# An Exploratory Study on the Impact of Usage of Screenshot in Software Inspection Recording Activity

Tatsuya Sasaki
*Graduate school of Information Science,*
*Nara Institute of Science and Technology*
*8916-5, Takayama, Ikoma, Nara, Japan 630-0192*
*tatsuya-s@is.naist.jp*

Shuji Morisaki
*Faculty of Informatics, Shizuoka University*
*3-5-1, Johoku, Naka, Hamamatsu,*
*Shizuoka, Japan 432-8011*
*ismoris@ipc.shizuoka.ac.jp*

Kenichi Matsumoto
*Graduate school of Information Science,*
*Nara Institute of Science and Technology*
*Email: matumoto@is.naist.jp*

***Abstract*—This paper describes an exploratory study on the use of screenshots for recording software inspection activities such as defect reproduction and correction. Although detected defects are usually recorded in writing, using screenshots to record detected defects should decrease the percentage of irreproducible defects and the time needed to reproduce defects during the defect correction phase. An experiment was conducted to clarify the efficiency of using screenshots to record detected defects. One practitioner group and two student groups participated in the experiment. The recorder in each group used a prototype support tool for capturing screenshots during the experiment. Each group conducted two trials: one with a general spreadsheet application to support recording, the other with the prototype tool that supports recording inspection activities. After the inspection meeting, the recorder was asked to reproduce the recorded defects. The percentage of reproduce defects and time to reproduce defects was measured. The results of the experiment show that use of screenshots increases the percentage of reproduced defects and decreases the time needed to reproduce the defects. The results also indicate that use of the recording tool affected the types of defects.**

***Keywords*-Software inspection, Recording activity, Tool support**

## I. Introduction

By detecting and fixing defects early[1], software inspection activities enable improved software quality and reduced development costs. Many techniques for reading support [8] [10] and estimation of the number of remaining defects [11] [9] have been proposed. However, relatively little research has focused on methods of recording software inspection activities. Both researchers and practitioners have identified the importance and need for support tools for software inspection [4].

Most existing support tools are specific to particular source code [6]. Some support geographically dispersed inspection [5], while others support inspectors in identifying false positives [2]. However, a general supporting tool for software inspection including requirements, design documents, and source code needs investigation.

As the first step of investigation for such a general support tool for software inspection, we focused on recording defect detection activities. Recording such activities is important because a wrong description or irreproducible record will cause faulty defect correction. The research question of this study was "Does the use of screenshots in recording software inspection activities increase the efficiency of recorders and inspectors?" In this paper, we describe an experiment using a prototype tool to help capture and record screenshots.

This research supposes that detected defects are documented for correction in writing with figures if needed. We also assume that target documents are inspected on the computer screen, and that documented defects are also recorded in files. This paper describes an investigation on the use of screenshots to help with recording and reproducing software inspection activities.

This paper is organized as follows. Section 2 describes support for recording in software inspections. Section 3 provides an overview of the experiment, while Section 4 shows the results of the experiment. Section 5 discusses the experimental results. Section 6 summarizes our findings.

## II. Support for recording of defect

To clarify our assumptions about software inspections, this section describes phases of inspection, the inspection meeting, and information usually recorded about defects. We also describe a prototype tool for recording defects. This tool supports recording of defect locations by capturing screenshots. The tool enables a user to capture a screenshots by clicking on a document displayed on the computer screen.

