An Experiment to Assess Cost-Benefits of Inspection Meetings and their Alternatives

Patricia McCarthy, Adam Porter*, Harvey Siy
Computer Science Department
University of Maryland
College Park, Maryland 20742
{trishmcc,aporter,harvey}@cs.umd.edu

Lawrence Votta
Software Production Research Department
AT&T Bell Laboratories
Naperville, Illinois 60566
votta@research.att.com

Abstract

We hypothesize that inspection meetings are far less effective than many people believe and that meetingless inspections are equally effective. However, two of our previous industrial case studies contradict each other on this issue. Therefore, we are conducting a multi-trial, controlled experiment to assess the benefits of inspection meetings and to evaluate alternative procedures.

The experiment manipulates four independent variables: (1) the inspection method used (two methods involve meetings, one method does not), (2) the requirements specification to be inspected (there are two), (3) the inspection round (each team participates in two inspections), and (4) the presentation order (either specification can be inspected first).

For each experiment we measure 3 dependent variables: (1) the individual fault detection rate, (2) the team fault detection rate, and (3) the percentage of faults originally discovered after the initial inspection phase (during which phase reviewers individually analyze the document).

So far we have completed one run of the experiment with 21 graduate students in the computer science at the University of Maryland as subjects, but we do not yet have enough data points to draw definite conclusions. Rather than presenting preliminary conclusions, this article (1) describes the experiment's design and the provocative hypotheses we are evaluating, (2) summarizes our observations from the experiment's initial run, and (3) discusses how we are using these observations to verify our data collection instruments and to refine future experimental runs.

1 Introduction

For almost twenty years inspections have been used to validate software. Although software development has changed substantially in that time, inspections are still performed in much the same way. First each member of the review team analyzes the document. Later the team meets to inspect the document as a group. Finally, the author repairs the faults that have been discovered.

Some researchers, notably Parnas and Weiss [10], have questioned the effectiveness of this approach. Specifically, they argued that group meetings are unnecessary, even detrimental, because they add cost and congest developers' already full schedules, yet they find few very faults.

*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

1.1 Does every inspection need a meeting?

Despite these objections many people are convinced that meetings are an essential part of successful inspections. This conviction stands on the argument that (1) many faults are found during meetings, and therefore they justify their cost; and (2) because of these meetings more faults are found than would be otherwise. Thus they believe that a group of reviewers is likely to be more effective working together than working separately.

Votta [14] evaluated this argument in a case study involving 20 design inspections at AT&T. To quantify the usefulness of inspection meetings, he determined the proportion of faults found during the inspection that were originally discovered at the meeting (the meeting gain rate). He reported that the average meeting gain rate for these inspections was $\approx 5\%$. This would mean that if 20 faults were discovered during the inspection, 19 were already known before the meeting ever started!

This result was striking, but a later data seemed to contradict it. Porter, Siy, Toman and Votta [11] conducted another study, also at AT&T, involving > 100 code inspections. Although their primary goal was not to study inspection meetings, they collected data on meeting gains in much the same way that Votta's earlier study had.[1] This time the average meeting gain rate was 33%, with considerable variance in the observations (i.e., many meetings produced no gains at all, while some had rates as high as 80%.)

We have been unable to find any simple explanation for these differences[2], but our attempts to make sense of these results generated hypotheses concerning the relationship between inspection meetings and the entire inspection process. And these hypotheses led us to some practical questions about the most cost-effective ways to conduct inspections.

Below we discuss how we arrived at these questions and working hypotheses, describe the design, execution and analysis of an experiment to test them, and discuss the implications of our experimental results.

1.2 Digging Deeper

Obviously, meetings that produce little gain are a large, unnecessary expense, therefore, it is important to determine exactly how meetings contribute to inspections and whether superior alternatives exist.

[1]We strongly believe that empirical research must be replicated. This experience illustrates an economical way to do this. We instrumented the study so that it provided not only the data we were immediately interested in, but also the data needed to replicate Votta's earlier study.

[2]This actually points out a fundamental limitation of case studies. They can be useful for describing a specific process, but they're often incapable of explaining differences among multiple processes because they lack the needed controls.

To do this we conducted the following analyses: (1) we surveyed several AT&T developers to find out how they practiced inspections, and (2) we reanalyzed the field notes we took from the code inspection study to better understand meeting performance.

1.2.1 Surveying Developers

One observation we made from the code inspection study was that high meeting gain rates were strongly correlated with specific developers. That is, if certain developers were on the inspection team, many faults were found at the meeting.

In talking with developers we learned that, even though inspections always involved an initial phase in which they worked individually and a second phase in which they worked as a group, different developers carried out the phases differently. For one set of reviewers the purposes of the two phases were **fault detection** and then **fault collection**. During the detection phase these reviewers individually analyzed the code to discover faults. The collection meeting was then held primarily to inform code unit's author of these discoveries – not necessarily to find more faults.

For other reviewers the purposes were **preparation** and then **inspection**. During the preparation phase these reviewers analyzed the code primarily to familiarize themselves with it, and not necessarily to find faults. This prepared them for the inspection meeting (a group activity), during which they would search for faults.

The obvious implication is that reviewers taking the second approach are likely to generate more meeting gains than those taking the first approach. This raises an important question, "What effect does each approach have on overall inspection performance?" Again, since meetings are expensive, if the approach that deemphasizes meetings works just as well, then that might be the most cost-effective option.

1.2.2 Analyzing the Field Notes

We also realized that if certain developers tend to produce high meeting gains (no matter which team they're on), then some factor related to individual effort might explain differences in gain rates as well as or better than any characteristics of the meetings themselves (e.g., group synergy or teamwork.)

