An Experiment to Observe the Impact of UML Diagrams on the Effectiveness of Software Requirements Inspections

Özlem Albayrak
Computer Technology and Information Systems
Bilkent University
Ankara, Turkey
ozlemal@bilkent.edu.tr

Abstract

Software inspections aim to find defects early in the development process and studies have found them to be effective. However, there is almost no data available regarding the impact of UML diagram utilization in software requirements specification documents on inspection effectiveness. This paper addresses this issue by investigating whether inclusion of UML diagrams impacts the effectiveness of requirements inspection. We conducted an experiment in an academic environment with 35 subjects to empirically investigate the impact of UML diagram inclusion on requirements inspections' effectiveness and the number of reported defects.

The results show that including UML diagrams in requirements specification document significantly impacts the number of reported defects, and there is no significant impact on the effectiveness of individual inspections.

1. Introduction

Since Fagan's initial work on inspections [1], a long history of experience and experimentation has produced a significant body of knowledge concerning the effectiveness of software inspections [2]. A properly conducted inspection may remove between 60% and 90% of existing defects [2]. Hence, software inspection provides a powerful way to improve the quality and productivity of the software process [3].

Various researchers have conducted studies to improve effectiveness of software inspections [4, 5]. Considerable research has been carried out on the structure of the inspection process. Researchers have developed several new process models that have been empirically evaluated and validated. Other studies have focused on particular methods, tools and models that support the structure of the inspection process.

Defect detection in requirements documents is one of the most effective and efficient quality assurance techniques in software engineering [6]. The Unified Modeling Language (UML) provides visualization and modeling support, and has its roots in object-oriented concepts [7]. Object-oriented modeling with UML diagrams has an important place in software development. Effects of defects in UML models were investigated [8]. In practice, it is common to see requirements documents that include UML diagrams.

Software inspections include an individual reading step, where inspectors read the artifacts alone and record any detected defects. Individual reading of artifacts relies on the reader's attributes, such as the inspector's educational background and experience [5]. In this paper, we focus on this individual defect detection step. To improve the output of the individual reading step, checklists and special reading guidelines such as special Object-Oriented Reading Techniques (OORTs) have been generated [9, 4, 10].

This paper does not propose a new method to improve effectiveness of requirements inspections. It presents an initial examination of whether inclusion of UML diagrams impacts requirements inspections' effectiveness. We undertook a study to investigate the following research questions:

RQ1) For requirements inspections, is the individual defect reporting rate affected by utilizing UML diagrams in the requirements specification document?

RQ2) For requirements inspections, is the individual inspector's defect detection rate affected by including UML diagrams in Software Requirements Specifications (SRSs)? (In other words, we study if individual inspection is more effective due to inclusion of UML diagrams (use cases and class diagrams) in the requirements specification document).

2. Study context

For initial work on UML diagrams' impact on requirements inspection effectiveness, we used two types of UML diagrams: Use cases and analysis class diagrams. We selected these diagrams because these are commonly utilized during the requirements specification phase.

3. Experimental setup

The study was conducted in classroom settings at Bilkent University during the Spring semester of 2009. In this study, we examined the following hypothesis about requirements inspection effectiveness, reporting defects and whether the requirements document included UML diagrams. For each hypothesis, the null hypothesis is followed by the alternative hypothesis:

$H1_0$: The number of defects reported by an inspector is not affected by including UML diagrams (use cases and class diagrams) in the requirements document inspected.

$H1_1$: The number of defects reported by an inspector is affected by including UML diagrams (use cases and class diagrams) in the requirements document inspected.

$H2_0$: The effectiveness of an inspector (the number of defects detected by an inspector) is not affected by including UML diagrams (use cases and class diagrams) in the requirements document inspected.

$H2_1$: The effectiveness of an inspector (the number of defects detected by an inspector) is affected by including UML diagrams (use cases and class diagrams) in the requirements document inspected.

3.1. Experimental variables

The independent variable in this experiment was whether the requirements document contained UML diagrams (use case and class diagrams) or not.

The dependent variables included:

Number of reported defects: The total number of defects reported by each subject reflects the number of possible improvements that should be made as a result of the individual inspection process.

