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

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

C. A. Toman
Global Data Development Department
L. G. Votta
Software Production Research Department
Bell Laboratories, Lucent Technologies
Naperville, Illinois 60566
ctoman@lucent.com
votta@research.bell-labs.com

May 16, 1997

Abstract

We conducted a long-term experiment to compare the costs and benefits of several different software inspection methods. These methods were applied by professional developers to a commercial software product they were creating. Because the laboratory for this experiment was 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 results and discuss their implications for both software practitioners and researchers, and (3) to discuss several new questions raised by our findings.

For each inspection we randomly assigned 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 were randomly selected without replacement from a pool of 11 experienced software developers. The dependent variables for each inspection included inspection interval (elapsed time), total effort, and the defect detection rate.

Our results showed that these treatments did not significantly influence the defect detection effectiveness, but that certain combinations of changes dramatically increased the inspection interval.

Keywords: Software Inspection, Controlled Experiments, Industrial Experimentation, ANOVA, Power Analysis.

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 argument is simplistic – for example, it doesn't consider the effect inspections have on schedule[23].

We have observed that a typical release of Lucent Technologies' 5ESS® switch[15] ($\approx$ .5M lines of added and changed code per release on a base of 5M lines) can require roughly 1500 inspections, each with four, five or even 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 complete development).[1] Since long development intervals risk substantial economic penalties, this hidden cost must be considered.

*This work is supported in part by a National Science Foundation Faculty Early Career Development Award, CCR-9501354. Mr. Siy was also partly supported by AT&T 's Summer Employment Program

[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[1].

We hypothesized that different inspection approaches create different tradeoffs between minimum interval, minimum effort and maximum effectiveness. But until now there have been no controlled experiments to identify the mechanisms that drive these tradeoffs. We conducted 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 examined, and present our experimental design, analysis, and conclusions.

1.1 Inspection Process Summary and Literature Review

To eliminate defects, many organizations use an iterative, three-step inspection procedure: Preparation, Collection, Repair[11]. 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.

The research literature shows that several variants of this approach have been proposed in order to improve inspection performance. These include Fagan Inspections[8], Active Design Reviews[16], N-Fold Inspections[19], Phased Inspections[13], and Two-Person Inspections[2].

Each of these is created by restructuring the basic process, e.g., rearranging the steps, changing the number of people working on each step, or the number of times each step is executed. Several also require the use of special defect detection methods. Although some of these variants have been evaluated empirically, the focus has been on their overall performance. Very few investigations have tried to isolate the effects of specific structural changes. We believe that we must know which changes cause which effects in order to determine the factors that drive inspection performance, to understand why one approach may be better than another, and to focus future research on high-payoff areas.

Team Size: Inspections are usually carried out by a team of four to six reviewers. Buck[3] 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 (calendar time to complete inspection).

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.[20] 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)[16] and Knight and Myers' phased inspections (PI)[14] 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[9]. 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[10] 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[23] found far lower meeting gain rates (about 5%). Porter et al.[18] 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.

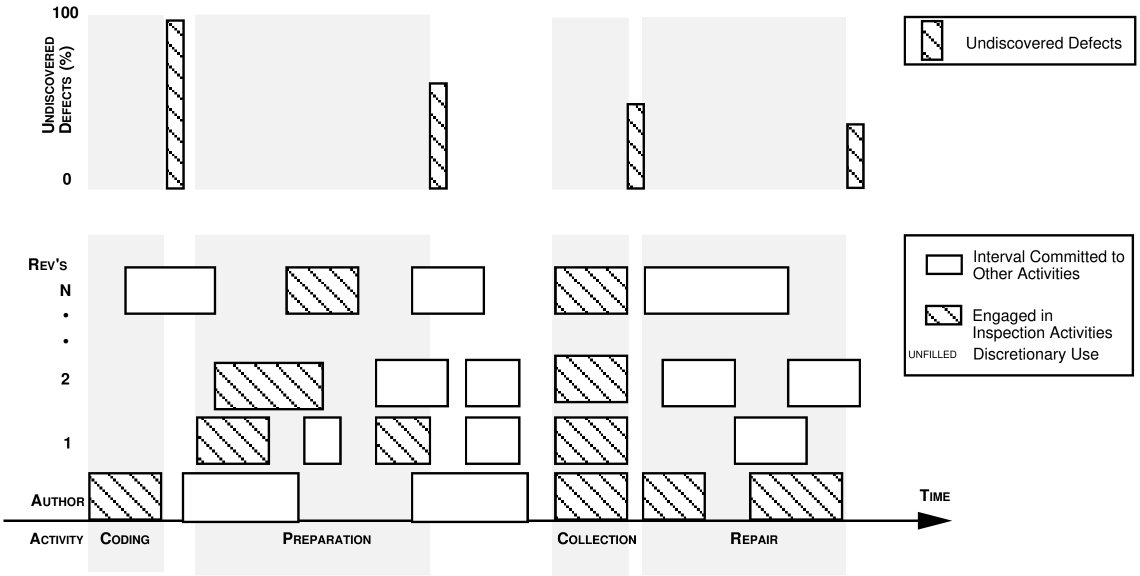

Figure 1: This figure depicts how inspection participants use time during the inspection process. The figure's lower panel summarizes the inspection's time usage. Specifically, it depicts the inspection's participants (an author and several reviewers), the activities they perform (coding, preparation, collection, repair, and other), the subinterval devoted to each activity (denoted by the shaded areas), and the total inspection interval (end of coding to completion of repair). It also suggests that in a software development organization, inspections must compete with other processes for limited time and resources. The upper portion of the figure suggests when and to what extent inspections remove defects from the code.

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.

2 The Experiment

2.1 Hypotheses

Inspection approaches are usually evaluated according to the number of defects they find. As a result, information has been collected about the effectiveness of different approaches, but far less 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

H1: Inspections with large teams have longer inspection intervals, but find no more defects than smaller teams.

H2: Multiple-session inspections are more effective than single-session inspections, but significantly increase inspection interval.

H3: Multiple-session inspections with sequential sessions (sessions happen in a specific order and all defects found at the i^{th} session must be repaired before the $i+1^{st}$ session begins) have a longer interval, but find more defects than multiple-session inspections with parallel sessions (sessions can happen in any order and defects are not repaired in between sessions).

2.2 Experimental Setting

We ran this experiment at Lucent Technologies (formerly AT&T Bell Labs) on a project developing a compiler and environment to support developers of the 5ESS® telephone switching system. The finished system contains over 55K new lines of C++ code, plus 10K which was reused from a prototype. Reused code was not inspected, which is the standard procedure in this organization.

The inspector pool consisted of the 6 developers building the compiler plus 5 developers working on other projects.[2] They had all been with the organization for at least 5 years, had similar development backgrounds, and had received inspection training within the 5 years prior to the experiment. However, most of their previous development efforts used the C programming language. Thus, few of them were highly experienced with C++. We collected data over a period of 18 months (June 1994 to December 1995), during which 88 code inspections were performed.

The first code units were inspected from July, 1994 to September, 1994, at which time the first integration build delivered the compiler's front end. After this there were few inspections as the development team tested and modified the front end and continued designing the back end. By Jan 1995, the back end code became available and there was a steady stream of inspections throughout 1995.

2.3 Operational Model

To test our hypotheses we needed to measure the effort, interval and effectiveness of every inspection. To do this we constructed two models; one for calculating inspection interval and effort, and another for estimating the number of defects in a code unit. These models are depicted in Figure 1.

2.3.1 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[24]. Whenever one of these events occurred it was timestamped and the event's participants were recorded. (In most cases this information was manually recorded on the forms described in Section 2.5.1.) These events occurred, for example, when code was ready for inspection, or when a reviewer started or finished his or her preparation. This information was entered into a database, and inspection intervals were calculated as the calendar time between two specific events. This includes time that is not spent on inspection activities. Inspection effort was calculated by summing only the calendar time attributed to inspection activities.

[2] In addition, 6 more developers were called in at one time or another to help inspect 1 or 2 pieces of code, mostly to relieve the regular pool during the peak development periods. It is common practice to get non-project developers to inspect code during peak periods.

2.3.2 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 was impossible to make this measurement directly, and therefore we were forced to approximate it.

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

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

- **Observed defect density:** We assumed that total defect density is constant for all code units and that we could compare the number of defects found per KNCSL (thousand non-commentary source lines). This was always available, but is inaccurate.
- **Partial estimation of detection ratio:** We tried capture-recapture methods to estimate pre-inspection defect content [4, 6, 7, 17, 22]. 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 defect density and is available immediately after every inspection. Since capture-recapture techniques make strong statistical assumptions, we tested our data to see whether or not this technique would be appropriate. We found that this method was inappropriate for our study. For example, inspectors often found completely disjoint sets of defects. Therefore we did not use it in our analysis.
- **Complete estimation of detection ratio:** We can 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. We are currently instrumenting the development process to capture this data, but it will not be available for some time. And even then, there may still be defects left undiscovered. In addition, it may be extremely difficult to determine whether additional defects found were due to mistakes in implementing the original requirements or in subsequent customer-requested enhancements.