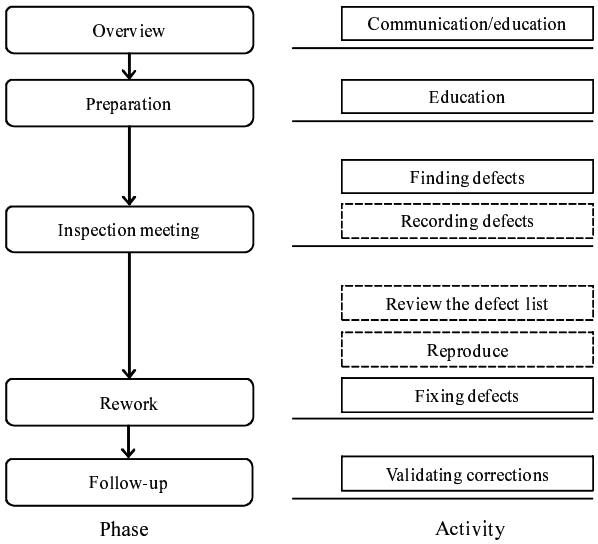


Figure 1: Inspection phase

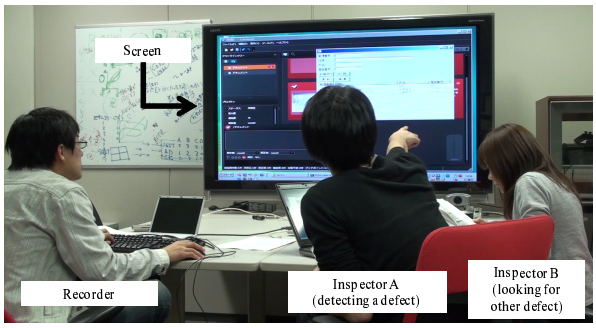


Figure 2: An example inspection meeting

| | |
|---|---|
| Time | 2010/07/01 13:21 |
| Location | p.121 line 24 and figure 4-2 |
| Description | External system is not described in figure 4-2 and no definition of interface of the external system |

Figure 3: Example record of defect

### A. Phases

Figure 1 shows phases of inspection. The rounded rectangles in Figure 1 represent the inspection phases defined in Fagan's article [1]. The squared rectangles in Figure 1 represent activities in each phase. We added dotted rectangles to clarify support for recording activity. In the "Inspection meeting" phase, when inspectors find a defect ("Finding defect"), a recorder writes down the location and symptoms of the defect ("Recording defects").

During the inspection meeting phase, the activities of finding defects and recording defects are repeated as many times as needed. After the "Inspection meeting," detected defects are corrected in the "Rework" phase. In the Rework phase, authors try to remember and identify a defect detected in the inspection meeting so that they can fix the defect in the artifact ("Review the defect list"). If the author can remember a detected defect and identify the defect in the artifact ("Reproduce"), the author will try to fix the defect ("Fixing defects"). These activities are also repeated as many times as the number of defects in the defect list.

### B. Inspection meeting

In the inspection meeting phase, one recorder and one or more inspectors participate. Depending on the target artifact, one or more authors may also participate. The inspectors detect defects, which are recorded by the recorder. The recorder also operates a computer connected to a shared large-screen that can be viewed by all participants in the inspection meeting. The inspection target is saved in a file and can be viewed on the shared screen. A printed target can be distributed to participants, but the target file must be stored in the recorder's computer for screenshots.

A list of defects detected and recorded in the inspection meeting is also displayed to a shared screen. Whenever an inspector finds a defect, the recorder inputs defect information to the defect list on the shared screen. If the inspector thinks the recorded information should be different, s/he can point that out.

Figure 2 shows an example scene from an inspection meeting. In figure 2, two inspectors and one recorder were participating. Inspector A explained a detected defect to the recorder, who input effect information based on the explanation. Inspector B was looking for other defects in the printed target. The screen shows the inspection target and defect list.

### C. Defect information

Recorded defect information includes fields such as:

- Time of detection
- Location
- Symptom (defect description)

Location and symptoms are recorded in natural language. Figure 3 shows an example description of a detected defect. In this paper, we expect to provide a clearer description of location and symptoms by adding screenshots to the description.

### D. Prototype tool

A prototype tool was developed for easily capturing screenshots of inspection targets. Capturing screenshots should reduce the recording effort in the inspection meeting phase, and the effort to recall and reproduce defects in the rework phase. The functions of the prototype tool are:

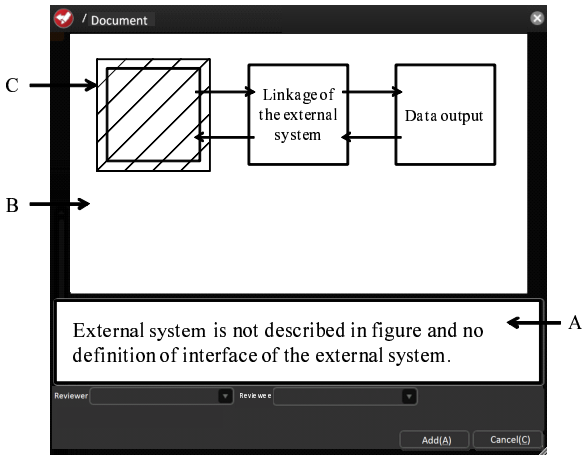


Figure 4: Screenshot of prototype

| index | Description | Location | Time and Date |
|---|---|---|---|
| 1 | The table of contents should not contain itself | First line in the table of contents in Specification | 17-Mar-09 15:05:06 |
| 2 | No revision record | Specification | 17-Mar-09 15:05:27 |
| 3 | No description of the way to add a new plan | Specification 1. add a new plan | 17-Mar-09 15:09:34 |
| 4 | No description of the scheme to delete a plan | Specification | 17-Mar-09 15:10:54 |

Figure 5: Defects list of spreadsheet

- Capture function: the prototype tool captures a specified area by clicking and adds the captured screenshot to the defect information.
- Marker information: the prototype tool highlights a specified area in the captured screenshot by clicking.

