Comparing Detection Methods For Software Requirements Inspections: A Replicated Experiment

Adam A. Porter Lawrence G. Votta, Jr. Victor R. Basili

Abstract— **Software requirements specifications (SRS) are often validated manually. One such process is inspection, in which several reviewers independently analyze all or part of the specification and search for faults. These faults are then collected at a meeting of the reviewers and author(s).**

Usually, reviewers use Ad Hoc or Checklist methods to uncover faults. These methods force all reviewers to rely on nonsystematic techniques to search for a wide variety of faults. We hypothesize that a Scenario-based method, in which each reviewer uses different, systematic techniques to search for different, specific classes of faults, will have a significantly higher success rate.

We evaluated this hypothesis using a 3×2^4 partial factorial, randomized experimental design. Forty eight graduate students in computer science participated in the experiment. They were assembled into sixteen, three-person teams. Each team inspected two SRS using some combination of Ad Hoc, Checklist or Scenario methods.

For each inspection we performed four measurements: (1) individual fault detection rate, (2) team fault detection rate, (3) percentage of faults first identified at the collection meeting (meeting gain rate), and (4) percentage of faults first identified by an individual, but never reported at the collection meeting (meeting loss rate).

The experimental results are that (1) the Scenario method had a higher fault detection rate than either Ad Hoc or Checklist methods, (2) Scenario reviewers were more effective at detecting the faults their scenarios are designed to uncover, and were no less effective at detecting other faults than both Ad Hoc or Checklist reviewers, (3) Checklist reviewers were no more effective than Ad Hoc reviewers, and (4) Collection meetings produced no net improvement in the fault detection rate – meeting gains were offset by meeting losses.

Keywords— **Controlled Experiments, Technique and Methodology Evaluation, Inspections, Reading Techniques**

I. Introduction

One way of validating a software requirements specification (SRS) is to submit it to an inspection by a team of reviewers. Many organizations use a three-step inspection procedure for eliminating faults : detection, collection, and repair[1]. [1], [3] A team of reviewers reads the SRS, identifying as many faults as possible. Newly identified faults are collected, usually at a team meeting, and then sent to the document's authors for repair.

We are focusing on the methods used to perform the first step in this process, fault detection. For this article, we define a fault detection method to be a set of fault detection techniques coupled with an assignment of responsibilities to individual reviewers.

Fault detection techniques may range in prescriptiveness from intuitive, nonsystematic procedures, such as Ad Hoc or Checklist techniques, to explicit and highly systematic procedures, such as formal proofs of correctness.

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

These individual responsibilities may be coordinated among the members of a review team. When they are not coordinated, all reviewers have identical responsibilities. In contrast, the reviewers in coordinated teams may have separate and distinct responsibilities.

In practice, reviewers often use Ad Hoc or Checklist detection techniques to discharge identical, general responsibilities. Some authors, notably Parnas and Weiss[4], have argued that inspections would be more effective if each reviewer used a different set of systematic detection techniques to discharge different, specific responsibilities.

Until now, however, there have been no reproducible, quantitative studies comparing alternative detection methods for software inspections. We have conducted such an experiment and our results demonstrate that the choice of fault detection method significantly affects inspection performance. Furthermore, our experimental design may be easily replicated by interested researchers.

Below we describe the relevant literature, several alternative fault detection methods which motivated our study, our research hypothesis, and our experimental observations, analysis and conclusions.

A. Inspection Literature

A summary of the origins and the current practice of inspections may be found in Humphrey [1]. Consequently, we will discuss only work directly related to our current efforts.

Fagan[5] defined the basic software inspection process. While most writers have endorsed his approach[6], [1], Parnas and Weiss are more critical [4]. In part, they argue that effectiveness suffers because individual reviewers are

This work is supported in part by the National Aeronautics and Space Administration under grant NSG–5123. Porter and Basili are with the Department of Computer Science, University of Maryland, College Park, Maryland 20472. Votta is with the Software Production Research Department, AT&T Bell Laboratories Naperville, IL 60566

[1] Depending on the exact form of the inspection, they are sometimes called reviews or walkthroughs. For a more thorough description of the taxonomy see [1] pp. 171*ff* and [2].

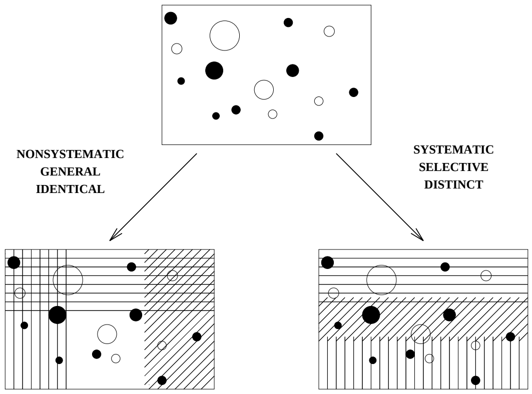


Fig. 1. **Systematic Inspection Research Hypothesis.** This figure represents a software requirements specification before and after a *nonsystematic* technique, *general* and *identical* responsibility inspection and a *systematic* technique, *specific* and *distinct* responsibility inspection. The points and holes represent various faults. The line-filled regions indicate the coverage achieved by different members of the inspection team. Our hypothesis is that systematic technique, specific and coordinated responsibility inspections achieve broader coverage and minimize reviewer overlap, resulting in higher fault detection rates and greater cost benefits than nonsystematic methods.

not assigned specific responsibilities and because they lack systematic techniques for meeting those responsibilities.

Some might argue that Checklists are systematic because they help define each reviewer's responsibilities and suggest ways to identify faults. Certainly, Checklists often pose questions that help reviewers discover faults. However, we argue that the generality of these questions and the lack of concrete strategies for answering them makes the approach nonsystematic.

To address these concerns – at least for software designs – Parnas and Weiss introduced the idea of active design reviews. The principal characteristic of an active design review is that each individual reviewer reads for a specific purpose, using specialized questionnaires. This proposal forms the motivation for the detection method proposed in Section II-B.2.

B. Detection Methods

Ad Hoc and Checklist methods are two frequently used fault detection methods. With Ad Hoc detection methods, all reviewers use nonsystematic techniques and are assigned the same general responsibilities.

Checklist methods are similar to Ad Hoc, but each reviewer receives a checklist. Checklist items capture important lessons learned from previous inspections within an environment or application. Individual checklist items may enumerate characteristic faults, prioritize different faults, or pose questions that help reviewers discover faults, such as "Are all interfaces clearly defined?" or "If input is received at a faster rate than can be processed, how is this handled?" The purpose of these items is to focus reviewer responsibilities and suggest ways for reviewers to identify faults.

C. Hypothesis

We believe that an alternative approach which gives individual reviewers specific, orthogonal detection responsibilities and specialized techniques for meeting them will result in more effective inspections.

To explore this alternative we developed a set of fault-specific techniques called Scenarios – collections of procedures for detecting particular classes of faults. Each reviewer executes a single scenario and multiple reviewers are coordinated to achieve broad coverage of the document.

Our underlying hypothesis is depicted in Figure 1: that nonsystematic techniques with general reviewer responsibility and no reviewer coordination, lead to overlap and gaps, thereby lowering the overall inspection effectiveness; while systematic approaches with specific, coordinated responsibilities reduce gaps, thereby increasing the overall effectiveness of the inspection.