To help us better understand this issue we reanalyzed the field notes that were recorded during the code inspection study. Our goal was to find out how much individual effort or group interaction might account for meeting gains. To examine the effect of synergy, we determined how often a fault was discovered during or immediately after group discussions. This is clearly imprecise since synergy might manifest itself in other ways. (For example, early in a meeting the reviewers might learn something that helps them discover a fault much later in the meeting.) Still, this analysis helps us generate hypotheses even if it doesn't help us prove them.

We were able to gather this data on 40% of the inspections. For these we saw that only 30% of meeting gains grew out of a visible group discussion. Despite the obvious limitations of this analysis, the results suggest that a non-negligible proportion of the faults discovered at an inspection meeting are found simply because individuals are given a second opportunity to inspect the artifact, not because working in groups enables more effective inspection.

This hypothesis led to another important question, "How would inspection performance be affected if the time given to meetings were devoted instead to additional individual analysis?"

1.3 Inspection Methods

The previous discussion suggests three methods for inspecting software.

- **Preparation-Inspection (PI).** Each reviewer individually analyzes the artifact in order to become familiar with it. The goal is not to discover faults but only to prepare for the inspection meeting. After all reviewers have completed this Preparation the team holds an Inspection meeting to find as many faults as possible.

- **Detection-Collection (DC).** Each reviewer individually analyzes the artifact with the goal of Detecting as many faults as possible. As with the PI approach, the team then meets (the Collection phase) to inspect the document. The results of the collection phase will, of course, contain many faults already found during the detection phase.

- **Detection-Detection (DD).** Each reviewer individually analyzes the artifact with the goal of Detecting as many faults as possible. After all reviewers complete this first Detection phase, each is asked to conduct fault Detection a second time, again individually, and again with the goal of detecting as many faults as possible. This approach does not involve a meeting, and instead the time is used by the reviewers to continue working individually.

1.4 Hypothesis

We hypothesize that inspection meetings are not nearly as cost-beneficial as many people believe; and that inspection methods that eliminate meetings (the DD method) are at least as cost-effective than methods that rely heavily on them (the PI and DC methods) and probably more so, and that we expect to see this result because we expect the benefit of additional individual analysis (as provided by the DD method) to be equal to or greater than holding inspection meetings.

2 The Experiment

To evaluate these hypotheses we designed and conducted a controlled experiment. The goals of this experiment were twofold: to characterize the behavior of existing approaches and to assess the potential benefits of meetingless inspections. We ran the experiment in the spring of 1995 with 21 subjects – students taking a graduate course in software engineering – who acted as reviewers. Each complete run consisted of (1) a training phase in which the subjects were taught inspection methods and the experimental procedures, and in which they inspected a sample SRS; and (2) an experimental phase in which the subjects conducted two inspections.

2.1 Experimental Design

2.1.1 Variables

The experiment manipulates four independent variables:

1. the inspection method used by each reviewer (PI, DC, or DD);
2. the inspection round (each reviewer participating in two inspections during the experiment);
3. the specification to be inspected (two are used during the experiment);
4. the order in which the specifications are inspected (Either specification can be inspected first.)

The inspection method is our treatment variable. The other variables allow us to assess several potential threats to the experiment's internal validity. For each inspection we measure three dependent variables:

1. the Individual Fault Detection Rate,

Round/Specification

Inspection Method	Round 1		Round 2	
	WLMS	CRUISE	WLMS	CRUISE
PI	G	B, D		E, F
DC	F		C	A
DD	A, E	C	D	G

Table 1: This table shows the settings of the independent variables. Each team inspects two documents, the WLMS and CRUISE, one per round, using one of the three inspection methods.

2. the Team Fault Detection Rate, [3]

3. the Gain Rate, i.e., the percentage of faults initially identified during the second phase of the inspection.(For the PI and DC methods the second phase is the team meeting; for the DD method it is the second individual detection activity.)

These calculations are explained in Section 3.

2.1.2 Design

The purpose of this experiment is to compare the PI, DC, and DD methods for inspecting software requirements specifications.

When comparing multiple treatments, experimenters frequently use fractional factorial designs. These designs systematically explore all combinations of the independent variables, allowing extraneous factors such as team ability, specification quality, and learning to be measured and eliminated from the experimental analysis.

Had we used such a design, each team would have participated in three inspection rounds, reviewing each of three specifications and using each of three methods exactly once. The order in which the methods are applied and in which the specifications are inspected would have been dictated by the experimental design.

However, to keep the duration of the experiment short, we chose a partial factorial design in which each team participates in two inspections, using some combination of the three inspection methods. Table 1 shows the settings of the independent variables.

[3]The Team and the Individual Fault Detection Rates are the number of faults detected by a team or individual divided by the total number of faults known to be in the specification. The closer these values are to 1, the more effective the detection method. No faults were intentionally seeded into the specifications. All faults are naturally occurring.

2.1.3 Threats to Internal Validity

A potential problem in any experiment is that some factor may affect the dependent variable without the researcher's knowledge. This possibility must be minimized. We considered four such threats: (1) selection effects, (2) maturation effects, (3) instrumentation effects, and (4) presentation effects.

Selection effects are caused by natural variation in human performance. For example, random assignment of subjects may accidentally create an elite team. Therefore, the difference in this team's natural ability will mask differences in the performances of the detection methods.

Our strategy is to assign teams and inspection methods on a completely random basis. This approach attempts to spread differences in natural ability across the inspection methods in an unbiased fashion. However, since each team uses only two of the three inspection methods, differences in the methods can't be completely separated from differences in ability.

Maturation effects are caused by subjects learning as the experiment proceeds. We have manipulated the inspection method used and the order in which the documents are inspected so that the presence of this effect can be discovered and taken into account.

Presentation effects can occur if inspecting one specification first makes it easier to inspect the remaining one. We control for this possibility by having half the teams inspect the documents in each of the two possible orders.

