a median of 2 years. None of them had yet participated in an inspection.

3.1.2 Materials Used in the Study

In order to use a set of diagrams to develop a system, it is necessary to follow a process. The development process defines the sequence of steps for building the various UML models. Moreover, a process relates the different models and diagrams to each other. This means that a development process must make it clear which documents (e.g., models, diagrams) contain relevant information about which software entities.

Of the leading object-oriented processes in widespread use, the one which best meets this criterion is the Fusion process (Coleman et al., 1994)[18]. Fusion is very precise about which specific models should be created as part of an object-oriented development project, and what information these models should contain. In contrast, other object-oriented processes typically give little prescriptive advice about what exactly to create during a project, and which activities to perform. As a consequence, when inspecting an entity, it is not easy to ensure that all the relevant information (i.e., models) describing properties of the entity have been found and checked. For these reasons, Fusion was used as the basis for developing the study materials.

Although Fusion uses diagram types that are quite similar to those of the UML (in fact, some of the UML models are derived from Fusion), these do conform precisely to the UML standard. Therefore, in the experiment a hybrid version of the Fusion method was used, known as FuML (Atkinson, 1998), which specifically adapts Fusion to exploit the UML. A comparison of FuML and UML was provided recently in (Laitenberger et al., 2000a), as well as an overview of the FuML models used in our study.

For our study, the FuML models were developed for one system. This was a point of sales system (see Section 2.4.1). The subjects were provided with the system's analysis and design documentation. The size of the point of sales system was 18 pages. It included 6 collaboration diagrams and 3 design class diagrams.

A set of defects were introduced into the design document prior to the study. Nineteen defects were inserted in the point of sales system. The defects primarily related to correctness, consistency, and completeness of the design models. Some of the defects were made while developing the artifact. However, some more defects were introduced in the design documents to have a larger set of defects. Based on our collective experience, these defects were perceived to be realistic for UML designs.

To ensure the feasibility of the required reading activities and the possibility for subjects to scrutinize the design document for defects in a 2-4 hour time frame, a trial run of the artifact inspection was performed. As part of the trial run, the artifact was inspected by the authors to scrutinize the documents for defects. After the trial run, the materials were improved based on their own experiences in using them.

3.1.3 Execution

An intensive exercise introducing the principles of software inspections and the Fusion development method was carried out during the inspection module. This included a brief explanation of the various UML models the subjects were to inspect, as well as the technique that they would use for reading the document during preparation. This is a checklist-based reading technique where the inspectors are given a list of Yes/No questions that they have to answer while scrutinizing the document. The checklist can be found in (Laitenberger et al., 2000a). Then, the subjects used the checklist for individual defect detection in the design document. While inspecting the design document, the subjects were asked to log all detected defects on a defect report form. After the reading exercise the subjects were also asked to fill out a debriefing questionnaire that collected information about their experience.

[18] The most popular object-oriented development process is probably the Rational Unified Process (RUP) (Jacobson et al., 1998). However, the Fusion process is more precise than RUP when it comes to defining when to create the various models and which set of models to create as part of the development effort.

3.2 Measurement of Efficiency

We defined efficiency as the cost savings from finding defects early. This is a similar concept to Return on Investment (ROI), except that that efficiecny deals with some inconsistent behaviors in typical ROI calculations. The complete derivation of our efficiency model is given in the appendix. For our purposes efficieny of any particular UML design inspection is defined as:

$$Efficiency = \hat{p}_{Design\ Inspection} \times \left(1 - \frac{\hat{\varepsilon}_{Design\ Inspection}}{\hat{\varepsilon}_{Post\ Design}}\right) \quad \textbf{Eqn. 2}$$

where $\hat{p}_{Design\ Inspection}$ is the proportion of defects in the document that were found during the design inspection, $\hat{\varepsilon}_{Design\ Inspection}$ is the detection effort per defect, and $\hat{\varepsilon}_{Post\ Design}$ is the effort to detect and correct defects after the design phase. The interpretation of this model is quite intuitive. It can be interpreted as the proportion of defect detection costs that are saved due to the performance of the design inspection. For example, if the efficiency is 0.54, then this means that 54% of the post-design defect detection costs have been saved by performing the design inspection.

The first two terms can be calculated easily from the data collected during our study. The third parameter, $\hat{\varepsilon}_{Post\ Design}$, is unknown, and therefore we vary it during our simulations. In this equation we consider only the detection effort for design inspections. Below we expand on this by adding the effort to perform a meeting, and the effort to fix a defect.

3.3 Simulation Parameters

3.3.1 Creating "Virtual Teams"

Since only a fixed number of inspectors were available, different team sizes were formed by creating "virtual teams". Virtual teams are combinations of individual inspectors that we create. For example, for a team size of 2 inspectors, 120 possible combinations are formed ($^{16}C_2$). Using 'virtual teams' for evaluating software engineering technologies (or nominal teams as they are sometimes called) is an approach that has been used in a number of recent studies (Basili et al., 1996; Briand et al., 2000; Miller, 1999).

3.3.2 Measurement

We measured two things: the effort for each individual during preparation in minutes, and the number of defects found by each individual. We also logged each defect found by each individual. This allows us to compute the number of unique defects found by a virtual team.

The subjects sometimes reported more defects on their defect report forms than were seeded in the design documents. When a true defect was reported that was not on the list of seeded defects, this defect was added to the list of known defects and then we reanalyzed the defect report forms of all the remaining subjects.

For virtual teams, the team defect detection effectiveness was computed as:

$$\text{Virtual team defect detection effectiveness} = \frac{\text{Unique no. of defect found by the team}}{\text{Total no. of defects in the design document}}$$

The effort per defect, $\hat{\varepsilon}_{Design\ Inspection}$, for each virtual inspection team was calculated by taking the total effort / unique number of defects found. For example, consider a virtual inspection with inspector I_1 and I_2 and having the data as in Table 2.

Inspector	Effort (mins)	Total Defects found	Unique No. of Defects
I_1	20	5	10
I_2	40	8	

Table 2: Example of data to calculate efficiency.

$\hat{p}_{Design\ Inspection}$ = 10/19

$\hat{\varepsilon}_{Design\ Inspection}$ = (20+40)/10 = 6

3.3.3 Varying the Post-Design Defect Detection Cost

Since we do not know the cost of post-design defect detection activities, $\hat{\varepsilon}_{Post\ Design}$, we simulated this value. Below we explain how this was done.