II. The Experiment

To evaluate our systematic inspection hypothesis we designed and conducted a multi-trial experiment. The goals of this experiment were twofold: to characterize the behavior of existing approaches and to assess the potential benefits of Scenario-based methods. We ran the experiment twice; once in the Spring of 1993, and once the following Fall. Both runs used 24 subjects – students taking a graduate course in formal methods 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 monitored inspections.

A. Experimental Design

The design of the experiment is somewhat unusual. To avoid misinterpreting the data it is important to understand the experiment and the reasons for certain elements of its design [2].

[2] See Judd, et al. [7], chapter 4 for an excellent discussion of randomized social experimental designs.

		Round/Specification			
		Round 1		Round 2	
		WLMS	CRUISE	WLMS	CRUISE
Detection Method	ad hoc	1B, 1D, 1G 1H, 2A	1A, 1C, 1E 1F, 2D	1A	1D, 2B
	checklist	2B	2E, 2G	1E, 2D, 2G	1B, 1H
	scenarios	2C, 2F	2H	1F, 1C, 2E 2H	1G, 2A, 2C 2F

TABLE I

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 detection methods. Teams from the first replication are denoted 1A–1H, teams from the second replication are denoted 2A–2H.

A.1 Variables

The experiment manipulates five independent variables:

1. the detection method used by a reviewer (Ad Hoc, Checklist, or Scenario);
2. the experimental replication (we conducted two separate replications);
3. the inspection round (each reviewer participates in two inspections during the experiment);
4. the specification to be inspected (two are used during the experiment).
5. the order in which the specifications are inspected (either specification can be inspected first).

The detection 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 four dependent variables:

1. the individual fault detection rate,
2. the team fault detection rate [3],
3. the percentage of faults first identified at the collection meeting (meeting gain rate), and
4. the percentage of faults first identified by an individual, but never reported at the collection meeting (meeting loss rate).

A.2 Design

The purpose of this experiment is to compare the Ad Hoc, Checklist, and Scenario detection 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 the specifications are inspected would have been dictated by the experimental design.

Such designs are unacceptable for this study because they require some teams to use the Ad Hoc or Checklist method after they have used the Scenario method. Since the Ad Hoc and Checklist reviewers create their own fault detection techniques during the inspection (based on their experience or their understanding of the checklist), our concern was that using the Scenario method in an early round might imperceptibly distort the use of the other methods in later rounds. Such influences would be undetectable because, unlike the Scenario methods, the Ad Hoc and Checklist methods do not require reviewers to perform specific, auditable tasks.

We chose a partial factorial design in which each team participates in two inspections, using some combination of the three detection methods, but teams using the Scenario method in the first round must continue to use it in the second round. Table I shows the settings of the independent variables.

A.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 five such threats: (1) selection effects, (2) maturation effects, (3) replication effects, (4) instrumentation effects, and (5) presentation effects.

Selection effects are due to 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 detection method performance. Two approaches are often taken to limit this effect:

1. Create teams with equal skills. For example, rate each participant's background knowledge and experience as either low, medium, or high and then form teams of three by selecting one individual at random from each experience category. Detection methods are then assigned to fit the needs of the experiment.
2. Compose teams randomly, but require each team to use all three methods. In this way, differences in team

[3] The team and 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 that value is to 1, the more effective the detection method. No faults were intentionally seeded into the specifications. All faults are naturally occurring.

skill are spread across all treatments.

Neither approach is entirely appropriate. Although we used the first approach in our initial replication, the approach is unacceptable for multiple replications, because even if teams within a given replication have equal skills, teams from different replications will not. As discussed in the previous section, the second approach is also unsuitable because using the Scenarios in the first inspection Round will certainly bias the application of the Ad Hoc or Checklist methods in the second inspection Round.

Our strategy for the second replication and future replications is to assign teams and detection methods on a random basis. However, teams that used Scenarios in the first round were constrained to use them again in the second round. This compromise provides more observations of the Scenario method and prevents the use of the Scenario method from affecting the use of the Ad Hoc or Checklist methods. However we can't determine whether or not the teams that used only the Scenarios have greater natural ability than the other teams.

Maturation effects are due to subjects learning as the experiment proceeds. We have manipulated the detection 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.

Replication effects are caused by differences in the materials, participants, or execution of multiple replications. We limit this effect by using only first and second year graduate students as subjects - rather than both undergraduate and graduate students. Also, we maintain consistency in our experimental procedures by packaging the experimental procedures as a classroom laboratory exercise. This helps us to ensure that similar steps are followed for all replications. As we will show in Section III, variation in the fault detection rate is not explained by selection, maturation, or replication effects.

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.

A.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 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 II-B.) Although several groups at AT&T and elsewhere are experimenting with formal notations [8], [9], it is not the industry's standard practice. Secondly, the specifications are considerably smaller than industrial ones.
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 one used in several development organizations within AT&T [10]. Although this process is similar to a Fagan-style inspection, there are some differences. One difference is that reviewers use the fault detection activity to to find faults, not just to prepare for the inspection meeting. Another difference is that during the collection meeting reviewers are given specific technical roles such as test expert or end-user only if the author feels there is a special need for them.

   Our process also differs slightly from the AT&T process. For example, the SRS authors are not present at our collection meetings, although, in practice, they normally would be. Also, industrial reviewers may bring more domain knowledge to an inspection than our student subjects did.

To surmount these threats we are currently replicating 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.

A.5 Analysis Strategy

Our analysis strategy had two steps. The first step was to find those independent variables that individually explain a significant amount of the variation in the team detection rate. The second step was 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 [11], pp. 165*ff and 210ff* or [12]). Once these relationships were discovered and their magnitude estimated, we examined other data, such as correlations between the categories of faults detected and the detection methods used that would confirm or reject (if possible) a causal relationship between detection methods and inspection performance.

B. Experiment Instrumentation

We developed several instruments for this experiment: three small software requirements specifications (SRS), instructions and aids for each detection method, and a data collection form.

B.1 Software Requirements Specifications

The SRS we used describe three event-driven process control systems: an elevator control system, a water level monitoring system, and an automobile cruise control system. Each specification has four sections: Overview, Specific Functional Requirements, External Interfaces, and a Glossary. The overview is written in natural language,

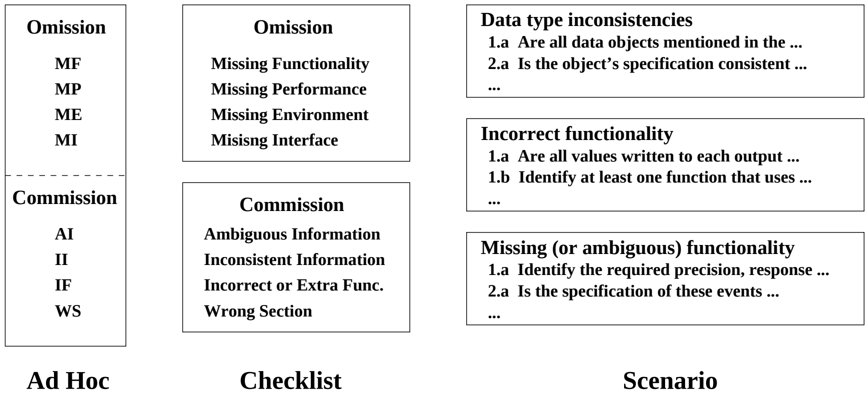


Fig. 2. **Relationship Between Fault Detection Methods.** The figure depicts the relationship between the fault detection methods used in this study. The vertical extent represents the coverage. The horizontal axis labels the method and represents the degree of detail (the greater the horizontal extent the greater the detail). Moving from Ad Hoc to Checklist to Scenario there is more detail and less coverage. The gaps in the Scenario and Checklist columns indicate that the Checklist is a subset of the Ad Hoc and the Scenarios are a subset of the Checklist.

while the other three sections are specified using the SCR tabular requirements notation [13].

