An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development

A. Porter[*]
H. Siy
Computer Science Department
University of Maryland
College Park, Maryland 20742
aporter@cs.umd.edu
harvey@cs.umd.edu

C. A. Toman
L. G. Votta
Software Production Research Department
AT&T Bell Laboratories
Naperville, Illinois 60566
cat@intgp1.att.com
votta@research.att.com

Abstract

We are conducting a long-term experiment (in progress) to compare the costs and benefits of several different software inspection methods. These methods are being applied by professional developers to a commercial software product they are currently writing.

Because the laboratory for this experiment is a live development effort, we took special care to minimize cost and risk to the project, while maximizing our ability to gather useful data.

This article has several goals: (1) to describe the experiment's design and show how we used simulation techniques to optimize it, (2) to present our preliminary results and discuss their implications for both software practitioners and researchers, and (3) to discuss how we expect to modify the experiment in order to reduce potential risks to the project.

For each inspection we randomly assign 3 independent variables: (1) the number of reviewers on each inspection team (1, 2 or 4), (2) the number of teams inspecting the code unit (1 or 2), and (3) the requirement that defects be repaired between the first and second team's inspections. The reviewers for each inspection are randomly selected without replacement from a pool of 11 experienced software developers. The dependent variables for each inspection include inspection interval (elapsed time), total effort, and the defect detection rate.

To date we have completed 34 of the planned 64 inspections. Our preliminary results challenge certain long-held beliefs about the most cost-effective ways to conduct inspections and raise some questions about the feasibility of recently proposed methods.

[*]This work is supported in part by the National Aeronautics and Space Administration under grant NSG–5123. Mr. Siy was also partly supported by AT&T 's Summer Employment Program

1 Introduction

For almost twenty years, software inspections have been promoted as a cost-effective way to improve software quality. Although the benefits of inspections have been well studied, their costs are often justified by simply observing that the longer a defect remains in a system, the more expensive it is to repair, and therefore the future cost of fixing defects is greater than the present cost of finding them.

However, this reasoning is naive because inspection costs are significantly higher than many people realize. In practice, large projects perform hundreds of inspections, each requiring five or more participants. Besides the obvious labor costs, holding such a large number of meetings can also cause delays which may significantly lengthen the development interval (calendar time to completion).[1] Since long development intervals risk substantial economic penalties, this hidden cost must be considered.

We hypothesize that different inspection approaches involve different tradeoffs between minimum interval, minimum effort and maximum effectiveness. But until now there have been no empirical studies to evaluate these tradeoffs. We are conducting such a study, and our results indicate that the choice of approach significantly affects the cost-effectiveness of the inspection.

Below, we review the relevant research literature, describe the various inspection approaches we exam-

[1]As developer's calendars fill up, it becomes increasingly difficult to schedule meetings. This pushes meeting dates farther and farther into the future, increasing the development interval.

ined, and present our experimental design, analysis, and conclusions.

1.1 Literature Review

To eliminate defects, many organizations use an iterative, three-step inspection procedure: Preparation, Collection, Repair[9] . First, a team of reviewers each reads the artifact separately, detecting as many defects as possible. Next, these newly discovered defects are collected, usually at a team meeting. They are then sent to the artifact's author for repair. Under some conditions the entire process may be repeated one or more times.

Many articles have been written about inspections. Most, however, are case studies describing their successful use [6, 7, 17, 15, 20, 10, 1] . Few critically analyze inspections or rigorously evaluate alternative inspection approaches. We believe that additional critical studies are necessary because the cost-effectiveness of inspections may well depend on such variables as team size, number of inspection sessions, and the ratio of individual contributions versus group efforts.

Team Size: Inspections are usually carried out by a team of four to six reviewers. Buck[2] provides data (from an uncontrolled experiment) that showed no difference in the effectiveness of three, four, and five-person teams. However, no studies have measured the effect of team size on inspection interval.

Single-Session vs. Multiple-Session Inspections: Traditionally, inspections are carried out in a single session. Additional sessions occur only if the original artifact or the inspection itself is believed to be seriously flawed. But some authors have argued that multiple session inspections might be more effective.

Tsai et al.[16] developed the N-fold inspection process, in which N teams each carry out independent inspections of the entire artifact. The results of each inspection are collated by a single moderator, who removes duplicate defect reports. N-fold inspections will find more defects than regular inspections as long as the teams don't completely duplicate each other's work. However, they are far more expensive than a single team inspection.

Parnas and Weiss' active design reviews (ADR)[12] and Knight and Myers' phased inspections (PI)[11] are also multiple-session inspection procedures. Each inspection is divided into several mini-inspections or "phases". ADR phases are independent, while PI phases are executed sequentially and all known defects are repaired after each phase. Usually each phase is carried out by one or more reviewers concentrating on a single type of defect.

The proponents of multiple-session inspections believe they will be much more effective than single-session inspections, but they have not shown this empirically, nor have they considered its effect on inspection interval.

Group-centered vs. Individual-centered Inspections: It is widely believed that most defects are first identified during the collection meeting as a result of group interaction[5] . Consequently, most research has focused on streamlining the collection meeting by determining who should attend, what roles they should play, how long the meeting should last, etc.

On the other hand, several recent studies have concluded that most defects are actually found by individuals prior to the collection meeting. Humphrey [8] claims that the percentage of defects first discovered at the collection meeting ("meeting gain rate") averages about 25%. In an industrial case study of 50 design inspections, Votta [18] found far lower meeting gain rates (about 5%). Porter et al.[14] conducted a controlled experiment in which graduate students in computer science inspected several requirements specifications. Their results show meeting gain rates consistent with Votta's. They also show that these gains are offset by "meeting losses" (defects first discovered during preparation but never reported at the collection meeting). Again, since this issue clearly affects both the research and practice of inspections, additional studies are needed.

