The Impact of Educational Background on the Effectiveness of Requirements Inspections: An Empirical Study

Jeffrey C. Carver, *Member*, *IEEE*, Nachiappan Nagappan, *Member*, *IEEE*, and Alan Page, *Member*, *IEEE*

Abstract—While the inspection of various software artifacts increases the quality of the end product, the effectiveness of an inspection largely depends on the individual inspectors involved. To address that issue, a large-scale controlled inspection experiment with over 70 professionals was conducted at Microsoft that focused on the relationship between an inspector's background and his effectiveness during a requirements inspection. The results of the study showed that inspectors with university degrees in majors not related to computer science found significantly more defects than those with degrees in computer science majors. We also observed that level of education (Masters, PhD), prior industrial experience, or other job-related experiences did not significantly impact the effectiveness of an inspector. The only other type of experience that had a significant impact on effectiveness was experience in writing requirements, i.e., professionals with prior experience writing requirements found statistically significant more defects than their counterparts.

Index Terms—Metrics/measurement, requirements/specifications, software quality/SQA.

1 Introduction and Background

The use of inspections throughout the software life cycle is an important factor in improving the overall quality of the resulting software. An inspection, originated in the mid-1970s at IBM by Fagan [10], is the static review of an artifact produced during the software development life cycle (e.g., requirements, design, or code). In the more than 30 years since the inception of inspections, researchers have made modifications to the original process with the goal of improving the effectiveness, efficiency, or applicability in various settings. This large body of research has left little doubt about the benefits provided by inspections when they are used in the proper environment and for appropriate tasks [3], [13], [18], [21], [25]. In fact, two studies report that inspections can help locate between 60 and 90 percent of the defects present in a software product [5], [11].

The main research goal driving much of the software inspection research is: improving the effectiveness and efficiency of the inspection process. Inspections have been extensively studied by a wide variety of researchers. This research has concentrated mainly on identifying areas for improvement in two important areas: 1) inspection team meetings and 2) individual preparation for subsequent team meetings. Despite the fact that inspections are a well-studied topic within software engineering, there are still open questions. In a paper summarizing the findings of 25 years of inspections research, Aurum et al. [1] provide an overview of the various improvements that have been made to the inspection process and associated support tools including: Perspective-Based Reading (PBR) [3], [8], [28], Usage-Based Reading [31], [32], checklists [24], Active Design Reviews [22], N-fold inspections [18], [27], and inspections without meetings [19], [33]. In addition, Aurum et al. raise a series of unanswered questions and suggest the need for additional research insights. Two questions are particularly relevant: 1) "What is the best way to plan and staff software inspections?" and 2) "Is it possible to identify relationships between different parameters in software inspections?"

The importance of these two questions emerges in light of previous work that highlights the wide variation in the effectiveness of different inspectors, even when they are using the same technique on the same artifact [6]. For example, Basili et al. reported that, during the investigation of PBR for detecting defects in natural language requirements, the effectiveness of individual inspectors ranged from finding only 10 percent of the defects up to finding 90 percent of the defects [3]. In a study of inspections of object-oriented design artifacts, Laitenberger et al. reported effectiveness ranging from 20 percent up to 70 percent [16]. In studies of code inspections at Lucent Technologies, Porter et al. found that a major influencing factor on the effectiveness of an inspection was the actual inspectors who participated. Some inspectors were up to two times more effective than others [23], [30]. Finally, in a study focused on inspection teams rather than individual inspectors, Schneider et al. reported that the least effective team found 22 percent of the defects while the most effective team found 50 percent of the defects [27]. In each of these cases,

- *J.C. Carver is with the Department of Computer Science, The University of Alabama, Box 870290, Tuscaloosa, AL 35487-0290. E-mail: carver@cs.ua.edu.*
- *N. Nagappan and A. Page are with Microsoft Research, One Microsoft Way, Redmond, WA 98052. E-mail: {nachin, alanpa}@microsoft.com.*

Manuscript received 18 June 2007; revised 3 Mar. 2008; accepted 19 May 2008; published online 29 June 2008.
Recommended for acceptance by D. Rombach.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2007-06-0193.
Digital Object Identifier no. 10.1109/TSE.2008.49.

the inspectors all used the same inspection approach. As a result, the wide variations cannot be attributed to process variation and must have other origins. Therefore, in addition to studying specific inspection techniques and methods, it is important to understand the variations among the individual inspectors that make them more or less effective.

Although researchers realize the importance of understanding the impacts of an inspector's background and experience, little previous research has focused specifically on identifying the characteristics that make an inspector particularly effective. In a paper that lays out a research program focused on understanding technical reviews and how to improve them, Sauer et al. highlight the importance of understanding how expertise impacts an inspector's effectiveness. This research, which makes use of behavioral theory, specifically identifies experience with the inspection task as being an important factor [26].

In the area of requirements inspections, two studies have focused on identifying the important characteristics. First, Biffl and Halling studied three characteristics specifically related to software engineering expertise. Using a large controlled study of 170 students, they investigated whether development skills, experience with different aspects of the software life cycle (projects, UML, requirements, and inspections), or performance on a pretest inspection were valid predictors of subsequent inspection effectiveness. The results of the study showed that only the subject's score on the pretest had a significant impact on performance [4]. Second, Miller and Yin investigated the impact of a characteristic that is not directly related to software engineering expertise, the Myers-Briggs personality types. The main goal of the study was to identify the impact of personality types on inspection team formation. But, they did an initial analysis on the impact of personality types on the effectiveness of inspectors during the individual preparation phase. The results of this analysis indicated that personality type was not a predictor of inspection effectiveness. Conversely, when forming teams, the results indicated that a mix of personality types on a team did impact the overall performance of the team [20]. An inference that can be drawn from the disparity in the individual and team results is that, while the number of defects identified did not vary based on personality type, the specific defects identified by each personality type were likely different leading to the increase in team performance. Unfortunately, a detailed analysis of this type was not reported by the authors of the study.

Finally, Hungerford et al. performed a study to understand the process used by experts when reviewing a series of entity-relationship diagrams and data-flow diagrams. The study had two goals, document the processes used by experts and determine if any approach was more effective than the others. The task required the reviewers to analyze information that was presented in multiple documents in order to identify defects. The two software engineering characteristics that were analyzed, IT experience and data-flow diagram experience, did not affect the performance of the inspectors. Conversely, a non-software engineering characteristic, ability to quickly switch between artifacts, did have an impact on performance [14].

These studies show that, often, characteristics related to software engineering are not a good predictor of inspection effectiveness, while non-software engineering characteristics (i.e., ability to switch between documents) were good predictors. The identification of specific, desirable characteristics or abilities, which could even be unrelated to software engineering, will provide inspection team leaders with information to guide the selection and training of potential inspection team members. This information will also be useful to educators in planning the curricula for software engineering and computer science degree programs. In addition, such a finding would provide insight into the first question raised by Aurum et al. [1]. Furthermore, as the impact of the background and experience of inspectors is better understood, that impact can be treated as a parameter that can interact with other parameters like inspection technique, problem domain, or format of the artifact. These interactions can then be studied to understand their effect on the outcome of an inspection.