We specified the effort per defect for post-design defect detection activities as:

$$\hat{\varepsilon}_{Post\ Design} = \bar{\varepsilon}_{Individual} \times \phi \qquad \textbf{Eqn. 3}$$

Where $\bar{\varepsilon}_{Individual}$ is the average effort per defect for an individual inspector to find a defect (i.e., we took the average across our 16 subjects). Therefore, ϕ *specifies how much larger the cost of post-design defect detection and correction was compared to the average cost for an individual inspector to find a defect*. We varied ϕ from 1 to 20[19]. A value of 1 indicates that the cost of finding a defect during post-design defect detection is equal to the cost of an individual inspector to find a defect. This is of course not likely to be the case in practice, but allows us investigate the behavior of optimal team size in general as ϕ increases.

3.3.4 Varying the Meeting Duration

Two main purposes for conducting a meeting during inspections as described by Fagan (Fagan, 1976) are defect detection and defect collection. In Fagan's inspection, the preparation is carried out for comprehension of the artifact, and defect discovery takes place during the meeting. Fagan found inspection meetings to be productive for defect discovery. For some authors, inspection meetings are essential for the effectiveness of formal technical review as they create "synergy" among the team that can lead to the discovery of defects not found through individual preparation (Ackerman et al., 1989). However, Parnas and Weiss (Parnas and Weiss, 1987) mentioned that a meeting was unnecessary for defect collection which is the stage where the defects during the inspections are combined together.

Debate continues over whether group meetings are effective or not. In design inspections performed on a large project, Votta (Votta, 1993) found that meetings increased the time interval by approximately 30%. He was unable to show the presence of synergy since the number of issues found during preparation were cancelled due to process loss[20]. Eick et al. (Eick et al., 1992) found that 90% of the defects were detected during the preparation phase while 10% were detected during the meeting. Eick et al.'s results are in contradiction with Fagan's claim that most

[19] ϕ was chosen to be within the range 1 to 20 since as can be seen from the results in Table 2, when $\phi \geq 13$, the optimal team size stays at 15.

[20] Process loss is the number of defects found during individual preparation that are lost during meetings (for example, they are not logged or overlooked).

(if not all) errors are found during the inspection meeting. The conflict is due to the differences in the goals and techniques for preparation and meeting.

An experiment was carried out by Johnson and Tjahjono (Johnson and Tjahjono, 1998) with undergraduate students where they assessed whether a real group[21] for defect detection outperformed a nominal group[22] for defect collection. Both methods used paraphrasing, where the reader attempts to explain and interpret the artifact, for defect detection. The results showed that the real group method was more costly than the nominal group in terms of total effort and effort per defect. However, less false positives were generated from the real group. In this experiment they were unable to show that inspection meetings for the purpose of discovery outperformed individuals working independently. Sauer et al. (Sauer et al., 2000) stated that behavioral theory supported the belief that group meetings did not achieve synergies, that is, group meetings did not discover significant number of new defects beyond those already found by the nominal groups. Rather, the group performance is dominated by task expertise.

Nevertheless, there are benefits attributed to meetings. For example, Johnson and Tjahjono (Johnson and Tjahjono, 1998) said that meetings helped the participants to improve their inspection skills, and increased their confidence in the outcome of the inspections. They also added that meetings could be applied in an organization that introduced inspections. After the participants became familiar with the inspection, then the organization could move to a meetingless approach to reduce costs without decreasing the detection effectiveness. Education and clarification are other reasons for holding a meeting (Gilb and Graham, 1993; Strauss and Ebeneau, 1994). Sauer et al. (Sauer et al., 2000) said that the advantage of a meeting was to discriminate true defects from false positives. Eick et al. (Eick et al., 1992) mentioned that inspectors tend to prepare more thoroughly with meetings than without meetings.

We therefore account for meetings in our simulation. However, an important question is what is the meeting duration ?

Gilb et.al (Gilb and Graham, 1993) proposed the duration of a meeting to last not more than 2 hours. Doolan (Doolan, 1992) also suggested not more than 2 hours meeting. Grady et al. (Grady, 1992) stated the logging meeting to be $1^1/_2$ hours. Barnard et al. (Barnard and Price, 1994) mentioned not more than 2 hrs of inspection meeting. Strauss et al. (Strauss and Ebeneau, 1994) recommended a meeting not to exceed 2 to 3 hours. Fagan (Fagan, 1976) said that inspection sessions should generally not exceed two hours. We therefore simulated inspections without meetings, and with meetings varying from 1 hour, 1.5 hours and 2 hours.

To account for meetings, we modify our equation of efficiency as follows:

$$Efficiency_{meeting} = \hat{p}_{Design\ Inspection} \times \left(1 - \frac{\left(\hat{\varepsilon}_{Design\ Inspection} + \eta\right)}{\bar{\varepsilon}_{Individual} \times \phi}\right) \qquad \textbf{Eqn. 4}$$

where η is the total meeting effort / unique number of defects detected.

As can be seen from the above equation, the addition of meeting duration to the inspections will increase the inspection effort. Hence, the efficiency for the meeting will decrease for a fixed cost of post-design defect detection activities (i.e., fixed ϕ).

Meeting effort is defined as q x t where q is the duration in minutes and t is the number of inspectors.

---

[21] A real group is one in which members meet and interact with each other to detect defects.

[22] A nominal group is one in which the members work individually without any interaction, and their individual results are aggregated as the group's result.

3.3.5 Varying the Effort to Fix Inspection Defects

In our simulation we also accounted for the effort for fixing a defect, $\hat{\varepsilon}_{FixingDefect}$, to the inspection effort. Since we do not know the effort for fixing a defects, this effort was defined as:

$$\hat{\varepsilon}_{FixingDefect} = c \times \bar{\varepsilon}_{Individual} \quad \textbf{Eqn. 5}$$

where c is a constant between 0 and 1. In other words, the effort to fix a defect is a multiple of the effort for an individual to find a defect. We took the worst case where c is 1. This is the worst case because typically detecting a defect is the time consuming activity, and during design inspections the cause of the problem is identified during detection, so the correction effort should be relatively small. In this way, the optimal team size will lie between the optimal team sizes without fixing effort and with fixing effort. Therefore, the equation of efficiency can be expressed as:

$$Efficiency = \hat{p}_{Design\ Inspection} \times \left(1 - \frac{\hat{\varepsilon}_{Design\ Inspection} + \eta}{\bar{\varepsilon}_{Individual} \times \phi} - \frac{c}{\phi}\right) \quad \textbf{Eqn. 6}$$

3.3.6 Summary

In summary, we vary the following parameters during our simulation:

- Insection team size (t): 2-15 (in increments of 1)
- Cost ratio (ϕ): 1-20 (in increments of 1)
- Meeting duration (q): 0,1,1.5 and 2 hours
- Relative cost of fixing defects during inspections (c): 0 and 1

By taking all possible combinations of the above parameters, our simulation consisted of 2240 different study points. For each study point, all possible combinations ${}^{16}C_t$ of inspection teams of size t were evaluated. By considering all the study points, we expect to obtain a broad perspective on the impact of team size on efficiency for UML inspections.

