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

A Preliminary Report

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

September 6, 1994

Abstract

This experiment (currently in progress) is designed to measure costs and benefits of different code inspection methods. It is being performed with a real development team writing software for a commercial product. The dependent variables for each code unit's inspection are the elapsed time and the number of defects detected. We manipulate the method of inspection by randomly assigning reviewers, varying the number of reviewers and the number of teams, and when using more than one team, randomly assigning author repair and non-repair of detected defects between code inspections.

After collecting and analyzing the first 17% of the data, we have discovered several interesting facts about reviewers, about the defects recorded during reviewer preparation and during the inspection collection meeting, and about the repairs that are eventually made. (1) Only 17% of the defects that reviewers record in their preparations are true defects that are later repaired. (2) Defects recorded at the inspection meetings fall into three categories: 18% false positives requiring no author repair, 57% soft maintenance where the author makes changes only for readability or code standard enforcement, and 25% true defects requiring repair. (3) The median elapsed calendar time for code inspections is 10 working days - 8 working days before the collection meeting and 2 after. (4) In the collection meetings, 31% of the defects discovered by reviewers during preparation are suppressed. (5) Finally, 33% of the true defects recorded are discovered at the collection meetings and not during any reviewer's preparation.

The results to date suggest that inspections with two sessions (two different teams) of two reviewers per session (2sX2p) are the most effective. These two-session inspections may be performed with author repair or with no author repair between the two sessions. We are finding that the two-session-two-person-with-repair (2sX2pR) inspections are the most expensive, taking 15 working days of calendar time from the time the code is ready for review until author repair is complete, whereas two-session-two-person-with-no-repair (2sX2pN) inspections take only 10 working days, but find about 10% fewer defects.

** 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. Their expense is often justified by 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. Holding such a large number of meetings can cause delays which may significantly lengthen the development interval (calendar time to completion).[1] Since long development intervals risk substantial economic penalties, the hidden cost of the current inspection process must be considered.

We hypothesize that different inspection approaches involve different tradeoffs between minimum interval and maximum effectiveness. But until now there have been no empirical studies to evaluate these tradeoffs. We have 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 Literature Review

To eliminate defects, many organizations use an iterative, three-step inspection procedure: Preparation, Collection, Repair[11] . First, a team of reviewers reads the artifact, 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 [8, 9, 20, 17, 24, 12, 1] . Few, critically analyze inspections or rigorously evaluate alternative 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.[18] 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)[14] and Knight and Myers' phased inspections (PI)[13] 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 authors believe that multiple-session inspections will be much more effective than single-session inspections, but they do not show this empirically, nor do they consider any effects 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[7] . 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.

[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.

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 [22] found far lower meeting gain rates (about 5%). Porter et. al[16] 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, concrete studies are needed.

1.2 Hypotheses

Inspection approaches are usually evaluated according to the number of defects they find. As a result, some information is available 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 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 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's 5ESS® telephone switching system. The finished system is expected to contain 30K lines of C++ code, of which about 6K is reused.

All of the team's six members are experienced developers, and all have received training on inspections. The project began coding during June, 1994, and will perform about 100 code inspections by the end of the year.

2.2 Operational Model

To test our hypotheses we must measure both the interval and the effectiveness of every inspection. We began by constructing models for calculating inspection interval and estimating the number of defects in a code unit. These models are depicted in Figure 1.

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 [23] . 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 is finished with his or her preparation. This information is entered into a database, and inspection intervals are reconstructed by performing queries against the database.

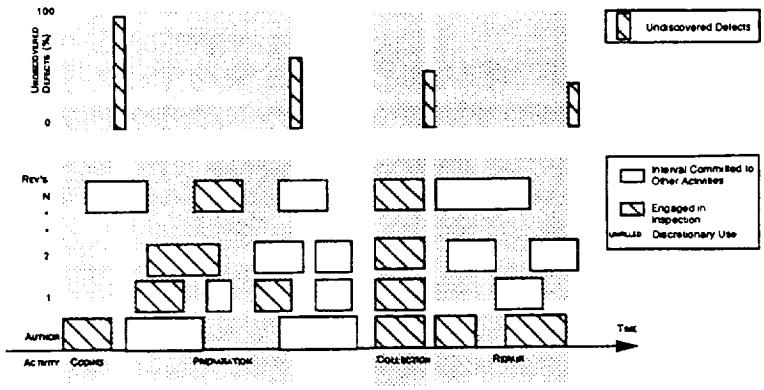


Figure 1: This figure depicts a simple model of the inspection process. The figure's lower panel summarizes the inspection's time usage. Specifically, it shows the inspection's participants (an author and several reviewers), the activities they perform (coding, preparation, collection, repair, and other),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

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.

We will use the following approaches to approximate the defect detection ratio.

- **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 very imprecise.
- **Complete estimation of detection ratio:** We track the code through testing and field deployment, recording new defects as they are found. This is more precise, but is not available until well after the project is completed.

2.3 Experimental Design

2.3.1 Variables

The experiment manipulates three independent variables:

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

The treatment distributions are shown in Table 1.

For each inspection we measured four dependent variables:

1. inspection intervals,
2. estimated defect detection ratio,
3. the percentage of defects first identified at the collection meeting (meeting gain rate),
4. the percentage of potential defects reported by an individual, but not recorded at the collection meeting (meeting suppression rate).