Number of correctly detected defects: The total number of defects that were correctly detected or found by an individual inspector is a measure the of inspection process's effectiveness.

Some subjects may report defects that do not exist in our defect list. Thus, not all of the reported defects are defects that are detected correctly. To evaluate if UML diagram inclusion in a requirements document impacts these variables, we measure them both.

3.2. Subjects

The 35 subjects were volunteer senior students of two different sections of CTIS494-Software Quality Assurance, a technical elective course. Overall, the subjects had programming experience in an academic setting and, an average of six months' experience in industrial settings. They had little experience with software inspections. They were previously trained on individual reading techniques, including checklist-based reading, perspective based reading and usage based reading. All of the subjects used the same checklist used in the study during a class exercise. The subjects had experience in software engineering, developing UML diagrams, and writing SRS documents.

3.3. Materials

The experiment used two major artifacts from: Cause Effect Graphing System (CEGS) and Quality Function Deployment Online System (QFDOS). For each system, there were two requirements documents: One with UML diagrams and another without UML diagrams. The UML diagrams used were use case diagrams and class diagrams.

The artifacts with UML diagrams did not include use case descriptions and explanations about class diagrams. For both CEGS and QFDOS systems, the artifacts with UML diagrams had the same requirements set, written in natural language, as the corresponding artifact without UML diagrams.

Table 1 presents UML diagram details of the artifacts.

Table 1. Artifacts with UML diagrams

Artifact	# of UseCase Diagrams	# of Use Cases	# of Class Diagrams	# of Classes
CEGS UML	1	5	1	4
OFDOS UML	3	20	1	10

CEGS had 15, and QFDOS included 18 requirements. The same defects were seeded to the UML and non-UML versions of the requirements documents. Five defects were seeded to QFDOS-related requirements and six defects were seeded to CEGS-related requirements. In non-UML versions, information presented by UML diagrams was provided in natural language as textual explanations and in the same place before the functional requirements.

The subjects were also given a checklist to refer to while inspecting the artifacts. The checklist used throughout the experiment was the same for all artifacts. We used the standard defect report form by Carver, Nagappan and Page as the checklist [5].

3.5 Study Design

The subjects previously were informed about software requirements inspections and different reading techniques. The subjects did not study descriptions of the systems previously. The study was conducted in two days. Two days after the study was conducted by Section 1 students, it was conducted by Section 2 students. Each student inspected two different artifacts (one with UML and one without UML diagrams) one after another on the same day. The order of artifact distribution is presented in Table 2.

Table 2. Artifact subject distribution

Section I	Section II
CEGS with UML: 2 pages	CEGS: 1 page
QFDOS: 2 pages	QFDOS with UML: 6 pages
Total subjects:13	Total subjects:22

For each artifact, the allotted time for the inspection task was 30 minutes. The subjects had to follow the checklist and analyze the given artifacts for defects. They were requested to document each defect in the same form used to describe defect classes (omission, ambiguous information, inconsistent information, incorrect fact, extraneous and miscellaneous).

For each defect reported, the subjects were requested to state the corresponding defect class and write descriptions of the defects.

4. Data analysis and results

We collected 70 valid inspection documents from the 35 subjects. Each subject inspected two requirements documents. We counted the number of defects reported and the number of defects detected by each inspector.

The descriptive statistics are presented in Table 3, while the box-plots in Fig. 1 shows graphically the number of detects reported and detected for different specification documents with or without UML diagrams.

Table 3. Descriptive statistics

	Defects Reported		Defects Detected		
	Mean	*Std. Deviation*	*Mean*	*Std. Deviation*	
0: No	9,54	2,934	1,66	1,259	35
1: Yes	7,77	2,197	2,17	1,424	35
Total	8,66	2,723	1,91	1,359	70

For all statistical tests reported in this paper, we have used an alpha value of 0.05.