3.4 Data Analysis

For each of the cases above where we simulated 'virtual teams' we computed the efficiency. We took the median value of efficiency as the value for that particular combinaton of simulation parameters. We then plotted the median effcieny values for numbers of inspectors varying from 2 to 15. Because of the tradeoff between effort and effectiveness, each efficieny curve has a maximum. The team size at the maximum is then taken as the optimal team size for that particular combination of simulation parameters.

Just to illustrate, Figure 3 shows an example from our results. This graph shows box-and-whisker plots. Here we have the efficiency values as the inspection team size (t) was varied for $\phi = 5$, no meetings ($q = 0$), and no accounting for fixing effort ($c = 0$). The median efficiency values clearly reach a maximum at a team size of 8.

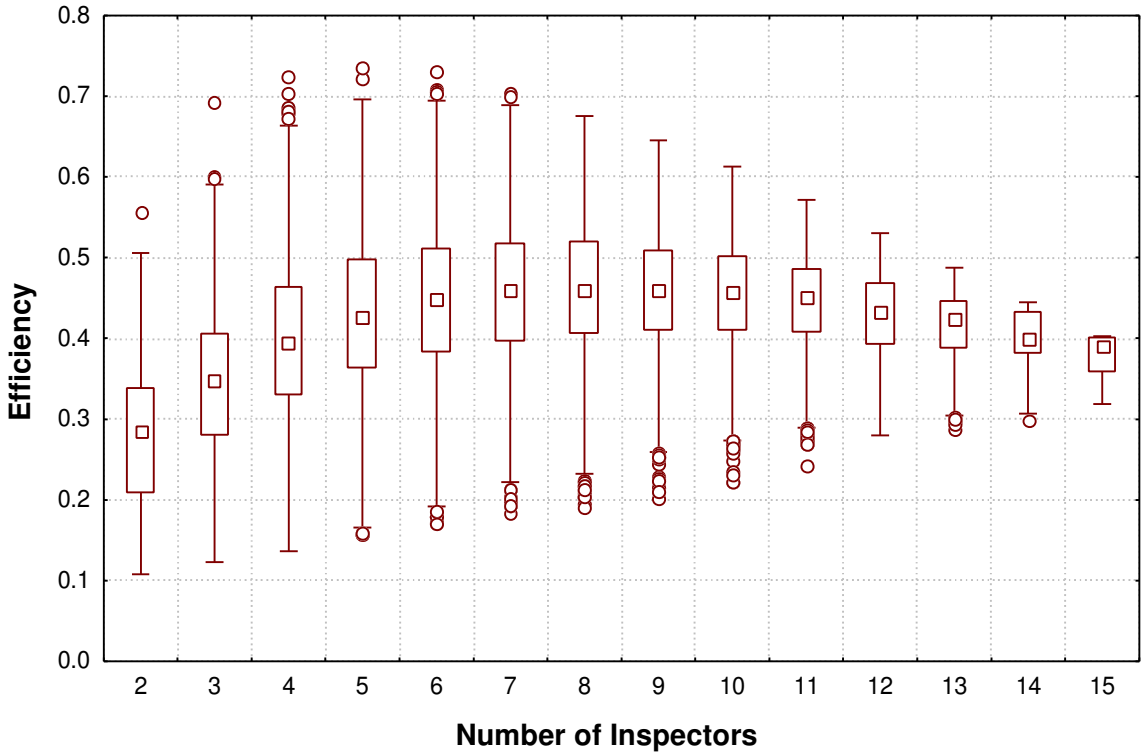

Figure 3: Efficiency plot for ϕ=5 when there are no meetings performed and no fixing effort.

4 Results

4.1 Optimal Team Size

Table 3 shows the results from all of the efficiency curves with meeting/no meeting, and with fixing effort/no fixing effort. It indicates that as the cost of post-design defect detection activities increases, more inspectors are needed to optimize efficiency. As the meeting duration increases, the optimal team sizes tend to decrease. The same trend is observed with fixing effort.

Figure 4 shows the upper bounds and lower bounds for the optimal team sizes for the different types of meetings. Note that in these plots the curve between ϕ and efficiency was smoothed to make the relationship clearer.[23]

[23] The smoothing method we used is locally weighted regression (Cleveland and Devlin, 1988) with Friedman's super smoother (Friedman, 1984).

φ	Optimal Team Size							
	Without Fixing Effort				With Fixing Effort			
	No meeting	**1 hr**	**1.5 hr**	**2 hrs**	**No Meeting**	**1 hr**	**1.5 hr**	**2 hrs**
1	2	2	2	2	2	2	2	2
2	3	2	2	2	3	2	2	2
3	5	4	3	3	3	2	2	2
4	6	5	5	5	5	4	3	3
5	8	7	6	5	6	5	4	5
6	10	9	8	7	8	7	6	5
7	11	9	9	9	11	9	8	7
8	13	9	9	9	11	9	9	9
9	13	11	9	9	13	9	9	9
10	15	15	15	9	15	11	9	9
11	15	15	15	15	15	15	11	9
12	15	15	15	15	15	15	15	11
13	15	15	15	15	15	15	15	15
14	15	15	15	15	15	15	15	15
15	15	15	15	15	15	15	15	15
16	15	15	15	15	15	15	15	15
17	15	15	15	15	15	15	15	15
18	15	15	15	15	15	15	15	15
19	15	15	15	15	15	15	15	15
20	15	15	15	15	15	15	15	15

Table 3 : Optimal Team Size for both Meeting/No Meeting and Fixing Effort/No Fixing Effort.