For this experiment, all three documents were adapted to adhere to the IEEE suggested format [2]. All faults present in these SRS appear in the original documents or were generated during the adaptation process; 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.

B.1.a Elevator Control System (ELEVATOR). [14] describes the functional and performance requirements of a system for monitoring the operation of a bank of elevators (16 pages).

B.1.b Water Level Monitoring System (WLMS). [15] describes the functional and performance requirements of a system for monitoring the operation of a steam generating system (24 pages).

B.1.c Automobile Cruise Control System (CRUISE). [16] describes the functional and performance requirements for an automobile cruise control system (31 pages).

B.2 Fault Detection Methods

To make a fair assessment of the three detection methods (Ad Hoc, Checklist, and Scenario) each method should search for a well-defined population of faults. To accomplish this, we used a general fault taxonomy to define the responsibilities of Ad Hoc reviewers.

The checklist used in this study is a refinement of the taxonomy. Consequently, Checklist responsibilities are a subset of the Ad Hoc responsibilities.

The Scenarios are derived from the checklist by replacing individual Checklist items with procedures designed to implement them. As a result, Scenario responsibilities are distinct subsets of Checklist and Ad Hoc responsibilities. The relationship between the three methods is depicted in Figure 2.

The taxonomy is a composite of two schemes developed by Schneider, et al. [17] and Basili and Weiss [18]. Faults are divided into two broad types: omission – in which important information is left unstated and commission – in which incorrect, redundant, or ambiguous information is put into the SRS by the author. Omission faults were further subdivided into four categories: Missing Functionality, Missing Performance, Missing Environment, and Missing Interface. Commission faults were also divided into four categories: Ambiguous Information, Inconsistent Information, Incorrect or Extra Functionality, and Wrong Section. (See Appendix A for complete taxonomy.) We provided a copy of the taxonomy to each reviewer. Ad Hoc reviewers received no further assistance.

Checklist reviewers received a single checklist derived from the fault taxonomy. To generate the checklist we populated the fault taxonomy with detailed questions culled from several industrial checklists. Thus, the checklist items are similar in style to those found in several large organizations. All Checklist reviewers used the same checklist. (See Appendix B for the complete checklist.)

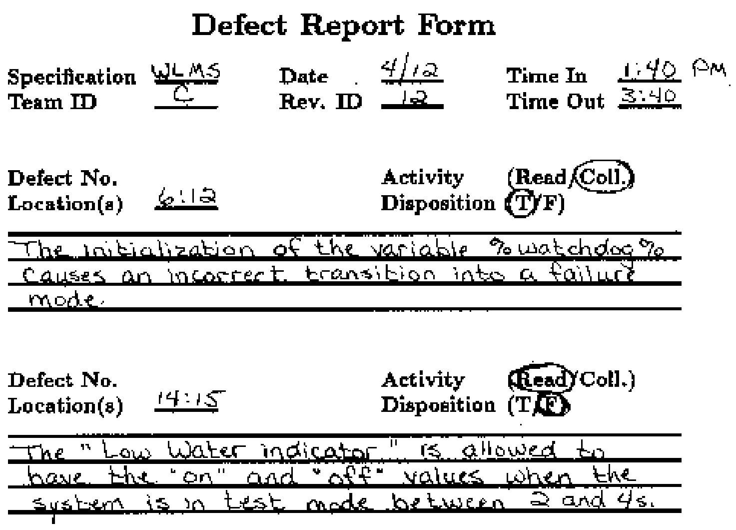
Finally, we developed three groups of Scenarios. Each group of Scenarios was designed for a specific subset of the Checklist items:

1. Data Type Inconsistencies (DF),
2. Incorrect Functionalities (IF),
3. Missing or Ambiguous Functionalities (MF).

After the experiment was finished we applied the Scenarios ourselves to estimate how broadly they covered the WLMS and CRUISE faults (i.e., what percentage of defects could be found if the Scenarios are properly applied.) We estimated that the Scenarios address about half of the faults that are covered by the Checklist. Appendix C contains the complete list of Scenarios.

B.3 Fault Report Forms

We also developed a Fault Report Form. Whenever a potential fault was discovered – during either the fault de-



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.

Fig. 3. **Reviewer Fault Report Form.** This is a small sample of the fault report form completed during each reviewer's fault detection. Faults number 10 and 11, found by reviewer 12 of team C for the WLMS specification are shown.

tection or the collection activities – an entry was made on the form. The entry included four kinds of information: Inspection Activity (Detection, Collection); Fault Location (Page and Line Numbers); Fault Disposition, (Faults can be True Faults or False Positives); and a prose Fault Description. A small sample of a Fault Report appears in Figure 3.

C. Experiment Preparation

The participants were given two, 75 minute lectures on software requirements specifications, the SCR tabular requirements notation, inspection procedures, the fault classification scheme, and the filling out of data collection forms. The references for these lectures were Fagan [5], Parnas [4], and the IEEE Guide to Software Requirements Specifications [19]. The participants were then assembled into three-person teams – see Section II-A.3 for details. Within each team, members were randomly assigned to act as the moderator, the recorder, or the reader during the collection meeting.

D. Conducting the Experiment

D.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. Afterwards, each team conducted a supervised collection meeting and filled out a master Fault Report Form for the entire team. The ELEVATOR SRS was not used in the remainder of the experiment.

D.2 Experimental Phase

This phase involved two inspection rounds. The instruments used were the WLMS and CRUISE specifications discussed in Section II-B.1, a checklist, three groups of fault-based scenarios, and the Fault Report Form. The development of the checklist and scenarios is described in Section II-B.2. The same checklist and scenarios were used for both documents.

During the first Round, four of the eight teams were asked to inspect the CRUISE specification; the remaining four teams inspected the WLMS specification. The detection methods used by each team are shown in Table I. Fault detection was limited to two hours, and all potential faults were reported on the Fault Report Form. After fault detection, all materials were collected.[4]

Once all team members had finished fault detection, the team's moderator arranged for the collection meeting. At the collection meeting, the reader paraphrases each requirement. During this paraphrasing activity, reviewers may bring up any issues found during preparation or discuss new issues. The team's recorder maintained the team's master Fault Report Form. Collection was also limited to 2 hours and the entire Round was completed in one week. The collection meeting process is the same regardless of which fault detection method was used during fault detection.

