ISSN- 2394-5125 VOL 08, ISSUE 02, 2021

An Experimental Analysis of the Use Case and Activity Diagrams' Efficiency in Software Requirement Inspection

Hamayoon Ghafory
Senior Lecturer, Faculty of Computer Science, Information System Department, Shahid Proff.Rbbani Education University, Kabul Afghanistan
E-mail:Hamayoon.ghafory@yahoo.com

Abstract
Software inspection is a technique for identifying errors in software artifacts such as documentation and source code. The purpose of an inspection is to visually locate and identify defects in a software product. Finding flaws early in the development life cycle is critical, as the expense of rectification grows as the development cycle progresses. Inspections can detect flaws early, as they may be done as soon as an artifact is generated. Diagrams created using the Unified Modeling Language (UML) have been widely utilized to model many elements of software systems throughout their life cycle. The purpose of this study is to determine the effect of integrating Use Case and Activity diagrams in software requirements specifications on the efficacy of inspections and the number of reported faults in an educational institution. The results indicated that students who inspected documents solely with text reported more faults than students who inspected documents with text + diagram, and those who inspected documents with text + diagram reported the same number of correct faults as those who inspected documents with text only. However, students who inspected documents solely with text reported more incorrect faults than students who inspected documents with text + diagram.
Keywords: Software Requirements, Software engineering.

1.0 Introduction

Inspection is a constant confirmation method and one of the real advantages of inspection is that, it can be applied to any artifact built through software development. Therefore, it is may be one of the few techniques that people really say yes that it is beneficial in improving software quality (Aurum et al, 2002). Detection of defect in documents requirements is one of the greatest useful quality assurance methods in Software Engineering (SE) (Porter et al, 1995). In software development the UML diagrams and object oriented modeling have an important role in specifying system requirements and it is usual to see the needs specification documents which includes UML diagrams (Albayrak, 2009). Software inspection involves individual step of reading, where the reviewer reads the artifact alone and record all defects when finds them. However, in the area of software requirements specification almost there is no data available about the effect of using UML diagrams in the specification document on the review effectiveness (Albayrak, 2009).Nowadays, ocular assessment happens during the preparation stage. The reading technique refers to the method through which the reviewer visually examines the artifact. The reading approach explains how the reviewer should inspect the artifact for flaws. Reading approaches such as checklist-based reading (CBR) and perspective-based reading (PBR) are examples. Additionally, ad hoc and checklist-based reading approaches are the most often utilized in business (Berling and Thelin, 2004).In this sense, this paper presents an experimental study in order to evaluate the effectiveness of use case and activity diagram in early stage of software development lifecycle. The following section describe the process of this experiment.

2.0 Related Work

Related work shows that what other researcher investigated in area of software requirement inspection. Therefore it is necessary to discuss about the experiment that executed in field of software requirement inspection.

Neiva et al (2009) executed an experimental study on requirement engineering for Software Product Lines (SPL). The goal of experiment was to analyze the requirement engineering process, to evaluate the efficiency, usability and impact of requirement engineering for SPL. At the result, the experiment showed which of the

ISSN- 2394-5125 VOL 08, ISSUE 02, 2021

processes allows performing requirement engineering in SPL context with good effectiveness. On the other hand, the efficiency can be improved with the use of an appropriate support tool.

Mendonca et al (2008) executed experimental replication in software engineering. They compared the use of specific inspection techniques Perspective Based Reading (PBR) and Checklist Based Reading (CBR) technique. The techniques that were used by participants in the experiment were to find defect during inspection (where the detection process is not by the experience of inspector) and Checklist Based Reading (CBR) (that each inspector focus on list of quality aspect). The result of this research has shown who had used the PBR technique found more defect in percentage and it was more useful than those who were used CBR.

Porter et al. (1994) ran a controlled study in which computer science major students examined numerous requirement specifications. Their findings indicated that meeting gain rates were comparable to Votta's gain rates. Additionally, they demonstrate that these advantages are negated by "meeting losses" (defects first discovered during preparation but never reported at the collection meeting). Again, because this problem obviously affects both inspection research and practice, greater study is necessary.