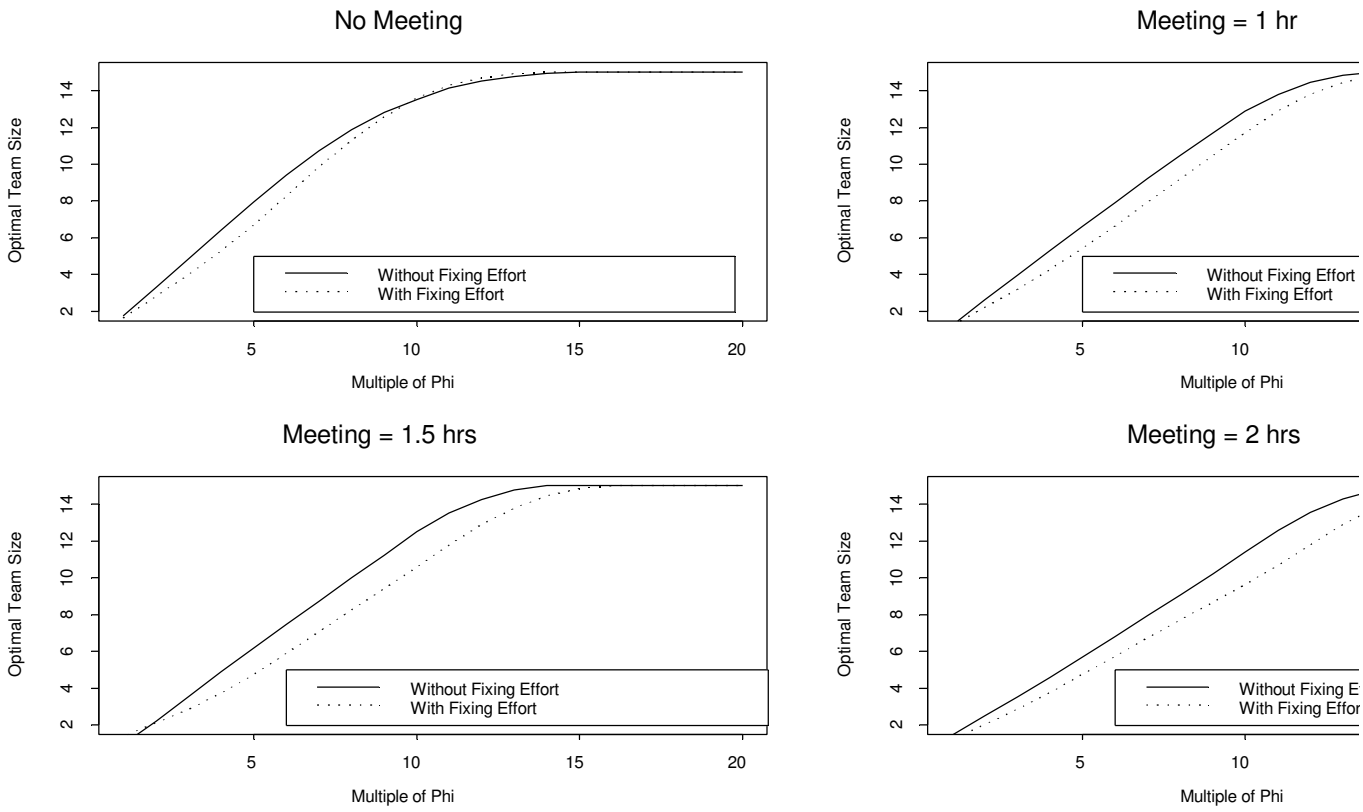

Figure 4: Optimal Team Size Curve for No Meeting and Meetings.

4.2 Discussion

We have evaluated the efficiency of inspections to determine the optimal team size. Our results represent an initial set of curves and tables that provide recommendation on the best team size. From the above results, we conclude that *there is no single optimal team size*. Broad claims about a single optimal team size or a recommended team size for software inspections do not receive support.

The optimal team size in fact depends on the relative cost of post-design defect detection activities. For a low cost of post-design defect detection activities, a small team size is needed. As this cost increases, a higher team size is required. However, if we look at Table 3, a team size of 2 is only optimal under the rather unrealistic conditions that $\phi<4$. Therefore, this casts doubt, at least for UML, on earlier prescriptions that only two inspectors are sufficient (Bisant and Lyle, 1989; Porter et al., 1996; Porter et al., 1997). Adding meeting effort would decrease the efficiency since the total effort of the inspections increases. For a fixed cost of post-design defect detection activities, it is more efficient to have a smaller team if meetings are held and the longer the team meeting. If a company knows the ratio of the cost of post-design defect detection activities to inspection, i.e., ϕ, reference can be made to our results to find the optimal team size.

However, some information cannot be easily obtained by measurements, observations, or experimentation. In this case, expert judgement is a solution to the problem. Expert judgement has been used in software engineering for cost estimation purposes (Briand et al., 1998a; Host and Wohlin, 1997, 1998). In a study at Siemens in Germany (Briand et al., 1999b), the authors tried to assess the efficiency of all inspections taking place in the organization using actual data and subjective estimates. For example, the effort during testing was not available. Since the experts often experience this effort, they were asked to estimate this parameter. Hence, ϕ can be determined. Similarly, if the total number of defects were collected, the experts were asked for the percentage breakdown of the defects origin in each activity in the life cycle, and hence the cost effectiveness can be calculated for each activity.

In retrospect it seems obvious that the patterns of efficiency increase/decrease as team size and ϕ change. However, as we saw in the literature review, in the past this has not resulted in clear guidance on optimal inspection team size. Therefore, we have quantified these effects to make precise recommendations which can serve as starting points for organizations implementing UML design inspections.

One may contest the claim that inspection team sizes should be as small or as large as we have found. There are three grounds for this *when team meetings are held*, which we will discuss below.

One argument that can be made is that inspection teams should be large, even larger than the optimal team size. For example, Johnson (Johnson, 1998) notes that inspections have a unique educational capability. He contends that the process of analyzing and critiquing software artifacts produced by others is a powerful way to learn about languages, design techniques, and application domains. A similar agument is made by Doolan (Doolan, 1992) in that he reports inspections are a useful mechanism for skills transfer. In such a case, one should add more new and junior staff to the inspection team to educate them. However, Votta (Votta, 1993), citing Deming, notes that education by observation and participation is not effective, and that proper training courses are a better option. Therefore, in this case it is not obvious that team sizes should be larger than the optimal efficiency teams.

It is possible that a larger team size will lead to a larger development interval when meetings are performed. Votta (Votta, 1993), expanding on the results in Bradac et al. (Bradac et al., 1993), notes that approximately 20% of the requirements and design interval is spent just waiting for inspectors to meet. He also notes that inspection interval accounts for approximately 10% of the development interval. The inspection interval expands as more inspectors are added. This prompted him to suggest that team size should be minimized if meetings are held. However, team

size minimization is not an obvious conclusion to draw from the increased interval result. Smaller teams will find fewer defects which may be found in subsequent defect detection phases. Isolating and fixing design defects in a subsequent phase takes more time and effort. If a design defect is found during testing, then retesting is required after a fix. If retesting requires setting up of a specific hardware and software configuration, then this will add further to the overall development interval. Therefore, subsequent rework due to defects escaping the design inspection may actually elongate the interval that we thought was being shrunk through smaller teams. Consequently, one cannot automatically conclude that smaller inspection teams are always better in terms of interval.