Finally, instrumentation effects may result from differences in the specification documents. Such variation is impossible to avoid, but we controlled for it by having each team inspect both documents.

As we will show in Section 3, variation in the fault detection rate is not explained by selection, maturation, or presentation effects.

2.1.4 Threats to External Validity

Threats to external validity limit our ability to generalize the results of our experiment to industrial practice. We identified three such threats:

1. The subjects in our experiment may not be representative of software programming professionals. Although more than half of the subjects have 2 or more years of industrial experience, they are graduate students, not software professionals. Furthermore, as students they may have different motivations for participating in the experiment.

2. The specification documents we used may not be representative of real programming problems. Our experimental specifications are atypical of industrial SRS in two ways. First, most of the experimental specification is written in a formal requirements notation (see Section 2.2). Although several groups at AT&T and elsewhere are experimenting with formal notations [2, 5], it is not the industry's standard practice. Second, the specifications used are considerably shorter than industrial specifications.

3. The inspection process in our experimental design may not be representative of software development practice. We have modeled our experiment's inspection process after the ones used in many development organizations, although each organization may adapt the process to fit its specific needs. Another difference is that the SRS authors are not present at our inspections, although in practice they normally would be. Finally, industrial reviewers may bring more domain knowledge to an inspection than our student subjects did.

To surmount these threats we will need to replicate our experiment using software professionals to inspect industrial work products. Nevertheless, laboratory experimentation is a necessary first step because it greatly reduces the risk of transferring immature technology and it is far less costly than using professional subjects while we refine our experimental design.

2.1.5 Analysis Strategy

Our analysis strategy has several steps. The first step is to find those independent variables that individually explain a significant amount of the variation in the team detection rate. The second step is to evaluate the combined effect of the variables shown to be significant in the initial analysis. Both analyses use standard analysis of variance methods (see [3], pp. 165*ff* and 210*ff* or [6]). Once these relationships were discovered and their magnitude estimated, we examined other data, such as the gain rates, that would confirm or reject (if possible) a causal relationship between the inspection methods and inspection performance.

2.2 Experiment Instrumentation

We used several instruments for this experiment: three small software requirements specifications (SRS), instructions for each inspection method, and a data collection form. (These specifications were originally developed for another experiment. See Porter et al.[12].)

2.2.1 Software Requirements Specifications

The SRSs we used describe three event-driven process control systems: an elevator control system (ELEVATOR), a water level monitoring system (WLMS), and an automobile cruise control system (CRUISE). Each specification has four sections: Overview, Specific Functional Requirements, External Interfaces, and a Glossary. The overview is written in natural language, while the other three sections are specified using the SCR tabular requirements notation [7].

For this experiment, all three documents were adapted to adhere to the IEEE suggested format [8]. All faults present in these SRS appear in the original documents or were generated during adaptation; no faults were intentionally seeded into the document. The authors discovered 42 faults in the WLMS SRS and 26 in the CRUISE SRS. The authors did not inspect the ELEVATOR SRS since it was used only for training exercises.

Elevator Control System [15] describes the functional and performance requirements of a system for monitoring the operation of a bank of elevators (16 pages).

Water Level Monitoring System [13] describes the functional and performance requirements of a system for monitoring the operation of a steam generating system (24 pages).

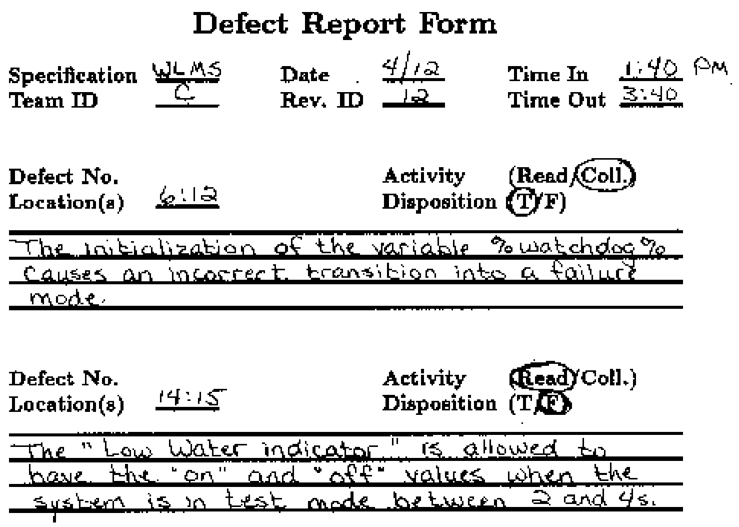
Automobile Cruise Control System [9] describes the functional and performance requirements for an automobile cruise control system (31 pages).

2.2.2 Fault Reporting Forms

We also developed a Fault Report Form. Whenever a potential fault was discovered – during either the fault detection or the collection activities – an entry was made on this form. The entry included four kinds of information: Inspection Activity (Detection or Collection), Fault Location (Page and Line Numbers), Fault Disposition (Faults can be True Faults or False Positives), and Fault Description (in prose). A small sample of a Fault Report appears in Figure 1.

2.3 Experiment Preparation

The participants were given two lectures of 75 minutes each on software requirements specifications, the SCR tabular requirements notation, inspection procedures, the fault classification scheme, and the filling out of data



Defect Report Form

Specification	WLMS	**Date**	4/12	**Time In**	1:40 PM
Team ID	C	**Rev. ID**	12	**Time Out**	3:40

Defect No.		**Activity**	(Read/[✓ Coll.])
Location(s)	6:12	**Disposition**	([✓ T]/F)

The initialization of the variable %watchdog% causes an incorrect transition into a failure mode.

Defect No.		**Activity**	([✓ Read]/Coll.)
Location(s)	14:15	**Disposition**	(T/[✓ F])

The "Low Water indicator" is allowed to have the "on" and "off" values when the system is in test mode between 2 and 4s.

