A Comparison of Use-Case Based and Checklist-Based Reading: A Controlled Experiment

Abstract—**Software quality can be defined as the customers' perception of how a system works. Inspection is a method to monitor and control the quality throughout the development cycle. Reading techniques applied to inspections help reviewers to stay focused on the important parts of an artifact when inspecting. However, many reading techniques focus on finding as many faults as possible, regardless of their importance. Use-case based reading helps reviewers to focus on the most important parts of a software artifact from a user's point of view. This study presents an experiment, which compares use-case based and checklist-based reading.**

Keywords—Controlled experiment; empirical study; reading techniques; software inspection

I. INTRODUCTION

Software inspections have since its inception [1] more than 25 years ago spawned quite some interest both from the research community and industrial practice. The research includes changes to the inspection process, e.g. [2], support to the process, e.g. [3] and empirical studies, e.g. [4]. The suggested improvements include active design reviews [5] and perspective-based reading [6]. Industry has studied the benefits of conducting software inspections [7].

The objective of this study is to compare and hence evaluate how well the use-case based reading performs in comparison to other methods. The study presents a controlled experiment where use-case based reading is compared to checklist-based reading.

II. EXPERIMENT PREPARATION

This section describes the preparation needed to conduct the experiment and the subjects acting in the experiment. The experimental package developed for this experiment can be found at http://www.thewebsite.com

A. Subjects

The students participating as reviewers in the study were 22 third-year Software Engineering and Management Bachelor's students at University of Gothenburg in Sweden. Many of the students have experience from software development. As part of their bachelor degree, they have obtained practical training in software development.

The objective of the experiment, from an educational perspective, was that the students should be exposed to an empirical study at the same time as they were introduced to some of the on-going research in the area.

B. Inspection Material

The material consists of four documents in structured text: one requirements document, one design document written in the *specification and description language* (SDL), one use case document, and one checklist.

The requirements document was written in natural language (English). The document is used as a reference document to show how the system is meant to work. The checklist consists of 18 check items and is based on a checklist presented by Laitenberger et al. [8]. It would have been preferable to use a checklist from an industrial application to check this kind of design, but no such checklist was found. Therefore, we used a modified version of a checklist utilized in experiments with the purpose of comparing Use-Case Based reading and Checklist-Based reading. The checklist items were modified to fit the taxi management system and they were sorted in order of importance.

The subjects inspected the design document using the requirements document as a reference. To guide the reading they used either a use-case document or a checklist. The design consists of software modules of a taxi management system and descriptions of signals in-between these modules. The modules are one taxi module for each vehicle, one central module for the operator and one communication link, see Figure 1. Furthermore, the design document consists of two *message sequence charts* (MSC) [9], which show signalling among the modules for two different cases, one normal case and one special case. The use cases are written in *task notation* [10] and are prioritized using the *analytic hierarchy process* (AHP) [11] from a user's point of view, i.e. the function of the first use case is the most important to the user while the last use case is least important. The use case document contains 19 use cases. The design document contains 38 faults.

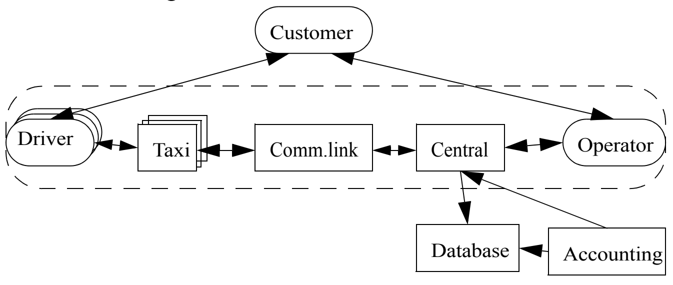

Figure 1. The taxi management system.

C. Fault Classification

The faults were divided into three classes depending on the importance for the user, which is a combination of the probability of the fault to manifest as a failure, and the severity of the fault considered from the user's point of view.