In an observational study of inspection meetings, Votta (Votta, 1993) noted that the amount of unproductive time during meetings is in the range of t-4 hours, where t is the number of participants in the meeting. This is partially due to the fact that there can only be two people interacting at any one time. If there are, say, 15 inspectors in a meeting and it lasts two hours, then this would amount to 11 person-hours of unproductive time per inspection. This is a potent argument against having large meetings. Furthermore, given that in our study defect detection was performed during preparation, then the meeting only adds overhead resulting in reduced efficiency. Therefore, one can argue that if meetings with all inspectors are going to be held, then it is reasonable to keep the team size small, even if it is not the optimal team size. Alternatively, one can perform the preparation with the optimal team size, even if it large, and have a smaller number of inspectors participate in the meeting.

4.3 Limitations

We have presented recommendations for the optimal team size for UML design inspections under various assumptions of cost, meeting duration, and defect fix effort. Our results are pertinent to the situation where defect detection is an individual activity and the meetings only serve the purpose of defect collection. Furthermore, we make the assumption that meeting gains and losses are negligible. At least in our experimental condition, these assumptions were true.

We do not claim that our findings of optimal team size are universal. The generalizability of these findings will depend on further replications of this study. In a similar manner to performing any empirical study in software engineering, the effects that we estimate need to be confirmed further. For instance, if one were to perform a study showing technique A being 50% better than technique B in the number of defects detected, this effect ought to be confirmed in replicated studies.

We do not find that team sizes larger than 15 are optimal because 15 is the maximum team size that we studied. However, inspection of our results indicates that the maximum was reached at a ϕ value of approximately 13. It is plausible that a study that investigates with a larger number of subjects will find that above this cost ratio team sizes larger than 15 are optimal.

We did not consider the impact of interval in computing efficiency. In principle we could incorporate interval into our efficiency model. However, this would have introduced a number of new parameters, such as the cost of missing a market opportunity through a delayed release, which is difficult to estimate. This is certainly an issue that deserves further study.

4.4 Threats To Validity

It is the nature of any empirical study that assumptions are made that later on may restrict the validity of the results. Here, we list these assumptions that impose threats to internal and external validity. The threats to validity concern the study from which the data that we use was obtained.

4.4.1 Threats to Internal Validity

A potential threat here is strictly related to the fact that seeded defects were used. Despite taking special care to alleviate this, there is always the danger that seeded defects are not the same as actual defects. Specifically, they may be easier to detect than actual defects.

A second potential threat which can influence the accuracy of an estimate is the total number of defects in the artifacts, which in our case is 19. It is arguable that the number of defects found by the inspectors only cover a narrow range.

4.4.2 Threats to External Validity

We consider three potential threats to the external validity of our results: subjects, the design document and the Fusion development process.

Our subjects may not be representative of the pool of software developers that professionally used the UML for the analysis and the design of object-oriented systems, although they were all professional developers rather than students.

The design document is not necessarily representative of the ones found in industry. The limitation primarily derives from the size of the created design documents. Components in industry are usually much larger in size than the ones we used in this experiment. However, we consider the amount of material that our subjects were required to inspect in a single inspection as appropriate.

The design document was developed according to the Fusion development process. Although this process is used at Hewlett-Packard (Cohen, 1988), other companies may follow another development process, e.g., the Rational Unified Process (Jacobson et al., 1998). Since all of the Fusion models apart from operation schemata can be found in other UML-based development processes as well, we believe that this represents a rather limited threat to validity.

5 Conclusions

Software inspections are considered to be one of the most effective methods for software quality improvement. In this paper, our focus was on inspections of UML design documents. We have been looking at the optimal inspection team size as a technique for maximizing the benefits of UML inspections.

The optimal team size takes into account the costs of inspections and the savings that accrue from finding faults during design as opposed to during post-design defect detection activities. Our results demonstrate that there is no single optimal team size. The latter depends on the cost of post-design defect detection activities. If an organization knows the ratio of the cost of post-design defect detection activities to inspection cost, it can determine the optimal team size through the set of curves and table we have provided.

To our best knowledge, this study represents the first empirical study that deals with controlling and improving UML design inspections through the selection of optimal teams sizes. Future work ought to replicate this study. Perhaps even more importantly, we have also presented a methodology for evaluating inspection team size which can be applied by other researchers and organizations. With a methodology readily available it is hoped that further investigations of this issue will be conducted.

6Appendix A: Proof of Generality of Simulation Results

In section 2.2 we performed a Monte Carlo simulation illustrating that the results of an experimental study comparing different team sizes will depend on the sample sizes used in the study. The simulation was done with the dependent variable being the number of defects found. We noted that the results are the same even if the dependent variable was defect density or the proportion of defects found. Defect density is defined as the number of defects found divided by size. The proportion of defects found is defined as the number of defects found divided by the actual number of defects in the inspected artifact.

The simulation used the t-test for two independent samples (from two groups). The t-test uses the t-value to determine statistical significance. This is defined as (Sheskin, 1997):

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{k} + \frac{s_2^2}{k}}} \qquad \textbf{Eqn. 7}$$

where $\overline{X}_j$ is the mean of group j, s_j^2 is the sample variance of group j, and k is the number of teams in each group. The mean is given by:

$$\overline{X}_j = \frac{1}{k}\sum_{i=1}^{k}\frac{D_{ij}}{\lambda} \qquad \textbf{Eqn. 8}$$

where $\frac{D_{ij}}{\lambda}$ is the dependent variable, with D_{ij} being the number of defects found during the inspection by team i in group j. The λ value is some constant that characterizes the artifact. If we are looking at the number of defects, then $\lambda = 1$, if we are looking at defect density, then λ is the size of the artifact, and if we are looking at the proportion of defects found then λ is the total number of defects in the artifact before the inspection. Also:

$$s_j^2 = \frac{1}{k-1} \times \frac{\sum_{i=1}^{k}\left(\frac{D_{ij}}{\lambda}\right)^2 - \frac{\left(\sum_{i=1}^{k}\frac{D_{ij}}{\lambda}\right)}{k}}{k-1} \qquad \textbf{Eqn. 9}$$

By substituting the variance and mean values into the equation for the t-value, we get:

$$t = \frac{\sum_{i=1}^{k} D_{i1} - \sum_{i=1}^{k} D_{i2}}{\sqrt{\frac{\sum_{j=1}^{2}\sum_{i=1}^{k}\left(D_{ij}\right)^2 - \frac{1}{k}\sum_{j=1}^{2}\left(\sum_{i=1}^{k} D_{ij}\right)^2}{1-\frac{1}{k}}}} \qquad \textbf{Eqn. 10}$$

As can be seen from the above equation, the λ value cancels out. Therefore, irrespective of the λ constant that we use (one, size, or the number of defects in the documents), we get exactly the same results.