We also capture repair statistics for every defect.

	Number of Sessions			Totals
	1	2		
Team Size		With Repair	Without 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 percentage of inspections allocated to each setting of the independent variables. Note: Since we cannot apply capture-recapture estimates to the data from the one-session-one-person or two-session-one-person-with-repair inspections, this data will be held out of the capture-recapture analysis.

2.3.2 Design

This experiment uses a $2^2 \times 3$ partial factorial design to compare the interval 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 with no 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 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 and so are not a random sample of the entire development population.

2.3.5 Analysis Strategy

Once the data are collected we will analyze the combined effect of the independent variables on the dependent variables to evaluate our principal hypothesis. 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. The reviewer then links this ID to his or her preparation form.

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).

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.

2.4.3 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.[2] 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.[3] The IQE attends the collection meeting to ensure that all the procedures have been correctly followed. 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.

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

[3] 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.

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.

As of this writing, only 17% of the planned inspections have been completed. Consequently, we do not yet have enough data to 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. In practice, many of these issues, even if they went unrepaired, would not lead to incorrect system behavior, and they are therefore of no interest to our analysis.

Based on information in the repair form and interviews with each author, we classified 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).

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 are then better able to make more reliable classifications.

The distribution of defect classifications for each treatment appears in Figure 2. Across all inspections, 18% of the issues are False Positives, 57% involve Soft Maintenance, and 25% are True Defects.

We consider only True Defects in our analysis of estimated defect detection ratio (a dependent variable).[4]

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 construct the inspection intervals (usually considered to be the calendar period between the submission of an inspection request and the completion of all repairs).

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

[4] 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.

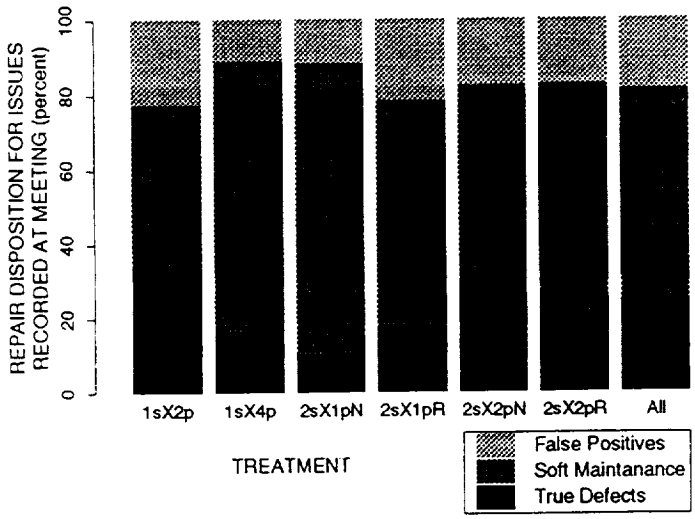


Figure 2: **Disposition of Issues Recorded at the Collection Meeting.** For each treatment, the stacked barchart shows the percentage of the issues recorded at collection meetings that turn out to be false positives, soft maintenance, or true defects. Across all treatments, only 25% of the issues are true defects.

	Number of Sessions			Totals
	1	2		
Team Size		With Repair	Without Repair	
1	0	3	2	5
2	5	2	4	11
4	1	0	0	1
Totals	6	5	6	17

Table 2: This table shows the number of observations we currently have for each treatment.

periods. To remove these cases from the analysis, we redefined the inspection interval to be the calendar period between the submission of an inspection request and the completion of the collection meeting.

When these reductions are made, two-session inspections have two inspection subintervals – one for each session. We equate the interval for such inspections with the longer of these two subintervals, since both of them begin at the same time.

Next, we removed all nonworking days from the interval. Nonworking days are defined as (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 the length of these reduced intervals in our analysis of the inspection interval.

Figure 3 is a boxplot[5] showing the number of working days from the issuance of the inspection request to the collection meeting, from the collection meeting to the completion of repair, and the total. The total inspection interval has a median of 10 working days, 8 before and 2 after the collection meeting.

3.2 Overview of Data

Table 2 shows the number of observations to date for each treatment. Figure 4 is a contrast plot showing the interval and effectiveness of all inspections and for every setting of each independent variable. This information is

[5]In this paper we have made extensive use of boxplots to represent prominent features of a distribution. Each set of data is represented by a box, the height of which corresponds to the spread of the central 50% of the data, with the upper and lower ends of the box marking the upper and lower quartiles. The data median is denoted by a bold point within the box. The lengths of the vertical dashed lines relative to the box indicate how stretched the tails of the distribution are; they extend to the standard range of the data, defined as 1.5 times the inter-quartile range. The detached points are "outliers" lying beyond this range.[4]