[4] For each round, we set aside 14 two-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	Method	Sum	1	2		21		32		41	42
42	Data inconsistency	9	0	0		0		0		0	0
43	Incorrect functionality	6	0	1	•••	0	•••	0	•••	0	0
44	Missing functionality	18	0	0		1		0		0	0
Team	Scenario	23	0	1		0		1		0	0
Key			AH	DT		MA		AH		DT	AH

Fig. 4. **Data Collection for each WLMS inspections.** This figure shows the data collected from one team's WLMS inspection. The first three rows identify the review team members, the detection 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. The fifth row contains the fault key which identifies those reviewers who were responsible for the fault (AH for Ad Hoc only; CH for Checklist or Ad Hoc; DT for data type inconsistencies, Checklist, and Ad Hoc; IF for incorrect functionality, Checklist and Ad Hoc; and MF for missing or ambiguous functionality, Checklist and Ad Hoc). 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 44 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 is first discovered at the collection meeting.

Rev	Method	Sum	1	2		14		17		25	26
42	Ad Hoc	7	0	1		0		0		1	0
43	Ad Hoc	6	0	1	•••	0	•••	0	•••	1	0
44	Ad Hoc	4	0	0		0		0		0	0
Team	Ad Hoc	10	0	1		1		0		1	0
Key			AH	MF		AH		AH		AH	DT

Fig. 5. **Individual and Team Fault Summaries (CRUISE).** This figure shows the data collected from one team's CRUISE inspection. The data is identical to that of the WLMS inspections except that the CRUISE has fewer faults – 26 versus 42 for the WLMS – and the fault key is different.

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.

III. Data and Analysis

A. Data

Three sets of data are important to our study: the fault key, the team fault summaries, and the individual fault summaries.

The fault key encodes which reviewers are responsible for each fault. In this study, reviewer responsibilities are defined by the detection techniques a reviewer uses. Ad Hoc reviewers are responsible (asked to search for) for all faults. Checklist reviewers are responsible for a large subset of the Ad Hoc faults[5]. Since each Scenario is a refinement of several Checklist items, each Scenario reviewer[6] is responsible for a distinct subset of the Checklist faults.

The team fault summary shows whether or not a team discovered a particular fault. This data is gathered from the fault report forms filled out at the collection meetings and is used to assess the effectiveness of each fault detection method.

The individual fault summary shows whether or not a reviewer discovered a particular fault. This data is gathered from the fault report forms each reviewer completed during their fault detection activity. Together with the fault key it is used to assess whether or not each detection technique improves the reviewer's ability to identify specific classes of faults.

We measure the value of collection meetings by comparing the team and individual fault summaries to determine the meeting gain and loss rates. One team's individual and team fault summaries, and the fault key are represented in Figures 4 and Figure 5.

Our analysis is done in three steps: (1) We compared the team fault detection rates to determine whether the detection methods have the same effectiveness. (2) We analyzed the performances of individual reviewers to understand why some methods performed better than others. (3) Finally, we analyzed the effectiveness of collection meetings to further understand differences in each method's performance.

B. Analysis of Team Performance

Figure 6 summarizes the team performance data. As depicted, the Scenario detection method resulted in the highest fault detection rates, followed by the Ad Hoc detection method, and finally by the Checklist detection method.

Table II presents a statistical analysis of the team performance data as outlined in Section II-A.5. The independent variables are listed from the most to the least significant. The Detection method and Specification are significant, but the Round, Replication, and Order are not.

[5] i.e., faults for which an Ad Hoc reviewer is responsible.
[6] i.e., reviewers using Scenarios.

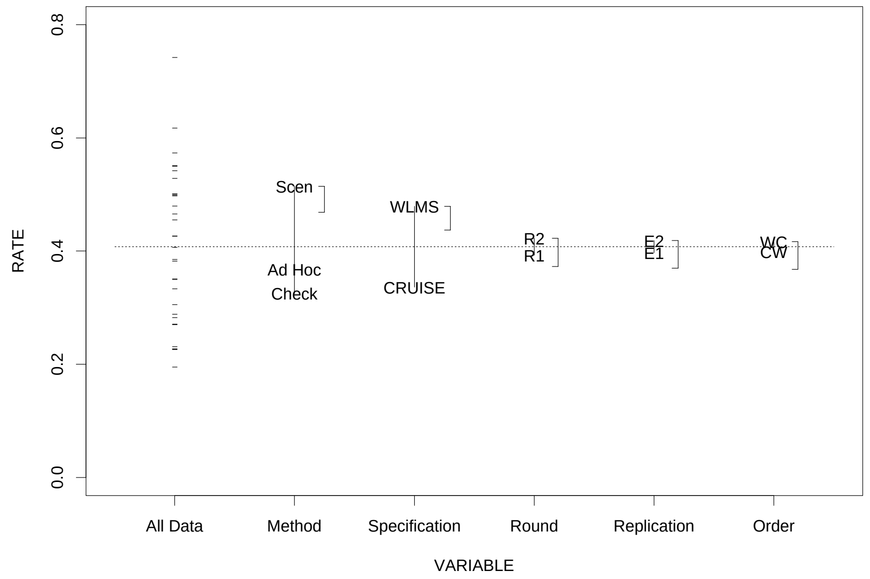


Fig. 6. **Fault Detection Rates by Independent Variable.** The dashes in the far left column show each team's fault detection rate for the WLMS and CRUISE. The horizontal line is the average fault detection rate. The plot demonstrates the ability of each variable to explain variation in the fault detection 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. The plot indicates that both the Method and Specification are significant; but Round, Replication, and Order are not.

Independent Variable	SS_T	ν_T	SS_R	ν_R	$(SS_T/\nu_T)(\nu_R/SS_R)$	Significance Level
Detection Method – treatment	.200	2	.359	29	8.064	$< .01$
Specification– instrumentation	.163	1	.396	30	12.338	$< .01$
Inspection round – maturation	.007	1	.551	30	.391	.54
Experimental run – replication	.007	1	.551	30	.391	.54
Order – presentation	.003	1	.003	30	.141	.71
Team composition – selection	.289	15	.268	16	1.151	.39

TABLE II
Analysis of Variance for Each Independent Variable. The analysis of variance shows that only the choice of detection method and specification significantly explain variation in the fault detection rate. Team composition is also not significant.

Next, we analyzed the combined Instrumentation and Treatment effects. Table III shows the input to this analysis. Six of the cells contain the average detection rate for teams using each detection method and specification (3 detection methods applied to 2 specifications). The results of this analysis, shown in Table IV, indicate that the interaction between Specification and Method is not significant. This means that although the average detection rates varied for the two specifications, the effect of the detection methods is not linked to these differences. Therefore, we reject the null hypothesis that the detection methods have no effect on inspection performance.

C. Effect of Scenarios on Individual Performance

We initially hypothesized that increasing the specialization and coordination of each reviewer's responsibilities would improve team performance. We proposed that the Scenario would be one way to achieve this. We have shown above that the teams using Scenarios were the most effective. However, this did not establish that the improvement was due to increases in specialization and coordination, and not to some other factor.

