An Experiment To Assess Different Defect Detection Methods For Software Requirements Inspections

Research Paper

A.A. Porter *
Computer Science Department
University of Maryland
Institute for Advanced Computer Science
College Park, Maryland 20472
aporter@cs.umd.edu

L.G. Votta
Software Production Research Department
AT&T Bell Laboratories
Naperville, IL 60566
votta@research.att.com

Abstract

Software requirements specifications (SRS) are usually validated by inspections, in which several reviewers read all or part of the specification and search for defects. We hypothesize that different methods for conducting these searches may have significantly different rates of success.

Using a controlled experiment, we show that a Scenario-based detection method, in which each reviewer executes a specific procedure to discover a particular class of defects has a higher defect detection rate than either Ad Hoc or Checklist methods.

We describe the design, execution, and analysis of the experiment so others may reproduce it and test our results for different kinds of software developments and different populations of software engineers.

1 Introduction

One of the most common ways 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 defects [1] : detection, collection, and repair[2]. [7, 16] A team of reviewers reads the SRS, identifying as many defects as possible. Newly identified defects 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, defect detection. For the purposes of this article, we define a defect detection method as a set of procedures coupled with the assignment of responsibilities to individual reviewers.

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

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

In practice, reviewers often use Ad Hoc or Checklist detection procedures to discharge general responsibilities. Some authors, notably Parnas and Weiss[12], have argued that using more systematic detection procedures with selective reviewer responsibilities would be more effective.

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 initial results strongly indicate that the choice of defect 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 defect detection methods which motivated our study, our research hypothesis, and our experi-

* *This work is supported in part by the National Aeronautics and Space Administration under grant NSG–5123*

[1] We use the word *defect* instead of the word *fault* even though this does not adhere to the IEEE Standards on Software Engineering Terminology [8]. We feel the word fault has a code-specific connotation – only one of the many places where inspections are used.

[2] Depending on the exact form of the inspection, they are sometimes called reviews or walkthroughs. For a more thorough description of the taxonomy see [7] pp. 171*ff* and [9].

To be Published in the Proceedings of the Sixteenth International Conference on Software Engineering, Sorento, Italy, May 1994

mental observations, analysis and conclusions.

1.1 Inspection Literature

A summary of the origins and the current practice of inspections may be found in Humphrey [7]. 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[2, 7], Parnas and Weiss are more critical [12]. In part, they argue that effectiveness suffers because individual reviewers are not assigned specific responsibilities and because they lack specialized techniques for meeting those responsibilities.

To address these concerns – at least for software designs – they 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 2.2.2.

1.2 Detection Methods

The two most frequently used defect detection methods employ either Ad Hoc or Checklist procedures, with no explicit assignment of reviewer responsibilities. Ad Hoc detection methods use no systematic detection techniques, and very general reviewer focus. Checklist methods reuse important "lessons learned" within an environment or application. Their purpose is to define the reviewer responsibilities and suggest ways for reviewers to identify defects. Individual checklist items might enumerate characteristic defects, prioritize different defects, or pose questions that help reviewers discover defects, such as "Are all interfaces clearly defined?" or "If input is received at a faster rate than can be processed, how is this handled?"

1.3 Hypothesis

We believe that an alternative approach which gave individual reviewers specific detection responsibilities and specialized techniques for meeting them will be more effective.

To explore this alternative we developed a set of defect-specific procedures called Scenarios – collections of procedures for detecting particular classes of defects. Each reviewer executes a single scenario, so multiple reviewers are needed to achieve broad coverage of the document.

Our underlying hypothesis is depicted in Figure 1: that nonsystematic procedures with general reviewer responsibility lack reviewer coordination, leading to overlap and gaps, thereby lowering the overall inspection effectiveness; while systematic approaches with specific responsibilities provide coordination and reduce gaps, thereby increasing the overall effectiveness of the inspection.

2 The Experiment

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 in the Spring of 1993 with 24 subjects – students from A. Porter's graduate course on formal methods who acted as reviewers. The 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.