Class A faults – The functions affected by these faults are crucial for the user, i.e. the functions affected are important for the user and are often used. An example of this kind of faults is: the operator cannot log in to the system.

Class B faults – The functions affected by these faults are important for the user, i.e. the functions affected are either important and rarely used or not as important but often used. An example of this kind of fault is: the operator cannot log out of the system.

Class C faults – The functions affected by these faults are *not* important for the user. An example of this kind of fault is: a signal is missing in a table summarizing all signals, but it is correctly defined in the section that describes the signals.

The design document contains 13 *class A faults*, 14 *class B faults* and *11 class C faults*. No syntax errors like spelling errors or grammatical errors were logged as faults. One person made the classification of the faults prior to the experiment.

III. Experimental Planning

A. Variables

Three types of variables are defined for the experiment, *independent*, *controlled*, and *dependent* variables.

a) Independent Variable: The independent variable is the reading technique used. The experiment groups used either Use-Cased Based Reading or Checklist-Based Reding.

b) Controlled Variable: The controll variable is the experience of the reviwers and it is measured on an ordinal scale. The reviwers were asked to fill in a questionnare comprising seven questions.

c) Dependent Variables: The dependent varialbles measured are faults. The four variables are: (1) Number of faults found by each reviwer, (2) Number of faults found by each experiment group, (3) efficiency (faults/hour) and is measured as: 60*(number of fault found/inspection time which is 45 min), and (4) effectivness (detection rate) and is mesured as: number of faults found/total number of faults.

B. Hypotheses

The general hypothesis of the experiment is that Use-Case Based Reading is more efficient and effective in finding faults of the most critical fault classes, i.e. Checklist-Based Reading is assumed to find more faults per time unit, and to find a larger rate of the critical faults.

The dependent variables are analyzed to evaluate the hypotheses of the experiment. The main null and alternative hypotheses [12] are stated below. These are evaluated for all faults, class A faults and class A&B faults. The hypotheses concern efficiency, effectiveness and fault detecting differences:

- $H_{0\ Eff}$ – There is no difference in *efficiency* (i.e. found faults per hour) between the reviewers applying use cases and the reviewers using a checklist.
- $H_{1\ Eff}$ – There is a difference in *efficiency* between the reviewers applying prioritized use cases and the reviewers using a checklist.
- $H_{0\ Rate}$ – There is no difference in *effectiveness* (i.e. rate of faults found) between the reviewers applying use cases and the reviewers using a checklist.
- $H_{1\ Rate}$ – There is a difference in *effectiveness* between the reviewers applying use cases and the reviewers using a checklist.
- $H_{0\ Fault}$ – The reviewers applying use cases do not *detect different* faults than the reviewers using a checklist.
- $H_{1\ Fault}$ – The reviewers applying use cases *detect different* faults than the reviewers using a checklist.

C. Design

The students were divided into two groups, one group using Use-Case Based Reading and one group using Checklist-Based Reading. Using the controlled variable to get a block design, the students were randomized, resulting in 11 students in the Use-Case Based Reading group and 11 students in the Checklist-Based Reading group. A questionnaire with seven questions was used to explore the students' experiences in programming, inspections, SDL, use cases and taxi systems. The questionnaire showed that the students had similar types of backgrounds based on the experience.

The instrumentation of the experiment consists of one requirements document, one design document, one use case document, one checklist and one inspection record. The inspection record contains fields to collect all the data used to analyze the experiment.

D. Threats to Validity

The threats to the validity of the experiment are analyzed below. As the purpose of the study is to compare two reading techniques, and more studies are needed for generalization purposes, the threats to internal and construct validity are most critical. When trying to generalize the results to a more general domain, the external validity becomes more important [13].

Threats to *conclusion validity* are considered under control. Robust statistical techniques are used, measures and treatment implementation are considered reliable. Random variation in the subject group is blocked, based on the controlled variable.