2.4 Experimental Design

2.4.1 Variables

The experiment manipulated 3 independent variables:

1. the number of reviewers per team (1, 2, or 4 reviewers, in addition to the author),
2. the number of inspection sessions (1-session or 2-sessions),
3. the coordination between sessions (in 2-session inspections the author was either required to or prohibited from repairing 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 <u>not</u> captured in our experiment is the choice of defect detection methods. (See Section 1.1 on Defect Detection Methods.) The methods used in Active Design Reviews and Phased Inspections involve systematic techniques, with specific and distinct responsibilities, while those used in Fagan and N-fold Inspection are normally nonsystematic techniques with general and identical responsibilities.

The treatments are arrived at by randomly selecting a value for each of the independent variables. The selection probabilities were weighted as shown in Table 1. These probabilities changed during the experiment because we discontinued some of the poorly performing or excessively expensive treatments. In this article particular treatments are denoted [<u>1</u>,or <u>2</u>] <u>s</u>essions <u>X</u> [<u>1</u>, <u>2</u>, or <u>4</u>] <u>p</u>ersons [<u>N</u>o-repair, <u>R</u>epair], so, for example, the label 2sX1pN indicates a two-session, one-person, without-repair inspection. The 1sX4p treatment is the standard inspection process in this software organization.

There are many ways in which inspections might be beneficial. The most obvious is defect detection. However, they may also help development teams share information quickly, train inexperienced personnel, etc. In this article, however, we restrict our attention to defect detection only. Consequently, we measured 5 dependent variables for each inspection.

	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 originally allocated to each treatment. These proportions changed during the experiment's execution because several poorly performing treatments were discontinued.

1. inspection interval,
2. inspection effort,
3. observed defect density (defects/KNCSL, see Section 2.3.2),
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 were determined not to be defects during the collection meeting (meeting suppression rate).

We also captured repair statistics for every defect (See Section 2.5.2). This information was 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.

2.4.2 Design

This experiment used 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).

2.4.3 Professional Developers as Subjects

We took special care to ensure that the experimental design did not inadvertently influence subject behavior (professional developers and inspectors). Each study participant was given a simple "bill of rights", reminding them of their right to withdraw from the study at anytime with no recriminations from the researchers or his/her management[12]. Each participant acknowledged this right at the beginning of the experiment by signing a release form. No subject used this right during the experiment.

2.4.4 Discontinuing Ineffective Treatments

In our initial briefings with the development team, we were asked, "What happens if a treatment cost too much or takes too long?" They were concerned that the experiment could jeopardize the budget or schedule of the product.

We took this concern seriously and realized that if a treatment was jeopardizing the project's budget, schedule, or quality, we would have to discontinue the treatment. However, the professional developers also realized that they were gaining some valuable knowledge from the study. So before we began the experiment we agreed to discontinue any treatment after enough inspections had been done to determine that the treatment was ineffective. We used simulation techniques to help determine the number of inspections we would need. See Appendix A for more details.

This specific problem of knowing when to stop experimenting is important for software engineering researchers. Experiments that use professional developers who are creating professional products are desirable for their strong external validity, but can put the participating project at risk. A similar problem confronts medical researchers when assessing the efficacy of drug treatments for diseases[12]. They solve the problem like we did through an agreement with their subjects in the study.

In the course of the experiment, several treatments were discontinued because they were either ineffective (1sX1p treatment), or because they were taking too long to complete (all 2-session treatments which required repair between sessions). See Appendix B for more details.

2.4.5 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 were spread across all treatments.

Maturation effects result because participants' skills improve 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. One important implication of this is that our analysis assumes that each treatment is applied to a uniform set of code units.

2.4.6 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 and artifact representativeness.

Experimental scale is 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 and artifact representativeness arise when the subject and artifact 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 and our artifacts are not representative of every type of software professional developers write.

2.4.7 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 (observed/total) 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 Appendix A for more details.)

Calibration. We continuously calibrated the experiment by monitoring the sample mean and variance of each treatment's detection ratio and inspection interval, and the number of observed inspections. (These data are the parameters of the simulation results described above.) After cross-referencing this information with the resolution analysis, we discontinued some treatments because their effectiveness was so low or their interval was so long that it put the project at risk. We also monitored the experiment to ensure that the distribution of treatments did not produce too few data points to identify statistically significant performance differences.[3]

[3] 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.

Hypothesis Testing. Once the data was collected we analyzed the combined effect of the independent variables on the dependent variables to evaluate our hypotheses. Once the significant explanatory variables were discovered and their magnitude estimated, we examined relevant subsets of the data to study specific hypotheses.

2.5 Experimental instrumentation

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

2.5.1 Data Collection Forms

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

The meeting form was 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 was filled in during both preparation and collection. During preparation, the reviewer recorded 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 decided which of the reviewer's issues were, in fact, real defects. At that time, real defects were recorded on the meeting form and given an ID. If a reviewer had discovered this defect during preparation, they recorded this ID on their preparation form.

2.5.2 Author Repair Forms

The author repair form captured information about each defect identified during the inspection. This information included 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).

2.5.3 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.

2.6 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 was responsible for tracking the experiment's progress, capturing and validating data, and observing all inspections. He also attended the development team's meetings, but had no development responsibilities.

When a code unit (a logical unit of code, on average about 300 LOC) has compiled successfully without warning, its author sent an inspection request to the IQE. He then randomly assigned a treatment (based on the treatment distributions given in Table 1) and randomly drew the review team from the reviewer pool.[4] These names were then given to the author, who scheduled the collection meeting. Once the meeting was scheduled, the IQE put together the team's inspection packets.[5]

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 the code. (Often this involves reading several lines of code at a time and emphasizing their function or purpose.) During

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

[5] 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. In addition, the inspectors have access to the appropriate design documents.

this activity, reviewers may bring up any issues found during preparation or discuss new issues. One reviewer acts as the moderator. This person runs the meeting and makes sure all required changes are made. The code unit's author compiles the master list of all defects and no other reviewer has a predefined role.

The IQE attended 125 of 130 collection meetings[6] to make sure the meeting data was reported accurately and that reviewers do not mistakenly add to their preparation forms any issues that were not found until collection. He also took extensive field notes to corroborate and supplement some of the data in the meeting forms. After the collection meeting he gave the preparation forms to the author, who then repaired the defects, filled out the author repair form, and returned all forms to him. After the forms were returned, he interviewed the author to validate any questionable 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. 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.

3.1 Significance Testing

Because most of the data have asymmetric (i.e., non-normal) distributions, significance tests that assume normality (e.g., t-test) may be unreliable. Thus we use nonparametric techniques which only require that the distribution be randomly sampled[21].

This decision has several implications. We considered two data distributions to be significantly different only if the Wilcoxon rank sum test[21] rejects the null hypothesis that the observations are drawn from the same population with a confidence level $\geq .9$.

Medians, not means, summarize the data. Therefore, care must be used when making inferences from the data presented in this article. In particular, since medians are ordinal, the median of two distributions is not equal to the sum of the medians of each distribution. Readers who wish to conduct their own significance tests can find the experiment's data at `http://www.cs.umd.edu/users/harvey/experiment`.

Note also the extensive use of boxplots (e.g., Figure 3) 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 data (1.5 times the inter-quartile range). All other detached points are "outliers"[5].

Finally, for expository convenience, we say that two distributions are "different" **only** if they are significantly different.

3.2 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.

[6] The unattended ones are due to schedule conflicts and illness.

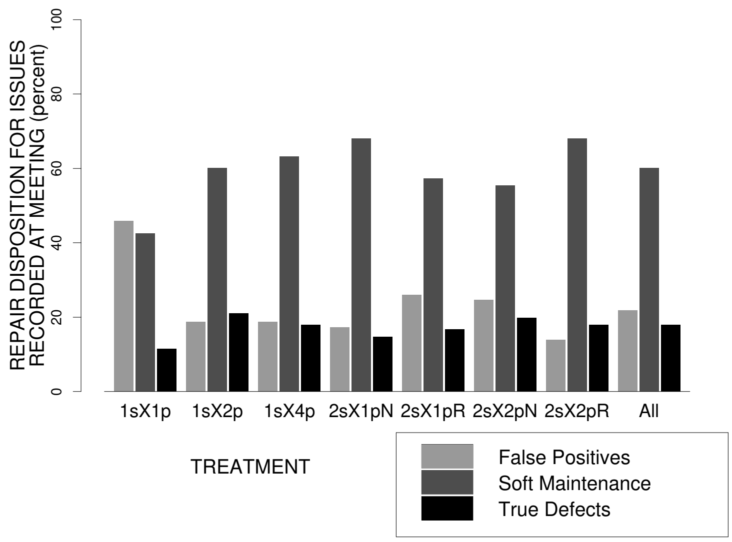

Figure 2: **Disposition of Issues Recorded at the Collection Meeting.** For each treatment, the barchart shows the percentage of the issues recorded at collection meetings that turn out to be false positives, soft maintenance, or true defects. Across all treatments, only 18% of the issues are true defects.

3.2.1 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 issues from 2-session-without-repair inspections. This task is performed by the IQE and the code unit's author.