To gain insight into the impact of an inspector's background and experience, a large-scale controlled experiment was conducted at Microsoft, Redmond, Washington. The goal of this study was to compare the effectiveness of inspectors from different educational backgrounds. More than 70 professionals participated in the inspection of a generic requirements document with the goal of identifying as many defects as possible. Some of those subjects had university degrees in computer science or a closely related major, while the rest of the subjects had degrees in non-computer science related majors. The specific focus of this study was comparing the performance of these different groups of subjects to determine whether one group was more effective than the other. The results provide insight into the types of knowledge or experience that are beneficial for inspectors.

Studying the effects of educational background on software engineering tasks is quite interesting and relevant due to the varied distribution of the backgrounds of practicing software engineers. Lethbridge et al. report that up to 60 percent of individuals employed in the computer industry do not have a computing-related education [17]. This figure alone provides adequate motivation for the need to better understand the impact that an inspector's educational background has on effectiveness.

The remainder of this paper is organized as follows: Section 2 provides a discussion of the experimental design, including the hypotheses, the subjects, and the procedure followed. The results of the study are explained in Section 3. Section 4 discusses the threats to validity that were present in the study. Finally, the conclusions are presented in Section 5.

2 Experiment Description

This section provides the pertinent information about the experiment. It describes the research questions, the hypotheses, the variables measured, the experimental design (including the subjects, artifacts, and procedures), and, finally, the data collection process.

2.1 Research Questions and Hypotheses

This study had a primary major research question and a secondary research question, which are given as follows:

1. *Are inspectors who have a degree in computer science more effective during a requirements inspection than inspectors with non-computer science degrees?*

The secondary research question was

2. *Do other variables (highest degree obtained, experience in the field, experience with requirements, and experience with inspections) impact the effectiveness of an inspector?*

To investigate these two questions, a more detailed set of five hypotheses was defined. For each hypothesis, the null hypothesis (HX_0) is presented, followed by the alternative hypothesis (HX_a):

- $H1_0$: *The effectiveness of an inspector is not affected by their educational background (computer-related or not computer-related).*
- $H1_a$: *The effectiveness of an inspector is affected by their educational background (computer-related or not computer-related).*
- $H2_0$: *Experience with requirements does not affect the effectiveness of an inspector.*
- $H2_a$: *Experience with requirements affects the effectiveness of an inspector.*
- $H3_0$: *The effectiveness of an inspector is not affected by their level of education (bachelor's versus master's degree).*
- $H3_a$: *The effectiveness of an inspector is affected by their level of education (bachelor's versus master's degree).*
- $H4_0$: *The effectiveness of an inspector is not affected by whether or not they have industrial software development experience.*
- $H4_a$: *The effectiveness of an inspector is affected by whether or not they have industrial software development experience.*
- $H5_0$: *Inspection experience does not affect the effectiveness of an inspector.*
- $H5_a$: *Inspection experience affects the effectiveness of an inspector.*

2.2 Variables

There were five independent variables measured to determine their impact on the one dependent variable. Each of these variables is defined in this section.

2.2.1 Independent Variables

- **Educational Background**—the field in which a subject's most advanced degree was awarded (computer-related or non-computer-related).
- **Educational Degree**—the highest degree earned (bachelor's or master's).
- **Industrial Experience**—whether the subject has industrial software development experience (Yes or No).
- **Requirements Experience**—experience writing requirements (originally measured on a five-point Likert scale ranging from 1—"No Experience" to 5—"Multiple Industrial Projects," but collapsed into two values for analysis: None and Some).
- **Inspection Experience**—experience reviewing requirements or with inspections in general (originally measured on a five-point Likert scale ranging from 1—"No Experience" to 5—"Multiple Industrial Projects," but collapsed into two values for analysis: None and Some).

2.2.2 Dependent Variable

- **Effectiveness**—the number of faults detected during the inspection.

2.3 Design

2.3.1 Subjects

The participants were self-selected, that is, they were drawn from the students enrolled in four training courses taught by the Engineering Excellence Group at Microsoft. Two of these courses were made up primarily of people with degrees in non-computer-related majors, while the other two were made up mostly of people with degrees in computer-related majors. Each course consisted of 15-20 students. The study was done as part of the course and the subjects did not receive any compensation for participation. The goals of this module of the course were given as follows:

1. Training the software engineers on inspections at Microsoft, explaining the context and history of inspections at Microsoft, and presenting an overview of some of the inspection studies conducted in academia.
2. Performing the controlled experiment to investigate the impact of background on inspections.

The 73 subjects were drawn from all major product groups within Microsoft: Windows, Office, SQL, Visual Studio, and so forth. In terms of their level of education, 52 of the subjects had bachelor's only degrees while the other 21 also had master's degrees. For the purpose of analyzing the effects of the inspectors' educational background, the subjects were split into two groups based on the field in which they obtained their most advanced degree. The two groups were: those with computer-related degrees and those with non-computer-related degrees. Due to the fact that the subject sample was drawn from a training course and not preselected, the number of subjects in the two groups is not equal. A detailed breakdown of the subjects' backgrounds is shown in Table 1. The average experience the subjects had working in the software field was 1.94 years (median = 1.00 year, standard deviation = 2.86 years). Fig. 1 illustrates the overall distribution of the subjects' experience, grouped by the field in which they received their highest degree. To provide further details about the subject population, Table 2 shows some descriptive statistics. For each of the two study groups (i.e., computer-related and non-computer-related), the distribution of the other variables is shown.

TABLE 1
Group Details

Group 1 – Computer Related		#	Group 2 – Non-Computer Related		#
1a – CS & SE		*27*	*2a – Math and Engineering*		*11*
Computer Science	26		Math / Applied Math	4	
Software Engineering	1		Other Engineering	7	
1b – Other		*21*	*2b - Science*		*8*
Electrical Engineering	11		Physics	5	
Computer Engineering	4		Biology	1	
Electrical & Computer Engineering	1		Psychology	1	
Computer Systems	2		Cognitive Science	1	
Management Information Systems	3				
			2c – Business and Arts		*6*
			Business (Econ, Finance, General)	4	
			Music	1	
			Spanish	1	
		48			25

2.3.2 Artifacts

This study was focused on the inspection of a requirements document written in unstructured, natural language (i.e., it did not contain any formal notation). The requirements document used in this study was for a generic (i.e., non-Microsoft) system: the Loan Arranger financial system. The Loan Arranger system provides functionality that allows a financial institution to sell groups of loans to other financial institutions. The institution purchasing the loans is provided with the capability of searching for loans based on outstanding value, remaining term of the loan, and risk. The Loan Arranger requirements document is 10 pages long and includes 49 detailed requirements. This document has been used in previous inspections studies and has been seeded with 30 defects [7], [29]. The subjects were given a standard checklist to guide their inspections.