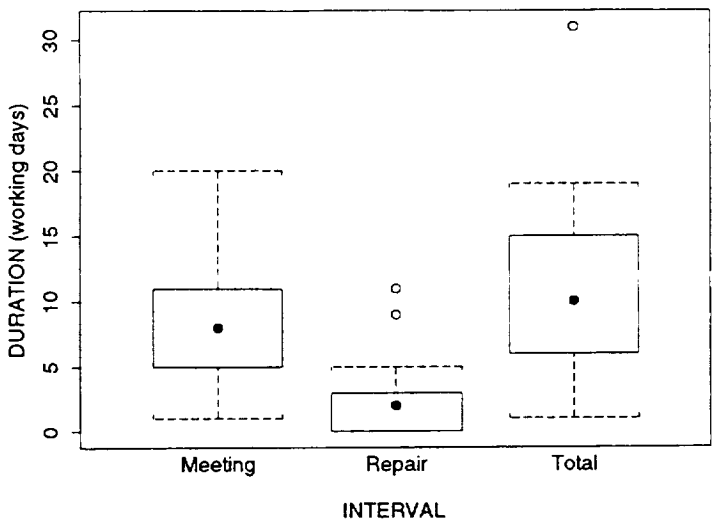


Figure 3: **Premeeting Inspection Interval.** These boxplots show all the interval data divided into two parts: time before the meeting and time after the meeting. The total inspection time has a median of 10 days, 80% of which is before the meeting.

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 5 shows the inspection interval (premeeting only) by treatment and for all treatments.

We draw several observations from this data. First, the interval of two-session-one-person inspections is no longer than the interval of one-session-two-person inspections. Second, 2sX2pN inspections also have no longer interval than 1sX2p inspections.

The cost of serializing two inspection sessions is suggested by comparing 2sX2pN inspections with 2sX2pR inspections. The 2sX2pR inspections have a 53% longer interval than the 2sX2pN inspections. This indicates that any multiple session inspections that require repair after each session will significantly increase the inspection interval.

The additional cost of multiple inspection sessions can be seen by comparing 1sX2p inspections with 2sX2pN inspections. The 2sX2pN interval is only slightly longer than the 1sX2p interval. However, since the author must be involved in each session, the interval is likely to grow as the number of sessions increases.

3.4 Analysis of Effectiveness Data

The benefit of inspections is that they find defects. This benefit will vary with the different inspection treatments. Figure 6 shows the observed defect density for all inspections and for each treatment separately.

Several interesting trends appear in the preliminary data. First, 1sX2p inspections are as effective as 1sX4p inspections. Second, 2sX2p inspections appear to be more effective than any one-session inspection, but 2sX1p inspections are *not* more effective than 1sX2p inspections. Finally, 2sX2pR inspections are more effective than 2sX2pN inspections.

The effectiveness of different team sizes is suggested by comparing 1sX2p, 1sX4p, and 2sx1pN inspections. The low effectiveness of 1sX4p inspections may indicate that current inspection teams are too large; however, with only a single 1sX4p inspection drawing any conclusions would be premature.

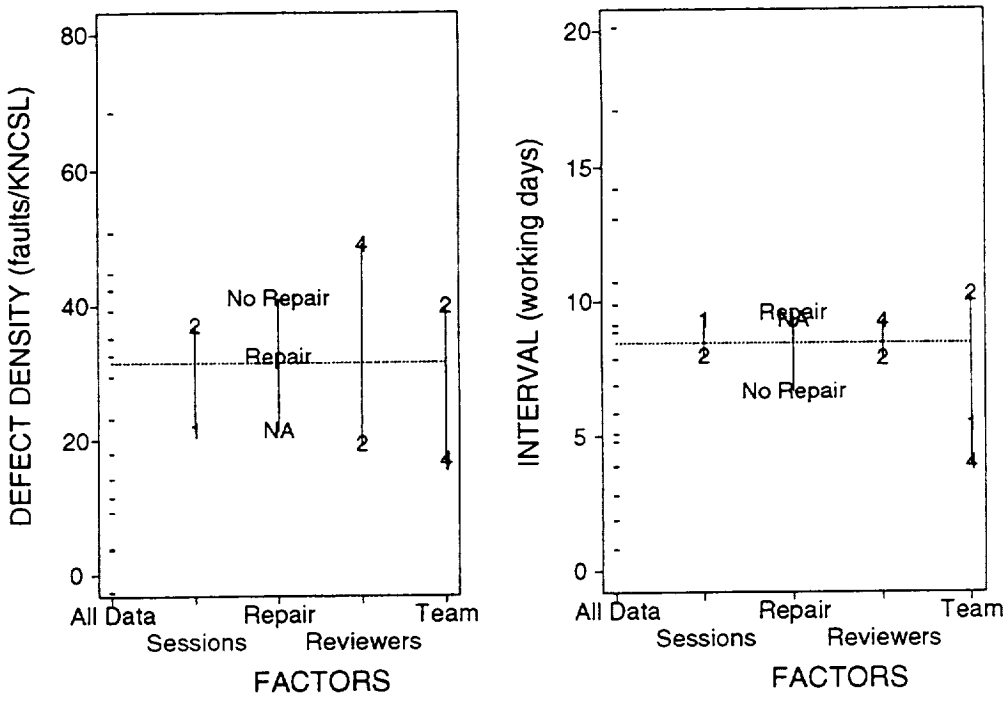


Figure 4: **Effectiveness and Interval by Independent Variables.** The dashes in the far left column of the first plot show the defect detection rates for all inspections. The dotted horizontal line marks the average defect detection rate. The other four columns indicate factors that may influence this 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 right plot shows similar information for inspection interval.