Defect Detection Methods. Preparation, the first step of the inspection process, is accomplished through the application of defect detection methods. These methods are composed of defect detection techniques, individual reviewer responsibilities, and a policy for coordinating responsibilities among the review team.

Defect detection techniques range in prescriptiveness from intuitive, nonsystematic procedures (such as ad hoc or checklist techniques) to explicit and highly systematic procedures (such as correctness proofs).

A reviewer's individual responsibility may be general, to identify as many defects as possible, or specific, to focus on a limited set of issues (such as ensuring appropriate use of hardware interfaces, identifying untestable requirements, or checking conformity to coding standards).

Individual responsibilities may or may not be coordinated among the review team members. When they are not coordinated, all reviewers have identical responsibilities. In contrast, the reviewers in coordinated teams have distinct responsibilities.

The most frequently used detection methods (Ad Hoc and Checklist) rely on nonsystematic techniques. Reviewer responsibilities are general and identical. Multiple-session inspection approaches normally require reviewers to carry out specific and distinct responsibilities. One reason these approaches are rarely used may be that many practitioners consider it too risky to remove the redundancy of general and identical responsibilities and to focus reviewers on narrow sets of issues that may or may not be present. Clearly, the advantages and disadvantages of alternative defect detection methods need to be understood before new methods can be safely applied.

1.2 Hypotheses

Inspection approaches are usually evaluated according to the number of defects they find. As a result, some information has been collected about the effectiveness of different approaches, but very little about their costs. We believe that cost is as important as effectiveness, and we hypothesize that different approaches have significantly different tradeoffs between development interval, development effort, and detection effectiveness. Specifically, we hypothesize that

- inspections with large teams have longer inspection intervals, but find no more defects than smaller teams;
- collection meetings do not significantly increase detection effectiveness;
- multiple-session inspections are more effective than single-session inspections, but significantly increase inspection interval.

2 The Experiment

To evaluate these hypotheses we designed and are conducting a controlled experiment. Our purpose is to compare the tradeoffs between minimum interval, minimum effort, and maximum effectiveness of several inspection approaches.

2.1 Experimental Setting

We are currently running this experiment at AT&T on a project that is developing a compiler and environment to support developers of the AT&T 5ESS$^{®}$ telephone switching system. The finished system is expected to contain 30K new lines of C++ code, plus 6K which will be reused from a prototype.

Our inspector pool consists of 11 experienced developers, each of which has received inspection training in the last 5 years. The project began coding during June, 1994, and will perform 64 code inspections by the middle of 1995.

2.2 Operational Model

To test our hypotheses we must measure both the interval and the effectiveness of every inspection. We began by constructing two models; one for calculating inspection interval and effort, and another for estimating the number of defects in a code unit.

2.2.1 Modeling the Inspection Interval The inspection process begins when a code unit is ready for inspection and ends when the author finishes repairing the defects found in the code. The elapsed time between these events is called the inspection interval.

The length of this interval depends on the time spent working (preparing, attending collection meetings, and repairing defects) and the time spent waiting (time during which the inspection does not progress due to process dependencies, higher priority work, scheduling conflicts, etc).

In order to measure inspection interval and its various subintervals, we devised an inspection time model based on visible inspection events [19] . Whenever one of these events occurs it is timestamped and the event's participants are recorded. (In most cases this information is manually recorded on the forms described in Section 2.4.1.) These events occur, for example, when code is ready for inspection, or when a reviewer starts or finishes his or her preparation. This information is entered into a database, and inspection intervals are reconstructed by performing queries against the database.

Inspection effort can be calculated by summing the appropriate subintervals. At this time, however, we haven't fully analyzed the effort data. Instead we are concentrating on inspection interval as our primary cost measure because of its schedule implications.

2.2.2 Modeling the Defect Detection Ratio One important measure of an inspection's effectiveness is its defect detection ratio – the number of defects found during the inspection divided by the total number of defects in the code. Because we never know exactly how many defects an artifact contains, it is impossible to make this measurement directly, and therefore we are forced to approximate it.

The estimation procedure must be (a) as accurate as possible and (b) available throughout the study because we are experimenting with a live project and must identify and eliminate dangerously ineffective approaches as soon as possible.

We found no single approximation that met both criteria. Therefore we will use three methods.

- **Observed detection ratio:** We assume that total defect density is constant for all code units and that we can compare the number of defects found per KNCSL. This is always available, but may be very inaccurate.
- **Partial estimation of detection ratio:** We use capture-recapture methods to estimate pre-inspection defect content. This estimation can be performed when there are at least two reviewers and they discover some defects in common. Under these conditions this method is more accurate than the observed detection ratio and is available immediately after every inspection. Since capture-recapture techniques make that strong statistical assumptions, we are currently testing our data to see whether or not this technique will be appropriate. (See Eick et al. [4] for more details.)
- **Complete estimation of detection ratio:** We will track the code through testing and field deployment, recording new defects as they are found. This is the most accurate method, but is not available until well after the project is completed.

2.3 Experimental Design

2.3.1 Variables The experiment manipulates 3 independent variables:

1. the number of reviewers per team (one, two, or four reviewers, in addition to the author),
2. the number of inspection sessions (one session or two sessions),
3. the coordination between sessions (in two-session inspections the author does or does not repair known defects between sessions).

These variables reflect many (but not all) of the differences between Fagan inspections, N-Fold inspections, Active Design Reviews, and Phased Inspections. One very important difference that is **not** captured in our experiment is the choice of defect detection methods. The methods used in Active Design Reviews and Phased Inspections involve systematic techniques, with specific and distinct responsibilities, while Fagan and N-fold inspection normally use non-systematic techniques with general and identical responsibilities.