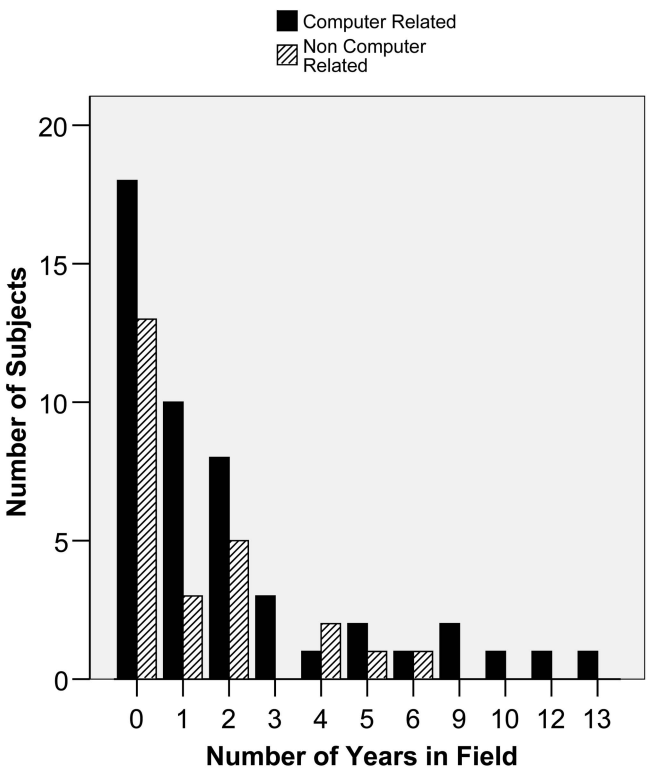

Fig. 1. Years of software work experience.

2.3.3 Procedure

Each course had an enrollment of 15-20 students. All four courses occurred within a three-week time period. The study took 3 hours and the subjects from each course followed the same sequence of steps as detailed in Fig. 2 and discussed in more detail below:

1. **Introduction to class, instructors**: In this step, the second and third authors introduced themselves to the class and gave an overall plan for the 3 hours. The class was conducted by the third author, while the second author observed the class and the experimental methodology for consistency (10 minutes).
2. **Introduction to software inspections**: In this step, the students were given a general introduction to software inspections to set the context of the study. This introduction began with the work of Fagan [10], then covered experiences and anecdotes from internal Microsoft usage (30 minutes).
3. **Prior case studies in inspections**: The use of inspections in NASA, results from case studies conducted by Basili et al. [3], was discussed (10 minutes).
4. **Explanation of experiment and procedures**: In this session, we explained to the participants how to use the documents. The participants had to inspect the requirements for the Loan Arranger system, as described in Section 2.3.2. During the inspection, the participants were to record anything they believed was a defect using the form shown in Fig. 3, including the defect number, page number, and requirement number from the Loan Arranger document. The defect class was selected from among the types shown in Table 3. A field for the description of the defect and the time it was found is also provided. An important point to be noted is

TABLE 2
Subject Characterization

		Educational Background		
		Computer-Related	Non-Computer Related	Total
Educational Degree	B.S.	32	20	52
	M.S.	16	5	21
Industrial Experience	None	18	13	31
	Some	30	12	42
Requirements Experience	None	17	16	33
	Some	31	9	40
Inspection Experience	None	16	16	32
	Some	32	9	41

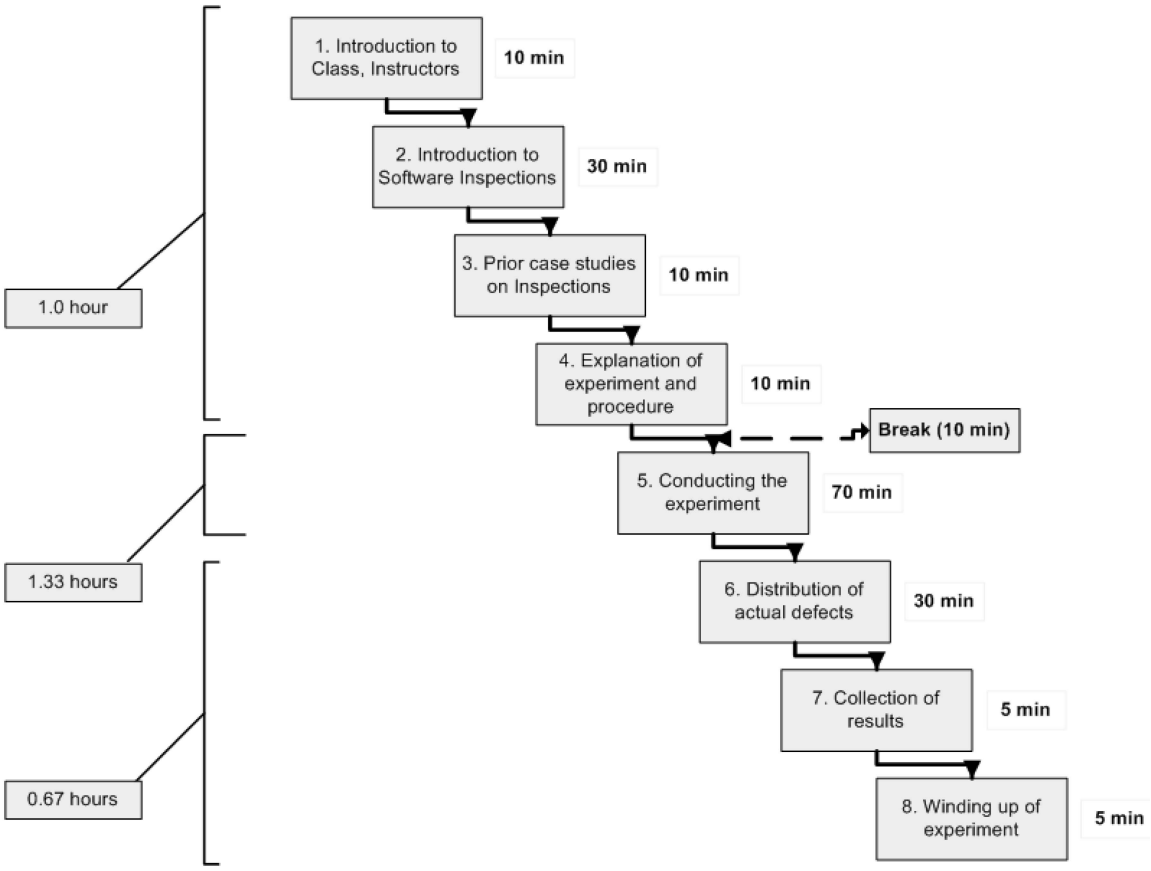

Fig. 2. Step-by step description of the inspections experiment.

that the participants were not told how many defects were seeded and were therefore encouraged to find as many problems as possible (10 minutes).

5. **Conducting the experiment**: The students performed the inspection using the materials described above. This inspection was limited to strictly 70 minutes to maintain consistency with Microsoft's corporate practice of inspection meetings, which last 60-75 minutes. Seven questions on basic demographics were also collected, which took 2-3 minutes.
6. **Distribution of actual defects**: A document containing the actual defects using the same form shown in Fig. 3 was distributed to the participants. They were asked to examine their own list and indicate which defects they believed they had found and provide any explanations they desired. While the researchers performed their own analysis of the subjects' defect lists to provide the data needed to assess the hypotheses, the information provided by the subjects served two important purposes. First, during the analysis phase (discussed at the end of this section), it provided the experimenters with a double check to ensure that the subjects' defects were properly matched to the master defect list. Second,

Defect No.	Page No.	Req. No.	Defect Class	Description	Time Found

Fig. 3. Defect collection form.

