Are Good Code Reviewers Also Good at Design Review?

Hidetake Uwano, Akito Monden, Ken-ichi Matsumoto
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama Ikoma, Japan
{hideta-u, akito-m, matumoto}@is.naist.jp

ABSTRACT

Software review is a necessity activity to build high reliability software in software development. In this paper, we experimentally analyze the difference in performance between two types of (checklist based) software reviews: design review and code review. If good code reviewers were also good at design review, then we should assign good code reviewers to the design review too. If not, that means these two reviews require different types of expertise. In our experiment, with ten review participants, we examined two hypotheses each related to the defect detection ratio and the required time to find a defect. As a result, we found that there was no correlation between two reviews, i.e. good code reviewers were not necessarily the good design reviewers. This suggests the need of a completely different training program for each review.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Code inspections and walk-throughs*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors*

General Terms

Human Factors

Keywords

Code review, design review, experimental evaluation

1. INTRODUCTION

Improving quality of software documents by reading them is essential to reduce development cost. Software review including software inspection and other reading techniques can be performed in early phases of software development without implemented system, therefore, rework costs can be reduced [2]. Especially in large scale projects, defect detection and defect correction consume huge resources, defect detection by review in early phases (e.g. requirements definition and/or architectural design) of the development is necessary. Many of studies about review have been conducted such as proposals of systematic reading techniques and experimental comparison of reading techniques [1, 4, 5].

In this paper, we experimentally analyze the difference in performance between two types of (checklist based) software reviews: design review and code review. If good code reviewers were also good at design review, then we should assign good code reviewers to the design review too. If not, that means these two reviews require different types of expertise, and we need a completely different training program for each review. So far, there are no past researches that compared the difference in individual performance between design review and code review.

We examined two hypotheses each related to the defect detection ratio and the required time to find a defect. In the experiment, ten participants reviewed a design document (containing defects) using its requirements specification (containing no defect), then they were asked to review source code (containing defects) using its requirements specification and the revised design document (containing no defect). During the experiment, performance measures (defect detection ratio and review time per defect) were recorded.

ESEM'08, October 9–10, 2008, Kaiserslautern, Germany..

2. EXPERIMENTAL EVALUATION

2.1 Hypotheses

In this paper, we examine the following two hypotheses about review performance between design and code review. These hypotheses indicate "good code reviewers" are "good design reviewers."

- H_{ratio}: Reviewers who detect more defects in design review detect more defects in code review, and, reviewers who detect less defects in design review detect less defects in code review.
- $H_{efficiency}$: Reviewers who require less time to detect a defect in design review require less time to detect a defect in code review, and, reviewers who require more time to detect a defect in design review require more time to detect a defect in code review.

2.2 Outline of Experiment

Subjects were asked to find defects from a target document in design review and code review. First, subjects performed design review with four documents (requirements specification, design document, data file, and a checklist for

design review) to detect defects injected in the design document beforehand. Then, the subjects performed code review with five documents (requirements specification, design document, C source code, data file, and a checklist for code review) to find injected defects in the source code. Reviews were finished when a subject (reviewer) concluded the target document had no more defects.

In both review, time spent for a review and the number of detected defects were collected for the analysis. Subjects were nine graduate students and one faculty member of Nara Institute of Science and Technology. An average of their programming experience was 7.6 years, and 2.4 years for programming with C language. Two of them had experience of software development in industry.

2.3 Materials

Documents used in the experiment were about a rental house search system actually used in an industrial training workshop. The documents consist of requirements specification, design document, source code and data file. This system reads a data file in which a set of rental houses are listed. A system user inputs a condition about rental houses (e.g. distance from a nearest train station, floor space and rent) that he/she wants to look at. According to the user input, the system outputs a list of rental houses that match the condition.

- **Requirements Specification:** This document consists of 40 lines of Japanese text, describing system functions and requirements.
- **Design Document:** This document describes details of each function's interface, data and processes. It consists of 30 lines of Japanese text.
- **Source Code:** This is 5 functions, 120 steps C language program. All comments were removed in advance.
- **Data File:** This file is read by the system when the system starts. The file consists of a list of rental houses.
- **Checklist:** We prepared two generic checklists, one for C source code review and the other for design document review. Both checklists were written based on existing literatures [3, 6].

2.4 Defects

We injected nine defects in total (three defects for each of three defect types) into the design document in advance. Defect types are described as follows.

- **Inconsistency with requirements:** This defect type means that the design document contains a function described in the requirements specification but it does not fulfills the requirements.
- **Omission of requirements:** This means, the design document has no description about a function described in the requirements specification.
- **Excess design:** This defect type indicates there exist excess descriptions in the design document, which has not described in the requirements specification. This defect can be also considered as insufficient description of the requirements specification.

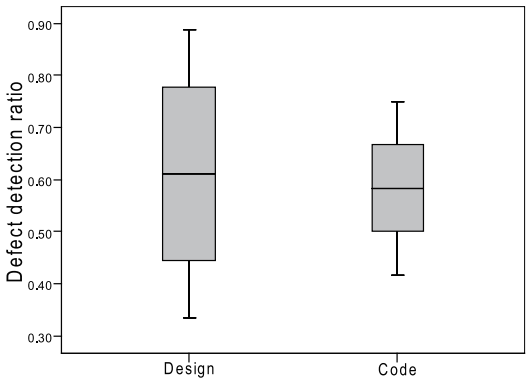

Figure 1: Defect detection ratio in code and design review.

We injected three types, eight defects in the source code. In addition, this source code had four excess functions that were not described in the requirements specification. We consider these excess implementation as defects. As a result, the source code contained twelve defects in total. Defect types are described as follows.