7 Appendix B: Measuring Inspection Efficiency

In this appendix we provide the rationale and details of the efficiency model that we use. One precondition for considering a model to be an efficiency model is that it must at least take into account the costs of performing an inspection.

7.1.1 Notation

Below we define a general notation for characterizing defect detection phases in a project (see (Briand et al., 1998b)). This allows us to subsequently describe existing efficiency models in a consistent and unambiguous manner.

Since ino our study we assume only two defect detection phases, design inspections and post-design defect detection activities (e.g., code inspections and testing), we limit our discussion below to the two phase case. Our assumed unit of observation is the instance of the defect detection activity.

We define the effort consumed to perform a defect detection activity f as:

$$\varepsilon_f = effort\ spent\ on\ performing\ defect\ detection\ activity\ f \qquad \textbf{Eqn. 11}$$

We define the set of unique defects existing in an artifact prior to a defect detection activity f as:

$$\alpha_f = \{x | x\ is\ a\ defect\ that\ exists\ in\ the\ artifact\ \ prior\ to\ defect\ det\ ection\ activity\ f\} \qquad \textbf{Eqn. 12}$$

We define the set of defects found during a defect detection activity f as:

$$D_f = \{x | x\ is\ a\ defect\ det\ ected\ during\ defect\ det\ ection\ activity\ f\} \qquad \textbf{Eqn. 13}$$

In the case of inspections, $|D_f|$ would be the number of defects logged during a collection meeting for example.

The estimated cost per defect of performing defect detection activity f is defined as:

$$\widehat{\varepsilon}_f = \frac{\varepsilon_f}{|D_f|} \qquad \textbf{Eqn. 14}$$

The *defect detection effectiveness* of a defect detection activity f is given by:

$$\widehat{p}_f = \frac{|D_f|}{|\alpha_f|} \qquad \textbf{Eqn. 15}$$

7.1.2 A Review of Efficiency Models

We now briefly present most of the efficiency models presented in the literature that we deemed relevant for our purpose. It should be noted that we only consider efficiency models that explicitly

take cost into account. Therefore, even though Fagan (Fagan, 1976), Jones (Jones, 1996), Remus (Remus, 1984) and Collofello and Woodfield (Collofello and Woodfield, 1989) introduce models which they call *error detection efficiency* or *defect removal efficiency*, these models are not efficiency models according to our conceptualization since they do not consider costs.

7.1.2.1 Basic Model

A simple model for the evaluation of efficiency is presented in (Gilb and Graham, 1993). This is a measure of how well effort consumed is made use of, and is defined as the number of defects found per some unit of time (e.g., work-hour). In our notation, it is:

$$A_f = \frac{|D_f|}{\varepsilon_f} \qquad \textbf{Eqn. 16}$$

However, this model does not account for defect detection phase f compared to subsequent defect detection phases in the development life cycle (i.e., what is the relative advantage of detecting defects earlier). Therefore, only the costs of phase f are taken into account, but not any potential savings.

7.1.2.2 Collofello and Woodfield Efficiency Model

Collofello and Woodfield (Collofello and Woodfield, 1989) defined efficiency in general as:

$$B_f = \frac{Costs\ saved\ by\ defect\ detection\ activity\ f}{Costs\ Consumed\ by\ defect\ detection\ activity\ f} \qquad \textbf{Eqn. 17}$$

Assuming a defect detection activity f detected and removed defects from a software artifact, we consider that if these defects had not been removed during f, they would have been removed in a later defect detection activity. Therefore, the potential costs associated with detecting and correcting the defects in later phases are saved by performing f.

Based on this assumption, the costs saved by some activity f are calculated as the sum of the costs that would be incurred with having to use activity $(f+1)$ to handle the defects detected by phase f.

The full equation for efficiency can be expressed as:-

$$B_f = \frac{\hat{\varepsilon}_{f+1} \times \hat{p}_{f+1} \times |D_f|}{\varepsilon_f} \qquad \textbf{Eqn. 18}$$

For activity $(f+1)$, we calculate the costs saved as the product of the effort per defect for activity $(f+1)$ multiplied by the number of defects expected to be detected by phase $(f+1)$. The number of defects expected to be detected by phase $(f+1)$ is calculated as the product of the effectiveness of phase $(f+1)$ and $|D_f|$

7.1.2.3 ROI Model

An alternative model that builds on the Collofello and Woodfield work is the Return on Investment model used at HP (Franz and Shih, 1994). In particular, the value of costs saved in the Collofello and Woodfield model does not consider the cost of detection phase *f* itself in calculating the

savings. Therefore, the costs saved numerator is changed by subtracting the costs consumed by *f:*

$$C_f = \frac{Costs\ saved\ by\ performing\ activity\ f - Costs\ consumed\ in\ activ}{Costs\ consumed\ in\ activity\ f} \qquad \textbf{Eqn. 19}$$

This is an improvement in that now we consider the real costs saved. It can be expressed as follows:

$$C_f = \frac{\left(\hat{\varepsilon}_{f+1} \times \hat{p}_{f+1} \times \left|D_f\right|\right) - \varepsilon_f}{\varepsilon_f} \qquad \textbf{Eqn. 20}$$

7.1.2.4 Kusumoto's Efficiency Model

Kusumoto (Kusumoto, 1993) noticed a discrepancy in the application of models such as C_f above. The discrepancy can be demonstrated with reference to the two projects in Figure 5. These projects only have two defect detection phases: inspections and testing. In both projects, if inspections had not been done, the cost of testing would be 1000 units. The first inspection consumes 10 units of cost, and saves 100 units. Therefore, the total cost is 910. The second inspection costs 60 units, and saves 600. The total cost is 460. In the second case, inspections saved much more of total defect detection costs than the first, therefore one would expect it to be more efficient. However, using the ROI model, for, example, would give them both an efficiency of 9.