TABLE 3
Defect Types

Omission (O)	*Necessary information about the system has been omitted from the software artifact.*
Ambiguous Information (A)	*Some information in the software artifact contradicts information in the requirements document or the general domain knowledge.*
Inconsistent Information (II)	*Information within one part of the software artifact is inconsistent with other information in the software artifact.*
Incorrect Fact (IF)	*Information within the software artifact is ambiguous, i.e. any of a number of interpretations may be derived that should not be the prerogative of the developer doing the implementation.*
Extraneous (E)	*Information is provided that is not needed or used.*
Miscellaneous (M)	*Other defects; e.g. a requirement may be found in an inappropriate section of the document.*

it provided qualitative information from the subjects about which items they thought really were defects (30 minutes).

7. **Collection of results**: The results were collected from the participants (5 minutes).
8. **Winding up of the experiment**: This session involved minor logistical issues with respect to winding up the experiment in terms of where to find resources for software inspection, a documentation of best practices, course notes for the lecture, and so forth (5 minutes).

To ensure consistency, the inspection time limit of 70 minutes was strictly followed for all subjects. After the completion of all four studies, the second author manually analyzed each of the 73 defect forms to match the reported defects to the master list of defects. This analysis was used to remove any false-positives (i.e., defect reports that did not match any of the seeded defects) and identify any new defects that were not originally seeded (note: only a few trivial defects were identified and, therefore, are not considered in the analysis). The information provided by the subjects in Step 6 of the procedure above was not used for the initial analysis; it was only used to ensure that no defect reports were overlooked. This analysis resulted in a spreadsheet of which defects were found by each participant. This analysis of each defect form took around 15 minutes. When the time required to prepare the data and store the data is added in, the total effort was around 40-45 hours spread over two weeks. The remainder of the analysis that produced the results described in Section 3 was then completed by the first author using this spreadsheet.

3 Results

This study has one major research question and one secondary research question. The major research question is: "Are inspectors who have a degree in computer science more effective during a requirements inspection than inspectors with non-computer science degrees?" The secondary research question is: "Do other variables (highest degree obtained, experience in the field, experience with requirements, and experience with inspections) impact the effectiveness of an inspector?"

Prior to conducting an ANOVA, the first step is to perform a data reduction exercise to ensure that the four secondary variables are all independent. If any of the variables are not independent, then they should be removed prior to conducting the ANOVA to increase the power of the analysis. To determine the independence of the variables, we conducted a factor analysis with *Educational Degree*, *Industrial Experience*, *Requirements Experience*, and *Inspection Experience*. Using principal components analysis with an equimax rotation, the factor analysis extracted two components that explain approximately 70 percent of the variation in the data. Factor 1 grouped together *Requirements Experience* and *Inspection Experience* with *Requirements Experience*. The two variables showed almost equal contribution to the factor, but *Requirements Experience* was slightly higher, so it was chosen to remain in the analysis. Factor 2 grouped together *Educational Degree* and *Industrial Experience*, with *Educational Degree* being the higher contributor to this factor, so it was chosen to remain in the analysis. Therefore, as a result of the factor analysis, we have reduced the four secondary variables to two, which are analyzed along with the main independent variable in the ANOVA test.

The most statistically appropriate way to investigate the effects of multiple variables is to conduct an N-way ANOVA. The results of such a test will indicate which main effects and interactions should be investigated in more detail. After the factor analysis, we were left with three independent variables; therefore, we conducted a three-way ANOVA with the following factors: 1) educational background (computer related or noncomputer related), 2) educational cegree (bachelor's or master's), and 3) experience writing requirements (some or none). The results of the ANOVA are shown in Table 4, including the sample size,

TABLE 4
ANOVA Results

Effect	*N*	*n*	*d*	*F*	*p*	f^2	*Power*
Educational Background	73	1	65	14.089	< .001	.216	.959
Requirements Experience	73	1	65	8.312	.005	.127	.811
Educational Degree	73	1	65	2.341	.131	.036	.326
Null hypothesis confidence interval		1	65			.111	.8

N is the number of data points analyzed, n, d are the numerator, denominator degrees of freedom, F is the value of the ANOVA F-test, p is the probability of the F-test, f^2 is Cohen's effect size, power is the post hoc power.

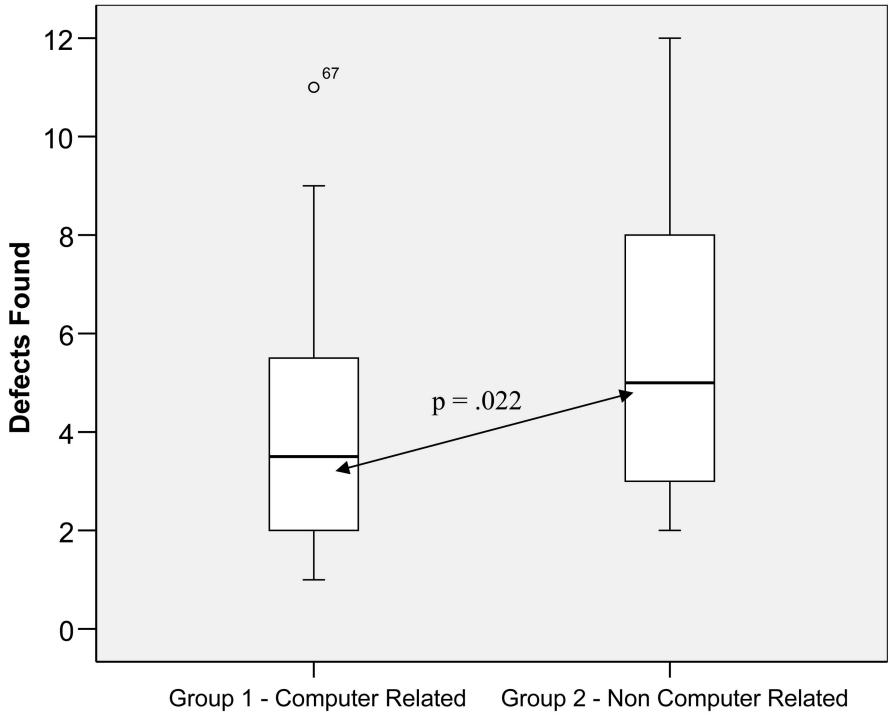

Fig. 4. Educational background.

F-value, p-value, effect size, and power. We used G*Power [12] and SPSS to perform the power and effect size calculations [9], [15]. For all statistical tests reported in this paper, we have used an alpha value of 0.05. These results show that two variables, *Educational Background* and *Requirements Experience*, were significant. Because *Educational Background* was the main independent variable of interest, the results relative to it are explored further in Section 3.1. Similarly, the *Requirements Experience* variable was significant in the ANOVA test and is further explored in Section 3.2. Finally, Section 3.3 provides further insight and analysis relative to the other independent variables.