The additional effectiveness of multiple sessions is suggested by comparing 1sX4p and 2sX2p and 2sX1p inspections. This data indicates that it is more effective to use two teams of two persons each than to use a single team of four persons. However, it appears that two teams of one person are *not* more effective than a single team of two persons. Many multiple session methods rely on the assumption that several one person teams can be more effective than a single large team. However, our results suggest that the performance of individual reviewers must be increased if multiple session methods are to be effective.

The additional effectiveness due to serializing multiple sessions is suggested by comparing 2sX2pR against 2sX2pN inspections. While the data shows that 2sX2pR inspections are the more effective, the difference in effectiveness between 2sX2pR and 2sX2pN inspections is small, about 2 defects per 300 NCSL.

3.5 Meeting Effects

During prepreparation, 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. These meetings have a significant effect 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 7 shows the percentage of defects reported by each reviewer that are eventually determined to be true defects. Across all inspections, only 25% of all reports turn out to true defects. This figure appears to be independent of inspection treatment.

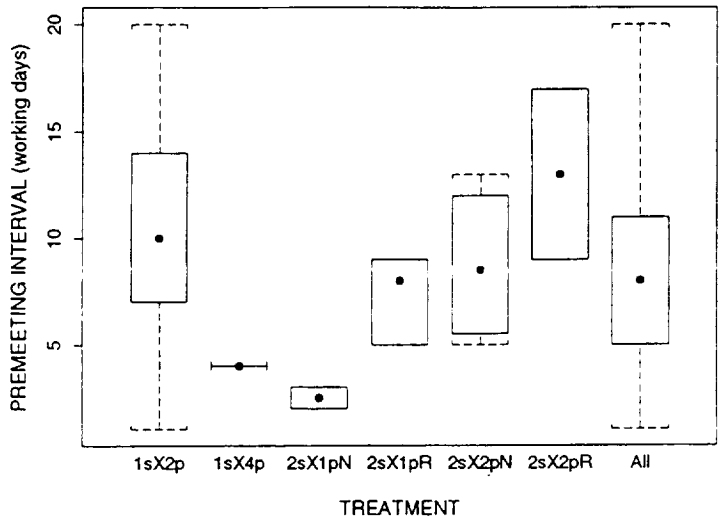


Figure 5: **Premeeting Interval by Treatment.** This plot shows the observed interval for each inspection treatment.

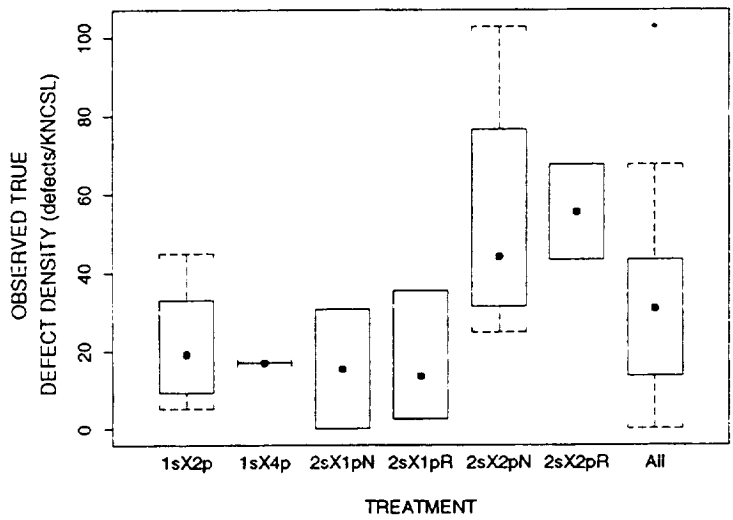


Figure 6: **Observed Defect Density by Treatment.** This plot shows the observed defect density for each inspection treatment. Across all inspections, 32 defects were found per KNCSL.

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.

Figure 8 shows the suppression rates for all inspections. One trend in the preliminary data is that four-person inspections consistently suppress more reports than any other treatment. (One-session-two-person inspections show considerable variability.)

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 are first discovered at the collection meeting are

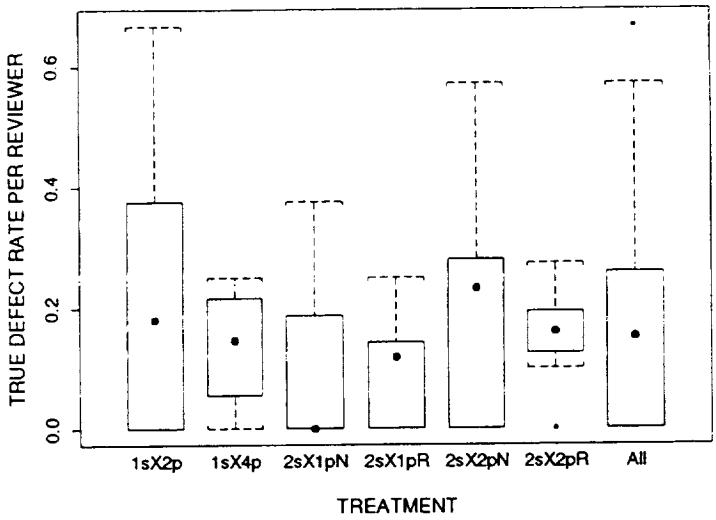


Figure 7: **True Defect Rate per Reviewer Prepartion Report by Treatment.** This boxplot shows the rate at which defects found during preparation are eventually considered to be true defects. Across all treatments, only 17% of the reports turn out to be true defects.