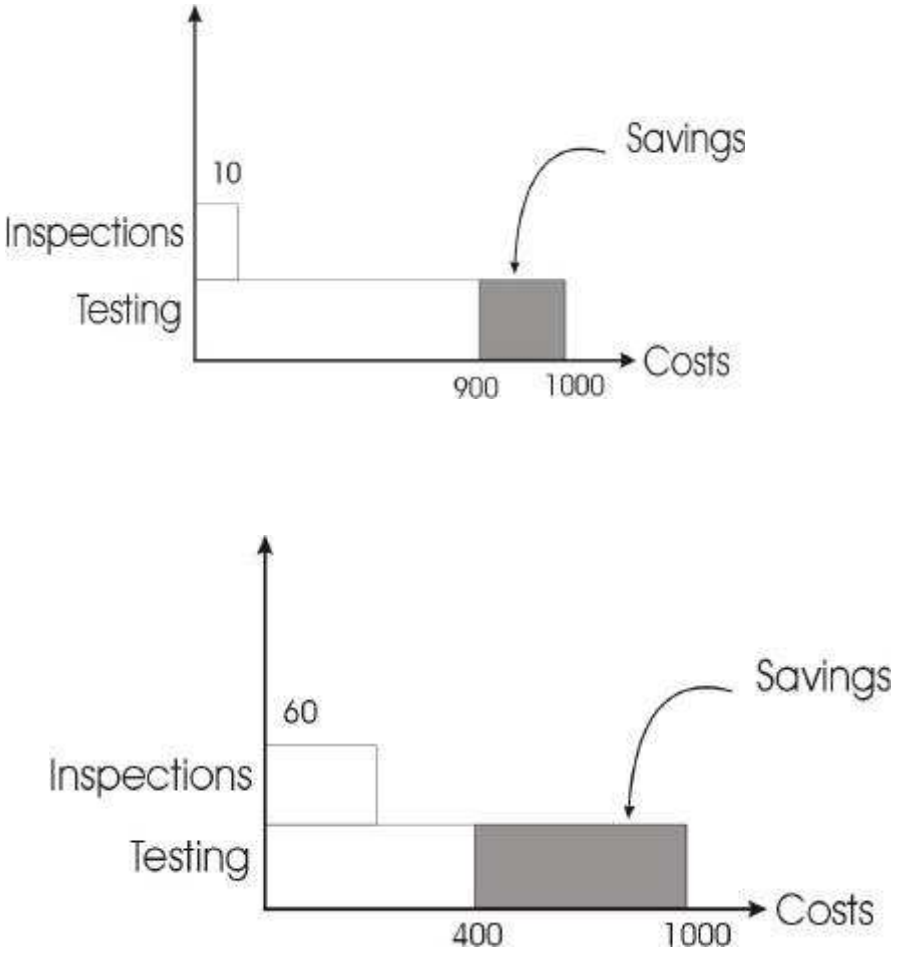

Figure 5: Comparing two different inspections.

To address this problem, Kusumoto proposes a new model. He first introduces the concept of *virtual testing cost*. This is the total cost of testing if no inspections were conducted at all. He suggests that instead of using the costs consumed as the denominator, we should use the *total* potential cost that would be consumed had inspections not been conducted (i.e., the virtual testing cost).

Using our notation, Kusumoto defined efficiency as:

$$E_f = \frac{\left(\hat{\varepsilon}_{f+1} \times \hat{p}_{f+1} \times \left|D_f\right|\right) - \varepsilon_f}{\left|\alpha_f\right| \times \varepsilon_{f+1}} \qquad \textbf{Eqn. 21}$$

We have used Kusumoto's model, and have made a distinction only between two activities: design and post-design defect detection activities. The efficiency for the design and post-design only life cycle is:

$$\hat{E}_f = \hat{p}_{Design\ Inspection} \times \left(1 - \frac{\hat{\varepsilon}_{Design\ Inspection}}{\hat{\varepsilon}_{Post\ Design}}\right) \qquad \textbf{Eqn. 22}$$

8 References

Ackerman, A.F, Buchwald, L.S, Lewski, F.H, (1989). Software Inspections: An Effective Verification Process. IEEE Software, vol. 6, No. 3, 31-36.

Adams, T., (1999). A Formula for the Re-Inspection Decision. Software Engineering Notes, 24:3, 80.

Atkinson, C. (1998). Adapting the Fusion Process to Support the Unified Modeling Language. Object Magazine**:** 32-39.

Barnard, J., Price, A., (1994). Managing Code Inspection Information. IEEE Software, 59-69.

Basili, V., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sorumgard, S., Zelkowitz, M., (1996). The Empirical Investigation of Perspective-based Reading. Empirical Software Engineering: An International Journal, 1:1, 133-164.

Bisant, D., Lyle, J., (1989). A Two-Person Inspection Method to Improve Programming Productivity. IEEE Transactions on Software Engineering, 1294-1304.

Booch, G., Rumbaugh, J., Jacobson, I. (1999). The Unified Modeling Language User Guide. Addison-Wesley.

Bottger, P.C, Yetton, P.W, (1988). An Intregration of Process and Decision Scheme Explanations of Group Problem Solving Performance. Organizational Behavior and Human Decision Processes, 42, 234-249.

Bradac, M., Perry, D., Votta, L. (1993). Prototyping a Process Monitoring Experiment. Proceedings of the International Conference on Software Engineering, 155-165.

Briand, L., Emam, K. El, Bomarius, F. (1998a). COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment. Proceedings of the 20th International Conference on Software Engineering, 390-399.

Briand, L., Emam, K. El, Fussbroich, T., Laitenberger, O. (1998b). Using Simulation to Build Inspection Efficiency Benchmarks for Development Projects. Proceedings of the Twentieth International Conference on Software Engineering, 340-349.

Briand, L., Kempkens, R., Ochs, M., Verlage, M., Lunenburger, K. (1999a). Modelling the Factors Driving the Quality of Meetings in the Software Development Process. Proceedings of the European Software Cost Estimation and Measurement (ESCOM'99), 17-26.

Briand, L. C., Freimut, B., Vollei, F., (1999b). Assessing the Cost-Effectiveness of Inspections by Combining Project Data and Expert Opinion. Technical Report-International Software Engineering Research Network ISERN-99-14.

Briand, L.C., Emam, K. El, Freimut, B.G., Laitenberger, O., (2000). A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect content. IEEE Transactions on Software Engineering, 26:6, 518-540.

Buck, F.O, (1981). Indicators of Quality Inspections. Technical Report 21.802, IBM, Kingston, NY.

Chao, A., S.Lee, Jeng, S., (1992). Estimation Population Size for Capture-Recapture Data when Capture Probabilities Vary by Time and Individual Animal. Biometrics, 48, 201-216.