Figure 1: **Reviewer Fault Report Form.** This is a small sample of the fault report form completed by each reviewer.

collection forms. The references for these lectures were Fagan [4], Parnas [10], and the IEEE Guide to Software Requirements Specifications [1]. The participants were then divided into three-person teams – (see Section 2.1.3 for details.) Within each team, members were randomly assigned to act as the moderator, the recorder, or the reader during the collection meeting.

2.4 Conducting the Experiment

2.4.1 Training

For the training exercise, each team inspected the ELEVATOR SRS. Individual team members read the specification and recorded all faults they found on a Fault Report Form. Their efforts were restricted to two hours. Later we met with the participants and answered questions about the experimental procedures. The ELEVATOR SRS was not used in the remainder of the experiment.

2.4.2 Experimental Phase

The experimental phase involved two inspection rounds. The instruments used were the WLMS and CRUISE specifications discussed in Section 2.2.1, and the Fault Report Form.

During the first Round, three of the seven teams were asked to inspect the CRUISE specification; the remaining four teams inspected the WLMS specification. Each inspection involved two phases that differed according to the method used. The inspection methods used by each team are shown in Table 1.

The first phase for the Detection-Collection and Detection-Detection methods lasted up to 2.5 hours, and all potential faults were reported on the Fault Report Form. The first phase for the Preparation-Inspection method took the same amount of time, but the reviewers were not allowed to report faults or take any notes. After the first phase all materials were collected.[4]

Once a team's members had finished the first phase, the team moderator arranged for the second phase which was also limited to 2.5 hours. This second phase involved a team meeting for the PI and DC methods during which the reader paraphrased each requirement, and the reviewers brought up any issues they had found earlier or had just discovered. The team recorder maintained the team's master Fault Report Form. The DD team did not hold a second meeting. Their second phase was exactly the same as the first – individual fault detection. The entire first Round was completed in one week. The second Round was similar to the first except that teams who had inspected the WLMS during Round 1 inspected the CRUISE in Round 2 and vice versa.

3 Data and Analysis

3.1 Data

Two sets of data are important to our study: the Individual Fault Summaries and the Team Fault Summaries.

An individual fault summary shows whether a reviewer discovered a particular fault. This data is gathered from the fault report forms the reviewers completed during fault detection.

A team fault summary shows whether a team discovered a particular fault. For the PI and DC methods this data is gathered from the fault report forms filled out at the collection meetings. For the DD method the team summary is just the set union of faults recorded by all reviewers. This data is used to assess the effectiveness of each fault detection method. Figure 5 depicts the team summaries, showing the fault detection rate and inspection method for each team.

One problem with the team summaries is that the DC and PI methods involve meetings, but the DD method

[4]For each round, we set aside 14 2-hour time slots during which inspection tasks could be done. Participants performed each task within a single two-hour session and were not allowed to work at other times.

Rev.	Activity	Sum	1	2		21		32		41	42
18	Detection	5	0	0		1		0		0	0
19	Detection	6	0	1	•••	0	•••	0	•••	0	0
20	Detection	2	0	0		0		0		0	0
Team	Collection	7	0	1		0		1		0	0

Figure 2: **Data Collection for each WLMS inspection.** This figure shows the data collected from one team's WLMS inspection (using the DC inspection method). The first three rows identify the review team members, the inspection methods they used, the number of faults they found, and shows their individual fault summaries. The fourth row contains the team fault summary. The fault summaries show a 1 (0) where the team or individual found (did not find) a fault. Meeting gain and loss rates can be calculated by comparing the individual and team fault summaries. For instance, fault 21 is an example of *meeting loss.* It was found by reviewer 18 during the fault detection activity, but the team did not report it at the collection meeting. Fault 32 is an example of *meeting gain*; it was first discovered at the collection meeting.

Rev.	Activity	Sum	1	2		14		17		25	26
1	Detection-1	4	0	0		0		0		0	0
2	Detection-1	1	0	1		0		0		0	0
3	Detection-1	3	0	0		0		1		0	0
1	Detection-2	1	0	0	•••	0	•••	1	•••	0	0
2	Detection-2	4	0	1		1		0		0	0
3	Detection-2	3	0	0		0		0		0	0
Team		11	0	1		1		1		0	0

Figure 3: **Data Collection for each CRUISE inspection.** This figure shows the data collected from one team's CRUISE inspection (using the DD inspection method). The data is identical to that of the WLMS inspections except that the CRUISE has fewer faults – 26 for the CRUISE versus 42 for the WLMS.

doesn't. Consequently, any meeting losses – faults discovered by individuals before the meeting that don't appear in the team fault report – will lower the fault detection rate for these two methods but not, of course, for the DD method. However, since meeting losses average only about 5% for the DC method (we can't measure them for the PI method) we consider them to be insignificant. One team's individual and team fault summaries are represented in Figure 2 and Figure 3.

We also compare the benefits of meetings (DC method) with the benefits of additional individual analysis (DD method). For the DC method we examine the individual and the team summaries to calculate meeting gains. For the DD method we examine the individual summaries to determine whether faults are originally discovered during the first or the second detection activity.