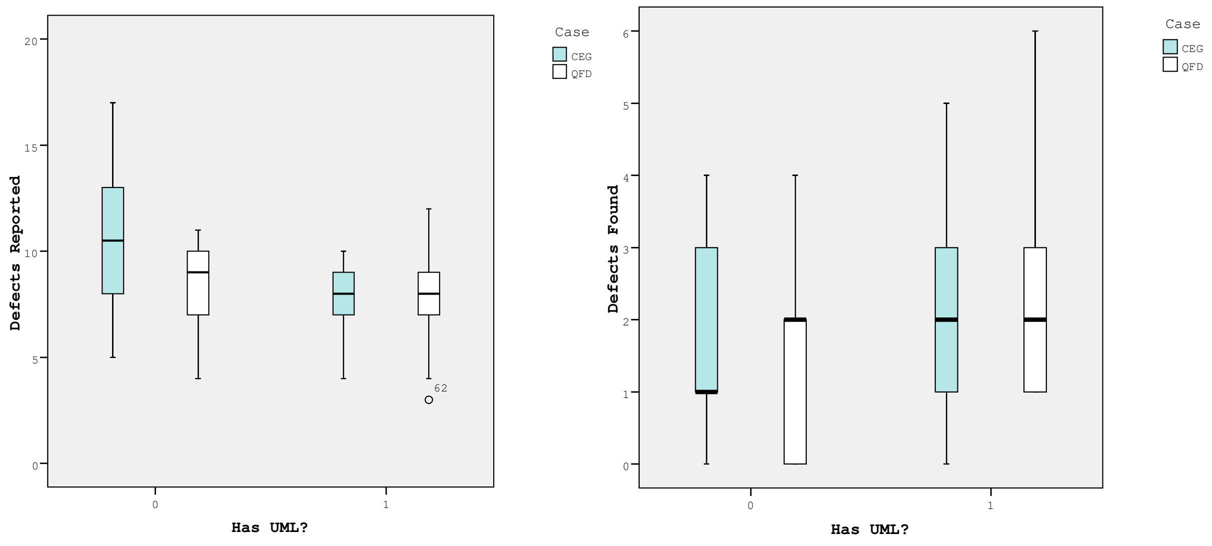

Figure 1. Defects reported and detected and requirements document with (Has UML:1) or without UML(Has UML:0).

4.1. H1: Defect report rate

We used one-way ANOVA to observe the impact of including UML diagrams in the requirements document on the number of defects reported by individual inspectors. All of the assumptions of ANOVA were satisfied.

The ANOVA was significant F (1. 68) = 8.174, p = 0.006, η2 = 0.107 (Table 4). 10.7% of variance in the number of defects reported is explained by the utilization of UML diagrams. This result allows $H1_0$ to be rejected in favor of $H1_1$. The inclusion of use cases and class diagrams in SRS impacts the number of defects reported by the subjects.

Table 4. Test of the ANOVA (Defects reported)

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	54.914	1	54.914	8.174	0.006
Within Groups	456.857	68	6.718		
Total	511.771	69			

R Squared = 0.107 (Adjusted R Squared=0.094)

4.2. H2: Defect detection rate (inspection effectiveness)

We used one-way ANOVA to observe the impact of including UML diagrams in the requirements document on the number of defects detected by individual inspectors. All of the assumptions of ANOVA were satisfied.

The ANOVA was not significant F (1. 68) = 2.562, p = 0.114, η2 = 0.036 (Table 5).

The subjects reported more defects when they inspect requirements documents without UML diagrams than when they inspect those with UML diagrams. The mean values for both CEGS and QFDOS are very close in the case of documents with UML diagrams' inspections.

Table 5. Test of the ANOVA (Defects detected)

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	4.629	1	4.629	2.562	0.114
Within Groups	122.857	68	1.807		
Total	127.486	69			

R Squared = 0.036 (Adjusted R Squared=0.022)

5. Threats to validity

As with any empirical study, this experiment exhibits a number of threats to internal and external validity. Internal validity investigates if the treatment causes the outcome; external validity deals with generalization [11].

The threat of a selection bias is the primary threat to internal validity. The subjects who participated in this study may have been the major source of the observed result. In this study, the subjects were selected based on their sections of an elective course. The participants did not receive any compensation for participation in the study. Thus, we expect the level of motivation of each subject to be similar.