2.1 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. [3]

2.1.1 Variables

The experiment manipulates four independent variables:

1. the detection method used by a reviewer (Ad Hoc, Checklist, or Scenario);
2. the team composition (who is assigned to which team);
3. the specification to be inspected (two are used during the experiment);
4. the order in which specifications are inspected by each team of reviewers.

Table 1 shows the settings of the independent variables.

[3]The interested reader should consult chapter 4 of reference [10] for an excellent discussion of randomized social experimental designs.

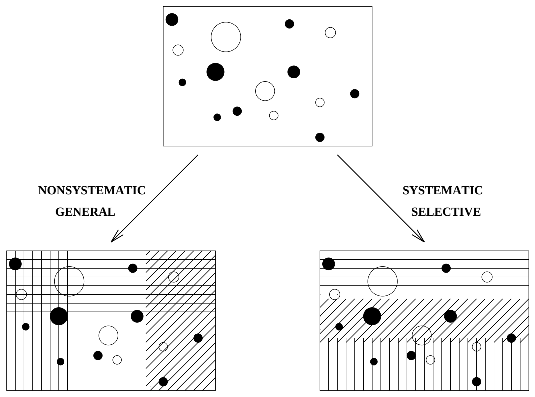


Figure 1: **Systematic Inspection Research Hypothesis.** This figure represents a software requirements specification before and after a *nonsystematic, general responsibility* inspection and a *systematic, selective responsibility* inspection. The points and holes portray various defects. The line-filled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

Trial/Specification

Detection Method		Trial 1		Trial 2	
		WLMS	CRUISE	WLMS	CRUISE
	Ad Hoc	B,D,G,H	A,C,E,F	A	D
	Checklist			E	B,H
	Scenarios			F,C	G

Table 1: This table shows the settings of the independent variables. Each team inspects both documents, one per round, using one of the three detection methods. For the first trial all eight teams used the *Ad Hoc* method. In the second trial, methods were randomly assigned.

The dependent variable is the team defect detection rate. That is the number of defects detected by the team divided by the total number of defects known [4] to be in the specification. The closer that value is to 1, the more effective the detection method.

2.1.2 Design

The experiment uses a randomized, partial factorial design. The design is randomized because the treatments of the teams are chosen at random. It is partial factorial because not all combinations of independent variables are present.

[4]No defects were intentionally seeded into the specifications. All defects 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 three such threats: (1) selection effects, (2) maturation effects, and (3) instrumentation effects.

Selection effects are due to natural variation in human performance. For example, random assignment of subjects may accidentally create an elite team. Therefore, differences in the team's natural ability will mask differences in the detection method performance. To limit this effect, we rated each reviewer's background knowledge and experience as either low, medium, or high. Teams of three were then formed by selecting one individual at random from each category.

This approach has its drawbacks. In future replications we will make purely random assignments to avoid any possibility, no matter how remote, that the method of categorizing inspection experience influences the experimental outcomes.

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.

As will be shown in Section 3, variation in the defect detection rate is not explained by selection or maturation.

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.

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 reviewers in the first run of our experiment may not be representative of software programming professionals;
2. the specification documents may not be representative of real programming problems;
3. the inspection process in our experimental design may not be representative of software development practice.

The first two threats are real. To surmount them we are planning to replicate our experiment with software programming professionals who will inspect industrial work products. Nevertheless, laboratory experimentation is a necessary first step because it greatly reduces the risk of transferring immature technology.

We avoided the third threat by modeling the experiment's inspection process after the design inspection process described in Eick, et al. [4], which is used by several development organizations at AT&T; therefore, we know that at least one professional software development organization practices inspections in this manner.

2.1.5 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 dependent variable. This was done by using an analysis of variance technique as discussed in Box, et al. ([3], pp. 165*ff*).

The second step was to evaluate the combined effect of the variables shown to be significant in the initial analysis. Again, we followed Box, et al. closely ([3], pp. 210*ff*).