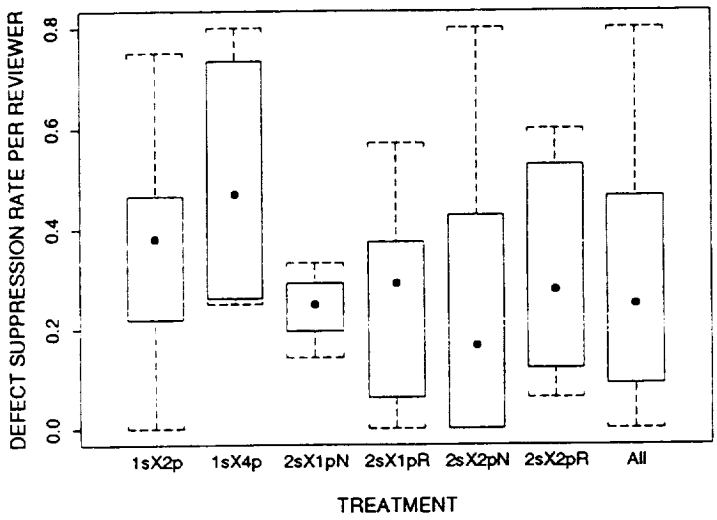


Figure 8: **Meeting Suppresion Rate by Treatment.** These boxplots show the suppression rate for each reviewer by treatment. The suppression rate for a reviewer is defined as the number of defects detected during preparation but not included in the collection meeting defect report, divided by the total number of defects recorded by the reviewer in his/her preparation. Across all inspections, 31% of the preparation reports are suppressed.

called meeting gains.

Figure 9 shows the meeting gain rates for all inspections. Across all inspections, 33% of all defects discovered are meeting gains. The data suggests that 1sX4p inspections have the lowest gain rates.

The effect of team size is suggested by comparing 1sX4p inspections to all others. Although most of the treatments show similar gain rates, one-session-four-person inspections appear to be the lowest. This may indicate

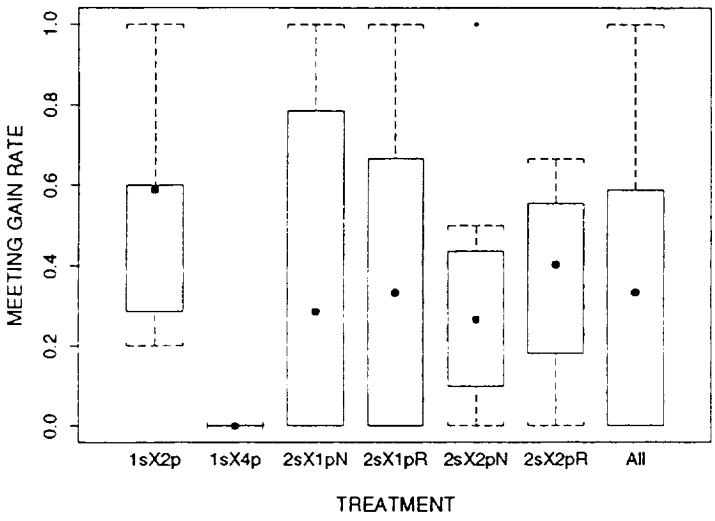


Figure 9: **Meeting Gain Rate by Treatment.** These boxplots shows the meeting gain rates for all inspections and for each treatment. The median rate was 33%.

that larger team size can be detrimental to effectiveness.

The data indicate that almost half of the defects reported during preparation turn out to be false positives. This suggests that much of the preparation effort is unproductive and that the development of improved preparation techniques may significantly increase overall effectiveness.

Our observed gain rates are much higher than those reported by Votta[22] . Explanations include three possible causes:

- Votta's study was concerned with design documents rather than code;
- the average team size for a design review is larger than for code inspections;
- design reviewers may prepare much more thoroughly since design defects have wider impact than code defects.

4 Conclusions and Future Work

We are in the midst of a long term software inspection experiment based on all of the code units in a real 5ESS software development product. We are assessing several inspections methods by randomly assigning different team sizes, combinations of reviewers, numbers of inspection sessions, and author repair activities to each code unit. To date we have completed 17 of the planned 100 inspections. We expect to finish the remaining 83 inspections by the end of 1994.

Preliminary results of our empirical study of the effectiveness of various software inspection methods challenge certain long-held beliefs about the most efficient way to conduct inspections.

Judging from the percentage of defects discovered, we are finding that two-session-two-person (2sX2p) inspections appear to be the most effective. The difference in effectiveness between 2sX2p inspections and other treatments also show that number of sessions and number of reviewers per session are important factors affecting efficiency.

Two-session-two-person-with-repair (2sX2pR) inspections seem to be slightly more effective than two-session-two-person-with-no-repair (2sX2pN) inspections. However, repairing defects between sessions costs 5 extra working days of inspection interval.