3.1 Impact of Educational Background ($H1$)

The results of the three-way ANOVA indicated that *Educational Background* was an important variable and worthy of further analysis. For the ANOVA, this variable was defined at two levels, subjects with degrees in computer-related majors and subjects with degrees in other majors (as described in Section 2.3.1). Of the 73 subjects, 48 had degrees in computer-related majors and 25 had degrees in non-computer-related majors. The box plot in Fig. 4 shows the result graphically for this variable in isolation. In the box plots, the thick line represents the mean value, the box contains the middle 50 percent of the data, and the whiskers indicate the extreme points (excluding outliers). The one-way ANOVA results for this variable in isolation indicate that the higher mean value for the subjects with degrees in non-computer-related majors is significant $[\mathrm{F}(1, 71) = 5.52,\ \mathrm{p} = 0.022]$. This result allows $H1_0$ to be rejected in favor of $H1_a$. Note that the small percentage of defects identified by reviewers (15 percent on average) was consistent with the results from previous uses of the Loan Arranger requirements document [7].

To further investigate this significant difference in overall effectiveness, the specific defect types (as defined in Table 3) are examined. Table 4 shows the detailed data for all defect types. The first column (count) indicates the number of subjects in each group. The remaining columns show the average number of defects of each type that were found. The number in parentheses at the top of the column provides the total number of defects of that type that were present in the document. Finally, the bottom row shows the p-value obtained from an ANOVA test to compare the means between the two groups. The inspectors with degrees in non-computer-related majors were more effective for defects of type omission and inconsistency, with the difference being statistically significant for inconsistencies $[\mathrm{F}(1, 71) = 19.068,\ \mathrm{p} < 0.001]$.

TABLE 5
Computer versus Non-Computer Majors

	Count	Overall (30)	Ambiguity (8)	Omission (16)	Inconsistency (4)	Incorrect Fact (2)
Group 1 - Computer Related	48	4.02	1.19	1.79	.77	.27
Group 2 – Non-Computer Related	25	5.52	1.16	2.48	1.64	.24
p-value	--	.022	.917	.103	< .001	.793

TABLE 6
Analysis of Non-Computer-Related Majors

	Count	Overall (30)	Ambiguity (8)	Omission (16)	Inconsistency (4)	Incorrect Fact (2)
Group 2a – Math/Engineering	11	5.73	1.27	2.73	1.45	.27
Group 2b – Science	8	6.13	1.38	2.38	2.25	.13
Group 2c – Business/Arts	6	4.33	.67	2.17	1.17	.33
p-value	--	.481	.307	.856	.089	.749

This result indicates that, at least for this group of subjects, the inspectors with degrees in non-computer-related majors possessed some type of knowledge or ability that gave them an advantage over the inspectors who had degrees in computer-related majors. We discuss this investigation further in Section 6. To better understand this interesting result, the subjects with degrees in computer-related majors and the subjects with degrees in non-computer-related majors were analyzed separately. The goal of this secondary analysis was to determine whether the subjects with degrees in any specific majors were overly good or overly bad. The purpose of this analysis was to provide insight into which types of inspectors an inspection manager may want to choose to compose the most effective inspection team.

For the subjects in Group 1 (computer-related majors), two subgroups were formed. Group 1a consisted of the subjects with degrees in computer science or software engineering (27 subjects), while Group 1b consisted of those with degrees in electrical and computer engineering, computer systems, and management information systems (21 subjects). The details of the analysis are shown in Table 5. The results of the one-way ANOVA indicate that the subjects in Group 1a were significantly less effective than the subjects in Group 1b both overall $[\mathrm{F}(1,46) = 3.025,\ \mathrm{p} = 0.089]$ and for inconsistency defects $[\mathrm{F}(1,46) = 4.396,\ \mathrm{p} = 0.042]$.

To analyze the subjects in Group 2 (non-computer-related majors) further, they were divided into three subgroups: Group 2a consisted of those with degrees in math or engineering (11 subjects), Group 2b consisted of those with degrees in science (eight subjects), and Group 2c consisted of those with degrees in business or the arts (six subjects). The detailed results are shown in Table 6. An ANOVA was performed to compare the means among the groups and no significant differences were found. In fact, the inspectors in each group were more effective than the inspectors in the other two groups for at least one of the defect types analyzed (highlighted in Table 6). So, unlike the subjects with degrees in computer-related majors, the subjects with degrees in non-computer-related majors were more consistent in terms of their overall effectiveness.

The previous two results indicated there was a significant difference in the effectiveness between Groups 1a and 1b within the computer-related majors, while the non-computer-related majors exhibited more similar overall performance. Therefore, the original three-way ANOVA test was conducted with one difference: Instead of only two levels for the Educational Background factor, now three levels were used (Group 1a—computer science/software engineering, Group 1b—other computer-related, and Group 2—non-computer-related). The results in Fig. 5 show that there is a significant increase in effectiveness $[\mathrm{F}(2,32) = 6.434,\ \mathrm{p} = 0.004]$ providing additional support for $H1_a$ (Table 7).

To summarize, overall the inspectors who had a background that was unrelated to computing (i.e., their degree was in engineering, math, science, business, or the arts) were significantly more effective in identifying requirements defects during an inspection. In addition, when the results were examined in more detail, the inspectors who had degrees in either computer science or software engineering were the least effective of all subjects. Based on this distribution of subject experience from Fig. 1 (computer-related subjects were more experienced) and the fact that Industrial Experience did not show up as a

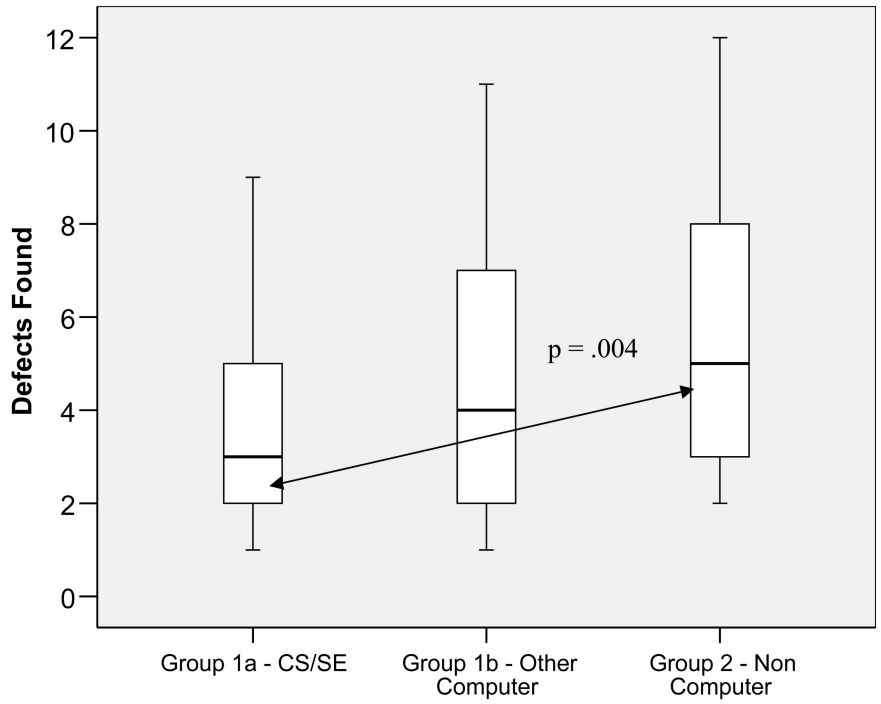

Fig. 5. Educational background (three levels).