Once these relationships were discovered and their magnitude estimated, we examined other data, such as correlations between the categories of defects detected and the detection methods used that would confirm or reject (if possible) a causal relationship between each independent variable and the dependent variable.

2.2 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.

2.2.1 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, while the other three sections are specified using the SCR tabular requirements notation [6].

For this experiment, all three documents were adapted to adhere to the IEEE suggested format [9]. All defects present in these SRS appear in the original documents or were generated during the adaptation process; no defects were intentionally seeded into the document. The authors discovered 42 defects in the WLMS SRS; and 26 in the CRUISE SRS. The authors did not inspect the ELEVATOR SRS since it was only used for training exercises.

Elevator Control System (ELEVATOR) [17] describes the functional and performance requirements of a system for monitoring the operation of a bank of elevators (16 pages).

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).

Automobile Cruise Control System (CRUISE) [11] describes the functional and performance requirements for an automobile cruise control system (31 pages).

2.2.2 Defect Detection Methods

To make a fair assessment of the three detection methods (Ad Hoc, Checklist, and Scenario) each method should search for the same population of defects. This population was defined using a general defect taxonomy.

The taxonomy is a composite of two schemes developed by Schneider, et al. [14] and Basili and Weiss [1]. Defects 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 defects were also subdivided into four categories: Missing Functionality (MF) , Missing Performance (MP), Missing Environment (ME), and Missing Interface (MI). Commission defects were divided into four categories: Ambiguous Information (AI), Inconsistent Information (II), Incorrect or Extra Functionality (IF), and Wrong Section (WS). We provided a copy of the taxonomy to each reviewer.

Reviewers using the Ad Hoc method received no further assistance.

For the Checklist reviewers, we constructed a single checklist based on the defect taxonomy. The checklist items are detailed questions culled from several industrial checklists. Thus, they are very similar to checklists used in practice. All Checklist reviewers used the same checklist.

Finally, we developed three groups of scenarios to cover the defect taxonomy. The scenarios are intended to detect data type inconsistencies, incorrect functionality, and ambiguous or missing functionality. The defects targeted by the procedures are a subset of those covered by the taxonomy. [5]

2.2.3 Defect Report Forms

We also developed a Defect Report Form. Whenever a potential defect was discovered – during either the defect detection or the collection activities – an entry was made on the form. The entry included four kinds of information: Inspection Activity (Detection, Collection); Defect Location (Page and Line Numbers); Defect Disposition, (Defect can be a True Defect or a False Positive); and a prose Defect Description.

A small sample of a Defect Report appears in Figure 2.

[5] The actual defect taxonomy, checklist and scenarios are presented in an earlier technical report [13]

2.3 Experiment Preparation

The participants were given a series of lectures on software requirements specifications, the SCR tabular requirements notation, inspection procedures, the defect classification scheme, and the filling out of data collection forms. The references for these lectures were Parnas [12], Fagan [5] and the IEEE Requirements Standard for Software Reviews and Audits [9].

The participants were then assembled 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

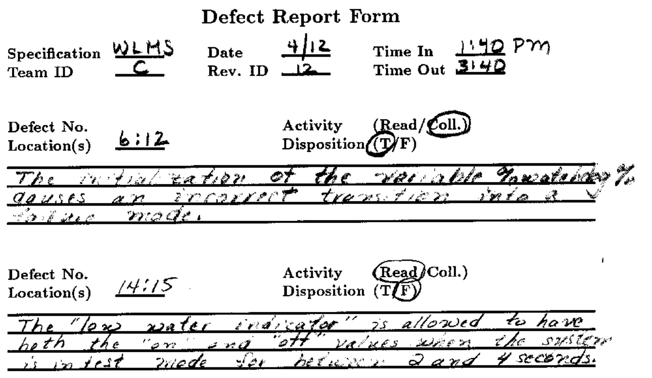
2.4.1 Training

For the training exercise, each team inspected the ELEVATOR SRS. Individual team members read the specification and recorded all defects they found on a Defect Report Form. Their efforts were restricted to two hours. Later we met with the students and answered questions about the procedure. Afterwards, each team conducted a supervised collection meeting and filled out a master Defect Report Form for the entire team. The ELEVATOR SRS was not used in the remainder of the experiment.