The recorder can capture an screenshot of the inspection target and highlight a specific area which needs a remark, if necessary. Figure 4 shows a screenshot of the prototype tool. In figure 4, area A is a text box for defect location and description. The captured screenshot is put in area B. the diagonal area indicated by C was marked using the marker function.

## III. Experiment

### A. Overview

The goal of the experiment was to identify improvements in the efficiency of recording and reproduction due to the use of screenshots in recording software inspection activities.

The experiment covered both the inspection meeting phase and rework phase as described previously. During the inspection meeting phase, to record detected defects, a general spreadsheet application and the prototype tool were used. The duration of recording was measured in the inspection meeting phase. The duration needed to recall and reproduce each defect in the defect list from the inspection meeting phase was measured in the rework phase.

### B. Settings

Table I shows the combinations of tools, subjects and inspection targets. Three groups of subjects participated in the experiment. Each group consists of three subjects, one recorder and two inspectors. The members of two groups are master students in information science. The members of one group are practitioners engaged in commercial software development. Due to the issue of cost, another practitioner group experiment has not been conducted yet.

Two inspection targets were prepared. Both of the inspection targets were design documents for small GUI applications. One is an address book application (“Address book” in Table I). The other is a time management application (“Time keeper” in Table I). Each trial in the experiment was conducted in a crossover way as defined in the article [3]. The spreadsheet application had a format prepared for recording defects. Figure 5 shows an example of the defects list. Where necessary, just as with the prototype, the recorder can capture screenshots of the defect descriptions.

## IV. Experimental result

### A. Use of Screenshots in Recording Defects

Table II shows the results. The number of records in Table II are total numbers from two trials (column 2). For student group 1 and the practitioner group, the average time to reproduce defects using records with screenshots is smaller than when using records with text only. Student group 2 did have an average time for reproducing defects using records with screenshots that is larger than when using records with text only.

However, when compared to the student group 1 and the practitioner group, the difference in the average time for reproducing defects is smaller (Student 2: 1.8 sec., Student 1: 11.4 sec., Practitioner: 7.4 sec.). Also, in student group 1, the difference in the number of records with screenshots and the number of records without screenshots is larger than those of student group 2 and the practitioner group. In the experiment with practitioners, two recorded defects could not be reproduced using the recorded information without Screenshots.

### B. Tool support for recording

Table III show the measurements comparing the prototype tool with the spreadsheet application. In Table III, row corresponds to a trial. The second from left column of the tables represents trial IDs. The ID of each trial in Table III corresponds with the IDs shown in Table I. There are several important points to notice in these tables. First, in both tables, S1-B has no screenshots recorded. Second, for student group 1 (S1-A, S1-B) and the practitioner group (P1-A, P1-B), the number of detected defects recorded with the spreadsheet application is larger than with the prototype tool. However, for these same groups, trials with the prototype tool have higher percentages of records with screenshots

Table I: Settings of the experiment

| | | Time keeper | Address book |
|---|---|---|---|
| Master student | Group 1 | Prototype(S1-A) | Spreadsheet(S1-B) |
| | Group 2 | Spreadsheet(S1-A) | Prototype(S2-B) |
| Practitioner | Group 1 | Spreadsheet(P1-A) | Prototype(P1-B) |

Table II: A comparison between defect records with Screenshots and without screenshots

| | Record with | Number of defect records | Number of irreproducible defect | Average of reproducing time (s) | Average of recording time(s) |
|---|---|---|---|---|---|
| Master student1 | Screenshot and text | 5 | 0 | 12.2 | 49.0 |
| | Text | 45 | 0 | 23.6 | 40.4 |
| | Total | 50 | 0 | | |
| Master student2 | Screenshot and text | 14 | 0 | 11.7 | 42.2 |
| | Text | 13 | 0 | 9.9 | 48.4 |
| | Total | 27 | 0 | | |
| Practitioner | Screenshot and text | 25 | 0 | 21.4 | 91.2 |
| | Text | 24 | 2 | 28.8 | 67.1 |
| | Total | 49 | 2 | | |