- **Data:** This defect type includes incorrect definition and usage of variables.
- **Process:** This defect type means incorrect functional logic (such as incorrect conditional statements.)
- **Incorrect message:** Incorrect message output to display, which causes user confusion and/or mistakes.
- **Excess implementation:** This type of defect indicates excess implementation in source code, which has not described in the requirements specification and the design document. This defect can be also considered as insufficient description of the requirements specification and the design document.

3. RESULT

The average review time was 58.9 minutes (24.9 minutes for design review, and 34.0 minutes for code review.) Figure 1 shows defect detection ratio (the percentage of detected defects of each subject) in design review and code review. The average defect detection ratio in design review was 62.2%, and 58.3% in code review.

Figure 2 and Figure 3 show correlation between design review performance and code review performance in terms of defect detection ratio and detection time per defect respectively. Similarly, Table 1 describes their correlation coefficients and p-values. Obviously, the correlation coefficient of defect detection ratio between two reviews had no statistical significance ($r = 0.052$, $p = 0.887$.) Also, the correlation coefficient of detection time per defect had no significance ($r = 0.201$, $p = 0.577$.)

From further analysis, it turned out that one subject who had industry experience achieved the best defect detection ratio in design review. This person also found defects more than the average in code review. However, the other subject who also had industry experience found defects less than the average in both design and code review. Moreover, as

Table 1: Correlation between code and design review.

		Detection ratio (code)	Review time / defect (code)
Detection ratio (design)	Pearsons' r	0.052	0.422
	p-value	0.887	0.225
Review time / defect (design)	Pearsons' r	0.299	0.201
	p-value	0.402	0.577

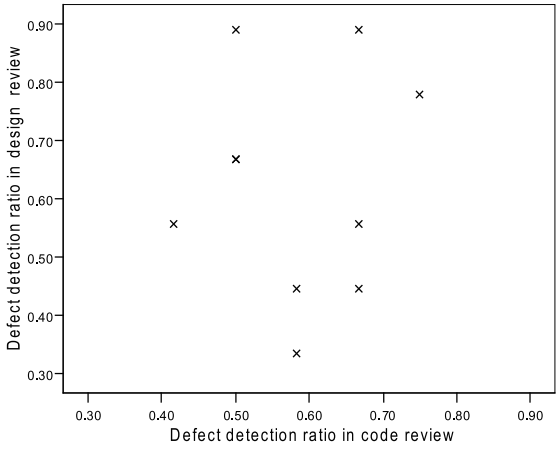

Figure 2: Correlation between defect detection ratio in code and design review.

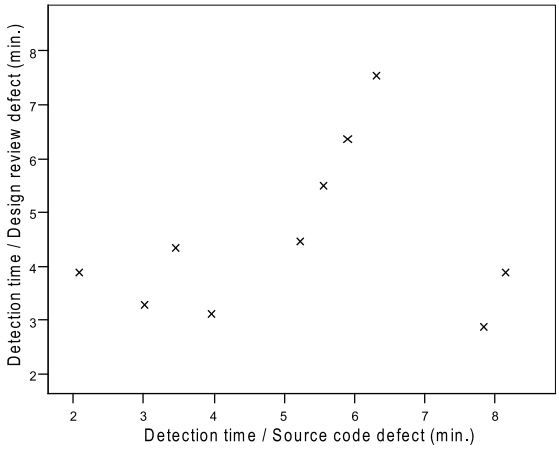

Figure 3: Correlation between spent time to detect a defect in code and design review.

for remaining eight subjects, those who found defects more than the average in design review found defects less than the average in code review and vice versa (seven of eight subjects).

We also conducted an analysis focusing on each defect type; however, there was no significant correlation in review performance between two reviews (Due to limited space, we do not describe its detail in this paper).

Obviously, these results rejected the hypothesis H_{ratio} (Reviewers who detect more defects in design review detect more defects in code review) as well as $H_{efficiency}$ (Reviewers who require less time to detect a defect in design review also require less time to detect a defect in code review.)

4. CONCLUSION

This paper experimentally analyzed the difference in performance between design and code review. As a result, we found that there was no correlation between these two reviews, i.e. good code reviewers were not necessarily the good design reviewers. This indicates we should not simply assign good code reviewers to design review (good design reviewers to code review as well). We believe there is a need of a completely different training program for each review.

The major limitation of our experiment is that the sample size was small (ten subjects). Also, we used just one software system to be reviewed. In the future, we are to increase the sample size with different software systems.

5. ACKNOWLEDGMENTS

This work was conducted as a part of the StagE Project, the Development of Next Generation IT Infrastructure, supported by Ministry of Education, Culture, Sports, Science and Technology.

6. REFERENCES

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

[2] O. Laitenberger, T. Beil, and T. Schwinn. An industrial case study to examine a non-traditional inspection implementation for requirements specifications. In *Proceedings of Eighth IEEE Symposium on Software Metrics*, pages 97–106, 2002.

[3] A. A. Porter, L. G. Votta, and V. R. Basili. Comparing detection methods for software requirements inspection - a replicated experiment. *IEEE Transaction on Software Engineering*, 21(6):563–575, 1995.

[4] G. Sabaliauskaite, F. Matsukawa, S. Kusumoto, and K. Inoue. An experimental comparison of checklist-based reading and perspective-based reading for UML design document inspection. In *Proceedings of the 2002 International Symposium on Empirical Software Engineering*, page 148, Washington, DC, USA, 2002.

[5] F. Shull, I. Rus, and V. Basili. How perspective-based reading can improve requirements inspections. *IEEE Computer*, 33(7):73–79, 2000.

[6] T. Thelin, P. Runeson, and C. Wohlin. An experimental comparison of usage-based and checklist-based reading. *IEEE Transaction on Software Engineering*, 29(8):687–704, 2003.