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 2. Across all inspections, 22% of the issues are False Positives, 60% involve Soft Maintenance, and 18% are True Defects. We consider only True Defects in our analysis of estimated defect detection ratio (a dependent variable). We made this second reduction because we observed that most of the soft maintenance issues are caused by conflicts between different reviewers about the coding style or conventions used. Since, in and of themselves, these are not true defects, some reviewers never reported them while others always did. In contrast, true defects have a clear definition, something that would cause proper execution to fail, and reviewers always reported them. We do not mean to imply that soft maintenance issues are unimportant, only that for this study we are restricting our focus to true defects.

3.2.2 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

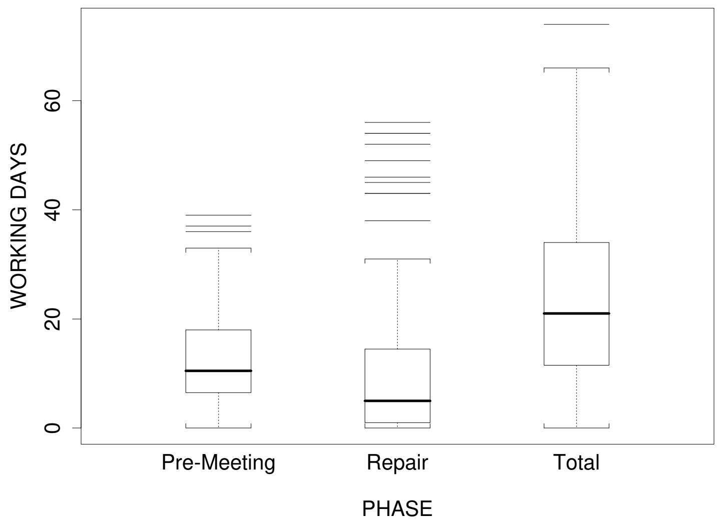

Figure 3: **Pre-meeting Inspection Interval.** These boxplots show all the interval data divided into two parts: time before the meeting and time after the meeting. The median inspection interval is 21 days.

	Number of Sessions			Totals
	1	2		
Team Size		With Repair	No Repair	
1	7	5	18	30
2	26	4	15	45
4	13	0	0	13
Totals	46	9	33	88

Table 2: This table shows the number of inspections allocated to each treatment.

the defects later, during slow work periods.[7] This happened regardless of treatment used, obscuring the effect any treatment may have on the inspection interval. Therefore 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 3 is a boxplot showing the number of working days from the issuance of the inspection request to the collection meeting (Pre-Meeting), from the collection meeting to the completion of repair (Repair), and the total (Total). The total inspection interval has a median of 21 working days.

3.3 Overview of Data

Table 2 shows the number of observations for each treatment. Figure 4 is a contrast plot showing the interval, effort, 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

[7] This interpretation is supported in Figure 3 by the larger number of outliers in the repair interval boxplot.

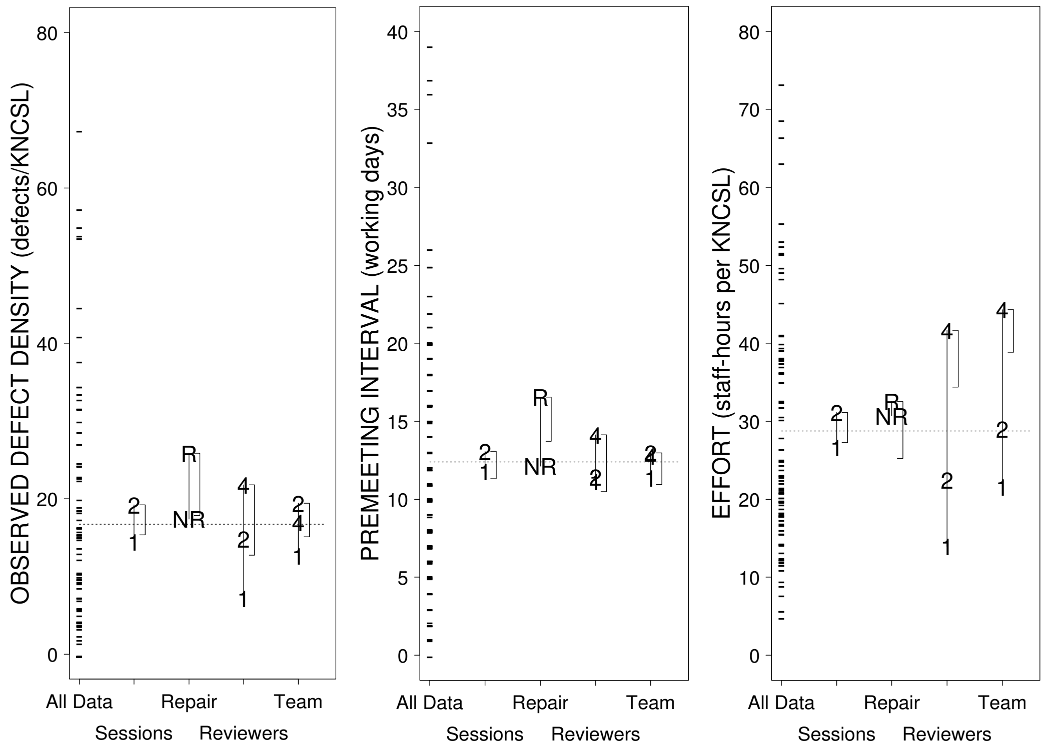

Figure 4: **Effectiveness, Interval, and Effort by Independent Variables.** The dashes in the far left column of the first plot show the observed defect densities for all inspections. The dotted horizontal line marks the average observed defect density. The other four columns indicate factors that may influence this dependent variable. The plot demonstrates the ability of each factor to explain variations in the dependent variable. For the Repair factor, the vertical locations of the symbols "N" and "NR" are determined by averaging the defect detection rates for all code inspections using 2-sessions with repair and 2-sessions without-repair. 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 middle and right panels show similar information for premeeting interval and effort.

by the number of sessions). This variable provides information about the relative influence of team size vs. number of sessions.

3.4 Defect Discovery by Inspection Phase

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 issues, and (2) finding new defects. In this section we analyze how defect discovery is distributed across the preparation and collection meeting activities.

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 5 shows the percentage of defects reported by each reviewer that are eventually determined to be true defects. We can find no clear relationship between treatment and preparation effectiveness. Across all 233 preparation reports, only 13% of all issues turn out to be true defects.

Analysis of Suppression. It is generally assumed that collection meetings suppress unimportant or incorrect issues, and that without these meetings, authors would have to process many spurious issues during repair. As we deduce from the previous section an average of 87% of reviewer issues (100% - 13%) do not involve true defects.

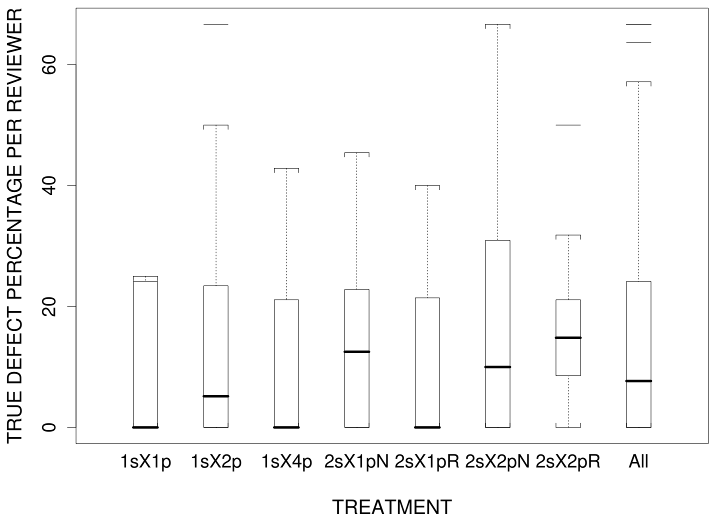

Figure 5: **Percentage of True Defects in Reviewer Preparation Forms by Treatment.** This boxplot shows the percentage of issues found during preparation that are eventually considered to be true defects. Across all inspections, an average of only 13% of the issues turn out to be true defects.

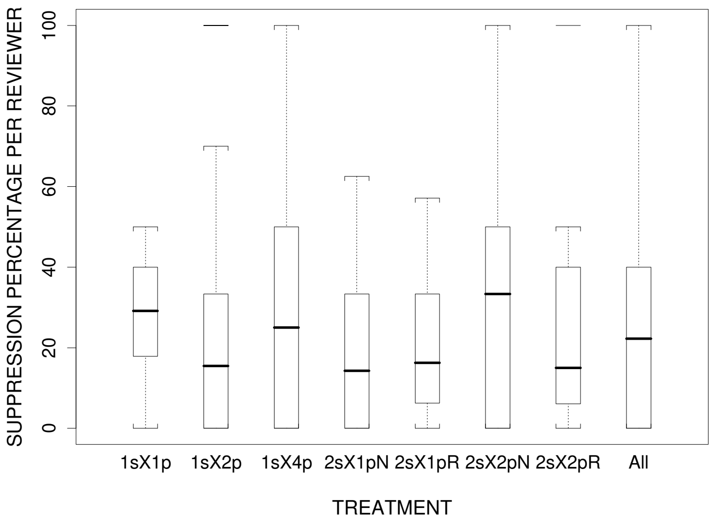

Figure 6: **Meeting Suppression Percentage by Treatment.** These boxplots show the suppression percentage for each reviewer by treatment. This is the number of defects detected during preparation but not included in the collection meeting defect report, divided by the total number of defects recorded by the reviewer in his/her preparation. Across all inspections, an average of 26% of the issues are suppressed.