	Number of Sessions			Totals
	1	2		
Reviewers		With Repair	No Repair	
1	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$
2	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$
4	$\frac{1}{3}$	0	0	$\frac{1}{3}$
Totals	$\frac{5}{9}$	$\frac{2}{9}$	$\frac{2}{9}$	1

Table 1: This table gives the proportion of inspections allocated to each setting of the independent variables.

The treatment distributions are shown in Table 1.

For each inspection we measured 5 dependent variables:

1. inspection interval,
2. inspection effort,
3. estimated defect detection ratio,
4. the percentage of defects first identified at the collection meeting (meeting gain rate),
5. the percentage of potential defects reported by an individual, that are determined not to be defects during the collection meeting (meeting suppression rate).

We also capture repair statistics for every defect (See Section 2.4.2). This information is used to discard certain defect reports from the analysis – i.e., those regarding defects that required no changes to fix them or concerned coding style rather than incorrect functionality.

2.3.2 Design This experiment uses a $2^2 \times 3$ partial factorial design to compare the interval, effort, and effectiveness of inspections with different team sizes, number of inspection sessions, and coordination strategies. We chose a partial factorial design because some treatment combinations were considered too expensive (e.g., two-session-four-person inspections with and without repair).

2.3.3 Threats to Internal Validity Threats to internal validity are influences that can affect the dependent variable without the researcher's knowledge. We considered three such influences: (1) selection effects, (2) maturation effects, and (3) instrumentation effects.

Selection effects are due to natural variation in human performance. For example, if one-person inspections are done only by highly experienced people, then their greater than average skill can be mistaken for a difference in the effectiveness of the treatments. We limited this effect by randomly assigning team members for each inspection. This way individual differences are spread across all treatments.

Maturation effects result from the participants' skills improving with experience. Again we randomly assigned the treatment for each inspection to spread any performance improvements across all treatments.

Instrumentation effects are caused by the code to be inspected, by differences in the data collection forms, or by other experimental materials. In this study, one set of data collection forms was used for all treatments. Since we could not control code quality or code size, we randomly assigned the treatment for each inspection.

2.3.4 Threats to External Validity

Threats to external validity are conditions that limit our ability to generalize the results of our experiment to industrial practice. We considered three sources of such threats: (1) experimental scale, (2) subject generalizability, and (3) subject representativeness.

Experimental scale becomes a threat when the experimental setting or the materials are not representative of industrial practice. We avoided this threat by conducting the experiment on a live software project.

A threat to subject generalizability may exist when the subject population is not drawn from the industrial population. This is not a concern here because our subjects are software professionals.

Threats regarding subject representativeness arise when the subject population is not representative of the industrial population. This may endanger our study because our subjects are members of a development team, not a random sample of the entire development population.

2.3.5 Analysis Strategy

Our strategy for analyzing the experiment has three steps: resolution analysis, calibration, and hypothesis testing.

Resolution Analysis. An experiment's resolution is the minimum difference in the effectiveness of two treatments that can be reliably detected.

We performed the resolution analysis using a Monte Carlo simulation. The simulation indicates that with as few as 5 observations per treatment the experiment can reliably detect a difference as small as .075 in the defect detection rate of any two treatments. The strongest influence on the experiment's resolution is the standard deviation of the code units' defect content – the smaller the standard deviation the finer the resolution. See Porter et al. [13] for more details.

Calibration. We continuously calibrate the experiment by monitoring the sample mean and variance of each treatment's detection ratio and inspection interval, and the number of observed inspections. Based on this information we may discontinue some treatments (1) if their effectiveness is so low or if their interval is so long that it puts the project at risk, or (2) if it is determined that the current distributions will produce too few data points to identify statistically significant differences in their performances[2]. In fact, we are planning to discontinue at least three treatments in the remainder of the study. (See Section 5.)

Hypothesis Testing. Once the data are collected we analyze the combined effect of the independent variables on the dependent variables to evaluate our hypotheses. Once the significant explanatory variables are discovered and their magnitude estimated, we will examine subsets of the data to study our specific hypotheses.

2.4 Experimental Instrumentation

We designed several instruments for this experiment: preparation and meeting forms, author repair forms, and participant reference cards.

2.4.1 Data Collection Forms

We designed two data collection forms, one for preparation and another for the collection meeting.

The meeting form is filled in at the collection meeting. When completed, it gives the time during which the meeting was held, and a page number, a line number, and an ID for each defect.

The preparation form is filled in during both preparation and collection. During preparation, the reviewer records the times during which he or she reviewed, and the page and line number of each issue ("suspected" defect). During the collection meeting the team will decide which of the reviewer's issues are, in fact, real defects. At this time, real defects are recorded on the meeting form and given an ID. If a reviewer had discovered this defect during preparation then they record this ID on their preparation form.

[2]For example, if two treatments have little within-treatment variance and very different mean performance, then few data points are needed to statistically establish the difference. Otherwise, more observations are necessary. If the number of data points needed is more than the number of inspections to be done, we will have to consider removing some of the treatments.

2.4.2 Author Repair Forms

The author repair form captures information about each defect identified during the inspection. This information includes Defect Disposition (no change required, repaired, deferred); Repair Effort ($\leq 1hr$, $\leq 4hr$, $\leq 8hr$, or $> 8hr$), Repair Locality (whether the repair was isolated to the inspected code unit), Repair Responsibility (whether the repair required other developers to change their code), Related Defect Flag (whether the repair triggered the detection of new defects), and Defect Characteristics (whether the defect required any change in the code, was changed to improve readability or to conform to coding standards, was changed to correct violations of requirements or design, or was changed to improve efficiency).