Our analysis is done in three steps: (1) We compared the team fault detection rates to ascertain whether the inspection methods have the same effectiveness. (2) We separately compared the first and second round

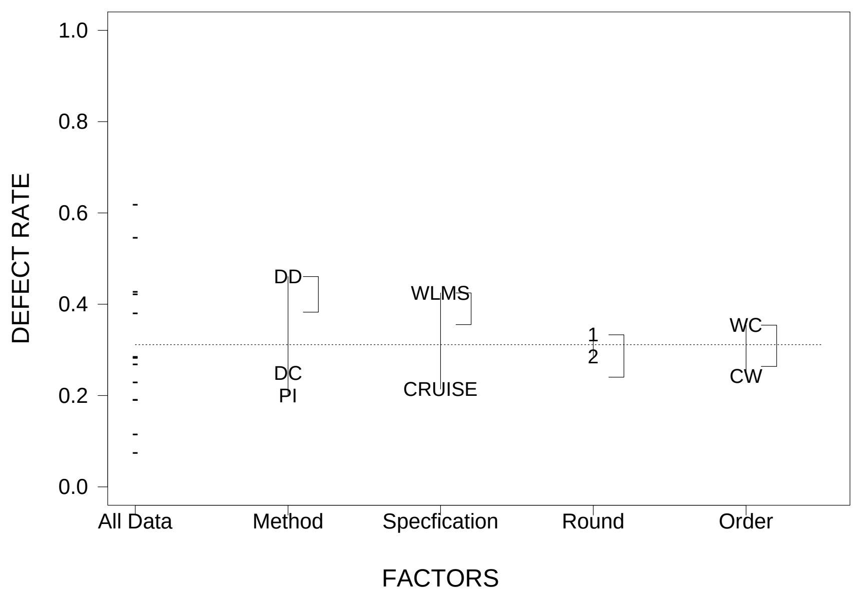


Figure 4: **Fault Detection Rates by Independent Variable.** The dashes in the far left column show each team's fault detection rate for the WLMS and CRUISE. The horizontal line is the average fault detection rate. The plot demonstrates the ability of each variable to explain variation in the fault detection rates. For the Specification variable, the vertical location of WLMS (CRUISE) is determined by averaging the fault detection rates for all teams inspecting WLMS (CRUISE). The vertical bracket,], to the right of each variable shows one standard error of the difference between two settings of the variable.

performance of individuals and teams to see how different treatments performed. (3) We analyzed the individual fault summaries determine whether different treatments found different faults.

3.2 Analysis of Team Performance

If our hypothesis that inspections without meetings are no less than effective inspection with meetings is true, we shouldn't find that the DD method's performances are significantly lower than those of the other two methods. Our analysis strategy is to determine whether various threats to the experiment's internal validity can be seen. And then to compare performances.

First we analyze each independent variable's contribution to the observed inspection performance data. Table 2 and Figure 4 show that Inspection Method and Specification are significant, but the Round, Order, and Team Composition are not.

Next, we analyzed the combined Instrumentation and Treatment effects. Table 3 shows the input to this analysis. Six of the cells contain the average detection rate for teams using each inspection method and specification

Independent Variable	SS_T	ν_T	SS_R	ν_R	$(SS_T/\nu_T)(\nu_R/SS_R)$	Significance Level
Detection Method – treatment	.19	2	.13	10	7.34	.01
Specification– instrumentation	.14	1	.17	11	9.250	.01
Inspection round – maturation	.007	1	.31	11	.026	.62
Order – presentation	.013	1	.240	10	.54	.48
Team composition – selection	.15	6	.16	6	.912	.54

Table 2: **Analysis of Variance for Each Independent Variable.** This analysis of variance shows that only the choice of detection method and specification significantly explain variation in the fault detection rate.

Specification	**Inspection Method**		
	PI	**DC**	**DD**
WLMS	.38	.29	.62 .29 .43 .55
(average)	.38	.29	.47
Cruise	.19 .12 .23 .077	.27 .19	.42
(average)	.15	.23	.42

Table 3: **Team Fault Detection Rate Data.** This table shows the nominal and average fault detection rates for all 7 teams. There are only 13 observations, however, since one team dropped out because of a team member's illness.

(3 detection methods applied to 2 specifications). The results indicate that the interaction between Method and Specification is not significant.

Finally, we compared the performance of each method. Figure 5 summarizes this data. As depicted, the DD inspection method resulted in the highest fault detection rates (46%), followed by the DC inspection method (23%), and finally by the PI detection method (19%).

3.3 Analysis of Second Phase Performance

The previous analysis shows that in the current experiment, inspections without meetings seem more effective than inspections with meetings. In this section we examine whether the current data support our original hypothesis that meetingless inspections would be at least as effective because the benefits of having a meeting wouldn't outweigh the benefits of additional individual analysis.

Our analysis strategy is to isolate first and second phase performances to see how well they explain differences in total inspection performance. If the hypothesis is true we should see little difference between the first phase performances of the DC and DD methods, but significant difference in the second phase performances.

Figure 6 contains a boxplot showing the number of faults found by each reviewer during the first phase of the DD and DC inspections. The rates for both the WLMS and the CRUISE appear to be similar.