We believe that when all the inspection data has been collected and analyzed, the answers to the following questions about software inspections will emerge:

1. Are some inspection methods significantly more effective than others?
2. What is the most efficient number of reviewers per inspection?
3. How many review sessions per inspection will give the best results?
4. Are multiple session inspections more cost beneficial than single session inspections?
5. Should author repair be done between review sessions?

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] 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.

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

[5] 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.

[6] 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.

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

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

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

[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] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

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

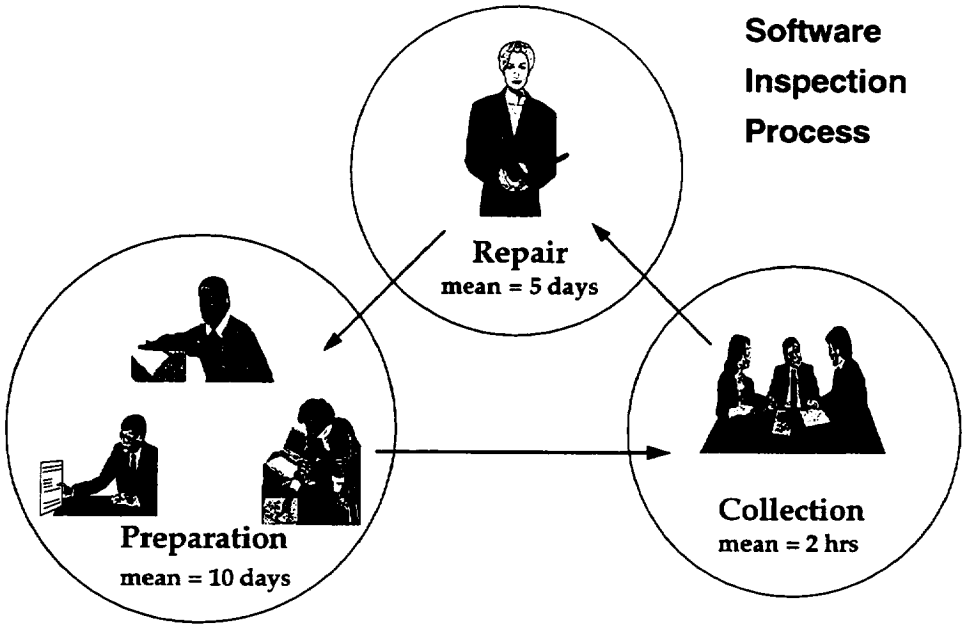
Adam Porter
Harvey Siy

Computer Science Dept.
University of Maryland
College Park, MD 20742
aporter@cs.umd.edu

Carol Toman
Lawrence Votta

Soft. Production Res. Dept.
AT&T Bell Laboratories
Naperville, IL 60566
votta@research.att.com

November 30, 1994



- Many organizations use a three-step inspection process
- Interval - ready for inspection -> completion of repair

Background/Motivation

- Competing views
  - number of sessions: single vs. multiple
  - collection meetings: yes vs. no
  - team size: large vs. small
  - coordination of multiple sessions: parallel vs. sequential
- Empirical validation
- Costs are ignored
  - interval is not normally considered an inspection cost

Hypotheses

- Inspections with larger teams have longer inspection intervals; but do not find significantly more defects.
- Collection meetings do not significantly increase detection effectiveness.
- Multiple-session inspections are more effective than single-session inspections, but significantly increase inspection interval.

Experimental Setting

- AT&T 5ESS
  - 3000 software developers
  - hierarchical organizational structure
- Legacy system
  - 1982
  - design lifetime 20 years
- Other
  - ISO 9001 certified, SEI Level 2
  - 5 MNCSL each in product and support tools
- Project
  - compiler 30K new C++, 6K reused from prototype
  - 6 software developers, plus 4 extra inspectors

Experiment Variables

- Independent
  - number of reviewers per session (1, 2, 4)
  - number of inspection sessions (1, 2)
  - repair between multiple sessions (N, Y)

- Dependent
  - inspection interval (working days)
  - observed defect density (defects/KNCSL)
  - meeting gain rate
  - meeting suppression rate