2.4.3 Participant Reference Cards

Each participant received a set of reference cards containing a concise description of the experimental procedures and the responsibilities of the authors and reviewers.

2.5 Conducting the Experiment

To support the experiment, Mr. Harvey Siy, a doctoral student working with Dr. Porter at the University of Maryland, joined the development team in the role of inspection quality engineer (IQE). The IQE is responsible for tracking the experiment's progress, capturing and validating data, and observing all inspections. The IQE also attends the development team's meetings, but has no development responsibilities.

When a code unit is ready for inspection, its author sends an inspection request to the IQE. The IQE then randomly assigns a treatment (based on the treatment distributions given in Table 1) and randomly draws a review team from the reviewer pool.[3] These names are then given to the author, who schedules the collection meeting. Once the meeting is scheduled, the IQE puts together the team's inspection packets.[4]

The inspection process used in this environment is similar to a Fagan inspection, but there are some differences. During preparation, reviewers analyze the code in order to find defects, not just to acquaint themselves with the code. During preparation reviewers have no specific technical roles (i.e., tester, or end-user) and have no checklists or other defect detection aids. All suspected defects are recorded on the preparation form. The experiment places no time limit on preparation, but a organizational limit of 300 LOC over a maximum of 2 hours is generally observed.

For the collection meeting one reviewer is selected to be the reader. This reviewer paraphrases each line of code. During this paraphrasing activity, reviewers may bring up any issues found during preparation or discuss new issues. The code unit's author compiles the master list of all defects. One reviewer is also assigned to be the moderator.

The IQE also attends the collection meeting to ensure that all the procedures are followed correctly. After the collection meeting he gives the preparation forms to the author, who then repairs the defects, fills out the author repair form, and returns all forms to the IQE. After the forms are returned, the IQE interviews the author to validate the data.

3 Data and Analysis

Four sets of data are important for this study: the team defect summaries, the individual defect summaries, the interval summaries, and the author repair summaries. This information is captured on the preparation, meeting, and repair forms.

The team defect summary forms show all the defects discovered by each team. This form is filled out by the author during the collection meeting and is used to assess the effectiveness of each treatment. It is also used to measure the added benefits of a second inspection session by comparing the meeting reports from both halves of two-session inspections with no repair.

The individual defect summary forms show whether or not a reviewer discovered a particular defect. This form is filled out during preparation to record all suspected defects. The data is gathered from the preparation form and is compiled during the collection meeting when reviewers cross-reference their suspected defects with those that are recorded on the meeting form. This information, together with the team summaries, is used to calculate the capture-recapture estimates and to measure the benefits of collection meetings.

The interval summaries describe the amount of calendar time that was needed to complete the inspection process. This information is used to compare the average inspection interval and the distribution of subintervals for each treatment.

The author repair summaries characterize all the defects and provide information about the effort required to repair them.

At this time 34 inspections have been completed. Consequently, we do not yet have enough data to

[3] We do not allow any single reviewer to be assigned to both teams in a two-session inspection.

[4] The inspection packet contains the code to be inspected, all required data collection forms and instructions, and a notice giving the time and location of the collection meeting.

definitively evaluate our hypotheses. However, we can look at the apparent trends in our preliminary data, explore the implications of this data for our hypotheses, and discuss how the resolution of these hypotheses at the completion of the experiment will help us answer several open research questions.

3.1 Data Reduction

Data reduction is the manipulation of data after its collection. We have reduced our data in order to (1) remove data that is not pertinent to our study, and to (2) adjust for systematic measurement errors.

3.1.1 Reducing the Defect Data The preparation and meeting forms capture the set of issues that were raised during each inspection. The reduction we made was to remove duplicate reports from 2-session-without-repair inspections. This task is performed by the IQE and the code unit's author.

Another reduction was made because, in practice, many issues, even if they went unrepaired, would not lead to incorrect system behavior, and they are therefore of no interest to our analysis.

Although defect classifications are usually made during the collection meeting, we feel that authors understand the issues better after they have attempted to repair them, and therefore, can make more reliable classifications. consequently, we use information in the repair form and interviews with each author to classify the issues into one of three categories:

- False Positives (issues for which no changes were made),
- Soft Maintenance (issues for which changes were made only to improve readability or enforce coding standards),
- True Defects (issues for which changes were made to fix requirements or design violations, or to improve system efficiency).

The distribution of defect classifications for each treatment appears in Figure 1. Across all inspections, 24% of the issues are False Positives, 55% involve Soft Maintenance, and 21% are True Defects. We consider only True Defects in our analysis of estimated defect detection ratio (a dependent variable).[5]

[5] We observed that most of the soft maintenance issues are caused by conflicts between the coding style or conventions used by different reviewers. In and of themselves, these are not true defects. We feel these issues might be more efficiently handled outside of the inspection process with automated tools or standards.

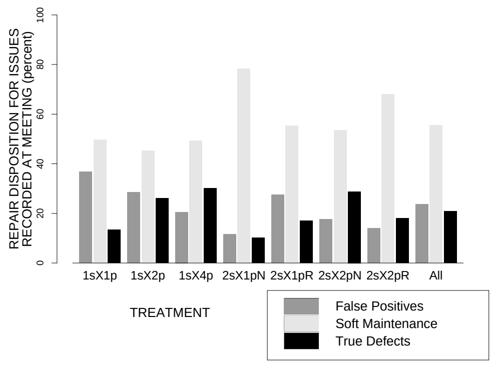

Figure 1: **Disposition of Issues Recorded at the Collection Meeting.** For each treatment, the barchart shows the percentage of the issues recorded at collection meetings that turn out to be false positives, soft maintenance, or true defects. Across all treatments, only 21% of the issues are true defects.