TABLE 7
Analysis of Computer-Related Majors

	Count	Overall (30)	Ambiguity (8)	Omission (16)	Inconsistency (4)	Incorrect Fact (2)
Group 1a - CS/SE	27	3.48	.96	1.7	.59	.22
Group 1b - Other	21	4.71	1.48	1.9	1	.33
p-value	--	.089	.124	.650	.042	.401

significant factor in the overall ANOVA test, we can observe that the results were not influenced by the industrial experience of the subjects. These results do not yet clearly indicate the cause of this result, but they do indicate the need for additional research to better understand the observed phenomenon.

3.2 Requirements Experience ($H2$)

The original ANOVA analysis indicated that experience in writing requirements was a significant factor in the effectiveness of the inspectors. For this variable, the subjects were classified into two groups: those with requirements experience and those without such experience. The box plot in Fig. 6 shows the result graphically for this variable in isolation. The one-way ANOVA results for this variable in isolation indicate that the higher mean value for subjects with requirements experience is significant $[\mathrm{F}(1, 71) = 3.958,\ \mathrm{p} = 0.05]$. Therefore, $H2_0$ can be rejected and $H2_a$ can be accepted.

As with $H1$, to further investigate this variable, the results were analyzed for each individual defect type. Table 8 shows the details for this analysis using the same notational conventions used for Table 5. The inspectors with experience writing requirements were more effective for all defect types, but none of the results were significant.

These results show that, while those with requirements experience were more effective overall, no specific defect type was a major contributor to this overall effectiveness. To more fully understand the group of subjects who had experience, we performed a secondary analysis in which that group was divided into two subgroups. Some of the subjects had experience in industry while others had only classroom experience. So, a second ANOVA test was conducted using only the experienced subjects to determine whether those with industrial experience were more effective than those with only classroom experience. The results of this secondary analysis showed that the 23 subjects with only classroom experience found 5.35 defects and the 16 with industrial experience found 4.75 on average. This difference was not significant $[\mathrm{F}(1, 38) = 3.372,\ \mathrm{p} = 0.52]$.

To summarize, the results show that inspectors who have requirements experience are significantly more effective in finding defects than those without such experience. But, there is no statistical difference between the subjects with industrial experience and those with only classroom experience.

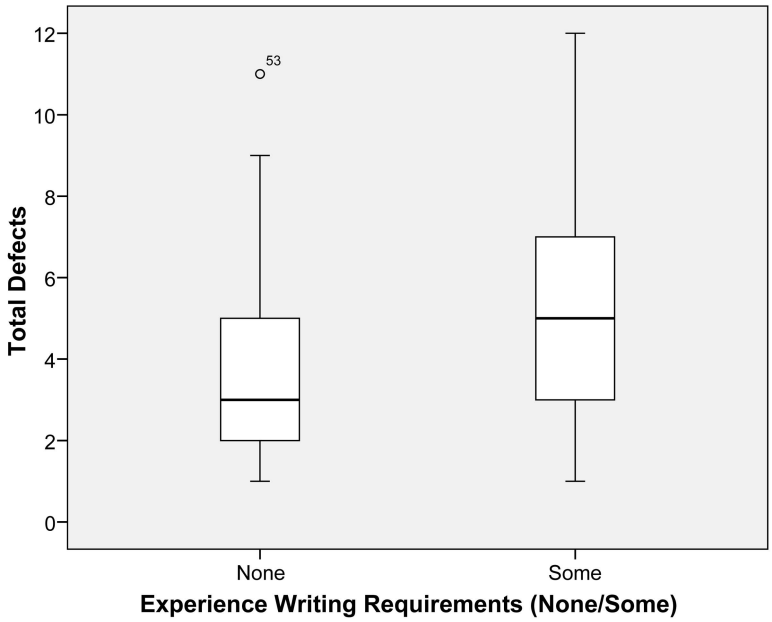

Fig. 6. Requirements experience.

TABLE 8
Requirements Experience

	Count	Overall (30)	Ambiguity (8)	Omission (16)	Inconsistency (4)	Incorrect Fact (2)
No Experience	34	3.88	1	1.71	1	.18
Some Experience	39	5.10	1.33	2.31	1.13	.33
p-value	--	.05	.181	.134	.549	.158

3.3 Other Variables ($H3$-$H5$)

Three other independent variables were considered in the overall study analysis. Two of those variables were eliminated during the factor analysis, *Industrial Experience* and *Inspection Experience*. Therefore, no conclusions can be drawn about their impact on inspection effectiveness. As a result, we were not able to evaluate $H4$ and $H5$ in this study.

The remaining variable, *Educational Degree* ($H3$), was not found to be significant in the overall three-way ANOVA. When the data is examined, this result is not surprising as the mean defect detection rate for the inspectors was identical, regardless of their highest degree obtained (4.5 defects).

To further investigate whether the three-way ANOVA test had enough power to detect a difference if it had been present, we conducted an analysis to determine the effect size that could have been detected while still maintaining a power of 0.8. The bottom row in Table 4 shows this calculation. According to these results, our experiment could have detected an effect size as small as 0.111. This effect size is smaller than the effect size that was detected for the other two main effects. Therefore, we can reject $H3_a$ in favor of $H3_0$ (the null hypothesis). In this study, the *Educational Degree* did not have a significant impact on defect detection effectiveness.

4 Threats to Validity

As with any empirical study, there are various threats to validity that must be discussed. This section explains the major threats to the validity of this study. The first threat is the threat of a selection bias in the subject population. The specific subjects who participated in this study could be the major source of the observed result and may not be repeatable by other researchers. This threat was alleviated to some degree by the fact that the participants were selected without any prior information about the composition of the class or participants. In addition, Microsoft (or, for that matter, any company) has uniform interviewing and hiring standards that should lead to a reflective sample of software engineers. Finally, the participants did not receive any compensation for participation in the study. They all participated as a part of their normal job responsibilities and therefore the level of motivation of each subject should have been similar.

The second threat, the representativeness of the artifact, is a threat to external validity. It is possible that the Loan Arranger document used in this study may not be reflective of an actual requirements document. This threat is addressed to some degree by the fact that such items of business interest and financial analysis are a major part of today's software systems. Hence, these requirements describe a realistic piece of software that is not a trivial system.

Third, there is the possibility of experimenter bias, that is, the results observed could have been influenced by the experimenters involved. In order to reduce the chance of this occurrence, the third author, who is not part of the research arm of Microsoft, conducted the experiment. The second author, who is part of Microsoft Research, helped

with logistic support like printing the documents and collecting the defect reports, but had no role in the actual execution of the experiment. He also then did all of the data collection, sorting, and mapping from all of the defect forms. The first author then did the data analysis. All three authors worked independently in order to minimize experimenter bias.