Table III: A comparison between Prototype and Spreadsheet

| | ID | Recording method | Number of defects | Number of records with screen shot | Average of reproducing time (s) | Average of recording time (s) |
|---|---|---|---|---|---|---|
| Master student1 | S1-A | Prototype | 24 | 5 | 16.0 | 57.6 |
| | S1-B | Spreadsheet | 26 | 0 | 28.4 | 25.5 |
| | | Total | 50 | 5 | | |
| Master student2 | S2-A | Spreadsheet | 13 | 8 | 9.8 | 81.5 |
| | S2-B | Prototype | 14 | 6 | 11.8 | 78.6 |
| | | Total | 27 | 14 | | |
| Practitioner | P1-A | Spreadsheet | 30 | 6 | 30.6 | 59.2 |
| | P1-B | Prototype | 19 | 19 | 16.4 | 37.3 |
| | | Total | 49 | 25 | | |

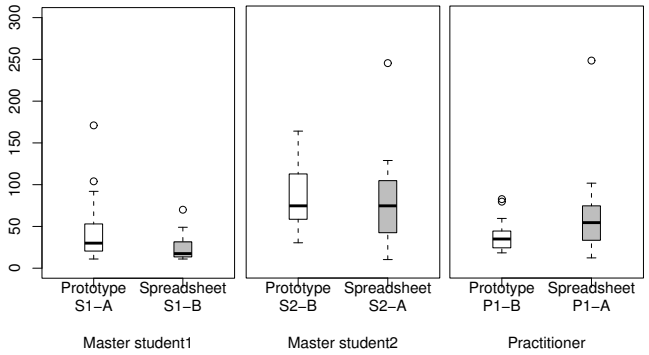


Figure 6: Distribution of durations recording time

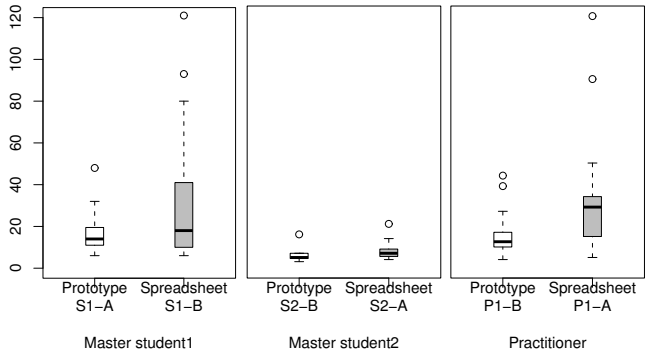


Figure 7: Distribution of durations reproducing time

than trials with the spreadsheet application. Also, for these two groups, the average time to reproduce defects with

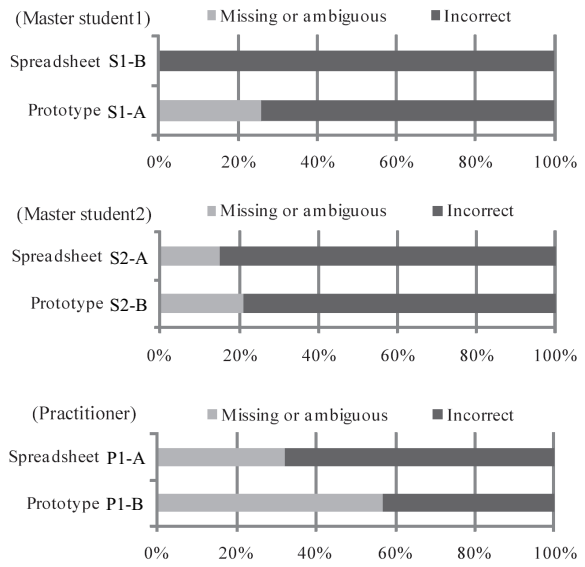


Figure 8: Classification of missing ambiguous and incorrect

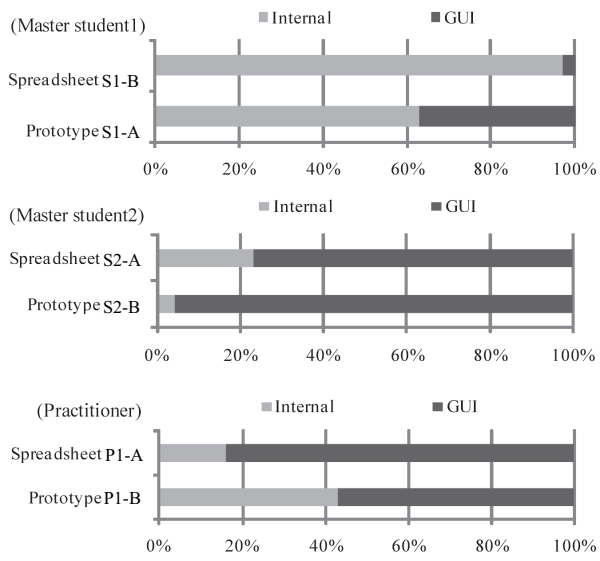


Figure 9: Classification of internal and external(GUI)

records from the prototype tool was smaller than that needed with records from the spreadsheet application.