3.1.2 Reducing the Interval Data The preparation, meeting, and repair forms show the dates on which important inspection events occur. This data is used to compute the inspection intervals.

We made two reductions to this data. First, we observed that some authors did not repair defects immediately following the collection meeting. Instead, they preferred to concentrate on other development activities, and fix the defects later, during slow work periods. To remove these cases from the analysis, we use only the pre-meeting interval (the calendar period between the submission of an inspection request and the completion of the collection meeting) as our initial measure of inspection interval.

When this reduction is made, two-session inspections have two inspection subintervals – one for each session. The interval for a two-session inspection is the longer of its two subintervals, since both of them begin at the same time.

Next, we removed all non-working days from the interval. Non-working days are defined as either (1) weekend days during which no inspection activities occur, or (2) days during which the author is on vacation and no reviewer performs any inspection activities. We use these reduced intervals as our measure of inspection interval.

Figure 2 is a boxplot[6] showing the number of working days from the issuance of the inspection request

[6] In this paper we have made extensive use of boxplots to represent data distributions. Each data set is represented by a box whose height spans the central 50% of the data. The upper and lower ends of the box marks the upper and lower quartiles. The data's median is denoted by a bold line within the box. The dashed vertical lines attached to the box indicate the tails of the distribution; they extend to the standard range of the

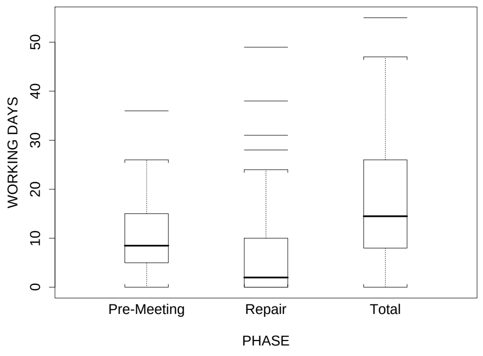

Figure 2: **Pre-meeting Inspection Interval.** These boxplots show all the interval data divided into two parts: time before the meeting and time after the meeting. The total inspection time has a median of 14.5 days, 59% of which is before the meeting.

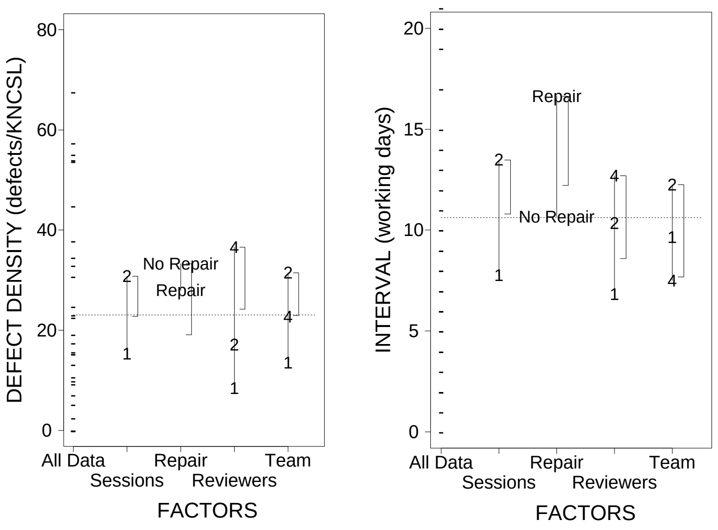

Figure 3: **Effectiveness and Interval by Independent Variables.** The dashes in the far left column of the first plot show the defect detection rates for all inspections. The dotted horizontal line marks the average defect detection rate. The other four columns indicate factors that may influence this dependent variable. The plot demonstrates the ability of each factor to explain variations in the dependent variable. For the Sessions factor, the vertical locations of the symbols "1" and "2" are determined by averaging the defect detection rates for all code inspection units having 1 or 2 sessions. The bracket at each factor represents one standard error of difference. If the actual difference is longer than the bracket, then that factor is statistically significant. The right plot shows similar information for inspection interval.

	Number of Sessions			Totals
	1	2		
Team Size		With Repair	No Repair	
1	6	4	4	14
2	7	4	5	16
4	4	0	0	4
Totals	17	8	9	34

Table 2: This table shows the number of inspections we have observed for each treatment.

to the collection meeting, from the collection meeting to the completion of repair, and the total. The total inspection interval has a median of 14.5 working days, 8.5 before and 6 after the collection meeting.

3.2 Overview of Data

Table 2 shows the number of observations to date for each treatment. Figure 3 is a contrast plot showing the interval and effectiveness of all inspections and for every setting of each independent variable. This information is used to determine the amount of the variation in the dependent variables that is explained by each independent variable. We also show another variable, total number of reviewers (the number of reviewers per session multiplied by the number of sessions). This variable provides information about the relative influence of team size vs. number of sessions.

3.3 Analysis of Interval Data

Inspection interval is an important measure of cost. Figure 4 shows the inspection interval (pre-meeting only) by treatment and for all treatments.

The cost of increasing team size is suggested by comparing 1-session inspections (1sX1p, 1sX2p, and 1sX4p). Since there is no difference[7] in the three intervals team size alone does not appear to affect interval.

The additional cost of multiple inspection sessions can be seen by comparing 1-session inspections with 2-session-without-repair inspections (1sX2p and 1sX1p with 2sX2pN and 2sX1pN inspections). We find that 2-session inspections don't take longer than

data (1.5 times the inter-quartile range). All other detached points are "outliers". [3]

[7]In this article, we consider two data distributions to be significantly different only if the Wilcoxon rank sum test rejects the null hypothesis that the observations are drawn from the same population with a confidence level $\geq .9$.