The last threat is one that is common to any empirical study. Researchers cannot draw a general conclusion based solely on the results of one study. Because of the presence of a large number of context variables, both known and unknown to the researchers, it cannot be assumed that results will always generalize beyond the setting in which the study was conducted. More confidence in a result comes from replication of a study [2]. Therefore, this study needs to be replicated, both within Microsoft and in other organizations, to build a body of empirical knowledge to allow concrete, general conclusions to be drawn.

5 Discussion of Results

The results of this study suggested that two variables are important to consider when performing requirements inspections: the educational background of the inspector and whether or not the inspector has experience writing requirements. In terms of experience writing requirements, the secondary analysis of the data did not provide any additional insight into why this variable has an effect (other than the fact that more experience is beneficial). This variable can be more specifically explored in future studies.

In terms of educational background, the results are more interesting and have some more practical applications. The findings showed that inspectors who have degrees in non-computing fields are more effective inspectors than those with degrees in a computing field. In further analysis, we found that, even within the group of subjects who had computer-related degrees, those with degrees specifically in computer science or software engineering were the least effective of all of the inspectors.

One question that arises as these results are considered is whether the subjects who had degrees in non-computing fields drawn from within Microsoft had any special characteristics or abilities that might have influenced their performance. There are two issues that were of particular concern. First, it is possible that Microsoft disproportionately assigns employees who do not have a computing background to a requirements engineering task, which would give them an unfair advantage in this study. According to a discussion with three senior Microsoft engineers, educational background does not affect the job assignments employees are given. This conclusion is supported by the background data collected in the study (62 percent of the inspectors with computing degrees had requirements experience while only 36 percent of the non-computing inspectors had such experience). Second, it is possible that there are some specific characteristics that Microsoft looks for when hiring someone with a non-computing background. To address this question, we posed these results to senior Microsoft engineers with significant experience. The experts relayed the following observations about employees from non-computing backgrounds:

- the employees have to be better than the average job applicant with a computer science-related background;
- the employee would tend not to think in terms of code immediately upon reading a requirements document as someone with a computing background would; and
- the employee is more precise in their writing due to their background training in a non-computer science field where stronger emphasis is placed on such soft skills.

These observations are purely the perception of the Microsoft engineers and were not empirically validated through any type of testing. Until we more completely understand what it means for the employee to be "better than average" and how that qualification might or might not translate to increased inspection performance, we will leave this issue as a potential threat to validity that must be tested further in future studies.

6 Conclusions and Future Work

This paper has described a study that is focused on whether the background and experience of an inspector (especially educational background) affects his performance during a requirements inspection. An important aspect of this study is that the subjects were 73 professional software developers employed by Microsoft. The results showed that inspectors with degrees in non-computer-related majors (i.e., engineering, math, science, or business) found significantly more defects during a requirements inspection than inspectors with degrees in computer-related majors (i.e., computer science, software engineering, electrical engineering, or management information systems). In addition, those with experience writing requirements were significantly more effective than those without such experience. These findings have implications on the planning of requirements inspections, on university curricula, and on corporate training programs, as summarized in Table 9.

Additional work is needed to better understand these results. First, the study described in this paper (or one that is similar) should be replicated both at Microsoft and at other companies. Additional replications will provide insight into the external validity of the results. We do hope that researchers from other companies will be interested in replicating the study. Researchers interested in replicating the study are requested to contact the authors to obtain a copy of the materials and for any other assistance needed to help replicate these studies.

Second, we plan to investigate in more detail the reasons why the inspectors with degrees in non-computer-related majors were more effective. To do this investigation, we will perform a thorough qualitative analysis of the data gathered when the master defect list was returned to the inspectors at the end of the study. The subjects were asked to go through the master list (prepared by the researchers) and indicate which items they thought they reported on their individual lists, which items they did not think were really defects, and explanations. This information should provide interesting insight into the thought process used by the subjects in each group. We also plan to follow up this study by collecting

TABLE 9
Implications of the Findings

Requirement Inspection Planning	Universities and Corporate Training Programs
1. Managers should include inspectors from varied educational backgrounds, especially those with non-computer backgrounds, to create the most effective inspection teams;	1. Graduates of many computing programs may not be acquiring the skills and background needed to be an effective requirements inspector;
2. It is important that some members of the inspection team have expertise writing requirements themselves.	2. Students need better instruction to understand the type of information and what level of detail is important at the requirements phase;
	3. Further work is needed to understand which specific skills or knowledge are possessed by inspectors with a non-computing background that are making them more effective.

additional qualitative data via one-on-one interviews that will provide insights into why the inspectors with degrees in non-computer-related majors were more effective. It will be important to understand what it means that they had to be better than the average job applicant with a computer science degree (as explained by the senior Microsoft engineers in Section 5) and whether that ability was the cause for the increased performance during the inspection.

Third, one major question that arises during the evaluation of the data from this study is whether the defects detected by the different groups were severe defects. Because the subjects in this study did not actually implement the requirements, it was difficult to determine which of the defects would have survived into the coding phase, causing serious problems. To investigate this specific question, we are preparing to conduct a follow-on study at Microsoft Research. In the new study, we will have inspectors review a real (or realistic) requirements artifact to identify defects as done in this study. Then, that requirements artifact will be given to a development team (without knowledge of the defects detected) who will create a development specification, which is the document that is used to guide the implementation process. One test of the severity of the defects will be to analyze which defects are caught by the development team when creating the development specification and which defects slip through into this document. Those that slip through can be considered major as they will directly affect the implementation. In this study, we will seek to understand whether the background of an inspector affects how many of these severe defects he or she is able to identify.

Finally, as these results have implications outside of computer science, beyond the scope of software engineering, we plan to collaborate with educational psychologists and social scientists to understand if there are any causal reasons for the results we observed.

ACKNOWLEDGMENTS

The authors would like to thank the employees of Microsoft for their participation in the study. The authors would also like to thank Edward B. Allen, Victor Basili, and Gursimran Walia for providing feedback on early drafts of this paper, J. Edward Swan for providing comments on the data analysis portion of this paper, and the anonymous referees for their useful feedback.

REFERENCES

[1] A. Aurum, H. Petersson, and C. Wohlin, "State-of-the-Art: Software Inspections after 25 Years," *Software Testing, Verification and Reliability*, vol. 12, no. 3, pp. 133-154, 2002.

[2] V. Basili, F. Shull, and F. Lanubile, "Building Knowledge through Families of Experiments," *IEEE Trans. Software Eng.*, vol. 25, no. 4, July/Aug. 1999.

[3] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, and M.V. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading," *Empirical Software Eng.: An Int'l J.*, vol. 1, no. 2, pp. 133-164, 1996.

[4] S. Biffl and M. Halling, "Investigating the Influence of Inspector Capability Factors with Four Inspection Techniques on Inspection Performance," *Proc. Eighth Int'l Symp. Software Metrics*, pp. 107-117, 2002.

[5] B. Boehm and V.R. Basili, "Software Defect Reduction Top 10 List," *Computer*, vol. 34, no. 1, pp. 135-137, Jan. 2001.