Figure 6 shows the percentage of issues suppressed for all 233 reviewer reports. Across all inspections about 26% of issues are suppressed. This 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.

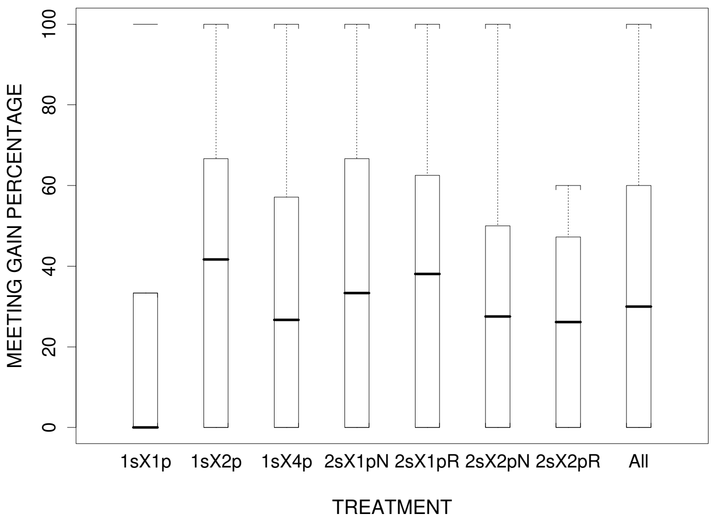

Figure 7: **Meeting Gain Percentage by Treatment.** These boxplots shows the percentage of defects discovered at the meeting for all inspections and for each treatment. The median is 30%.

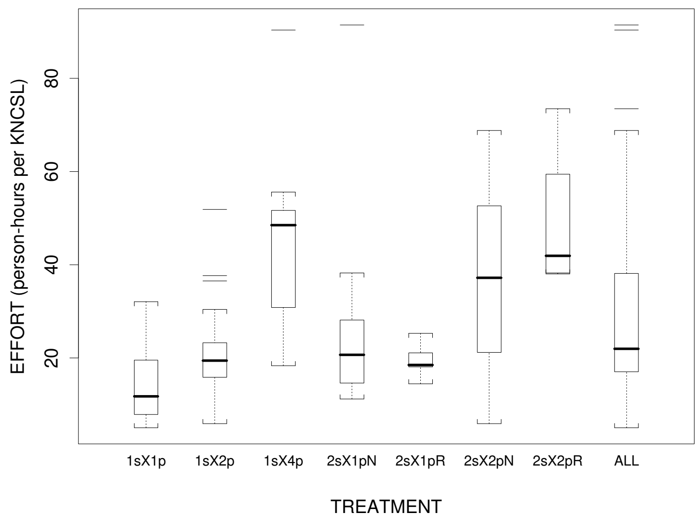

Figure 8: **Total Inspection Effort by Treatment.** This plot shows the total inspection effort per KNCSL for each treatment. Across all treatments, the median effort is 22 person-hours per KNCSL.

Figure 7 shows the meeting gain percentages for all 130 collection meetings. Across all inspections, 30% of all defects discovered are meeting gains. The data suggests that, except for the 1sX1p treatment, meeting gains are independent of treatment.

3.5 Analysis of Effort Data

The common measure of inspection cost is total effort – the number of hours spent in preparation and meeting by each reviewer and author. Figure 8 shows the effort spent per KNCSL for each inspection by treatment and for all treatments. Across all treatments, the median effort is about 22 person-hours per KNCSL.

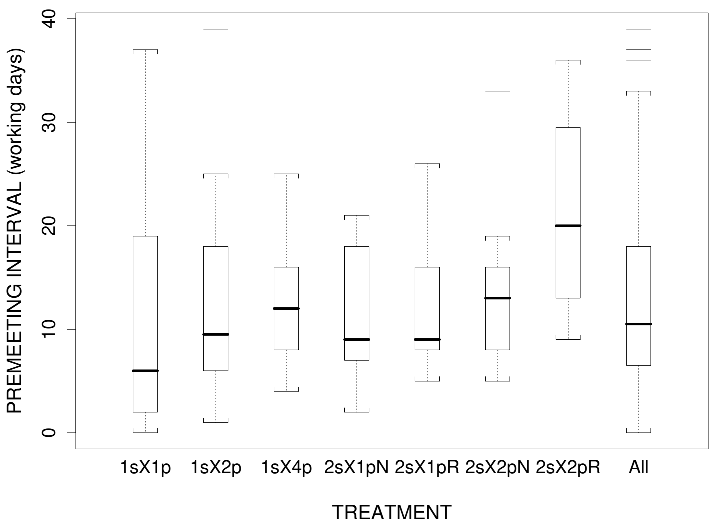

Figure 9: **Pre-meeting Interval by Treatment.** This plot shows the observed pre-meeting interval for each inspection treatment. Across all treatments, the median interval is 10.5 days.

The data suggest that effort increases with the total number of reviewers while the number of sessions and the repair between sessions have no effect. That is, inspections involving 4 reviewers (1sX4p, 2sX2pN, and 2sX2pR) required significantly more effort than inspections involving 2 reviewers. Likewise, inspections involving 2 reviewers (1sX2p, 2sX1pN, and 2sX1pR) required significantly more effort than inspections involving 1 reviewer.

3.6 Analysis of Interval Data

Inspection interval is another important, but often overlooked cost. Figure 9 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 between the intervals, team size alone did not affect interval.

The additional cost of multiple inspection sessions can be seen by comparing 1-session inspections with 2-session inspections (1sX2p and 1sX1p with 2sX2p and 2sX1p inspections). We find that 2sX1p inspections didn't take longer to conduct than 1sX1p inspections, but that 2sX2p inspections took longer to complete than 1sX2p inspections. (This effect is caused solely by the 2sX2pR treatment, since there was no difference between 1sX2p and 2sX2pN 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 had only 1 reviewer we found no difference in interval, however, we did see a difference for 2-reviewer teams. This suggests that requiring repair between sessions only increases interval as the team size grows.

Another interesting observation is that the median interval for the 2sX2pR treatment is extremely long (20 days), while all others have a median of only 10 days. Since this treatment took much longer to complete than did the others we discontinued it early in the experiment. Consequently, we conducted only four of these inspections. Nevertheless, we are convinced that this finding warrants further study, because it suggests that relatively straightforward changes to a process can have dramatic, negative effects on interval.

3.7 Analysis of Effectiveness Data

The primary benefit of inspections is that they find defects. This benefit varied with different inspection treatments. Figure 10 shows the observed defect density for all inspections and for each treatment separately.

The effect of increasing team size is suggested by comparing the effectiveness of all 1-session inspections (1sX1p, 1sX2p, and 1sX4p inspections). There was no difference between 2- and 4-person inspections, but both

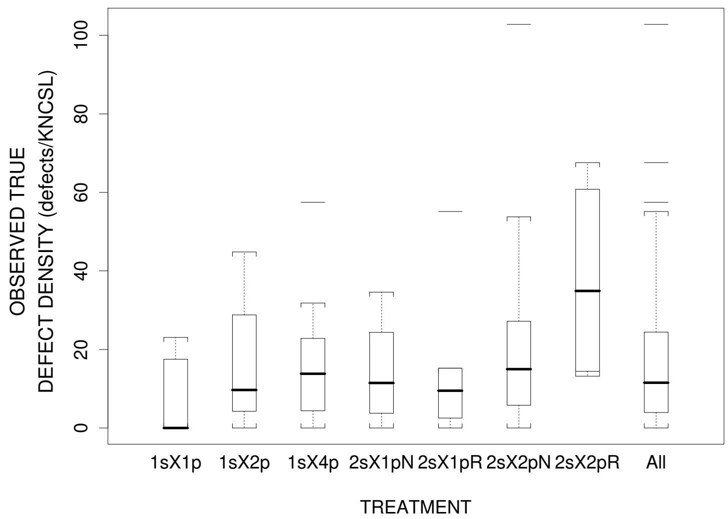

Figure 10: **Observed Defect Density by Treatment.** This plot shows the observed defect density for each inspection treatment. Across all inspections, the median defect detection rate was 12 defects per KNCSL.

performed better than 1-person inspections.

The effect of multiple sessions is suggested by comparing 1-session inspections with 2-session inspections. When team size is held constant (1sX2p vs. 2sX2p and 1sX1p vs. 2sX1p inspections), 2-session inspections were more effective than 1-session inspection only for 1-person teams. However, when total number of reviewers is held constant (1sX2p vs. 2sX1p and 1sX4p vs. 2sX2p) there were no differences in effectiveness.

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 didn't increase effectiveness when the team size was one, but did when the team size was two. This result should be viewed with caution, however, because there are only four 2sX2pR and five 2sX1pR inspections, respectively. Also, during the time in which the with-repair treatments were used they performed no differently than did without-repair treatments, and furthermore the overall mean dropped steadily as the experiment progressed possibly exaggerating the differences between the 2sX2pR and 2sX2pN treatments. (See Appendix B for more details.)