Some alternative explanations for the observed improvement could be (1) the Scenario reviewers responded to some perceived expectation that their performance should improve; or (2) the Scenario approach improves individual

Specification	Detection Method		
	Ad Hoc	**Checklist**	**Scenario**
WLMS	.5 .38 .29 .5 .48 .45	.29 .52 .5 .33	.74 .57 .55 .4 .62 .55
(average)	.43	.41	.57
Cruise	.46 .27 .27 .23 .38 .23 .35	.19 .31 .23 .23	.5 .42 .42 .54 .35
(average)	.31	.24	.45

TABLE III

Team Fault Detection Rate Data. The nominal and average fault detection rates for all 16 teams.

Effect	SS_T	ν_T	SS_R	ν_R	$(SS_T/\nu_T)(\nu_R/SS_R)$	Significance Level
Detection Method	.200	2	.212	26	12.235	$< .01$
Specification	.143	1	.212	26	17.556	$< .01$
Meth$\times$Spec	.004	2	.212	26	.217	.806

TABLE IV

Analysis of Variance of Detection Method and Specification. This table displays the results of an analysis of the variance of the average detection rates given in Table III.

performance regardless of Scenario content. Consequently, our concern is to determine exactly how the use of Scenarios affected the reviewer's performance. To examine this, we formulated two hypothesis schemas.

- **H1: Method X reviewers do not find any more X faults than do method Y reviewers.**
- **H2: Method X reviewers find either a greater or smaller number of non X faults than do method Y reviewers.**

C.1 Rejecting the Perceived Expectation Argument

If Scenario reviewers performed better than Checklist and Ad Hoc reviewers on both scenario-targeted and non-scenario-targeted faults, then we must consider the possibility that their improvement was caused by something other than the scenarios themselves.

One possibility was that the Scenario reviewers were merely reacting to the novelty of using a clearly different approach, or to a perceived expectation on our part that their performance **should** improve. To examine this we analyzed the individual fault summaries to see how Scenario reviewers differed from other reviewers.

The detection rates of Scenario reviewers are compared with those of all other reviewers in Tables V, VI, VII and VIII. Using the one and two-sided Wilcoxon-Mann-Whitney tests [20], we found that in most cases Scenario reviewers were more effective than Checklist or Ad Hoc reviewers at finding the faults the scenario was designed to uncover. At the same time, all reviewers, regardless of which detection method each used, were equally effective at finding those faults not targeted by any of the Scenarios.

Since Scenario reviewers could not have known the fault classifications, it is unlikely that their reporting could have been biased. Therefore these results suggest that the detection rate of Scenario reviewers shows improvement only with regard to those faults for which they are explicitly responsible. Consequently, the argument that the Scenario reviewers' improved performance was primarily due to raised expectations or unknown motivational factors is not supported by the data.

C.2 Rejecting the General Improvement Argument

Another possibility is that the Scenario approach rather than the content of the Scenarios was responsible for the improvement.

Each Scenario targets a specific set of faults. If the reviewers using a type X Scenario had been no more effective at finding type X faults than had reviewers using non-X Scenarios, then the content of the Scenarios did not significantly influence reviewer performance. If the reviewers using a type X Scenario had been more effective at finding non-X faults than had reviewers using other Scenarios, then some factor beyond content caused the improvement. To explore these possibilities we compared the Scenario reviewers' individual fault summaries with each other.

Looking again at Tables V, VI, VII, and VIII we see that each group of Scenario reviewers was the most effective at finding the faults their Scenarios were designed to detect, but was generally no more effective than other Scenario reviewers at finding faults their Scenarios were not designed to detect. Since Scenario reviewers showed improvement in finding only the faults for which they were explicitly responsible, we conclude that the content of the Scenario was primarily responsible for the improved reviewer performance.

D. Analysis of Checklists on Individual Performance

The scenarios used in this study were derived from the checklist. Although this checklist targeted a large number of existing faults, our analysis shows that the perfor-

Reviewers Using Method		Finding Faults of Type		Compared with Reviewers using Method				
Detection Method	Number Reviewers	Fault Population	Number Present	DT	MF	IF	CH	AH
DT	6	DT	14	- (6.5)	.02 (3)	.06 (4.5)	.01 (4)	.02 (4)
MF	6	MF	5	.07 (0.5)	- (2)	.12 (1)	.02 (0)	.04 (1)
IF	6	IF	5	.01 (0)	.01 (1)	- (1.5)	.04 (1)	.01 (1)
CH	12	CH	38	.95 (10.5)	.86 (11)	.89 (12.5)	- (8)	.51 (10)
AH	18	AH	42	.91 (12)	.84 (12.5)	.75 (13)	.37 (9.5)	- (11)

TABLE V

Significance Table for H1 hypotheses: WLMS inspections. This table tests the H1 hypothesis - Method X reviewers do not find any more X faults than do method Y reviewers - for all pairs of detection methods. Each row in the table corresponds to a population of reviewers and the population of faults for which they were responsible, i.e., method X reviewers and X faults. The last five columns correspond to a second reviewer population, i.e., method Y reviewers. Each cell in the last five columns contains two values. The first value is the probability that H1 is true, using the one-sided Wilcoxon-Mann-Whitney test. The second value – in parentheses – is the median number of faults found by the method Y reviewers.

Reviewers Using Method		Finding Faults of Type		Compared with Reviewers using Method				
Detection Method	Number Reviewers	Fault Population	Number Present	DT	MF	IF	CH	AH
DT	5	DT	10	- (6)	.05 (3)	.03 (2)	$< .01$ (1)	.02 (3)
MF	5	MF	1	NA (0)	- (0)	NA (0)	NA (0)	NA (0)
IF	5	IF	3	NA (0)	NA (0)	- (0)	NA (0)	NA (0)
CH	12	CH	24	$> .99$ (8)	.95 (5)	.93 (4)	- (2.5)	.98 (5)
AH	21	AH	26	.96 (8)	.50 (5)	.41 (5)	.02 (3)	- (5)

TABLE VI

Significance Table for H1 hypotheses: CRUISE inspections. This analysis is identical to that performed for WLMS inspections. However, we chose not to perform any statistical analysis for the Missing Functionality and Incorrect Functionality faults because there are too few faults of those types.

mance of Checklist teams were no more effective than Ad Hoc teams. One explanation for this is that nonsystematic techniques are difficult for reviewers to implement.

To study this explanation we again tested the H1 hypothesis that Checklist reviewers were no more effective than Ad Hoc reviewers at finding Checklist faults. From Tables V and VI we see that even though the Checklist targets a large number of faults, it does not actually improve a reviewer's ability to find those faults.

E. Analysis of Collection Meetings

In his original paper on software inspections Fagan [5] asserts that

> Sometimes flagrant errors are found during ... [fault detection], but in general, the number of errors found is not nearly as high as in the ... [collection meeting] operation.

From a study of over 50 inspections, Votta [3] collected data that strongly contradicts this assertion. In this Section, we measure the benefits of collection meetings by comparing the team and individual fault summaries to determine the meeting gain and meeting loss rates. (See Figure 4 and Figure 5).