(Biffi, 2000) experimented the impact of inspector ability and the change of inspection process for a software requirements specification. In addition, this research showed controlled experiment which investigated the power of reading techniques and ability of inspector on the effectiveness to find set of defect in requirement specification document. Finally, the result of experiment showed that inspector ability played an important role for inspection performance, in the experiment with varying influence depending on reading technique and documents part.

Rong et al (2012) examined the effectiveness of checklist technique by conducting semi controlled experiments in code review for inexperienced student. They prepared two groups of students for experiment. The first group didn't use checklist during code review and second group used checklist during code review. Then, they collected the data from both groups. Finally, the result showed that Checklist is helpful for inexperienced students because checklist guides them to start and conduct code review.

3.0 Experiment

3.1 RESEARCH QUESTION & HYPOTHESES

This paper has two research questions as follows:

RQ1: For requirements inspections, what is the effect of including Unified Modeling Language (UML) diagrams (Use Case and Activity) in the Software Requirement Specification (SRS) on the rate of faults reported by inspectors?

RQ2: For requirements inspections, what is the effect of including Unified Modeling Language (UML) diagrams (Use Case and Activity) in the Software Requirement Specification (SRS) on the rate of correct faults detected report by inspectors?

To investigate these two research questions, a more detailed set of two hypotheses were defined. For each hypothesis, the null hypothesis Ha0 is presented, followed by the alternative hypothesis Ha1:

Ha0) Including UML diagrams has little effect on the amount of problems reported by inspectors. The requirements document's use case and activity diagrams were examined.

Ha1) Including UML diagrams has an effect on the amount of errors reported by inspectors. The requirements document's use case and activity diagrams were examined.

Hb0) Including UML diagrams has no influence on an inspector's efficacy, on the number of correct errors found by an inspection. The requirements document's use case and activity diagrams were examined.

Hb1) Including UML diagrams improves an inspector's effectiveness and the number of correct errors found by an inspection. The requirements document's use case and activity diagrams were examined.

3.2 VARIABLES

There were two independent variables measured to specify their effect on the two dependents variables. Each of these variables is defined in the following section

ISSN- 2394-5125 VOL 08, ISSUE 02, 2021

3.2.1 INDEPENDENT VARIABLES

- SRS with UML diagrams: the Software Requirements Specification (SRS) documents contained UML diagrams (Use Case and Activity) diagrams.
- SRS without UML diagrams : the Software Requirements Specification (SRS) documents do not contain UML diagrams (Use Case and Activity) diagrams

3.2.2 DEPENDENT VARIABLES

- The overall number of problems identified or discovered by the inspector shows the overall number of potential changes that should be done as a consequence of the inspection procedure.
- Correct faults reported by inspector: the total number of correct defects recognized or discovered by the inspector is a proxy for the inspection process's performance.

3.3 EXPERIMENT DESIGN

3.3.1 PARTICIPANTS

There were 40 participants were selected from fourth class of software engineering faculty at Shahid Prof.Rabbani Education University 20 of them were female and the other 20 of them were male. The students didn't have industries experience but they were familiar with programing, data base, data analysis, software engineering and UML diagrams. 20 of them were inspected the two documents with text only and half them were inspected the documents with UML diagrams use case and activity (text + diagram). Firstly, students were guided how to transfer the faults to the checklist.

3.3.2 MATERIALS

In this study two software system were used: Parking Garage Control System (PGCS) and Online Registration System (ORS), each system was contained two requirements documents, one without UML diagrams (Text only) and another with UML diagrams use case and activity diagram (text+diagrams). The documents with UML diagrams did not describe the use case and activity diagram for both systems (PGCS AND ORS). In non UML version, the information of requirements document was provided in natural language as textual explanation before the functional requirements. In UML version, the diagrams were used after information of the system before the functional requirements. They had the same requirements set that were written in natural language as the corresponding documents without UML diagrams. Table 4.1 shows the number of use case diagram, activity diagram, use cases and activities which were used in requirements documents.