We draw several observations from this data: (1) increasing the number of reviewers did not necessarily lead to increased defect discovery, (2) splitting one large team into two smaller teams did not increase effectiveness, and (3) repairing defects in between 2-session inspections doesn't guarantee increased effectiveness.

4 Low Level Analysis

Several software inspection researchers have proposed changes to the structure of the process, hoping to improve its performance. For example, some researchers claimed that large teams bring a wide diversity of expertise to an inspection, and, therefore find more defects than smaller teams. But others believed that smaller teams are better because they minimize the inefficiencies of large team meetings. Some argued further that multiple sessions with small teams are more effective than a single session with a larger team because the small teams are nearly as effective as large ones, won't duplicate each other's effort and have more effective collection meetings. Finally, some authors told us that repairing defects in between multiple sessions would be more effective than not repairing because repair improves the ability of the second team to find defects.

Our initial analysis suggests, however, that many of these changes did not have the hypothesized effect on observed defect density. For example,

- Increasing team size does not always improve performance. (1sX1p < 1sX2p, but 1sX2p = 1sX4p),

Defect ID	1	2	3	4	5	6	7	8	9	10	...	N	Number Found
R_1	$\surd$		$\surd$							$\surd$	...		p_1
R_2		$\surd$	$\surd$		X			$\surd$			...		p_2
R_3								$\surd$			...	$\surd$	p_3
$\vdots$													$\vdots$
R_n						$\surd$				$\surd$	...	$\surd$	p_n
$Preparation$	$\surd$	$\surd$	$\surd$			$\surd$		$\surd$		$\surd$	...	$\surd$	P
$Meeting$				$\surd$			$\surd$				...		M
$TotalDefects$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$		$\surd$	...	$\surd$	$D = P + M$

Figure 11: **A Defect Detection Model.** During preparation, reviewer R_i finds p_i defects. Each $\surd$ mark in row R_i indicates one of these defects. Each X mark indicates a defect that was found by R_i, but was suppressed at the meeting. The row labeled $Preparation$ contains one $\surd$ mark for each defect that found by at least one reviewer during preparation and the M defects found at the meeting are indicated by a $\surd$ mark in the row labeled $Meeting$. Finally, the row labeled $TotalDefects$ contains a $\surd$ mark for each of the D defects that are known to the artifact's author at the end of the inspection.

- Creating two smaller teams is not an effective way to reorganize a large group. (2sX2p = 1sX4p and 2sX1p = 1sX2p), and
- Repairing defects between sessions does not guarantee improved inspection performance. (2sX2pR = 2sX2pN[8] and 2sX1pR = 2sX1pN).

One possible explanation is that the assumptions driving inspection process changes didn't hold in practice. (e.g., that repairing defects between multiple sessions didn't improve the ability of the second team to find defects.) Another possible explanation is that the treatments had unintended, negative side effects (i.e., the treatment improved some aspect of the inspection while degrading another).

To evaluate these potential explanations we examined the effect of each treatment on several inspection sub-activities.

4.1 Modeling Defect Detection in an Inspection Artifact

First, we have developed a model to measure defect discovery in each inspection subtask. The model, shown in Figure 11, assumes that the inspection artifact contains N undiscovered defects. Each reviewer, R_i, finds some number of defects, p_i, during preparation. Some number of these, $common$, may be found by more than one reviewer so the number of unique defects found in preparation, P, may be less than $\sum p_i$. Some number of additional defects, M, may found at the meeting. During the meeting some of the defects found in preparation may be suppressed (determined not to be defects). These are called meeting losses.

Although we don't know how many true defects are suppressed, we will assume the number to be small and will, therefore, ignore meeting losses for now. Given this assumption, the number of defects found in one inspection, D, is just $P + M$ and the observed defect density is $\frac{D}{NCSL}$, where $NCSL$ is the number of noncommentary lines of code in the artifact.

Using this model and the data from our experiment we can calculate several statistics.

1. the average number of defects found by individual reviewers during preparation: $\bar{p_i}$,
2. the number of unique preparation defects: P,
3. the number of defects found by more than one reviewer during preparation: $common$,
4. the overlap in preparation defects: $\frac{common}{P}$, and
5. meeting gains: M.

[8] Comparing only the inspections that occurred while the 2sX2pR treatment was being used.

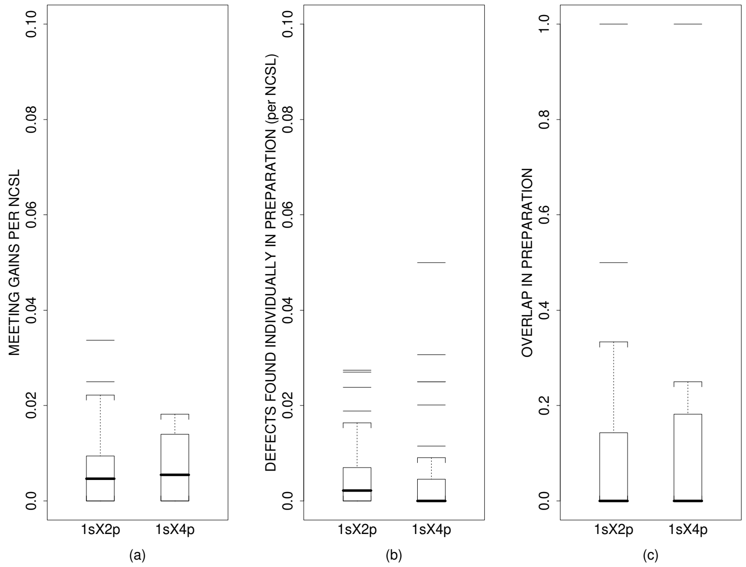

Figure 12: **Effect of Team Size on Inspection Subtasks.** (a) meeting gains, (b) mean individual preparation performance, and (c) overlap of defects found in preparation.

Our goal in this analysis is to determine whether treatments with similar inspection performances show significant differences in these lower-level activities. For example, if one treatment has higher preparation defect densities ($\frac{P}{NCSL}$) than another, but the same observed defect densities, then we'd expect to find worse performance in some other subtasks, (e.g., lower meeting gain densities ($\frac{M}{NCSL}$)).

4.2 Large Teams vs. Small Teams

As long as additional reviewers find some new defects and don't negatively affect collection meeting performance, we would expect larger teams to find more defects than smaller teams, yet we found that 1sX2p inspections performed the same as 1sX4p inspections. Somewhere the supposed advantage of having more reviewers didn't materialize, so we investigated how team size affected both preparation and meeting performance.

First, we investigated two aspects of preparation performance: individual preparation and amount of overlap in the defects found by the reviewers.

Figure12(b) shows the number of defects per NCSL found in preparation by reviewers in 1sX2p and 1sX4p inspections, $\frac{p_i}{NCSL}$. There was no difference between the two treatments.

Then we examined the amount of overlap in the reviewer's defect reports. This is the number of defects found by more than one reviewer divided by the total number found in preparation, $\frac{common}{P}$. There was no difference in overlap between 1sX2p and 1sX4p inspections and both distributions had a median of 0. (See Figure 12(c)).

Next we examined two aspects of meeting performance: defect suppression and meeting gains. We found that defect suppression rates were higher for 1sX4p than for 1sX2p inspections. (See Figure 6).

Finally, Figure 12(a)) shows that there is no difference in the meeting gains per NCSL, $\frac{M}{NCSL}$, for 1sX2p and 1sX4p inspections.

Our interpretation of these results is that larger teams don't improve inspection performance because meeting gains do not increase as the number of reviewers increases, and because larger teams may suppress a large number of (possibly true?) defects.

4.3 One Large Team vs. Two Small Teams

Another recommendation that has appeared in the literature is to substitute several small (1- or 2-person) teams for one larger team. This approach should be more effective if the combined defect detection of the smaller teams is greater than that of the single larger team, and if the small teams don't significantly duplicate each other's efforts.

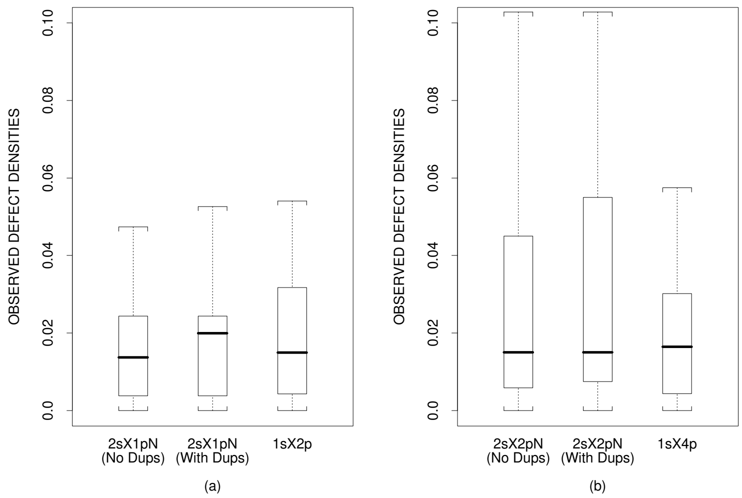

Figure 13: **The Effect of Splitting One Large Team.** This figure compares the distribution of observed defect densities of 2-session inspections before (Dups) and after (No Dups) accounting for overlap with that of 1-session inspections.