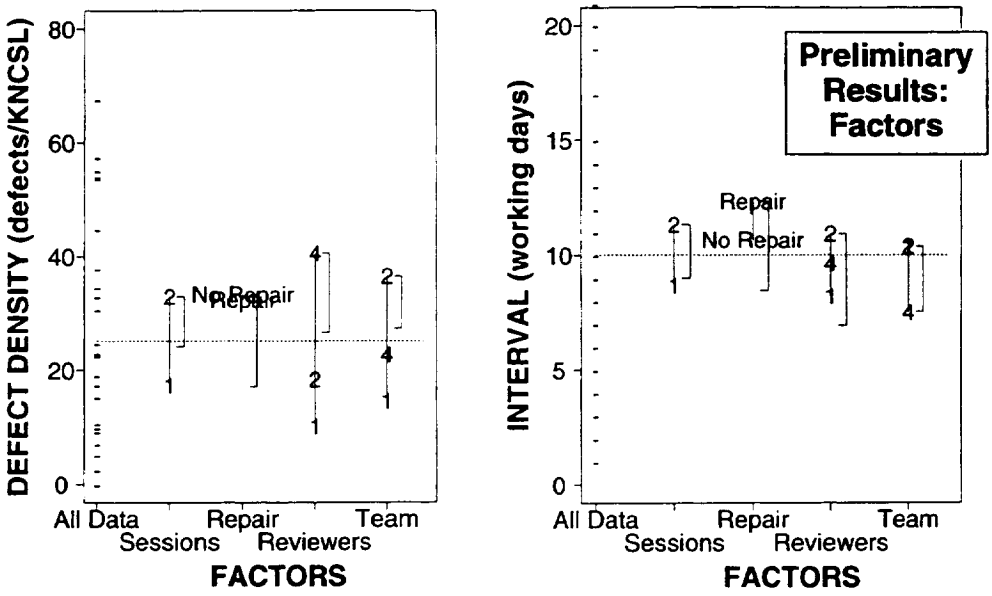
Experimental Design

Number of Reviewers	Number of Sessions			Totals
	1	2		
		Repair	No Repair	
1	1/9	1/9	1/9	1/3
2	1/9	1/9	1/9	1/3
4	1/3	0	0	1/3
Totals	5/9	2/9	2/9	1

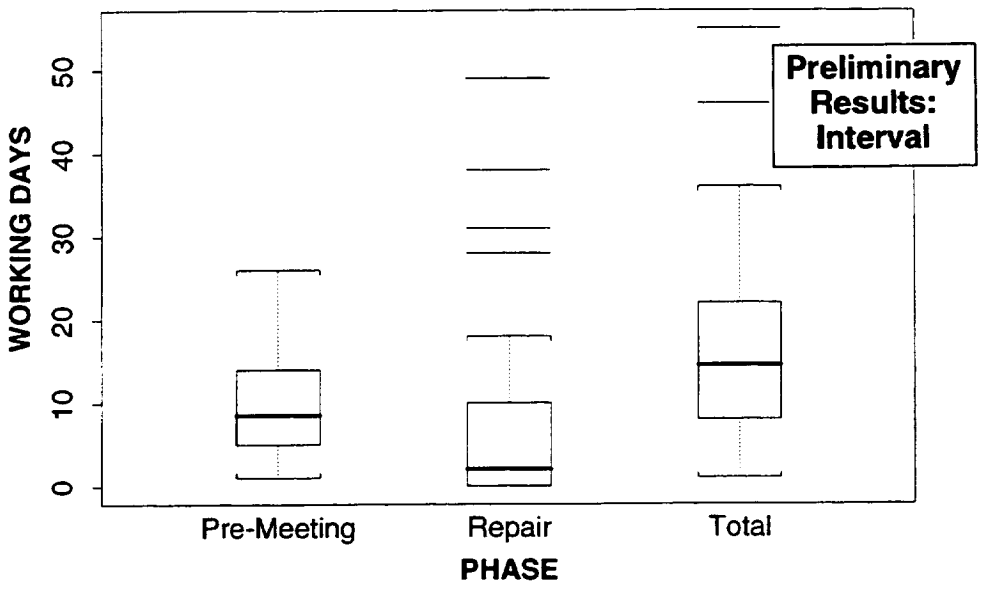
- 2 session, 4 person treatments too expensive
- 1 session, 4 person treatment is common practice

Experimental Validity

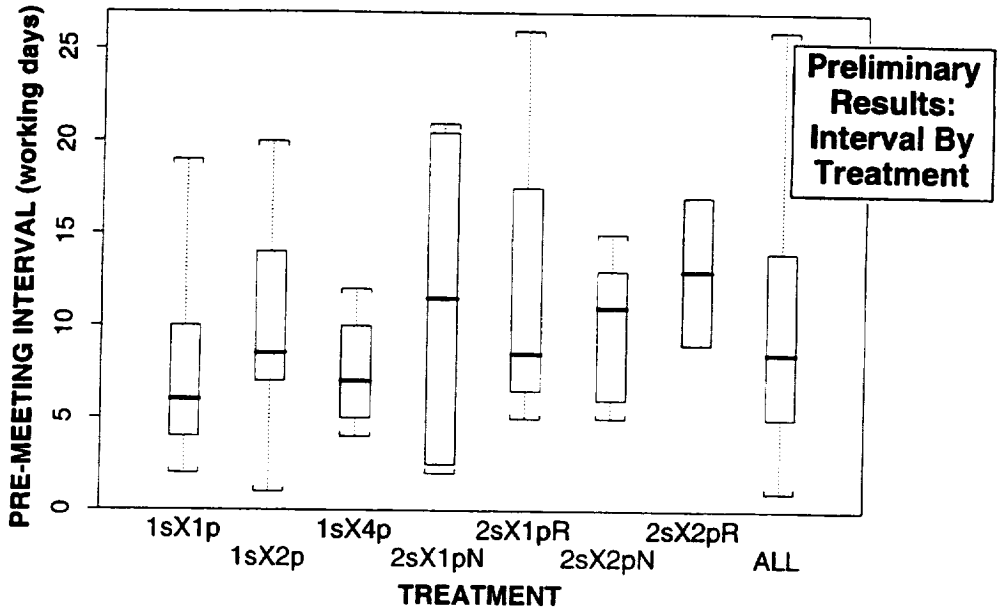
- Internal
  - selection (natural ability)
  - maturation (learning)
  - instrumentation (code quality)
- External
  - scale (project size)
  - subject generalizability (experience)
  - subject representativeness (random draw from population)