2.4.2 First Trial: Characterizing Ad Hoc Detection

The instruments used in the first trial are the WLMS and CRUISE specifications discussed in Section 2.2.1 and the defect report forms described in Section 2.2.3. Four of the eight teams were asked to inspect the CRUISE specification; the remaining four teams inspected the WLMS specification. The defect detection step was limited to two hours and all potential defects were to be reported on the Defect Report Form. After defect detection, all materials were collected. [6] Once all team members had finished the detection phase, the team's moderator arranged for the collection meeting. At the meeting, the documents were reread and defects discussed. The team's recorder maintained the master defect report form. The collection meeting was also limited to 2 hours. The entire trial was completed within one week.

[6] 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.



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 [illegible] causes an incorrect transition into a [illegible] mode.

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

The "low water indicator" is allowed to have both the "on" and "off" values when the system is in test mode for between 2 and 4 seconds.

Figure 2: **Reviewer Defect Report Form.** This is a small sample of the defect report each reviewer completed during the reviewer's detection phase of the experiment. Defects number 10 and 11, found by reviewer 12 of team C for the WLMS specification document are shown.

2.4.3 Second Trial: Detection Methods

The instruments used in the second trial were the specifications used in Trial 1, a checklist, three defect-based scenarios, and the Defect Report Form. The development of the checklist and scenarios is described in Section 2.2.2. The same checklist and scenarios were used for both documents.

The eight teams were assigned to experimental groups as follows: teams that inspected the CRUISE specification in Trial 1 inspected the WLMS specification in Trial 2, and vice versa. Of the four teams inspecting the CRUISE specification, one used the Ad Hoc method, two used the Checklist, and the fourth team used the Scenarios. For the WLMS specification, one team used the Ad Hoc method, one team used the Checklist, and two teams used Scenarios. The experimental procedures and data collection were exactly the same as for Trial 1. After the trial was completed we gathered and analyzed all the defect report forms.

3 Data and Analysis

3.1 Data

Two sets of data are important to our study. The first set shows whether or not a team discovered a particular defect. This data is created at the team meetings and is used to assess the effectiveness of each defect detection method. Figure 3 summarizes this data graphically, showing for each team the defect detection rate and the detection method used. As Figure 3 shows, the Scenario detection methods resulted in the highest defect detection rates, followed by Ad Hoc detection methods, and finally by Checklist detection methods.

The second data set comprises the summaries of the individual Defect Report Forms filled out when each reviewer read the specification document. This data provides a log of the defects each reviewer discovered and is used to assess whether Scenarios improve a reviewer's ability to identify specific classes of defects.

3.2 Analysis

Table 2 contains the results of the analysis outlined in Section 2.1.5. The independent variables are listed in order from the least to the most significant. The maturation, and selection effects are negligible. The anticipated instrument bias is present, as is the significant effect of the detection method.

We used an analysis of variance as described in Section 2.1.5 to assess the simultaneous effect of instrumentation and detection methods on the defect detection rate.

Table 3 shows the input to this analysis. The table contains 6 cells (3 detection methods, 2 specification documents), each cell contains the average detection rates for teams using each detection method and each specification document. As the Table shows, the Scenario detection methods resulted in the highest defect detection rates, followed by Ad Hoc detection methods, and finally by Checklist detection methods.