Nevertheless we saw that 2sX1p (2sX2p) inspections did not perform better than 1sX2p (1sX4p) inspections. To investigate this, we compared the distribution of observed defect densities for 1-session inspections with the sum of the defect densities found in both sessions of the 2-session inspections (defects found by both teams are counted twice). We found that the combined defect densities of 2sX1p (2sX2p) inspections are not greater than the defect densities of 1sX2p (1sX4p) inspections. (Compare the second and third boxplots in Figures 13(a) and (b).) We also found that there was effectively no overlap in the defects found by the two sessions. (Compare the first and second boxplots in Figures 13(a) and (b).)

This data suggests that for our experimental setting overlap among reviewers is a rare occurrence, but that splitting teams did not improve performance because the two smaller teams found no more defects than the one larger team.

4.4 Repair vs. No Repair

Repairing defects between sessions of a multiple session inspection should result in greater defect detection than not repairing if (1) the teams in the with-repair inspections perform as well as the teams in the without-repair inspections, (2) there are significantly more defects than one team can find alone, and (3) the teams doing without-repair inspection find many of the same defects.

However, we saw that during the period in which with-repair inspections were conducted they did not perform better than without-repair inspections. One or more of the assumptions may have been violated. To investigate this, we extended our inspection model with a second session such that D_1 and D_2 are the number of defects found in the first and second session, respectively.

To test whether with-repair teams perform as well as without-repair teams we compared defect densities per session, D_1 and D_2, of with-repair inspections with those of without-repair inspections. We found no differences in the performances (see Figures 14(a) and 15(a)), suggesting that the with-repair teams perform no differently than without-repair teams.

To test whether there are enough defects to warrant two inspection teams we compared the performance of with-repair teams inspecting the same unit. If the second team (inspecting after the repair) consistently found fewer defects than the first team, (i.e., $D_1 - D_2$ is significantly higher than 0), then the first team may have found most of the defects that can be found with current inspection techniques. If not, this suggests that that there are more than enough defects to be found by two teams, and that on the average, one team is as good as the other. We found that the number of defects found by the second team of 2sX1pR inspections are generally

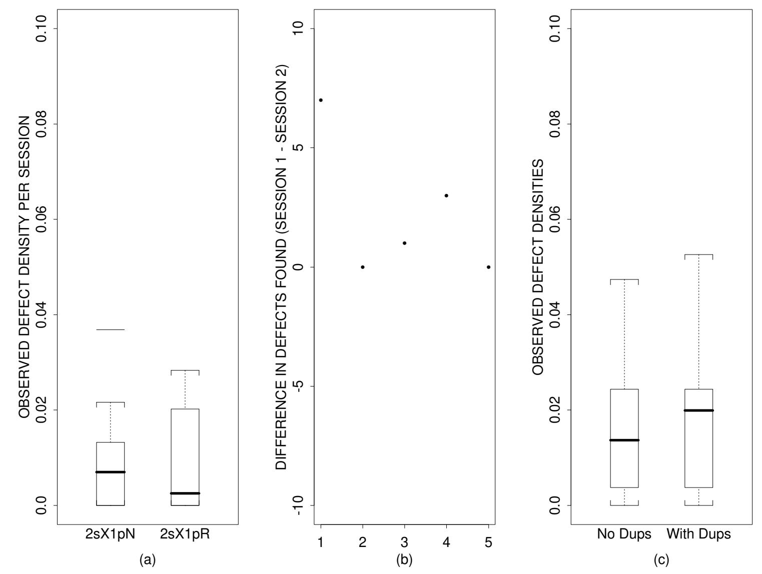

Figure 14: **Effect of Repairing In Between Sessions for 1-Reviewer Teams.** (a) comparing session performance with same team size, (b) difference in number of defects found between session 1 and session 2, and (c) counting the duplicates for 2sX1pN inspections.

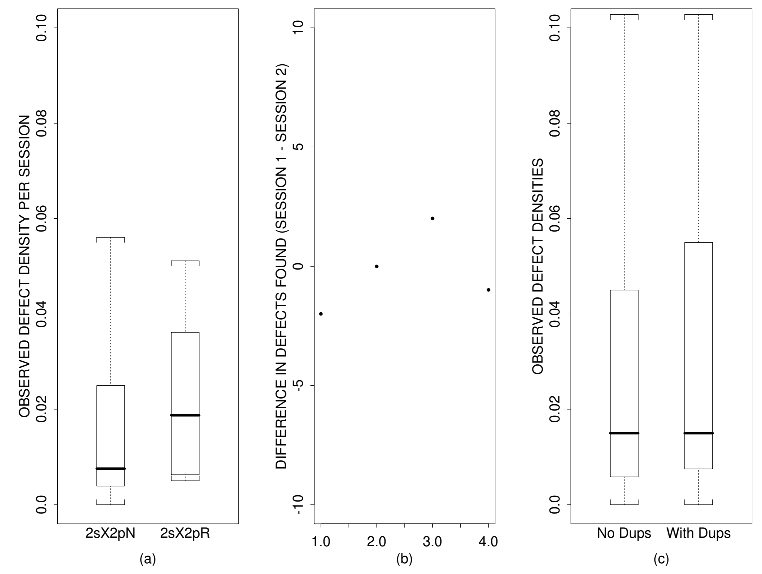

Figure 15: **Effect of Repairing In Between Sessions for 2-Reviewer Teams.** (a) comparing session performance with same team size, (b) difference in number of defects found between session 1 and session 2, and (c) counting the duplicates for 2sX2pN inspections.

lower (Figure 14(b)), but that was not the case for 2sX2pR inspections (Figure 15(b)).

To test whether overlap has a significant influence on without-repair inspections we first calculated the number of defects identified by the first team that were subsequently rediscovered by the second team. If we assume that an equal number of new defects would have been found had repair been done prior to the second inspection, then an approximation for the total number of defects that would have been found by the two sessions would be $D_1 + D_2$. We found that this approximate defect density was not different than defect density of the actual without-repair inspections (see Figures 14(c) and 15(c)).

These results are based on a very small number of observations and should be viewed with considerable caution. Tentatively, it suggests that multiple-session inspections will improve performance only when there is an excess of defects to be found, and that repairing defects in between multiple sessions may not improve the performance of a second inspection team.

5 Conclusions

We have run an 18-month experiment in which we applied different software inspection methods to all the code units produced during a professional software development. We assessed the methods by randomly assigning different team sizes, numbers of inspection sessions, author repair activities, and reviewers to each code unit.

5.1 Fundamental Issues in Industrial Experimentation

This study and its results have several implications for the design and analysis of industrial experiments.

Measurement Problems. In any empirical study, it is important to precisely define the variables to be measured. An important measure of an inspection's effectiveness is the proportion of defects it was able to detect. As we discussed in Section 2.3.2, it is very difficult to create a reliable and readily-available measure for this because we never know how many defects are in the original code unit. Longitudinal studies may help, but they have many problems as well. For example, they are not available until long after the project is finished and thus cannot help identify ineffective treatments in the course of an industrial experiment. Also our attempts to use statistical methods to estimate the original defect content were unsuccessful. Future research should look into better estimation methods.

Another issue is that defects can have different levels of severity. Therefore, defect detection ratio may not reflect the true value of different treatments. For example, a treatment that finds fewer, but more severe, defects may be preferable to another that finds more, but less severe, defects. Future research should look into measures that capture these issues.

Discontinuing Treatments. Because we performed this experiment on a real project, with real deadlines, budgets, and customers, we would have put the project at risk if any one of our treatments turned out to be too costly or too ineffective. Prior to running the experiment we agreed to terminate such treatments. At the same time, we wanted to be reasonably certain that we had enough points to determine that the ineffectiveness or costliness was due to the treatment and not to random chance. Therefore we simulated the experiment and inferred that, with as few as 5 observations, we could tell whether two treatments were different (if the true distributions are sufficiently different. See Appendix A). At the end of the first calendar quarter of the experiment, we indeed discontinued several treatments because they were either significantly less effective or costlier than other treatments. Industrial experimenters must be aware of the risks they introduce to the projects under study.

Nonparametric Statistical Analysis. We used nonparametric statistical techniques because the data distributions were asymmetric and we felt that parametric statistical techniques might be misleading. As we discussed in Section 3.1, this has several implications that the experimenter (and the reader) must be aware of when interpreting the data.

Simplified Instrumentation. We needed to balance our objective of collecting precise, detailed information with our subject's needs to complete their work on time. This is always important in long running experiments with professional subjects. To manage these constraints, the subjects helped us design the data collection forms, trading some precision for ease of reporting. For example, repair effort for each defect (see Section 2.5.2) is reported as ($\leq 1hr$, $\leq 4hr$, $\leq 8hr$, or $> 8hr$) because these categories correspond to less than 1 hour, half-day, full day, and more than one day. Steps like these made it easier for the subjects to fill in the information even in busy times.

In the following Sections we summarize our specific results and discuss their implications from the points of view of both practitioners and researchers.