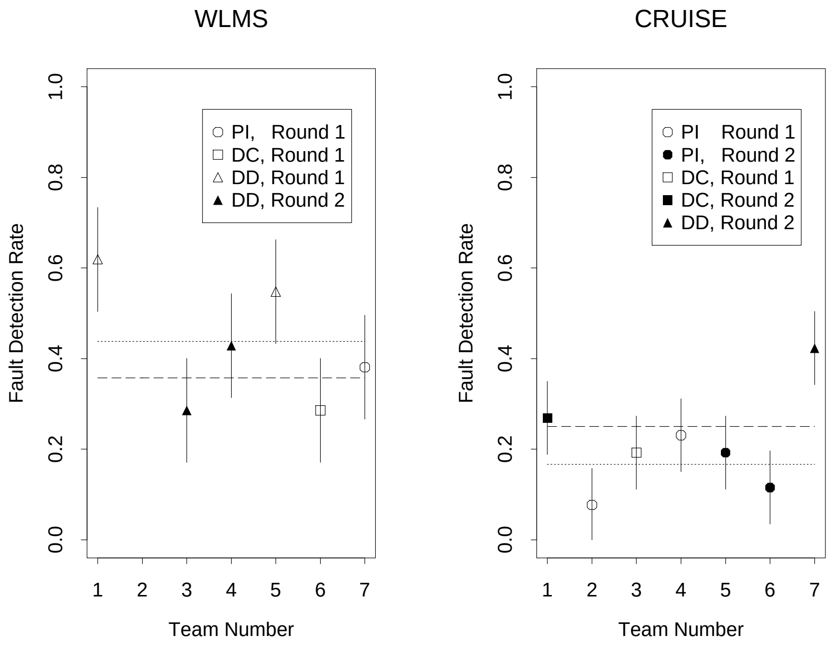


Figure 5: **Fault detection by inspection method.** The observed fault detection rates are displayed above. The unfilled symbols indicate observations from Round 1; the filled symbols those from round 2. The vertical line through each point indicates one standard deviation in the rate's estimate (modeling fault detection as Bernoulli trials). The dashed (dotted) lines display the average detection rates for Round 1 (Round 2).

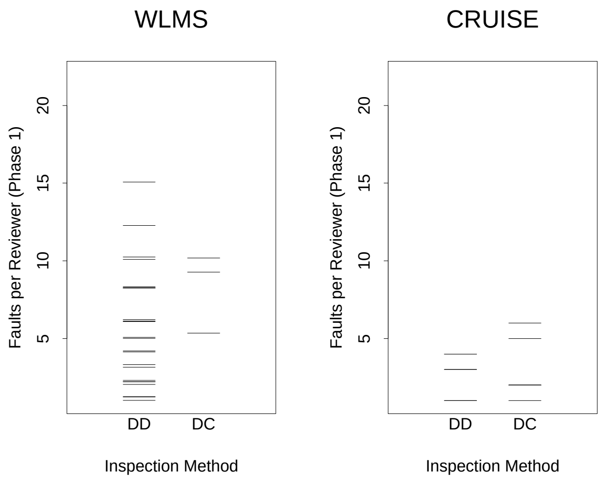


Figure 6: **Individual Fault Detection Rate (Phase 1).** This figure shows the number faults discovered by each reviewer during the first phase of a DD or DC inspection.

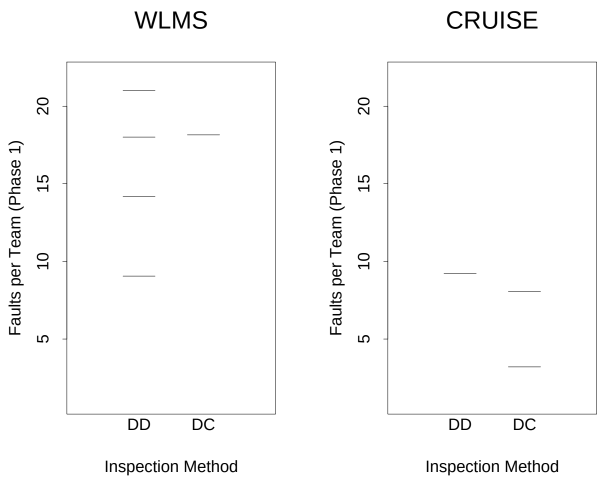


Figure 7: **Team Fault Detection Rate (Phase 1).** This figure shows the number faults discovered by each team during the first phase of a DD or DC inspection.

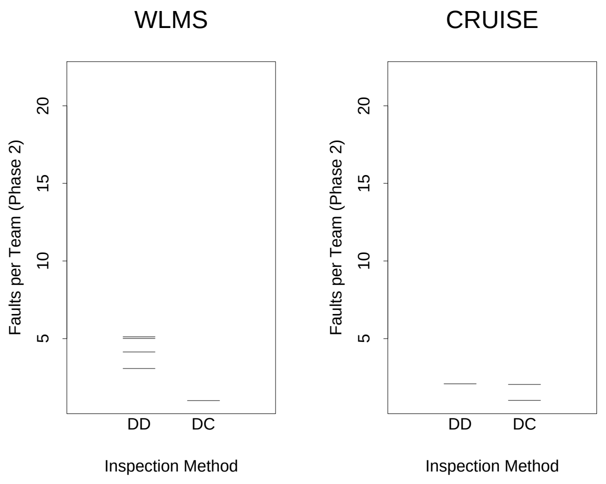


Figure 8: **Team Fault Detection Rate (Phase 2).** This figure shows the number of faults discovered by each team during the second phase of a DD or DC inspection.

Figure 7 shows the number of unique faults discovered by each team during the first phase of a DD or DC inspection. This is the set union of faults found by all team members. For the WLMS the rates again appear to be similar, but for the CRUISE the data is inconclusive.

Figure 8 shows the number of unique faults discovered by each team during the second phase of a DC or DD inspection. For the WLMS the rates may be higher for the DD method than for the DC method, but the rates for the CRUISE are again inconclusive.

3.4 Analysis of Detection Rates for Specific Faults

Even if inspections with meetings are less effective than inspections without meetings, team meetings might still be beneficial if they promote detection of classes of faults that individuals seldom or never find. This section analyzes the individual and team fault summaries to determine whether the current data supports the hypothesis that this is indeed the case.

If this hypothesis is true, then there should be a subset of faults for which the probability of their being found by the DC and PI methods is greater than by the DD method.

For example, 9 faults in the WLMS had a higher detection probability with meetings than without. There were 3 such faults in the CRUISE. We want to know whether a subset of these faults can be considered "meeting sensitive".

If we knew which faults should be meeting sensitive, we could test whether they are so in this experiment. For example, if we had a classification scheme that could distinguish meeting sensitive from meeting insensitive faults we could separate each specification's faults into two populations and then compare the detection probabilities of each population using each inspection method. The faults suspected of being meeting sensitive should have a higher probability of being found at a meeting.