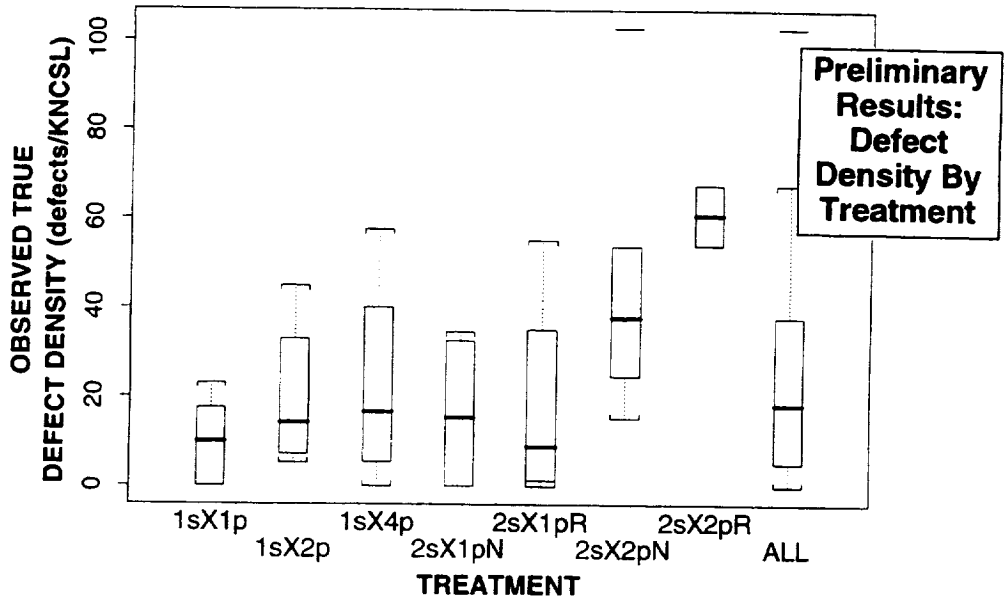
- **Density: Sessions, Reviewers, Team significant**
- **Interval: no significant factors**



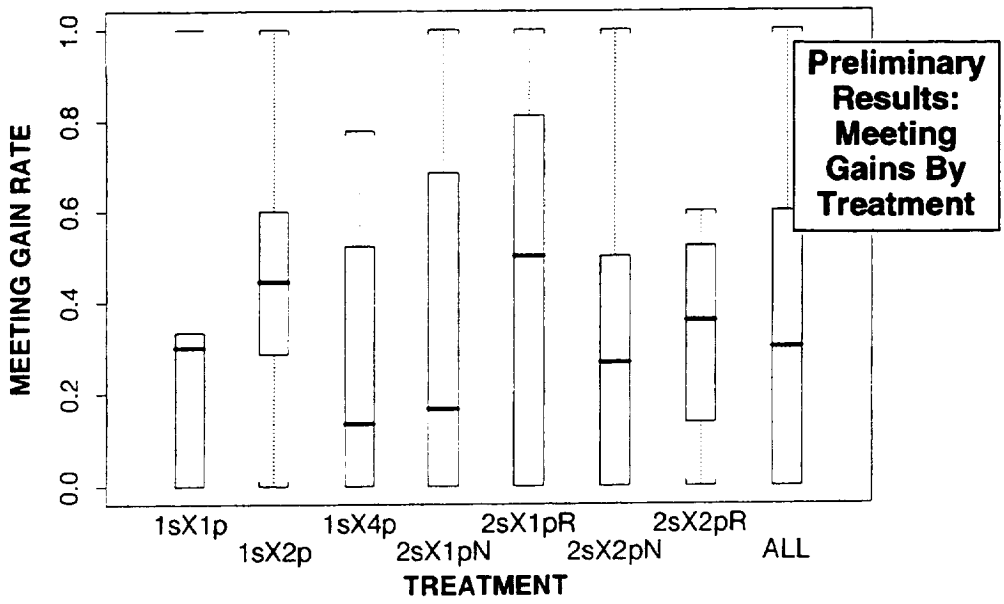
- **Medians: pre-meeting = 8.5 days, total = 14.5 days**
- **Delayed repair sometimes inflates interval data**



- **Distributions are similar**
- **2sX2pR takes longest: median of 13 days**
- **2sX2pN has median of 11 days**



- **2sX2p treatments are the best**
- **2sX2pR better than 2sX2pN by 35%**
- **Team size makes no difference for 1-session treatments**



- Overall median meeting gain rate is 0.3
- 1sX4p has lowest meeting gain rate

Results To Date

- 2sX2p are most effective inspection method.
- Repair between improves detection effectiveness by 35 %; at a cost of 2 additional working days of interval.
- 1sX2p are as effective as 1sX4p but most 1sX4p defects are found at preparation.

Next Steps

- What makes 2sX2p inspections more effective than 1sX2p?
- Why are the interval differences not more pronounced?
- What are the cost-benefit tradeoffs of meetingless inspections?