5.2 Main Results

Team Size (H1). We found no difference in the interval or effectiveness of inspections of 2-, or 4-person teams. The effectiveness of 1-reviewer teams was poorer than both of the others.

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. Maybe better inspection techniques would have found more defects, maybe the code was relatively defect-free, or maybe problems with group interaction become more pronounced as team size grows. We will explore this issue further by tracking the system as it is tested and deployed in the field.

Multiple Sessions (H2). We found that two, 2-person teams weren't more effective than 1, 2-person team. We found that two, 2-person (1-person) teams were not more effective than one, 4-person (2-person) team. We also found that 2-session inspections without repair have the same interval as 1-session inspections.

In practice this suggests that 2-session inspections may not be worth their extra effort.

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 if these techniques are in fact more effective than simpler approaches, the improvement will not be due to the structural organization of the process, but will come from the defect detection techniques they employ.

Serializing Multiple Sessions (H3). We found that repairing defects in between multiple sessions had no effect on observed defect density, but in some cases increased interval dramatically.

In practice, this argues against repairing 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. In particular, why did one treatment have such a long interval? And why weren't we able to predict this effect?

5.3 Other Results

Individual Preparation. Our data indicate that about one-half of the issues reported during preparation turn out to be false positives, Approximately 35–40% pertain to nonfunctional style and maintenance issues. Finally, only 13% 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[18].

Meeting Gains. 30% of defects were meeting gains. These meeting gain rates are higher than those reported by Votta[23] (5%) but are consistent with Humphrey[10] (25%). Since meetings are expensive it's important for researchers to better understand this issue. Also, it is extremely important that contradictory findings be examined and resolved. 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 (so more defects are found in preparation), or (3) design reviewers may prepare much more thoroughly since design defects are likely to be more damaging than code defects. We are currently conducting further experiments to help resolve these discrepancies.

5.4 Interpretation

Our 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.

In particular, two of our major findings are that:

- Although a significant amount of software inspection research has focused on making structural changes (team size, number of sessions, etc.) to the process, these changes did not always have the intended effect. Consequently, we believe that significant improvements to the inspection process are unlikely to come from just reorganizing the process, but rather will depend on the development of new defect detection techniques.
- The 2sX2pR treatment had an interval twice that of the other treatments. Although we were able to gather only four observations, the magnitude of this difference surprises us. Furthermore, it highlights the fact that although researchers frequently argue for changes to software development processes, we have no reliable methods for predicting the effect of these changes on development interval.

6 Future Work

Our continuing work will focus on deepening our analysis in several areas. Some of the questions we will be addressing include:

- How much variation in the observed performance did our experimental design successfully control?
- How much variation in the observed performance can be explained by natural variation in factors outside our control like inspector skill, code quality and author skill?
- What factors outside of our experimental control affected inspection interval? For example, the number of inspections in which each reviewer was already participating, proximity to project deadlines, etc.
- This work suggests that there are general and identifiable mechanisms driving the costs and benefits of inspections. However, we lack a comprehensive theory bringing these principles together. We are currently exploring this issue.

Finally, we remind the reader that all our results were obtained from one project, in one application domain, using one language and environment, within one software organization. Thus we feel it is important that others attempt to replicate our work, and we have prepared materials to facilitate this. These are available online. (See `http://www.cs.umd.edu/users/harvey/experiment`.) Although we have rigorously defined our experiment and tried to limit the external threats to validity, it is only through replication that we can gain confidence that they have been adequately 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] Karla Ballman and Lawrence G. Votta. Organizational congestion in large scale software development. In *Third International Conference on Software Process*, pages 123–134, October 1994.

[2] David B. Bisant and James R. Lyle. A two-person inspection method to improve programming productivity. *IEEE Trans. Software Eng.*, 15(10):1294–1304, Oct. 1989.

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

[4] K P Burnham and W S Overton. Estimation of the size of a closed population when capture probabilities vary among animals. *Biometrika*, 65:625–633, 1978.

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

[6] Stephen G. Eick, Clive R. Loader, M. David Long, Scott A. Vander Wiel, and Lawrence G. Votta. Estimating software fault content before coding. In *Proceedings of the 14th International Conference on Software Engineering*, pages 59–65, May 1992.

[7] 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.

[8] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.

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

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

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

[12] Charles M. Judd, Eliot R. Smith, and Louise H. Kidder. *Research Methods in Social Relations.* Holt, Rinehart and Winston, Inc., Fort Worth, TX, sixth edition, 1991.

[13] John C. Knight and E. Ann Meyers. An improved inspection technique. *Communications of the ACM*, 36(11):50–61, Nov. 1993.

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

[15] K.E. Martersteck and A.E. Spencer. Introduction to the 5ESS(TM) switching system. *AT&T Technical Journal*, 64(6 part 2):1305–1314, July-August 1985.

[16] 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.

[17] Kenneth H. Pollock. Modeling capture, recapture, and removal statistics for estimation of demographic parameters for fish and wildlife populations: Past, present, and future. *Journal of the American Statistical Association*, 86(413):225–238, March 1991.

[18] 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.

[19] 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.

[20] 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.

[21] Sidney Siegel and Jr. N. John Castellan. *Nonparametric Statistics For the Behavioral Sciences.* McGraw-Hill Inc., New York, NY, second edition, 1988.

[22] Scott A. Vander Wiel and Lawrence G. Votta. Assessing software design using capture-recapture methods. *IEEE Trans. Software Eng.*, SE-19:1045–1054, November 1993.

[23] 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.

[24] 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.

A Resolution Analysis

In this section, we describe a simulation conducted to determine the resolution of the experiment, i.e., how different must two treatments be before the significance tests can tell them apart? We simulated the data that would have been generated by inspections using two hypothetical treatments, under different resolutions, sample sizes and variances, and for each combination, we performed a power analysis to determine the significance test's probability of telling apart the two treatment distributions when they are actually different. By using different parameters, we were able to determine the minimum number of data points needed to tell two treatments apart, for a given resolution and variance.

The simulation involves just two treatments, T_a and T_b, whose defect detection probabilities are p_a and p_b. It comprises three distinct steps:

1. **Creation of code units.** We create a number of code units with known size and defect density. The defect density is randomly drawn from a normal distribution with mean μ and standard deviation σ. The number of defects in the code, N, is just the defect density multiplied by the code size.

2. **Application of treatments.** We apply treatments T_a and T_b to different groups of code units. Each group contains sets of 5, 10, and 15 code units. The number of defects found, n_a, by applying T_a to a code unit containing N defects, is determined by a random draw from a Binomial distribution with parameters N and p_a (p_b when applying treatment T_b finds n_b defects).

3. **Comparison of results.** We use the Wilcoxon rank sum test[21] to determine the probability that the n_a's are drawn from the same population as the n_b's.[9]

This process is repeated a hundred times for each experimental setting. Even though the two treatments have different detection probabilities, under some conditions the test may fail to recognize the difference. Running the simulation in a wide variety of experimental settings helps us to determine when and how confidently we can say that two treatments are different.

We created 600 experimental settings consisting of 25 different combinations of means (53,67,80,93,107) and standard deviations (3,7,13,27,40) to generate defect densities, and 24 different pairs of $p_a(.2, .4, .6, .8)$ and p_b ($p_b = p_a + .0, .025, .05, .075, .1, .15$).

Figure 16 shows some (108 out of 600 settings) of the simulation results. The x-axis shows the true difference between p_a and p_b and the y-axis shows the probability that the null hypothesis ($p_a = p_b$) will be rejected. Each combination of a symbol and a line segment represents the outcomes of 100 simulation runs of one experimental setting. The symbol indicates the median, and the line segment through the symbol spans the .25 through the .75 quantiles.

We define the experimental resolution as the value when more than 50% of the 100 outcomes have a significance greater than .9 (the symbol in Figure 16 lies *above* the resolution line), and the next smaller true difference value has the symbol with less than 50% of the 100 outcomes greater than .9 (the symbol in Figure 16 lies *below* the resolution line).

[9] Although the Wilcoxon rank sum test is not as powerful as a t distribution test, the Wilcoxon rank sum test does not require the n_a's and n_b's to be normally distributed – an assumption that is difficult to test with small samples of data.