Table 4 shows the analysis of variance for the specification and detection method independent variables. The analysis assumes that there is no interaction be-

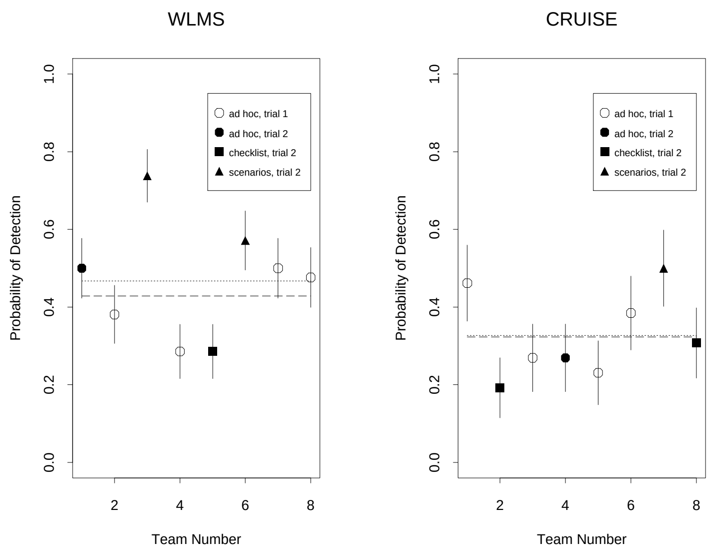


Figure 3: **Defect Detection by Detection Method.** The observed defect detection rates are displayed. The open symbols are trial 1, the filled symbols are trial 2. The vertical line segment through each point indicates one standard deviation in the estimate of the rate (assuming each defect was a Bernoulli trial). The dashed (dotted) lines display the average of the defect detection rates for trial 1 (trial 2).

tween the independent variables [7]. The analysis of variance confirms our first analysis. The detection methods explain roughly 3 to 4 times more variance in the defect detection rate than do the instruments (the difference between the specification documents).

From both these analyses we conclude that the Scenario detection method is the most effective and the Checklist method the least effective.

An analysis of the second data set shows one reason for the Scenario's increased defect detection rates. Table 5 categorizes the defects in both documents and compares the average detection rates of Scenario reviewers with those of all other reviewers. We see that in most cases reviewers using a scenario are more effective at finding the defects the scenario was designed to uncover. At the same time, all reviewers are equally effective at detecting defects that are not targeted by the scenarios. ("Not Observable" category).

[7]This assumption may appear naive; but, until we can replicate the experiment, we do not have enough data points to analyze nonlinear interactions.

4 Summary and Conclusions

Our experimental design for comparing defect detection methods is flexible and economical, and allows the researcher to measure the effect of several potential threats to the experiment's internal validity. In particular, we determined that neither the experience gained by the reviewers as the experiment progressed nor the composition of the teams had any significant influence on inspection effectiveness. However, differences in the SRS did.

The results from our initial experiment are drawn from only sixteen observations. We are currently replicating the experiment to increase the size of our data sets.

Keeping in mind the limited number of observations, we tentatively draw three conclusions.

1. The defect detection probability when using Scenarios is superior to that obtained with Ad Hoc or Checklist methods – an improvement of roughly 35%.

2. Scenarios are effective partly because reviewers are able to focus on specific classes of defects. Readers should note however, that the scenarios appeared to be better suited to the defect profile

Independent Variable	SS_T	ν_T	SS_R	ν_R	$(SS_T/\nu_T)(\nu_R/SS_R)$	Significance Level
Inspection round–maturation	.009	1	.315	14	.40	.537
Team composition–selection	.160	7	.164	8	1.120	.434
Specification– instrumentation	.081	1	.243	14	4.678	.048
Detection Method	.186	2	.139	13	8.71	.004

Table 2: **Analysis of Variance for Each Independent Variable.** The analysis of variance of team defect detection rate on the four independent variables shows the significance of the detection method and specification for explaining variation in the dependent variable.

Instrument	**Detection Method**		
	Ad Hoc	*Checklist*	*Scenario*
WLMS	.38, .29, .50, .48, .50*	.29*	.50*, .74*
(average)	.43	.29	.62
Cruise	.39, .23, .27, .46, .27*	.19*, .31*	.50*
(average)	.32	.25	.50

Table 3: **Team Defect Detection Rate Data.** The nominal and mean defect detection rates for all 16 teams. The rates without asterisks are for trial 1. Those with asterisks are for trial 2.