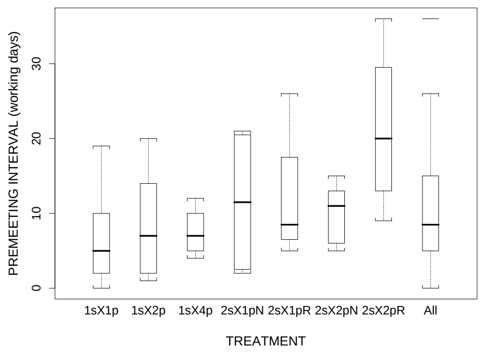

Figure 4: **Pre-meeting Interval by Treatment.** This plot shows the observed pre-meeting interval for each inspection treatment. Across all treatments, the median interval is 8.5 days.

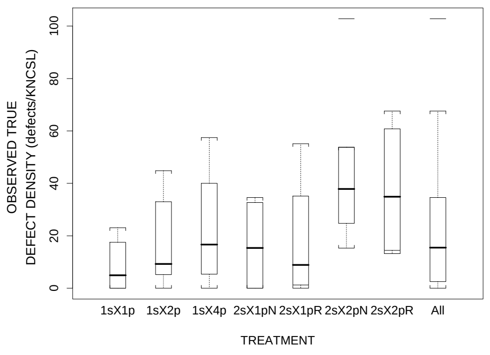

Figure 5: **Observed Defect Density by Treatment.** This plot shows the observed defect density for each inspection treatment. Across all inspections, the median defect detection rate was 24 defects per KNCSL.

1-session inspections. However, we also note that the intervals of 2-session inspections have much more variance than do 1-session inspections.

The cost of serializing two inspection sessions is suggested by comparing 2-session-with-repair inspections to 2-session-without-repair inspections (2sX2pN and 2sX1pN with 2sX2pR and 2sX1pR inspections). When the teams have only 1 reviewer we find no difference in interval, but difference is found for 2-reviewer teams. This may indicate that requiring repair between sessions only increases interval as the team size grows.

We can draw other observations from this data. For instance, except for the 2sX2pR treatment (20 days), all treatments have similar median intervals (7.5 days).

3.4 Analysis of Effectiveness Data

The primary benefit of inspections is that they find defects. This benefit will vary with the different inspection treatments. Figure 5 shows the observed defect density for all inspections and for each treatment separately.

The effect of increasing team size is suggested by noting that there is no difference in the effectiveness of any 1-session inspection (1sX1p, 1sX2p, and 1sX4p inspections). Although the median defect detection rate increases with the number of reviewers, the variance increases also.

The effective of having multiple sessions is suggested by comparing 1-session inspections with 2-session-without-repair inspections (1sX2p and 1sX1p with 2sX2pN and 2sX1pN inspections). 2-session inspections are more effective than 1-session inspection only when there are two reviewers on the team.

The effect of serializing multiple sessions is suggested by comparing 2-session-with-repair inspections to 2-session-without-repair inspections (2sX2pN and 2sX1pN with 2sX2pR and 2sX1pR inspections). The data show that repairing defects between multiple sessions doesn't increase effectiveness.

Several other trends appear in the preliminary data. First, 2sX2p inspections are more effective than 1sX4p, suggesting that 4 people will be more effective as 2 small groups than as 1 larger group. On the other hand 2sX1p inspections aren't more effective than 1sX2p inspections.

3.5 Meeting Effects

During preparation, reviewers analyze the code units to discover defects. After all reviewers are finished preparing, a collection meeting is held. These meetings are believed to serve at least two important functions: (1) suppressing unimportant or incorrect defect reports, and (2) finding new defects. In this section we analyze the effect collection meetings on inspection performance.

Analysis of Preparation Reports. One input to the collection meeting is the list of defects found by each reviewer during his or her preparation. Figure 6 shows the percentage of defects reported by each reviewer that are eventually determined to be true defects. Across all 88 preparation reports, only 15% of all issues turn out to true defects.

The only differences we see are that 2sX2pN inspections have more true reports than 1sX1p, 1sX4p, 2sX1pN and 2sX1pR inspections, and 2sX2pR inspections have more true reports than 2sX1pN and

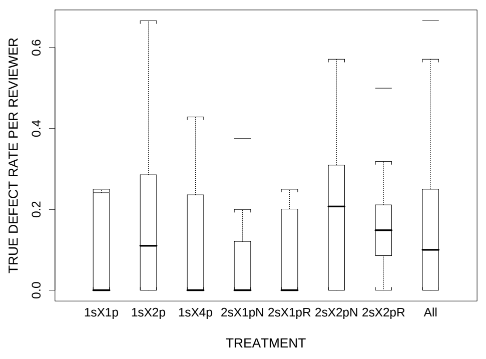

Figure 6: **True Defect Rate per Reviewer Preparation Report by Treatment.** This boxplot shows the rate at which defects found during preparation are eventually considered to be true defects. Across all treatments, only 15% of the reports turn out to be true defects.

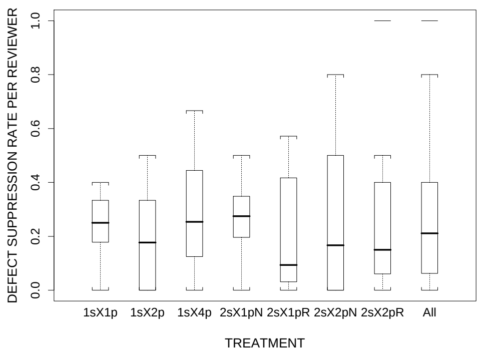

Figure 7: **Meeting Suppression Rate by Treatment.** These boxplots show the suppression rate for each reviewer by treatment. The suppression rate for a reviewer is the number of defects detected during preparation but not included in the collection meeting defect report, divided by the total number of defects recorded by the reviewer in his/her preparation. Across all inspections, 25% of the preparation reports are suppressed.