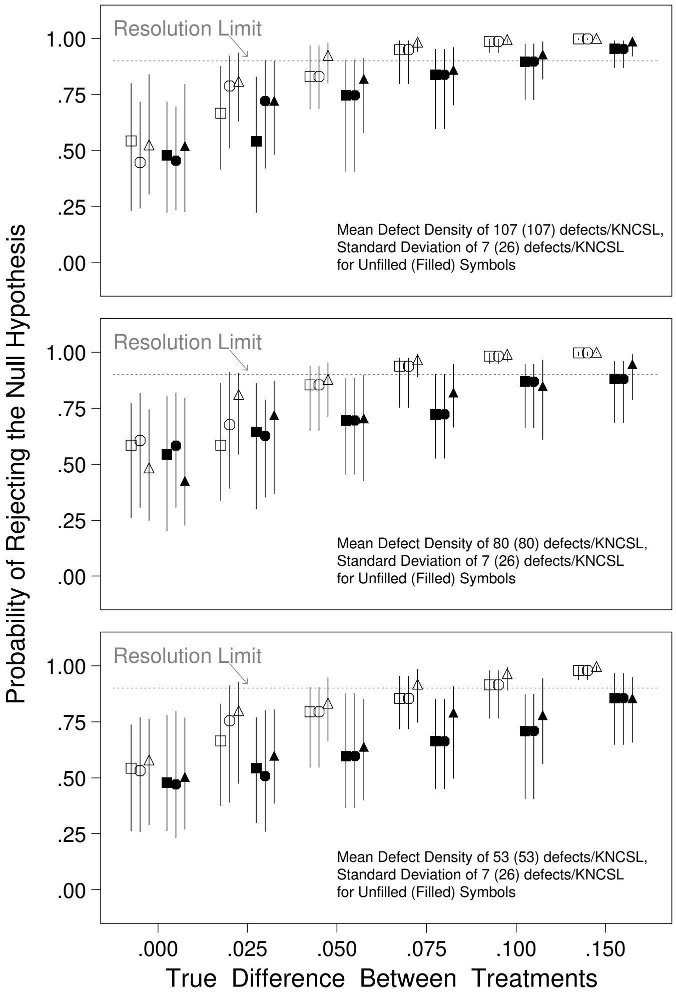

Figure 16: **Resolution of the Experiment.** This plot shows the results of applying treatments T_a and T_b to sets of 5, 10, or 15 code units (marked by the square, circle, and triangle). Each simulated unit has 300 NCSL and a mean defect density of 53, 80, or 107 defects per 1000 NCSL, with a standard deviation of 7 or 26 defects per 1000 NCSL. p_a is set to .6. The x-axis shows the true difference between p_a and p_b and the y-axis shows the probability that the null hypothesis (i.e., that all the treatments have the same effectiveness) will be rejected. Each combination of a symbol and a line segment represents the outcome of 100 simulation runs for one experimental setting. The symbol indicates the median and the line segment runs from the lower to the upper quartile. Symbols plotted above the dotted horizontal line in each panel indicate experimental situations where true differences in treatment effectiveness can be reliably detected. The simulation results indicate a resolution as fine as .05. The resolution does not become substantially finer as the number of observations increases; however, it does become finer as the standard deviation decreases.

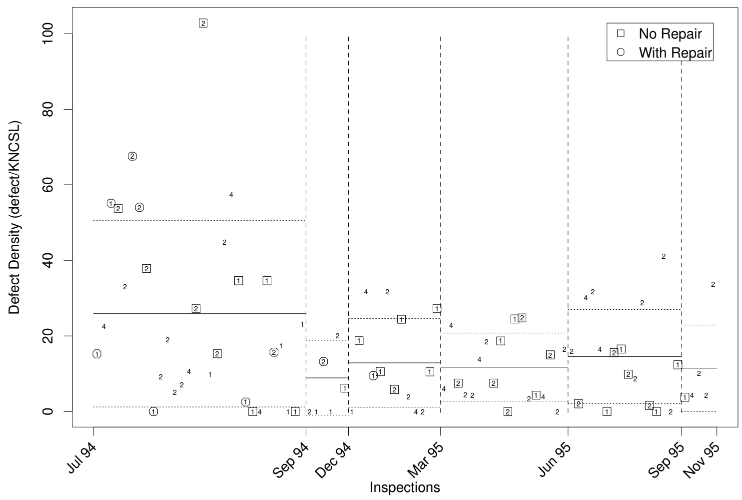

Figure 17: **Inspection Performance Over Time.** This is a time series plot showing the trends in observed defect densities of inspections as time passed. The vertical lines partition the plot into quarters. Within each quarter, the solid horizontal line marks the mean of that quarter's distribution. The dashed lines mark one standard deviation above and below the mean. The treatment used by the inspection is encoded in the plotting symbol. The plotted numbers represent the team size of the inspection. The open ones are 1-session inspections, the circled ones are 2-session inspections with repair and the square ones are 2-session inspections with no repair.

B Inspection Performance over Time

B.1 Chronological Overview

Initially, the experiment involved seven treatments. At the beginning of 1995, we evaluated the existing results and discussed them with the project's management. Although we would have preferred to gather more data, it would have been risky for the project to continue performing expensive or ineffective treatments. Therefore, we discontinued three treatments: 1sX1p, 2sX1pR, and 2sX2pR.

The 1sX1p treatment was dropped because it was ineffective relative to the others while the two with-repair treatments (2sX1pR and 2sX2pR) were dropped because the authors shared with us that they felt they were doing twice as much work, thus congesting their schedules unnecessarily, and because we saw that they were no more effective than the without-repair treatments (Figure 18). In addition, the 2sX2pR treatment was, by far, the most expensive treatment in terms of interval. Figure 17 confirms that the last instances of these discontinued treatments were held in the first quarter of 1995.

Our primary concern is that discontinuing treatments may compromise the experiment's internal validity (i.e., factors that affected all treatments early in the experiment, will affect only the remaining treatments later in the experiment). Consequently, we must be careful when we compare treatments that were discontinued with those that were not.

Figures 17 and 19 show inspection effectiveness and interval over time, with observations sorted according to the time at which the code unit became available for inspection.

B.2 Analysis of Inspection Performance Over Time

The data presented in Figure 17 suggests that there are two distinct performance distributions. That is, that the first quarter (July - September, 1994) – during which about one-third of the inspections occurred – has a significantly higher mean and variance than the remaining quarters (October, 1994 – December, 1995).

One reason for this may be that the end of the first quarter coincides with the system's first integration build. Our records show that with the compiler's front end in place, the developers were able to do more thorough unit testing for the back end code than they could for front end code alone.

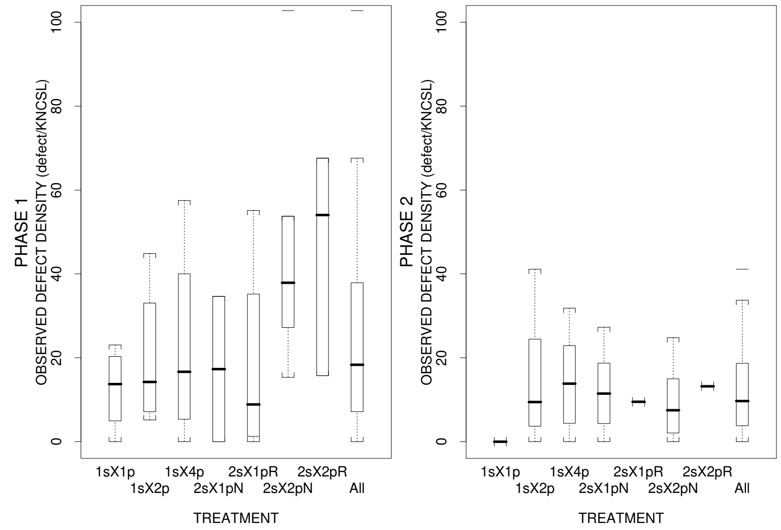

Figure 18: **Observed Defect Density by Treatment and Phase.** These two plots show the observed defect density for each inspection treatment during the first and second phase of the project. Across all inspections, the median observed defect density was 18 defects per KNCSL for the first phase and 10 defects per KNCSL for the second phase.

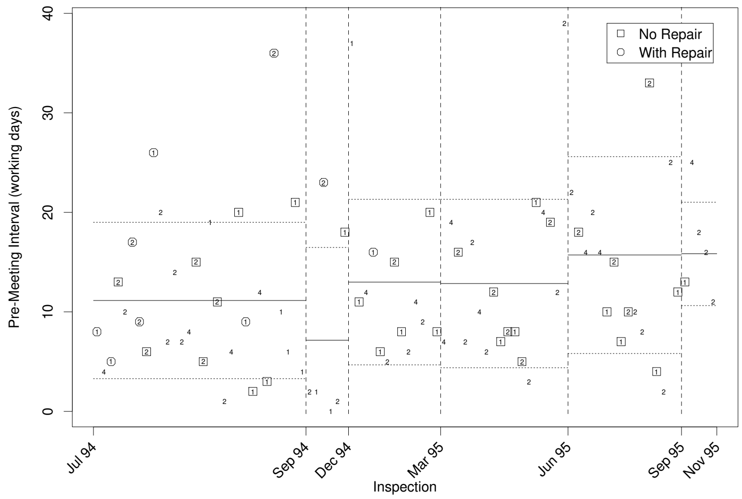

Figure 19: **Inspection Intervals Over Time.** This is a time series plot showing the trends in inspection intervals as time passed.

Other factors may be that the reviewers had become more familiar with the programming language as the project progressed, that the requirements for the front-end (language definition, parsing, and intermediate code generation) were more prone to misinterpretation than the final code generation and optimization.

In particular, this suggests to us that had we continued using the 2sX2pR treatment its effectiveness would have dropped in a manner consistent with the other treatments.

B.3 Analysis of Inspection Interval Over Time

Figure 19 is a time series plot showing inspection interval as project progressed. We see that the mean inspection interval did not vary significantly throughout the project, although there is a gradual increase as the project nears completion.

Although there were only four 2sX2pR inspections, the stability of the interval for the other treatments suggests that had we continued the treatment, its interval would not have changed significantly.