Table4.1: Number of Use case diagram, Activity diagram, Use cases and activities

Documents	Number of Use-Case Diagrams	Number of Use Cases	Number of Activity Diagram	Number of Activity
PGCS	1	6	1	13
ORS	1	5	1	5

The Both two systems (PGCS and ORS) had 12 requirements in ORS and PGCS related requirements 7 defects were seeded. The same defects were seeded to the requirements documents with UML diagrams and without UML diagram. There were three types of defects were seeded in requirements documents omission, incorrect fact and ambiguous information. The document requirements of PGCS were prepared in three pages without UML diagram and in four pages with UML diagram. The document requirements of ORS were prepared in two pages without UML diagram and in three pages with UML diagram as shown in table 4.2.

Table4. 2:document requirements

Non-UML-diagram	Number of pages	Participants	With-UML-diagram	Number of pages	Participants
PGCS	3	22	PGCS	4	22
ORS	2	22	ORS	3	22

ISSN- 2394-5125 VOL 08, ISSUE 02, 2021

The students had checklist while they inspecting the documents and the checklist was the same for both documents. The checklist standard was used from albayrak (2009) and sample of checklist has shown in table 4.3.

Table4.3: Sample of checklist

Defect No	Functional Requirement NO	Defect Type	Description

3.3.3 PROCEDURE

There were two software system Online Registration System (ORS) and Parking Garage Control System (PGCS), each software was written in two documents one with text only and other text with diagram. The ORS was clear and understandable for students, they didn't need for explanation. The PGCS was described for students and students were guided how to transfer faults to the checklist. The experiment was done in two days; in first day 20 students participated and inspected the two documents with text only. The students were divided in two groups 10 by 10 and the documents prepared 10 by 10 with text only the first group inspected the ORS and then PGCS. The second group inspected the PGCS and then ORS. The second day 20 students were participated and inspected the documents with UML diagrams use case and activity (text + diagram). The students were divided in to two groups the first group inspected the ORS documents with diagram and then PGCS with diagram. The second group inspected first the PGCS document and then ORS documents with diagram. Each student received two documents and checklist for reporting of faults with defect types for information. The SRS document without UML diagrams were inspected about 2 hours by inspectors and the SRS document with UML diagrams were inspected about 1 and half hours by inspectors.

4.0 Data Analysis

The participants inspected the ORS with text only reported 98 faults that 55 was correct reported and 43 was incorrect reported faults. For PGCS with text only they reported 109 faults that 55 were correct faults and 54 was incorrect report faults. The participants that inspected ORS with text + diagram they reported 70 faults that they reported 47 correct faults and 23 was incorrect faults. When they inspected the PGCS with text + diagrams reported 91 faults that 63 was correct faults and 28 was incorrect faults. Therefore, participants that reviewed text only reported more faults than participants inspected the text + diagram.

5.0 Discussion

The participants that inspected the SRS documents with text only reported more faults and they used about 2 hours for inspection that start from 10:00 am to 12:00. The faults that they reported were consist of correct faults and incorrect faults. The number of correct and incorrect faults that participants found and reported for SRS with text only in both systems was approximately same. In contrast, the students that inspected the SRS documents with text + diagrams (use case and activity diagrams) reported less faults in less time and they inspected the both SRS about in 1 and half hour which start from 2:30 pm to 4:00. The faults that they reported were consist of correct faults and incorrect faults. The number of correct faults that participants found and reported were more than the number incorrect faults that participants found and reported for SRS with text + diagrams in both systems. In addition, the inspectors that inspected the SRS with text only found more faults than those inspectors that inspected the documents with text + diagrams but the inspectors that inspected the SRS with text only reported more incorrect faults from total of reported faults. The reason that participants inspected the documents with text only in long time and reported more faults were:

ISSN- 2394-5125 VOL 08, ISSUE 02, 2021

- The inspectors read many times the documents to understand the system and functional of the system.
- It was difficult for inspectors to compare the functional requirement of the system with description of the system.
- The inspectors could not decide soon about error in functional requirements.