A "meeting gain" occurs when a fault is found for the first time at the collection meeting. A "meeting loss" occurs when a fault is first found during an individual's fault

Reviewers Using Method		Finding Faults of Type		Compared with Reviewers using Method				
Detection Method	Number Reviewers	Fault Population	Number Present	DT	MF	IF	CH	AH
DT	6	DT^c	28	- (4.5)	.92 (9)	.82 (7.5)	.50 (5.5)	.64 (6)
MF	6	MF^c	37	.87 (11)	- (9.5)	.83 (12.5)	.56 (8.5)	.64 (10)
IF	6	IF^c	37	.66 (11)	.53 (12)	- (11.5)	.24 (8.5)	.27 (10)
CH	12	CH^c	4	.12 (0.5)	.28 (1)	.35 (1)	- (1)	.07 (1)
AH	18	AH^c	0	NA (0)	NA (0)	NA (0)	NA (0)	– (0)

TABLE VII

Significance Table for H2 hypothesis: WLMS inspections. This table tests the H2 hypothesis - Method X reviewers find a greater or smaller number of non X faults than do method Y reviewers - for all pairs of detection methods. Each row in the table corresponds to a population of reviewers and the population of faults for which they were **not** responsible - i.e., method X reviewers and non X faults (the complement of the set of X faults). The last five columns correspond to a second reviewer population, i.e., method Y reviewers. Each cell in the last five columns contains two values. The first value is the probability that H2 is true, using the two-sided Wilcoxon-Mann-Whitney test. The second value is the median number of faults found by the method Y reviewers.

Reviewers Using Method		Finding Faults of Type		Compared with Reviewers using Method				
Detection Method	Number Reviewers	Fault Population	Number Present	DT	MF	IF	CH	AH
DT	5	DT^c	16	- (2)	.59 (2)	.86 (3)	.37 (2)	.46 (2)
MF	5	MF^c	25	.96 (8)	– (5)	.33 (4)	.06 (3)	.62 (5)
IF	5	IF^c	23	.96 (8)	.41 (4)	– (5)	.44 (2.5)	.57 (5)
CH	12	CH^c	2	NA (0)	NA (1)	NA (0)	– (0)	NA (0)
AH	21	AH^c	0	NA (0)	NA (0)	NA (0)	NA (0)	– (0)

TABLE VIII

Significance Table for H2 hypothesis: CRUISE inspections. This analysis is identical to that performed for WLMS inspections. However, we chose not to perform statistical analysis for the non Checklist faults because there are too few faults of that type.

detection activity, but it is subsequently not recorded during the collection meeting. Meeting gains may thus be offset by meeting losses and the difference between meeting gains and meeting losses is the net improvement due to collection meetings. Our results indicate that collection meetings produce no net improvement.

E.1 Meeting Gains

The meeting gain rates reported by Votta were a negligible $3.9 \pm .7\%$. Our data tells a similar story. (Figure 7 displays the meeting gain rates for WLMS inspections.) The mean gain rate is $4.7 \pm 1.3\%$ for WLMS inspections and $3.1 \pm 1.1\%$ for CRUISE inspections. The rates are not significantly different. It is interesting to note that these results are consistent with Votta's earlier study even though Votta's reviewers were professional software developers and not students.

E.2 Meeting Losses

The average meeting loss rates were $6.8 \pm 1.6\%$ and $7.7 \pm 1.7\%$ for the WLMS and CRUISE respectively. (See Figure 8.) One cause of meeting loss might be that reviewers are talked out of the belief that something is a fault. Another cause may be that during the meeting reviewers forget or can not reconstruct a fault found earlier.

This effect has not been previously reported in the lit-

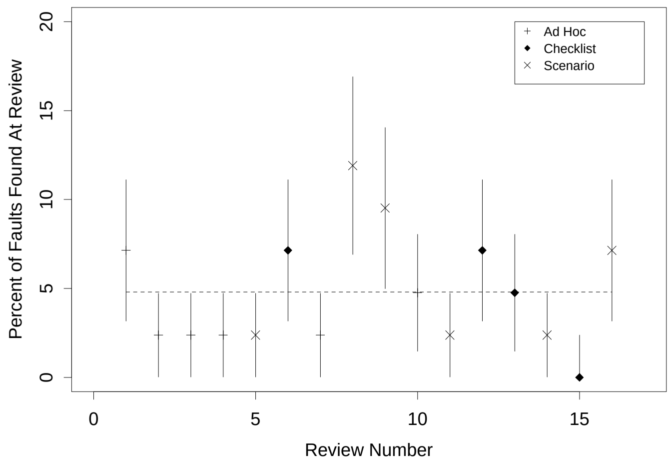


Fig. 7. **Meeting Gains for WLMS Inspections.** Each point represents the meeting gain rate for a single inspection, i.e., the number of faults first identified at a collection meeting divided by the total number of faults in the specification. Each rate is marked with symbol indicating the inspection method used. The vertical line segment through each symbol indicates one standard deviation in the estimate (assuming each fault was a Bernoulli trial). This information helps in assessing the significance of any one rate. The average meeting gain rate is $4.7 \pm 1.3\%$ for the WLMS. ($3.1 \pm 1.1\%$ for the CRUISE.)

erature. However, since the interval between the detection and collection activities is usually longer in practice than it was in our experiment (one to two days in our study versus one or two weeks in practice), this effect may be quite significant.

E.3 Net Meeting Improvement

The average net meeting improvement is $-.9 \pm 2.2$ for WLMS inspections and -1.2 ± 1.7 for CRUISE inspections. (Figure 9 displays the net meeting improvement for WLMS inspections.) We found no correlations between the loss, gain, or net improvement rates and any of our experiment's independent variables.

IV. Summary and Conclusions

Our experimental design for comparing fault detection methods is flexible and economical, and allows the experimenter to assess several potential threats to the experiment's internal validity. In particular, we determined that neither maturation, replication, selection, or presentation effects had any significant influence on inspection performance. However, differences in the SRS did.

From our analysis of the experimental data we drew several conclusions. As with any experiment these conclusions apply only to the experimental setting from which they are drawn. Readers must carefully consider the threats to external validity described in Section II-A.4 before attempting to generalize these results.

1. **The fault detection rate when using Scenarios was superior to that obtained with Ad Hoc or Checklist methods – an improvement of roughly 35%.**
2. **Scenarios helped reviewers focus on specific fault classes.** Furthermore, in comparison to Ad Hoc or Checklist methods, the Scenario method did not compromise their ability to detect other classes of faults. (however, the scenarios appeared to be better suited to the fault profile of the WLMS than the CRUISE. This indicates that poorly designed scenarios may lead to poor inspection performance.)
3. **The Checklist method – the industry standard, was no more effective than the Ad Hoc detection method.**
4. **On the average, collection meetings contributed nothing to fault detection effectiveness.**

The results of this work have important implications for software practitioners. The indications are that overall inspection performance can be improved when individual reviewers use systematic procedures to address a small set of specific issues. This contrasts with the usual practice, in which reviewers have neither systematic procedures nor clearly defined responsibilities.

Economical experimental designs are necessary to allow replication in other environments with different populations. For software researchers, this work demonstrates the feasibility of constructing and executing inexpensive experiments to validate fundamental research recommendations.

V. Future Work

The experimental data raise many interesting questions for future study.

- In many instances a single reviewer found a fault, but the fault was not subsequently recorded at the collection meeting. Are single reviewers sometimes forgetting to mention faults they observed, or is the reviewer being talked out of the fault at the team meeting?