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 the poorest of the three detection methods. Since the scenarios were constructed from the checklist, the obvious explanation that the checklist was poorly designed to begin with seems incorrect.

The results of this work have important implications for software practitioners. They indicate 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.

From the point of view of software researchers, our work demonstrates the feasibility of constructing and executing inexpensive experiments to validate fundamental research recommendations. Since economical designs are necessary to allow replication in other environments with different populations of software engineers, this is an important consideration.

5 Future Work

The experimental data raise many interesting questions for future study.

- In many instances a single reviewer identified a defect at preparation, but the defect was not subsequently recorded in the team log. Are single reviewers who observe a defect sometimes forgetting to mention a defect they observed, or is the reviewer being "talked out" of the defect at the team meeting?

- In the same analysis we notice that very few defects are initially discovered during the defect collection step. As the collection activity is normally carried out at a meeting, we should ask whether these meetings are worth holding, especially in view of their impact on production interval. Interestingly, this data is consistent with an industrial study performed by L. Votta [16].

- More than half of the defects in both specifications are not addressed by the scenarios used in this study. What other scenarios are necessary to achieve a broader defect coverage?

- Previous work by Schneider, et al., has shown that an N-fold inspection (i.e., performing an inspection with several independent teams working

Source	**Sum of Squares**	**Degrees of Freedom**	**Mean Square**
average	.9680	1	.9680
specification	.0122	1	.0122
detection method	.0862	2	.0431
residuals	.0019	2	.0010
TOTALS	1.0683	6	

Table 4: **Analysis of Variance of Detection Method and Document.** The table displays the results of an analysis of variance of the average team defect detection rates given in Table 3

	WLMS			CRUISE		
Scenario	Total	Scen.	Other	Total	Scen.	Other
Data type checking	14	7.5	4.1	10	6	2.6
Incorrect functionality	5	1.5	0.7	3	0	0.3
Missing functionality	5	3.5	0.8	1	0	0
Not Observable	18	5.0	3.9	12	2.6	2.0

Table 5: **Average Number of Defects Found by Defect Type**. The table shows the average defect detection rate per reviewer for both Scenario and all other detection methods. For the defects that the scenario was designed to detect the reviewers using the Scenario method were more effective than reviewers using either the Ad Hoc or Checklist methods. All reviewers were equally effective at finding the defects in the "Not Observable" category–defects that no scenario was designed to uncover.

in parallel) can improve performance. Our data support this conclusion, but it appears that single teams using improved methods can be as effective as several teams using Ad Hoc or Checklist methods.

The results of any individual experiment can be challenged. Experiments gain credibility only through replication and reproduction. Each new run reduces the probability that results can be explained by human variation or experimental error. Consequently, we are replicating the experiment at the University of Maryland.

We are also creating a laboratory kit (i.e., a package containing all the experimental materials) to facilitate replication. We expect to have it publicly available by June 1994, and plans are being made to reproduce the experiments with other university researchers in Japan, Germany, and Australia and with industrial developers at AT&T Bell Laboratories and Motorola Inc.

Acknowledgments

The work reported in this article is part of a long-term effort to create software development processes that can be engineered to fit the goals and needs of their users. The research is the product of collaboration between Victor Basili, Adam Porter, and Lawrence Votta. Due to a mistake on our part, Victor Basili does not appear as an author on this work. Therefore, we would like to take this opportunity to acknowledge his many contributions to this research.

We would also 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] 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*, pages 314–323, San Diego, CA, March 1981.

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

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

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

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

[6] 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.

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

[8] *IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software.* Soft. Eng. Tech. Comm. of the IEEE Computer Society, 1989. IEEE Std 982.2-1988.

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

[10] 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.

[11] 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.

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

[13] Adam A. Porter and Lawrence G. Votta. An experiment to assess different defect detection methods for software requriements inspections. Technical Report CS-TR-3130, University of Maryland, College Park, MA, September 1993.

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

[15] 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.

[16] 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.

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