Concerning the *internal validity*, the risk of rivalry between groups is considered the largest one. Their grade on the course was not affected by the performance in the experiment. This could lead to lack of motivation. The students were introduced and motivated to empirical research before the experiment. In order to discuss the results in the course, they had to give good inputs to the empirical work. Furthermore, the students were randomly assigned. Thus, the threat of lack of motivation was minimized in the experiment.

Threats to the *construct validity* are not considered very harmful. The development of the textual requirements document was performed after the development of the use cases. Hence, there is a risk that the use cases may have affected the requirements document to make it suitable for the use cases. On the other hand, the inspection object was the design document and the requirements document was just a reference.

Concerning the *external validity*, the use of students as subjects is a threat. However, the students are third year bachelor students in software engineering and management. Another threat to the external validity is the design document used in the experiment. Only one domain is considered and the size of the inspected document is in the smaller range for real-world problem, even though it describes a real-world problem.

IV. Experimental Operation

The experiment was run over a two-hour lecture during spring 2014. During the first hour, the students were given an introductory of the taxi management system. All students were reading the same material of the system, which included the same material for both groups. Then they were divided into two groups depending on the method they were going to use during the inspection experiment. The second hour included performing the experiment and the students were not allowed to discuss with each other.

The package for the experiment contained:

1. Inspection record, including:
    a. A description of the fault classification
    b. A fault log. For each fault found, the students logged the use case/checklist item used, the class and severity of the fault.
2. A requirements document
3. A design document
4. Either use-case document or an inspection checklist

The instructions for the students were:

1. The textual requirements are assumed to be correct
2. Read through all documents briefly before starting to inspect
3. The inspection experiment is finished when everything is checked or after 45 minutes. When finished, verify that the logged data seem to be correct and hand them in.

After each student handed in their inspection record, an inspection moderator checked for errors and missing data in the record in order to get as accurate data as possible.

References

[1] Fagan, M. E. "Design and Code Inspections to Reduce Errors in Program Development", *IBM System Journal*, 15(3):182-211, 1976.

[2] Bisant, D. B. and Lyle, J. R., "A Two-Person Inspection Method to Improve Programming Productiv- ity", *IEEE Transactions on Software Engineering*, 15(10):1294-1304, 1989.

[3] Basili,V.R.,Green,S.,Laitenberger,O.,Lanubile,F.,Shull,F.,Sørumgård,S. andZelkowitz,M.V., "The Empirical Investigation of Perspective-Based Reading", *Empirical Software Engineering: An In- ternational Journal*, 1(2):133-164, 1996.

[4] Basili,V.R.,Shull,F.andLanubile,F.,"BuildingKnowledgethroughFamilie sofExperiments", *IEEE Transactions on Software Engineering*, 25(4):456-473, 1999.

[5] Parnas, D. L. and Weiss, D. M., "Active Design Reviews: Principles and Practices", *Proc. of the 8th* International Conference on Software Engineering, pp. 418-426, 1985.

[6] Shull, F., Ioana, R. and Basili, V. R., "How Perspective-Based Reading Can Improve Requirements In- spections", *IEEE Computer*, 33(7):73-79, 2000.

[7] Weller, E. F., "Lessons from Three Years of Inspection Data", *IEEE Software*, 10(5):38-45, 1993.

[8] Laitenberger, O., Atkinson, C., Schlich, M. and El Emam, K., "An Experimental Comparison of Read- ing Techniques for Defect Detection in UML Design Documents", *Journal of Systems and Software*, 53(2):183-204 2000.

[9] ITU-T Z.120 *Message Sequence Charts*, MSC, ITU-T Recommendation Z.120, 1996.

[10] Lauesen, S., Software Requirements – Styles and Techniques, Addison-Wesley, UK, 2002.

[11] Saaty, T. L. and Vargas, L. G., Models, Methods, Concepts & Applications of the Analytic Hierarchy *Process*, Kluwer Academic Publishers, Netherlands, 2001.

[12] Montgomery, D., *Design and Analysis of Experiments*, John Wiley & Sons, USA, 2000.

[13] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. and Wesslén, A., *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publisher, USA, 2000.