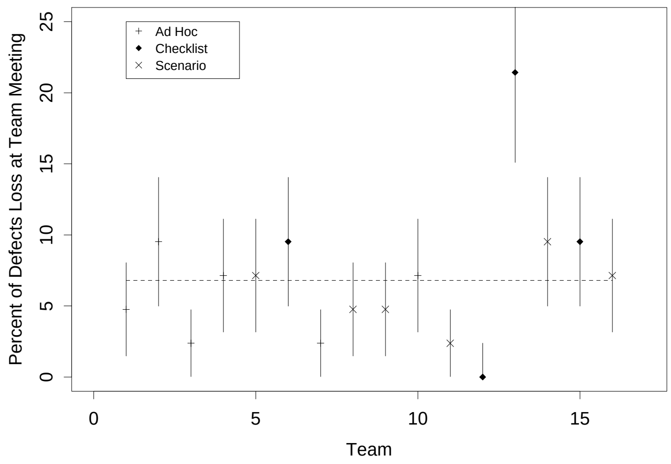


Fig. 8. **Meeting Loss Rate for WLMS Inspections.** Each point represents the meeting loss rate for a single inspection. The meeting loss rate is the number of faults first detected by an individual reviewer divided by the total number of faults in the specification. Each rate is marked with a symbol indicating the inspection method used. The vertical line segment through each symbol indicates one standard deviation in the estimate of the rate (assuming each fault was a Bernoulli trial). This information helps in determining the significance of any one rate. The average team loss rate is $6.8 \pm 1.6\%$ for the WLMS. ($7.7 \pm 1.7\%$ for CRUISE).

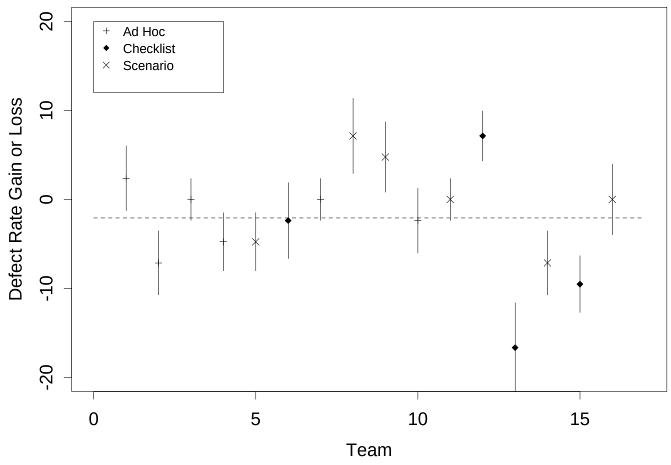


Fig. 9. **Net Meeting Improvement for WLMS.** Each symbol indicates the net meeting improvement for a single inspection. The average net meeting improvement rate is $-.9 \pm 2.2$ for the WLMS. (-1.2 ± 1.7 for the CRUISE). These rates are not significantly different from 0.

What are the significant suppression mechanisms affecting collection meetings?

- Very few faults are initially discovered during collection meetings. Therefore, in view of their impact on development interval (calendar time to complete development), are these meetings worth holding?
- More than half of the faults are not addressed by the Scenarios used in this study. What other Scenarios are necessary to achieve a broader fault coverage?
- There are several threats to this experiment's external validity. These threats can only be addressed by replicating and reproducing these studies. Each new run reduces the probability that our results can be explained by human variation or experimental error. Consequently, we are creating a laboratory kit (i.e., a package containing all the experimental materials, data, and analysis) to facilitate replication. The kit is available via anonymous ftp at ftp.cs.umd.edu.
- Finally, we are using the lab kit to reproduce the experiments with other university researchers in Japan, Germany, Italy, and Australia and with industrial developers at AT&T Bell Laboratories and Motorola Inc. These studies will allow us to evaluate our hypotheses with different populations of programmers and different software artifacts.

Acknowledgments

We would like to recognize the efforts of the experimental participants – an excellent job was done by all. Our thanks to Mark Ardis, John Kelly, and David Weiss, who helped us to identify sample requirements specifications and inspection checklists, and to John Gannon, Richard Gerber, Clive Loader, Eric Slud and Scott VanderWeil for their valuable technical comments. Finally, Art Caso's editing is greatly appreciated.

References

[1] Watts S. Humphery, *Managing the Software Process*, Addison-Wesley Publishing Co., 1989, Reading, Massachusetts.

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

[3] 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.

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

[5] M. E. Fagan, "Design and code inspections to reduce errors in program development", *IBM Systems Journal*, vol. 15, no. 3, pp. 182–211, 1976.

[6] Barry W. Boehm, *Software Engineering Economics*, Prentice Hall, Englewood Cliffs, NJ, 1981.

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

[8] Mark A. Ardis, "Lessons from using basic lotos", in *Proceedings of the Sixteenth International Conference on Software Engineering*, Sorrento, Italy, May 1994, pp. 5–14.

[9] S. Gerhart, D. Craigen, and T. Ralston, "Experience with formal methods in critical systems", *IEEE Software*, vol. 11, no. 1, pp. 21–28, January 1994.

[10] 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*, May 1992, pp. 59–65.

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

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

[13] Kathryn L. Heninger, "Specifying Software Requirements for Complex Systems: New Techniques and their Application", *IEEE Transactions on Software Engineering*, vol. SE-6, no. 1, pp. 2–13, January 1980.

[14] William G. Wood, "Temporal logic case study", Tech. Rep. CMU/SEI-89-TR-24, Software Engineering Institute, Pittsburgh, PA, August 1989.

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

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

[17] G. Michael Schneider, Johnny Martin, and W. T. Tsai, "An experimental study of fault detection in user requirements", *ACM Transactions on Software Engineering and Methodology*, vol. 1, no. 2, pp. 188–204, April 1992.

[18] V. R. Basili and D. M. Weiss, "Evaluation of a software requirements document by analysis of change data", in *Proceedings of the Fifth International Conference on Software Engineering*, San Diego, CA, March 1981, pp. 314–323.

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

[20] S. Siegel and N.J. Castellan, Jr., *Nonparametric Statistics For the Behavioral Sciences*, McGraw-Hill Book Company, New York, NY, second edition, 1988.

Appendix

I. Ad Hoc Detection

The fault taxonomy is due to the work of Schneider, et al., and Basili and Weiss.

- Omission
  - Missing Functionality: Information describing the desired internal operational behavior of the system has been omitted from the SRS.
  - Missing Performance: Information describing the desired performance specifications has either been omitted or described in a way that is unacceptable for acceptance testing.
  - Missing Interface: Information describing how the proposed system will interface and communicate with objects outside the the scope of the system has been omitted from the SRS.
  - Missing Environment: Information describing the required hardware, software, database, or personnel environment in which the system will run has been omitted from the SRS
- Commission
  - Ambiguous Information: An important term, phrase or sentence essential to the understanding of system behavior has either been left undefined or defined in a way that can cause confusion and misunderstanding.
  - Inconsistent Information: Two sentences contained in the SRS directly contradict each other or express actions that cannot both be correct or cannot both be carried out.
  - Incorrect Fact: Some sentence contained in the SRS asserts a facts that cannot be true under the conditions specified in the SRS.
  - Wrong Section: Essential information is misplaced within the SRS

II. Checklist Method

- General
  - Are the goals of the system defined?
  - Are the requirements clear and unambiguous?
  - Is a functional overview of the system provided?
  - Is an overview of the operational modes provided?
  - Have the software and hardware environments been specified?
  - If assumptions that affect implementation have been made, are they stated?
  - Have the requirements been stated in terms of inputs, outputs, and processing for each function?
  - Are all functions, devices, constraints traced to requirements and vice versa?
  - Are the required attributes, assumptions and constraints of the system completely listed?
- Omission
  - Missing Functionality
    * Are the described functions sufficient to meet the system objectives?