2sX1pR inspections. There is no difference between any 2-person method. The largest diffference is between 2sX2pN inspections (20% true reports) and 2sX1pN inspections (8% true reports).

Analysis of Suppression. It is generally assumed that collection meetings suppress unimportant or incorrect defect reports, and that without these meetings, authors would have to process many spurious reports during repair. As we showed in the previous section an average of 85% of reviewer reports do not involve true defects.

Figure 7 shows the suppression rates for all 88 reviewer reports. Across all inspections about 25% of issues are suppressed. One trend in the preliminary data is that suppression appears to be independent of the treatment.

Analysis of Meeting Gains Another function of the collection meeting is to find new defects in addition to those discovered by the individual reviewers. Defects that are first discovered at the collection meeting are called meeting gains.

Figure 8 shows the meeting gain rates for all 51 collection meetings. Across all inspections, 33% of all defects discovered are meeting gains. The data suggests meeting gains are independent of treatment.

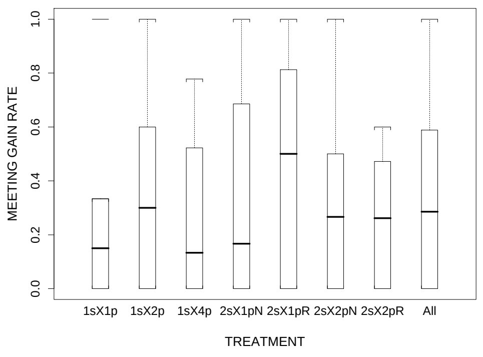

Figure 8: **Meeting Gain Rate by Treatment.** These boxplots shows the meeting gain rates for all inspections and for each treatment. The average rate was 33%.

4 Conclusions

We are in the midst of a long term experiment in which we apply different software inspection methods to all the code units produced during a professional software development. We are assessing the methods by randomly assigning different team sizes, numbers of inspection sessions, author repair activities, and reviewers to each code unit. To date we have completed 34 of the planned 64 inspections. We expect to finish the remaining 30 inspections by the middle of 1995.

Our preliminary results challenge certain long-held beliefs about the most cost-effective ways to conduct inspections. It also raises some questions about the feasibility of recently proposed methods.

In this section we summarize our preliminary results and discuss their implications from points of view of both practitioners and researchers.

Individual Preparation. Our data indicate that almost one-half of the issues reported during preparation turn out to be false positives, Another 35% pertain to nonfunctional style and maintenance issues. Finally, only 15% concern defects that will compromise the functionality of the delivered system.

For practitioners this suggests that a good deal of effort is currently being expended on issues that might better be handled by automated tools or standards.

For researchers this suggests that developing better defect detection techniques may be much more important than any of the organizational issues discussed in this article [14] .

Meeting Gains. On the average 33% of defects were meeting gains.

One implication of this result is that it may be worthwhile to study the cost-benefits of meeting-less inspections. For example, 2sX2pN inspections are more than 33% more effective than 1sX4p inspections. Without a collection meeting 2sX2pN inspections would still be more effective, but might require less total effort and have a shorter interval.

These meeting gain rates are higher than those reported by Votta[18] (5%). Since meetings without meeting gains are a large, unnecessary expense, it's important for researchers to better understand this issue. Some possible explanations for this are (1) Votta's study focused on design inspections rather than code inspections, (2) the average team size for a design inspection is considerably larger than for code inspections, or (3) design reviewers may prepare much more thoroughly since design defects are likely to be more damaging than code defects. We will be examining this issue more closely in the remainder of the experiment.

Team Size. We found no difference in the interval or effectiveness of inspections with 1, 2, or 4-reviewer teams. The effectiveness of 1-reviewer teams was considered poorer than that of larger teams with a confidence level of about .2. Although this difference is not normally considered statistically significant, it does suggest that 1-reviewer teams are not as effective as 2 or 4-reviewer teams.

For practitioners this suggests that reducing the default number of reviewers from 4 to 2 may significantly reduce effort without increasing interval or reducing effectiveness.

The implications of this result for researchers is unclear. We need to develop a better understanding of why 4-reviewer teams weren't more effective than 2-reviewer teams. We will explore this issue further during the remainder of the experiment.

Multiple Sessions. We found that 2, 2-person teams were more effective than 1, 4-person team, but 2, 1-person teams weren't more effective than 1, 2-person team. We also found that 2-session inspections without repair have the same interval as 1-session inspections.

In practice this indicates that 2-session-without-repair inspections should be used if their increased effectiveness is considered to be worth the extra effort (not interval).

These results are significant for researchers as well. Multiple session methods such as active design reviews (ADR) and phased inspections (PI) rely on the assumption that several one person teams using specially developed defect detection techniques can be more effective than a single large team without special techniques. Some of our experimental treatments mimic the ADR and PI methods (without special defect detection techniques). This suggests that any improvement offered by these techniques will not come just from the structural organization of the inspection, but will depend heavily on the development of defect detection techniques.

The performance of 2sX2pN inspections partially supports use of multiple sessions methods such as N-fold inspections. Of course, our data doesn't indicate to what degree these methods will scale up (i.e., as the number of sessions grows beyond 2).

Serializing Multiple Sessions. We found that repairing defects in between multiple sessions had no effect on defect detection rate, but in some cases increased interval dramatically.