Cleveland, W., Devlin, S., (1988). Locally-weighted Regression: An Approach to Regression Analysis by Local Fitting. Journal of the American Statistical Association, 83, 596-610.

Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences. Lawrence Erlbaum Associate Publishers.

Coleman, D., Arnold, P., Godoff, S., Dollin, C., Gilchrist, H., Hayes, F., Jeremaes, P. (1994). Object-Oriented Development: The Fusion Method. Englewood Cliffs, NJ:Prentice Hall.

Collofello, J., Woodfield, S., (1989). Evaluating the Effectiveness of Reliability-Assurance Techniques. Journal of Systems and Software, 191-195.

Doolan, E., (1992). Experience with Fagan's Inspection Method. Software - Practice and Experience, 22:2, 173-182.

Eick, S., Loader, C., Long, M., Votta, L., Weil, S. Vander (1992). Estimating Software Fault Content Before Coding. Proceedings of the 14th International Conference on Software Engineering, 529-523.

Fagan, M., (1976). Design and Code Inspections to Reduce Errors in Program Deveolpment. IBM Systems Journal, 15:3, 182-211.

Fagan, M., (1986). Advances in Software Inspections. IEEE Transactions on Software Engineering, 12:7, 744-751.

Franz, L., Shih, J., (1994). Estimating the Value of Inspections and Early Testing for Software Projects. Hewlett-Packard Journal, 60-67.

Friedman, J., (1984). A Variable Span Smoother. Laboratory for Computational Statistics, Dept. of Statistics, Stanford Univ., California, Technical Report No. 5.

Gilb, T., Graham, D. (1993). Software Inspection. Addison-Wesley Publishing Company.

Grady, R.B (1992). Practical Software Metrics for Project Management and Process Improvement. Englewood Cliffs, NJ, Prentice-Hall.

Host, M., Wohlin, C., (1997). A Subjective Effort Estimation Experiment. International Journal of Information and Software Technology, 39:11, 755-762.

Host, M., Wohlin, C. (1998). An Experimental Study of Individual Subjective Effort Estimations and Combinations of the Estimates. Proceedings of the 20th International Conference on Software Engineering, 332-339, Japan.

Jacobson, I., Booch, G., Rumbaugh, J. (1998). The Unified Software Development Process. Addison-Wesley.

Johnson, P., (1998). Reengineering Inspection. Communications of the ACM, 49-52.

Johnson, P., Tjahjono, D., (1998). Does Every Inspection Really Need a Meeting? Empirical Software Engineering, 9-35.

Johnson, R., Hardgrave, B., (1999). Object-Oriented Methods: Current Practices and Attitudes. Systems and Software, 48, 5-12.

Jones, C. (1996). Applied Software Measurement. McGraw-Hill.

Kelly, J.C, , Sheriff, J.S, Hops, J., (1992). An Analysis of Defect Densities Found During Software Inspections. Journal of Systems Software, 17, 111-117.

Kusumoto, S., (1993). Quantitative Evaluation of Software Reviews and Testing Processes. PhD. Thesis, Osaka University.

Laitenberger, O., Atkinson, C., Schlich, M., Emam, K. El, (2000a). An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents. Journal of Systems and Software, 53:2, 183-204.

Laitenberger, O., DeBaud, J.M., (2000). An Encompassing Life-cycle Centric Survey of Software Inspection. Journal of Systems and Software, 50, 5-31.

Laitenberger, O., Emam, K. El, Harbich, T., (2000b). An Internally Replicated Qasi-Experimental Comparison of Checklist and Perspective-based Reading of Code Documents. To appear in IEEE Transactions on Software Engineering.

Land, L., Sauer, C., Jeffery, R. (1997). Validating the Defect Detection Performance Advantage of Group Designs for Software Reviews: Report of a Laboratory Experiment Using Program Code. 6th European Software Engineering Conference, 294-309.

Lau, L., Jeffery, R., Sauer, C., (1996). Some Empirical Support for Software Development Technical Reviews. Unpublished manuscript.

Madachy, R., Little, L., Fan, S. (1993). Analysis of a Successful Inspection Program. Proceedings of the 18th Annual NASA Software Engineering Laboratory Workshop, 176-198.

Miller, J., (1999). Estimating the Number of Remaining Defects after inspection. Software Testing, Verification and Reliability, 9, 167-189.

Parnas, D., Weiss, D.M, (1987). Active Design Reviews: Principles and practices. Journal of System and Software, 7, 259-265.

Porter, A., Votta, L., Basili, V., (1995). Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment. IEEE Transactions on Software Engineering, 21:6, 563-575.

Porter, A., Siy, H., Votta, L., (1996). A Review of Software Inspections. Technical Report.

Porter, A., Votta, L.G, Siy, H.P, Toman, C.A, (1997). An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development. IEEE Transactions on Software Engineering, 23, 329-346.

Porter, A., Votta, L., (1998). Comparing Detection Methods for Software Requirements Inspections: A Replication Using Professional Subjects. Empirical Software Engineering, 3, 355-379.

Remus, H. (1984). Integrated Software Validation in the View of Inspections/Reviews. Proceedings of the Symposium on Software Validation, 57-64, H.L. Hausen(ed.), Elsevier Science Publishers.

Sauer, C., Jeffery, D.R, Land, L., Yetton, P., (2000). The Effectiveness of Software Development Technical Reviews: Behaviorally Motivated Program of Research. IEEE Transactions on Software Enginnering, 26:1, 1-14.

Schneider, G. M., Martin, J., Tsai, W.T., (1992). An Experimental Study of Fault Detection in User Requirements Documents. ACM Transactions on Software Engineering and Methodology, 1:2, 188-204.

Sheskin, D. (1997). handbook of Parametric and Nonparametric Statistical Procedures. CRC Press.

STD, IEEE, (1989). IEEE Standard for Software Reviews and Audits. ANSI/IEEE STD 1028-1988.

Strauss, H., Ebeneau, R. (1994). Software Inspection Process.

Votta, L., (1993). Does Every Inspection Need a Meeting? Proceedings of the ACM SIGSOFT 1993 Symposium on Foundations of Software Engineering, 18:5, 107-114.

Weller, E.F, (1993). Lessons from Three Years of Inspection Data. IEEE Software, 10:5, 38-45.

Yetton, P.W, Bottger, P.C, (1983). The Relationship among Group Size, Member Ability, Social Decision Schemes, and Performance. Organizational Behavior and Human Performance, 32, 145-159.