The subjects are students. They may not be representative of real developers. We believe that this study helped not only the researchers but provided benefits to the students as suggested by [12]. This study may be considered as a pilot study during which the subjects have had the opportunity to learn and apply requirements inspections reading techniques. In practice, not the most experienced and qualified engineers perform inspections, thus this threat may be alleviated to some degree by assuming that the student subjects are representative to a reasonable degree [13].

The representativeness of the artifacts used is a threat to external validity. The requirements documents used in the study may not be reflective of an actual requirements document. This threat is addressed by the fact that the artifacts used in the study are shortened versions of real projects on QFDOS and CEGS.

There is the possibility of experimenter bias. The researcher who conducted the experiment and the instructor who trained the students on software inspections is the same person. Thus, the results of the experiment might be influenced by the experimenter. On the other hand, since the researcher and the instructor is the same, their goals did not conflict [12].

Common to any empirical study, researchers cannot draw general conclusions based solely on the results of one study.

6. Conclusions and future work

The main goal of this study is to investigate the impact of UML diagram utilization in SRSs on inspection effectiveness and defects reported by individual inspectors.

The analysis is based on empirical data collected during an experiment in an academic setting with 35 participating inspectors.

We analyzed the impact of UML diagram utilization in SRSs on the individual effectiveness and defect reporting rate during inspection. The analysis showed that inspectors report more defects when they inspect documents without UML diagrams. We also observed that the mean values for the number of defects detected are almost the same for both CEGS and QFDOS. On the average, the inspectors reading documents including UML diagrams detected more defects than those who inspected requirements documents without UML diagrams.

From a practical point of view we see these results as important because they reveal initial information on impact of including UML diagrams in SRS documents influencing the number of defects reported and detected by individual inspectors. However, we need to collect more data to establish greater external validity to these results. Therefore, we encourage the external replication of this study by different researchers.

We are planning to run the experiment at different academic settings and with professional subjects at software companies.

Acknowledgment

We wish to thank the subjects who collaborated with us.

References

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

[2] Ackermann, C., F. Shull, R. Carbon, C. Denger, and M. Lindvall, "Assessing the Quality Impact of Design Inspections," *Proc. of the First International Symposium on Empirical Software Engineering and Measurement.* 2007, pp.470-472.

[3] A. Aurum, H. Petersson, and C. Wohlin, 2002. "State-of-the-art: Software Inspections After 25 Years," *Software Testing, Verification and Reliability*.vol. 12, 2002, pp.133-154.

[4] R. Conradi, P. Mohagheghi, T. Arif, L. C. Hegde, G. A. Bunde, and A. Pedersen, "Object-oriented Reading Techniques for Inspection of UML Models . An Industrial Experiment," *Proc. ECOOP 2003*, LNCS 2743, 2003, pp. 483-500.

[5] J. Carver, N. Nagappan, and A. Page, "The Impact of Educational Background on the Effectiveness of Requirements Inspections: An Empirical Study", *IEEE Trans. on Software Engineering*, vol.34, 2008, pp.800-812.

[6] A. Porter, L. Votta, V. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," *IEEE Trans. on Software Engineering*, vol. 21, 1995, pp.563-575.

[7] Booch, G., J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[8] C.F.J. Lange, and M.R.V. Chaudron, "Effects of Defects in UML Models – An Experimental Investigation," *Proc. ICSE 2006,* May 20–28, 2006, pp.401-410.

[9] G.H. Travassos, F. Shull, J. Carver, and V.R. Basili, "Reading Techniques for OO Design Inspections," *Proc. Twenty-Forth Annual Software Engineering Workshop*, NASA-SEL, Greenbelt, MD, Dec. 1999.

[10] O. Laitenberger, C. Atkison, and K. El-Emam, "Using Inspection Technology in Object-Oriented Development Projects" Technical Report NRC/ERB-1077, June 2000.

[11] Wohlin, C., P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers, 1999.

[12] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "Using empirical studies during software courses," *ESERNET 2001-2003*, 2003, pp.81-103.

[13] S. Biffl, and M. Halling, "Investigating the Influence of Inspector Capability Factors With Four Inspection Techniques on Inspection Performance," *Proc. IEEE Symp. Software Metrics (METRICS 02*), 2002, pp.1-11.