In practice, we see no reason to repair defects between multiple sessions. Furthermore, some of the developers in our study felt that the 2-session-with-repair treatments caused the greatest disruption in their schedule. For example, they had to explicitly schedule their repairs although they would normally have used repair to fill slow work periods.

This result raises several research questions as well. Why aren't more defects found relative to the without-repair inspections? Is it because different defects are found in the 2 sessions of without-repair inspections? Is it because the defect detection techniques being used are finding all the defects they can find (producing an upper bound on any treatment's effectiveness)?

This result also provides some information about the recently proposed phased inspection method. This method requires small teams each using specially defect detection techniques to perform several inspections in serial, repairing defects between each session. Our data shows no improvement due solely to the presence of repair. Consequently, without special defect detection techniques this approach in unlikely to be effective.

5 Future Work

During the remainder of this experiment we will be making several changes to the experimental design. In particular we will remove treatments that require defect repair in between a 2-session inspection (2sX2pR and 2sX1pR). As discussed previously, these treatments are no more effective than the without repair variety, but can affect interval considerably.

We are also considering removing the 1sX1p treatment. As discussed above, this treatment is potentially the poorest performing treatment and continuing to use it poses some threats to the projects. Therefore, we will be monitoring it closely as the experiment continues.

We will also deepen our data analysis. For example, we will explore the following questions.

- How much variation in the observed performance can be explained by natural variation in the code being inspected?
- How much variation in the observed performance can be explained by natural variation in the individual inspectors?
- How many defects are found in common by both teams in a 2-session inspection?
- Why are 2-sessions better than 1? Is it because there is little overlap between the 2 teams or is it because with 2 teams there is a greater chance of selecting a high performance team?

Finally, we feel it is important that others attempt to replicate our work, and we are preparing materials to facilitate this. Although we have rigorously defined our experiment and tried to remove the external threats to validity, it is only through replication that we can be sure all of them have been addressed.

Acknowledgments

We would like to recognize the efforts of the experimental participants – an excellent job is being done by all. Our special thanks to Nancy Staudenmayer for her many helpful comments on the experimental design. Our thanks to Dave Weiss and Mary Zajac who did much to ensure we had all the necessary resources and to Clive Loader and Scott VanderWiel for their valuable technical comments. Finally, Art Caso's editing is greatly appreciated.

References

[1] Barry Boehm. Verifying and validating software requirements and design specifications. *IEEE Software*, 1(1):75–88, January 1984.

[2] F. O. Buck. Indicators of quality inspections. Technical Report 21.802, IBM Systems Products Division, Kingston, NY, September 1981.

[3] John M. Chambers, William S. Cleveland, Beat Kleiner, and Paul A. Tukey. *Graphical Methods for Data Analysis.* Wadsworth International Group, Belmont, California, 1983.

[4] Stephen G Eick, Clive R Loader, M. David Long, Scott A Vander Wiel, and Lawrence G Votta. Capture-recapture and other statistical methods for software inspection data. In *Computing Science and Statistics: Proceedings of the 25th Symposium on the Interface*, San Diego, California, March 1993. Interface Foundation of North America.

[5] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):216–245, 1976.

[6] P. J. Fowler. In-process inspections of work products at at&t. *AT&T Technical Journal*, March-April 1986.

[7] D. P. Freeman and G. M. Weinberg. *Handbook of Walkthroughs, Inspections and Technical Reviews.* Little, Brown, Boston, MA, 1982.

[8] Watts Humphrey. *Managing the Software Process.* Addison-Wesley, New York, 1989.

[9] *IEEE Standard for software reviews and audits.* Soft. Eng. Tech. Comm. of the IEEE Computer Society, 1989. IEEE Std 1028-1988.

[10] John C. Kelly, Joseph S. Sherif, and Jonathan Hops. An analysis of defect densities found during software inspecitons. In *SEL Workshop Number 15*, Goddard Space Flight Center, Greenbelt, MD, nov 1990.

[11] John C. Knight and E. Ann Myers. An improved inspection technique. *Communications of the ACM*, 36(11):51–61, November 1993.

[12] Dave L. Parnas and David M. Weiss. Active design reviews: principles and practices. In *Proceedings of the 8th International Conference on Software Engineering*, pages 215–222, Aug. 1985.

[13] Adam A. Porter, Harvey Siy, Carol Toman, and Lawrence G. Votta. An experiment to assess the cost-benefits of code inspection in large scale software development. Technical Report CS-TR-3410, University of Maryland, College Park, MD.

[14] Adam A. Porter and Lawrence G. Votta. An experiment to assess different defect detection methods for software requirements inspections. In *Sixteenth International Conference on Software Engineering*, Sorrento, Italy, May 1994.

[15] Glen W. Russel. Experience with inspections in ultralarge-scale developments. *IEEE Software*, 8(1):25–31, January 1991.

[16] G. Michael Schnieder, Johnny Martin, and W. T. Tsai. An experimental study of fault detection in user requirements. *ACM Trans. on Software Engineering and Methodology*, 1(2):188–204, April 1992.

[17] T. A. Thayer, M. Lipow, and E. C. Nelson. *Software reliability, a study of large project reality*, volume 2 of *TRW series of Software Technology*. North-Holland, Amsterdam, 1978.

[18] Lawrence G. Votta. Does every inspection need a meeting? In *Proceedings of ACM SIGSOFT '93 Symposium on Foundations of Software Engineering*, pages 107–114. Association for Computing Machinery, December 1993.

[19] Alexander L. Wolf and David S. Rosenblum. A study in software process data capture and analysis. In *Proceedings of the Second International Conference on Software Process*, pages 115–124, February 1993.

[20] E. Yourdon. *Structured Walkthroughs*. Prentice-Hall, Englewood, NJ, 1979.