Since at present we don't have any such classification scheme we can't do this and therefore our approach is to look for "markers" or "indicators" that are associated with the presence or absence of meeting sensitive faults. For example, if the detection probabilities of meeting sensitive faults are statistically distinguishable from those of meeting insensitive faults, then we can identify those faults which are very likely or very unlikely to be meeting sensitive.

To estimate these detection probabilities we conducted a Monte Carlo simulation of inspections under a variety of different conditions (e.g., different proportions of total faults that are meeting sensitive). The outcome of each simulation run is the expected detection probabilities for meeting sensitive and insensitive faults when the inspection is conducted with and without meetings.

This information allows us to answer three questions.

1. If none of the faults are meeting sensitive, what is the inter-treatment difference in detection probabilities?

2. As the proportion of meeting sensitive faults grows, how does this affect the inter-treatment difference in detection probabilities?

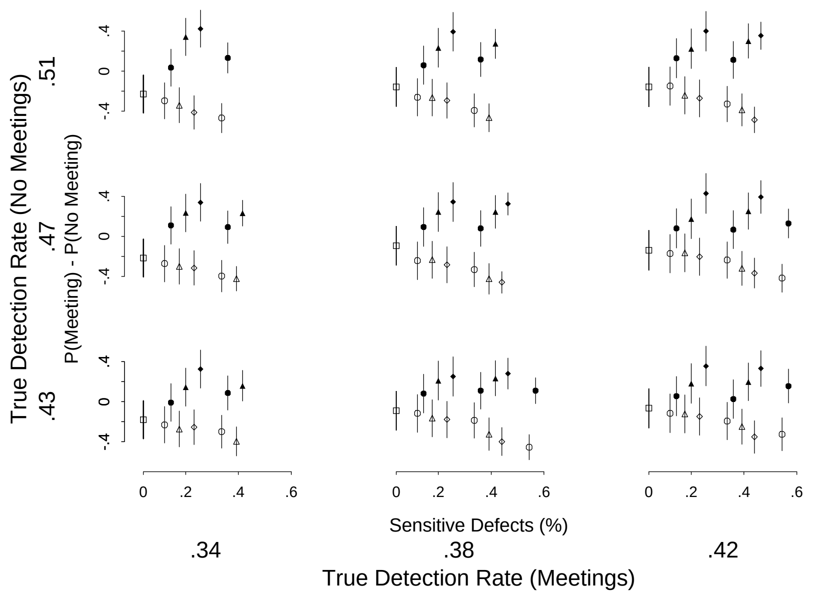


Figure 9: **Inter-treatment Differences for Sensitive and Insensitive Faults. (WLMS)** This figure shows a 3 × 3 matrix of subplots, with each row corresponding to a fixed value of p_a (.43, .47, or .51) and each column to a fixed value of p_b (.34, .38, and .42). The cells in the matrix contain some of the simulation results for each combination of p_a and p_b. Within each cell the y-axis encodes the inter-treatment differences for several fault populations. Each of these populations has both a meeting sensitive and meeting insensitive subpopulations and is defined by two parameters: the proportion of meeting sensitive faults in the population and the detection probability for meeting sensitive faults. The proportion of meeting sensitive faults (0, .2, .4, or .6) is encoded on the subplot's x-axis. Different detection probabilities are plotted using differently shaped symbols (a square for $pa \times 1$, an octagon for $pa \times 1.25$, a triangle for $pa \times 1.5$, and a diamond for $pa \times 1.75$). The results for the meeting sensitive subpopulation are plotted with filled symbols and open symbols are used for the meeting insensitive subpopulation. The line segments running through each symbol mark one standard deviation in the detection rate's estimate. Since the detection probabilities of the meeting sensitive and meeting insensitive faults are constrained by the total detection probability some combinations are mathematically impossible. In these cases no symbols are plotted.

3. At what magnitude does the inter-treatment difference in detection probabilities become statistically significant?

Each run simulates using two treatments, T_a and T_b, to inspect either the WLMS or CRUISE specification. The fault detection probabilities for each treatment are p_a and p_b and are based on the probabilities p_{DD} and $p_{DC,PI}$ observed during the experiment.

The simulation manipulates five independent variables:

1. the specification (WLMS or CRUISE),

2. the detection rate for T_a. ($p_a = p_{DD}$ + { -4, -2, 0, 2, or 4 } $\times factor$, where $factor = \frac{1}{42} = 0.023$ for the WLMS and $factor = \frac{1}{26} = 0.038$ for the CRUISE),

3. the detection rate for T_b. ($p_b = p_{DC,PI}$ + { -4, -2, 0, 2, or 4 } $\times factor$),

4. the proportion of faults that are meeting sensitive (0, .1, .2, .25, .4, .5, .6, .75, .9 or 1 }),

5. the detection rate for meeting sensitive defects ($p_{sens} = p_a \times$ {1.0, 1.1, 1.2, 1.25, 1.4, 1.5, 1.6, 1.75, 1.9, or 2}. Note that the detection probability for insensitive faults, p_{insens}, is now constrained by the values of p_b and p_{sens}.),

For each combination of independent variables we simulate 50 inspections of a single specification. Half of the inspections use treatment T_a; the other half use treatment T_b. For T_a, each fault's detection probability is estimated by the proportion of successes in a random draw from a Bernoulli distribution with parameters 25 and p_a. A similar approach is used for T_b, except that meeting sensitive faults use a Bernoulli distribution with parameters 25 and p_{sens} and the insensitive faults use parameters 25 and p_{insens}. The difference between the detection probability for a given fault when using T_a and when using T_b is called the inter-treatment difference:

For each run we calculate two dependent variables:

1. the average inter-treatment difference for meeting sensitive faults,

2. the average inter-treatment difference for meeting insensitive faults.