[6] J. Carver, "The Impact of Background and Experience on Software Inspections," PhD thesis, Dept. of Computer Science, Univ. of Maryland, 2003.

[7] J. Carver, F. Shull, and V. Basili, "Observational Studies to Accelerate Process Experience in Classroom Studies: An Evaluation," *Proc. Second Int'l Symp. Empirical Software Eng.*, pp. 72-79, 2003.

[8] J. Carver, F. Shull, and V.R. Basili, "Can Observational Techniques Help Novices Overcome the Software Inspection Learning Curve? An Empirical Investigation," *Empirical Software Eng.: An Int'l J.*, vol. 11, no. 4, pp. 523-539, 2006.

[9] T. Dyba, V.B. Kampenes, and D.I.K. Sjoberg, "A Systematic Review of Statistical Power in Software Engineering Experiments," *Information and Software Technology*, vol. 48, no. 8, pp. 745-755, 2006.

[10] M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, vol. 15, no. 3, pp. 182-211, 1976.

[11] M.E. Fagan, "Advances in Software Inspections," *IEEE Trans. Software Eng.*, vol. 12, no. 7, pp. 744-751, July 1986.

[12] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner, "G*Power 3: A Flexible Statistical Power Analysis Program for the Social, Behavioral, and Biomedical Sciences," *Behavior Research Methods*, vol. 39, pp. 175-191, 2007.

[13] T. Gilb and D. Graham, *Software Inspections.* Addison-Wesley, 1993.

[14] B.C. Hungerford, A.R. Hevner, and R.W. Collins, "Reviewing Software Diagrams: A Cognitive Study," *IEEE Trans. Software Eng.*, vol. 30, no. 2, pp. 82-96, Feb. 2004.

[15] V.B. Kampenes, T. Dyba, J.E. Hannay, and D.I.K. Sjoberg, "A Systematic Review of Effect Size in Software Engineering Experiments," *Information and Software Technology,* vol. 49, nos. 11-12, pp. 1073-1086, 2007.
[16] O. Laitenberger, C. Atkinson, M. Schlich, and K. El Emam, "An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents," *J. Systems and Software,* vol. 53, no. 2, pp. 183-204, 2000.
[17] T.C. Lethbridge, J. Diaz-Herrera, R.J. LeBlanc Jr., and J.B. Thompson, "Improving Software Practice through Education: Challenges and Future Trends," *Proc. 29th Int'l Conf. Software Eng., Future of Software Eng. Track,* pp. 12-28, 2007.
[18] J. Martin and W. Tsai, "N-Fold Inspection: A Requirements Analysis Technique," *Comm. ACM,* vol. 33, no. 2, pp. 223-232, 1990.
[19] P. McCarthy, A. Porter, H. Siy, and L.G. Votta, "An Experiment to Assess Cost-Benefits of Inspection Meetings and Their Alternatives: A Pilot Study," *Proc. Third IEEE Int'l Software Metrics Symp.,* p. 100, 1996.
[20] J. Miller and Z. Yin, "A Cognitive-Based Mechanism for Constructing Software Inspection Teams," *IEEE Trans. Software Eng.,* vol. 30, no. 11, pp. 811-825, Nov. 2004.
[21] D.L. Parnas and D. Weiss, "Active Design Reviews: Principles and Practice," *Proc. Eighth Int'l Conf. Software Eng.,* pp. 132-136, 1985.
[22] D.L. Parnas and D.M. Weiss, "Active Design Reviews: Principles and Practices," *Proc. Eighth Int'l Conf. Software Eng.,* pp. 132-136, 1985.
[23] A. Porter, H. Siy, A. Mockus, and L. Votta, "Understanding the Sources of Variation in Software Inspections," *ACM Trans. Software Eng. and Methodology,* vol. 7, no. 1, pp. 41-79, 1998.
[24] A.A. Porter, L.G. Votta Jr., and V.R. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," *IEEE Trans. Software Eng.,* vol. 21, no. 6, pp. 563-575, June 1995.
[25] A.A. Porter and P.M. Johnson, "Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies," *IEEE Trans. Software Eng.,* vol. 23, no. 3, pp. 129-145, Mar. 1997.
[26] C. Sauer, D.R. Jeffery, L. Land, and P. Yetton, "The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research," *IEEE Trans. Software Eng.,* vol. 26, no. 1, pp. 1-14, Jan. 2000.
[27] G.M. Schneider, J. Martin, and W.T. Tsai, "An Experimental Study of Fault Detection in User Requirements Documents," *ACM Trans. Software Eng. and Methodology,* vol. 1, no. 2, pp. 188-204, 1992.
[28] F. Shull, I. Rus, and V. Basili, "How Perspective-Based Reading Can Improve Requirements Inspections," *Computer,* vol. 33, no. 7, pp. 73-79, July 2000.
[29] F. Shull, J. Carver, and G. Travassos, "An Empirical Methodology for Introducing Software Processes," *Proc. Joint Eighth European Software Eng. Conf. and Ninth ACM SIGSOFT Symp. Foundations of Software Eng.,* pp. 288-296, Sept. 2001.
[30] H. Siy, "Identifying the Mechanisms Driving Code Inspection Cost and Benefits," PhD thesis, Dept. of Computer Science, Univ. of Maryland, 1996.
[31] T. Thelin, P. Runeson, and C. Wohlin, "An Experimental Comparison of Usage-Based and Checklist-Based Reading," *IEEE Trans. Software Eng.,* vol. 29, no. 8, pp. 687-704, Aug. 2003.
[32] T. Thelin, P. Runeson, C. Wohlin, T. Olsson, and C. Andersson, "Evaluation of Usage-Based Reading—Conclusions after Three Experiments," *Empirical Software Eng.: An Int'l J.,* vol. 9, no. 1, pp. 77-110, 2004.
[33] L. Votta, "Does Every Inspection Need a Meeting?" *Proc. First ACM SIGSOFT Symp. Foundations of Software Eng.,* pp. 107-114, 1993.

Jeffrey C. Carver received the PhD degree in computer science from the University of Maryland. He is an assistant professor in the Department of Computer Science at the University of Alabama. From August 2004 to August 2008, he was an assistant professor in the Department of Computer Science and Engineeringa at Mississippi State University. His main research interests include empirical software engineering, software quality, software inspections, software architecture, and human factors in software engineering. He is a member of the IEEE, the IEEE Computer Society, and the ACM.

Nachiappan Nagappan received the BTech degree from the University of Madras in 2001 and the MS and PhD degrees from North Carolina State University in 2002 and 2005, respectively. He is a researcher at Microsoft Research, Redmond, Washington. His research interests are in software reliability, software measurement, and empirical software engineering. He is a member of the IEEE and the ACM.

Alan Page is one of Microsoft's first test architects and is currently the director of the test group in Microsoft's Engineering Excellence team. Among other things, this means that he teaches testers how to be better testers and he has designed and taught courses for Microsoft's most senior testers and test managers. Beyond teaching, he creates and updates technical courses, working with test teams across Microsoft to help them reach their goals, and writes on his blog and elsewhere. He has been involved in software testing for more than 15 years and has been at Microsoft since 1995. He is a member of the IEEE and the IEEE Computer Society.