- * Are all inputs to a function sufficient to perform the required function?
  * Are undesired events considered and their required responses specified?
  * Are the initial and special states considered (e.g., system initiation, abnormal termination)?
- – Missing Performance
  * Can the system be tested, demonstrated, analyzed, or inspected to show that it satisfies the requirements?
  * Have the data type, rate, units, accuracy, resolution, limits, range and critical values
  * for all internal data items been specified?
  * Have the accuracy, precision, range, type, rate, units, frequency, and volume of inputs and outputs been specified for each function?
- – Missing Interface
  * Are the inputs and outputs for all interfaces sufficient?
  * Are the interface requirements between hardware, software, personnel, and procedures included?
- – Missing Environment
  * Have the functionality of hardware or software interacting with the system been properly specified?
- • Commission
  - – Ambiguous Information
    * Are the individual requirements stated so that they are discrete, unambiguous, and testable?
    * Are all mode transitions specified deterministicly?
  - – Inconsistent Information
    * Are the requirements mutually consistent?
    * Are the functional requirements consistent with the overview?
    * Are the functional requirements consistent with the actual operating environment?
  - – Incorrect or Extra Functionality
    * Are all the described functions necessary to meet the system objectives?
    * Are all inputs to a function necessary to perform the required function?
    * Are the inputs and outputs for all interfaces necessary?
    * Are all the outputs produced by a function used by another function or transferred across an external interface?
  - – Wrong Section
    * Are all the requirements, interfaces, constraints, etc. listed in the appropriate sections.

III. Scenarios

A. Data Type Consistency Scenario

1. Identify all data objects mentioned in the overview (e.g., hardware component, application variable, abbreviated term or function)
   (a) Are all data objects mentioned in the overview listed in the external interface section?
2. For each data object appearing in the external interface section determine the following information:
   - Object name:
   - Class: (e.g., input port, output port, application variable, abbreviated term, function)
   - Data type: (e.g., integer, time, boolean, enumeration)
   - Acceptable values: Are there any constraints, ranges, limits for the values of this object
   - Failure value: Does the object have a special failure value?
   - Units or rates:
   - Initial value:

   (a) Is the object's specification consistent with its description in the overview?
   (b) If object represents a physical quantity, are its units properly specified?
   (c) If the object's value is computed, can that computation generate a non-acceptable value?
3. For each functional requirement identify all data object references:
   (a) Do all data object references obey formatting conventions?
   (b) Are all data objects referenced in this requirement listed in the input or output sections?
   (c) Can any data object use be inconsistent with the data object's type, acceptable values, failure value, etc.?
   (d) Can any data object definition be inconsistent with the data object's type, acceptable values, failure value, etc.?

B. Incorrect Functionality Scenario

1. For each functional requirement identify all input/output data objects:
   (a) Are all values written to each output data object consistent with its intended function?
   (b) Identify at least one function that uses each output data object.
2. For each functional requirement identify all specified system events:
   (a) Is the specification of these events consistent with their intended interpretation?
3. Develop an invariant for each system mode (i.e. Under what conditions must the system exit or remain in a given mode)?
   (a) Can the system's initial conditions fail to satisfy the initial mode's invariant?
   (b) Identify a sequence of events that allows the system to enter a mode without satisfying the mode's invariant.
   (c) Identify a sequence of events that allows the system to enter a mode, but never leave (deadlock).

C. Ambiguities Or Missing Functionality Scenario

1. Identify the required precision, response time, etc. for each functional requirement.

   (a) Are all required precisions indicated?
2. For each requirement, identify all monitored events.
   (a) Does a sequence of events exist for which multiple output values can be computed?
   (b) Does a sequence of events exist for which no output value will be computed?
3. For each system mode, identify all monitored events.
   (a) Does a sequence of events exist for which transitions into two or more system modes is allowed?

Adam A. Porter Adam A. Porter earned his B.S. degree summa cum laude in Computer Science from the California State University at Dominguez Hills, Carson, California in 1986. In 1988 and 1991 he earned his M.S. and Ph.D. degrees from the University of California at Irvine.
Since 1992 he has been an assistant professor with the department of Computer Science and the Institute for Advanced Computer Studies at the University of Maryland. His current research interests include empirical methods for identifying and eliminating bottlenecks in industrial development processes, experimental evaluation of fundamental software engineering hypotheses, and development of tools that demonstrably improve the software development process.
Dr. Porter is a member of the ACM and IEEE Computer Society.

Lawrence G. Votta Jr. Lawrence G. Votta received his B.S. degree in Physics from the University of Maryland, College Park, Maryland in 1973, and his Ph.D. degree in Physics from the Massachusetts Institute of Technology, Cambridge, Massachusetts in 1979.
Since 1979 he has been both a member of technical staff and manager at AT&T Bell Laboratories, working and managing development groups in switching and computer products. Currently, he is a member of technical staff in the Software Production Research Department at Naperville, Illinois. His current research interest include understanding how to measure, model, and do experiments with large and complex software development processes.
Dr. Votta is a member of the ACM and IEEE Computer Society.

Victor R. Basili Dr. Victor R. Basili received his B.S. in mathematics from Fordham College, MS. in mathematics from Syracuse University, and his Ph.D in computer science from the University of Texas at Austin. He is currently a Professor in the Institute for Advanced Computer Studies and the Computer Science Department at the University of Maryland, College Park, Maryland, were he served as chairman for six years. He is one of the founders and principals in the Software Engineering Laboratory, a joint venture between NASA Goddard Space Flight Center, the University of Maryland and Computer Sciences Corporation, established in 1976. The SEL was the first winner of the IEEE Process Achievement Award in 1994.
He has been working on the development of quantitative approaches for software management, engineering and quality assurance by developing models and metrics for improving the software development process and product. He was one of the recipients of the NASA Group Achievement Award for the GRO Ada experiment in 1989 and the NASA/GSFC Productivity Improvement and Quality Enhancement Award, for the Cleanroom project in 1990. Dr. Basili has authored over 150 papers. In 1982, he received the Outstanding Paper Award from the IEEE Transactions on Software Engineering for his paper on the evaluation of methodologies.
He is an IEEE Fellow and serves on the editorial board of the Journal of Systems and Software. He has been Editor-in-Chief of the IEEE Transactions on Software Engineering, Program Chairman for several conferences including the 6th International Conference on Software Engineering in Japan, General Chairman of the 15th International Conference on Software Engineering, a Governing Board member of the IEEE Computer Society, and Treasurer of the Computing Research Association.