Figures 9 and 10 show some of the simulation results, about which we make several observations. (1) The larger the difference between p_a and p_b, the smaller the proportion of meeting sensitive faults can be, but the easier it will be to differentiate sensitive faults from insensitive ones. (2) The meeting sensitivity will have to be roughly $p_a \times 1.5$ for a statistically significant difference to be detectable. (3) In many cases the two populations of meeting sensitive and insensitive faults will be statistically indistinguishable from a population having no meeting sensitive faults.

4 Conclusions

This article describes the first run of a planned multi-trial experiment whose goal is to resolve the conflicting results of two earlier industrial case studies. One of these studies found that inspection meetings consistently

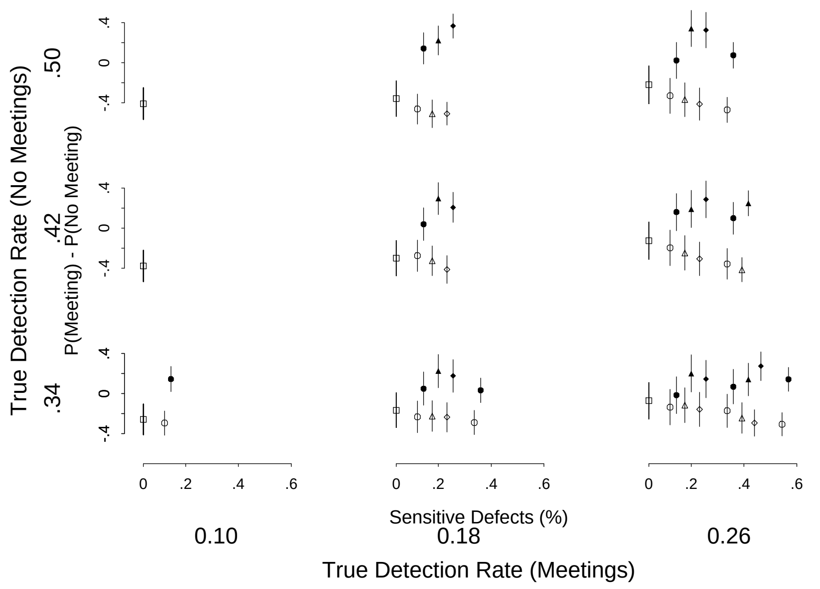


Figure 10: **Inter-treatment Differences for Sensitive and Insensitive Faults. (CRUISE)** For detailed explanation of the plot see Figure 9.

produced few gains, but the other found just the opposite – the gain rates varied widely and could be quite high. This situation illustrates that every empirical study is at best an approximation, needs to be checked against previous observations, and differences resolved through continued experimentation.

While attempting to understand the causes of these discrepancies we uncovered anecdotal evidence that pointed to two possible explanations: (1) differences in the type of artifact being inspected (design documents vs. code units) led to the use of different "implicit" inspection processes, and (2) faults found at the meeting might be explained by factors other than meeting synergy or teamwork. Initially we are concentrating on the second explanation. Our current observation is that the inspection method used can't be ignored as a significant source of variation in the meeting gain rates.

In addition to comparing two inspection methods that involved meetings (DC and PI), we also included a control treatment that had no inspection meeting (DD) . These meetingless inspections detected more new faults in the second phase of the inspection than the DC PI methods and more total faults than either.

From the perspective of the software practitioner, this outcome suggests that meetings are not necessarily essential to successful inspections and that further study is warranted.

Another result of conducting and analyzing the initial experimental run is confidence that we are collecting the appropriate data for evaluating our hypotheses. We are also confident that the experimental design is adequately controlling any threats to internal validity.

Our primary concern at this time is the need for more data. To address this concern we will replicate the experiment this fall. We are also constructing a laboratory manual to help other researchers replicate the experiment as well.

References

[1] *IEEE Guide to Software Requirements Specifications.* Soft. Eng. Tech. Comm. of the IEEE Computer Society, 1984. IEEE Std 830-1984.

[2] Mark A. Ardis. Lessons from using basic lotos. In *Sixteenth International Conference on Software Engineering*, pages 5–14, Sorrento, Italy, May 1994.

[3] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters.* John Wiley & Sons, New York, 1978.

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

[5] S. Gerhart, D. Craigen, and T. Ralston. Experience with formal methods in critical systems. *IEEE Software*, 11(1):21–28, January 1994.

[6] R. M. Heiberger. *Computation for the Analysis of Designed Experiments.* Wiley & Sons, New York, New York, 1989.

[7] Kathryn L. Heninger. Specifying Software Requirements for Complex Systems: New Techniques and their Application. *IEEE Transactions on Software Engineering*, SE-6(1):2–13, January 1980.

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

[9] J. Kirby. Example NRL/SCR software requirements for an automobile cruise control and monitoring system. Technical Report TR-87-07, Wang Institute of Graduate Studies, July 1984.

[10] 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.

[11] Adam Porter, Harvey Siy, Carol A. Toman, and Lawrence G. Votta. An experiment to assess the cost-benefits of code inspections in large scale software development. In *ACM SIGSOFT Software Engineering Notes*, volume 20, October 1995.

[12] Adam Porter, Lawrence G. Votta, and Victor Basili. Comparing detection methods for software requirement inspections: A replicated experim ent. *IEEE Transactions on Software Engineering*, 21(6):563–575, June 1995.

[13] J. vanSchouwen. The A-7 requirements model: Re-examination for real-time systems and an application to monitoring systems. Technical Report TR-90-276, Queen's University, Kingston, Ontario, Canada, May 1990.

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

[15] William G. Wood. Temporal logic case study. Technical Report CMU/SEI-89-TR-24, Software Engineering Institute, Pittsburgh, PA, August 1989.