The reason of inspected the documents with text + diagrams (use case and activity) in less time and reported lest faults were:

- The diagrams help inspectors to understand the main functional requirement of the system and avoid from waste of time.
- The inspectors easily find the correct faults when see the diagrams.
- The inspectors do not need to read many times the description of the system.
- The diagrams help inspectors to compare the functional requirement of the system with description of the system to find faults in less time.

6.0 Conclusion

This paper investigate the effectiveness of use case and activity diagram on software requirement inspection at early phase of software development life cycle by conducting an experimental study at Shahid prof Rabbani Education University Kabul Afghanistan

This study will be carried out to achieve the following objectives:

- To conduct an experiment and evaluate the effect of including UML diagrams, Use Case diagram and Activity diagram in the usefulness of software inspections.
- To evaluate the effectiveness of faults finding in the early phase of software development life cycle using UML diagrams (Use case and Activity diagrams).

As a conclusion, according to research question-1what is the effect of including Unified Modeling Language (UML) diagrams (Use Case and Activity) in the Software Requirement Specification (SRS) on the rate of faults reported by inspectors? The UML diagrams does not affect to the number of total faults reported by inspectors, but the UML diagrams (use case and activity) help inspectors to understand the main functions of SRS documents and avoid them from reporting incorrect faults, because without UML diagrams the participants reported more incorrect faults. In addition, the UML diagrams help the inspectors to understand the main function of the system during the inspection and find or reported the correct faults in less time than the SRS documents with text only. According to research question-2: what is the effect of including Unified Modeling Language (UML) diagrams (Use Case and Activity) in the Software Requirement Specification (SRS) on the rate of correct faults detected report by inspectors? The participants that inspected the documents without UML diagrams reported same correct faults as those participants that inspected the documents with UML diagrams (use case and activity) but those participants that inspected the documents without UML diagrams spent more time than those participants that inspected the documents with UML diagrams. It means that use case and activity diagrams help the inspectors that find correct faults in less time during inspection.

References

Albayrak, O. 2009. An experiment to observe the impact of UML diagrams on the effectiveness of software requirements inspections. *3rd International Symposium,* pp. 506-510.

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

Berling, T., and Thelin, T. 2004. A case study of reading techniques in a software company. *In Empirical Software Engineering, ISESE'04. Proceedings. 2004 International Symposium on*, pp. 229-238.

ISSN- 2394-5125 VOL 08, ISSUE 02, 2021

Biffl, S. 2000. Analysis of the impact of reading technique and inspector capability on individual inspection performance. *In Software Engineering Conference. Proceedings. Seventh Asia-Pacific,* pp. 136-145.

Mendonça, M. G., Maldonado, J. C., de Oliveira, M. C., Carver, J., Fabbri, C. P. F., Shull, F., and Basili, V. R. 2008. A framework for software engineering experimental replications. *In Engineering of Complex Computer Systems, 13th IEEE International Conference on,* pp. 203-212.

Neiva, D. F. S., de Almeida, E. S., & de Lemos Meira, S. R. 2009. An experimental study on requirements engineering for software product lines. *In Software Engineering and Advanced Applications,* pp. 251-254.

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

Rong, G., Li, J., Xie, M., & Zheng, T. 2012. The Effect of Checklist in Code Review for Inexperienced Students: An Empirical Study. *In Software Engineering Education and Training (CSEE&T), 2012 IEEE 25th Conference on,*. pp. 120-124.

AUTHORSPROFILES

Hamayoon Ghafory is a senior lecturer at Shahid Prof. Rabbani Education University at the Computer Science Faculty Kabul Afghanistan. He received his Master Degreein Software Engineering in 2014. His interests include software Engineering, Mobile Application and Web Engineering.