Figure 6 and 7 show the distributions of recording time durations and reproducing time durations. The vertical axes represent time in second.

### C. Defect classification

We categorized the detected defects in two ways. One is missing/ambiguous and incorrect [8]. Defects classified as "missing/ambiguous" lacked a description or had an unclear description. Defects classified as "incorrect" had a wrong or inconsistent description. The other categorization is internal or GUI (external). Defects classified as "internal" are defects in the internal design including the architecture, functional interface definitions, and data structure. Defects classified as "GUI" are defects in the external design including lack of records for layout of GUI components, messages, and labels for the user.

Figures 8 and 9 show the classification results of the experiment. Figure 8 shows the ratio of defects classified as "missing/ambiguous" and "incorrect." For all the groups in this experiment, with the prototype tool, the ratio of defects classified as "missing/ambiguous" is larger than with the spreadsheet application. Figure 9 shows the ratio of defects classified as "internal" or "GUI." For student groups 1 and 2, the ratio of defects classified as "internal" with the spreadsheet application is larger than with the prototype tool. With the practitioner group, however, this is reversed.

## V. Discussion

### A. Defects failed to be reproduced

The practitioner group had two records of defects that could not be reproduced after the inspection meeting. We asked the recorder why he could not remember the recorded defects. The reasons were ambiguous locations and ambiguous descriptions. The recorder said that with a captured screenshot he might have been able to recall and reproduce the defects.

### B. Improved efficiency due to the use of screenshots

Table II, along with Figures 6 and 7, indicate:

1) The use of screenshots in recording defects reduces the time to reproduce defects. (All trials)
2) The use of screenshots in recording defects prevents failure to reproduce defects. (practitioner trials)

Table III suggests that the prototype tool may reduce the time needed to reproduce defects based on comparing the average times for reproducing defects from each recording method. This result may be due to pictorial superiority [7] reducing the time needed to reproduce defects.

### C. Defects categorization

In all groups, the defect categorization results indicate that use of the prototype tool increases the number of defects classified as missing/ambiguous. In two of the groups, the defect categorization results indicate that use of the prototype tool may increase the number of defects classified as GUI. These results may indicate that the goal of the software inspection should determine the use of support tools for recording software inspection activities. For example, when performing requirements and design inspections to identify missing/ambiguous defects, the use of a support tool such as the prototype tool may be desirable.

### D. Limitations

Due to the effort and cost necessary to conduct the experiment, the experiment was conducted with only one group of practitioners. Further investigation and experiments with practitioner groups is required.

## VI. CONCLUSION

This paper reports on an investigation of the efficiency of using screenshots in recording software inspection activities and reproducing defect reports after the inspection meeting. One practitioner group and two student groups participated in an experiment. Every group conducted two trials: one trial with a general spreadsheet application, and the other trial with a prototype tool providing support to capture screenshots to record defects.

The results of the experiment showed that the use of the screenshots in recording defects increased the accuracy of reproducing the defects. The results from two groups showed that the use of screenshots decreased the time needed to reproduce defects from the defect record. However, the results also indicated a negative relationship between the time needed to record defects and the time needed to reproduce defects, so that increasing the percentage of screenshots captured with records also increases the time required to record defects.

Results of the experiment also indicated that support for capturing screenshots may affect the classification of defects in such categories as missing/ambiguous and incorrect. Although further investigations are needed, we believe that the use of screenshots increases the efficiency of recording activities in software inspection and that the use of screenshots also increases the accuracy of defect classification. Future work in this area should include measurement of the duration of recording to clarify the required costs and benefits of the use of screenshots.

## VII. ACKNOWLEDGMENT

The work is being conducted as a part of the Development of Next Generation IT Infrastructure Program. This work was supported by Grant-in-Aid for Scienticfic Research (B) 23300009 and Young Scientists (B) 21700033 by Ministry of Education, Culture, Sports, Science and Technology, Japan.

## REFERENCES

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

[2] P. Grunbacher, M. Halling, S. Biffl An empirical study on groupware support for software inspection meetings. In *Proceedings of the Eighteenth IEEE International Conference on Automated Software Engineering*, 2003.

[3] J. Dolado, M. Harman, M. Otero, L. Hu An empirical investigation of the influence of a type of side effects on program comprehension. *IEEE Transactions on Software Engineering*, 29:665–670, 2003. crossover design.

[4] O. Laitenberger, J. DeBaud An encompassing life cycle centric survey of software inspection. *Journal of Systems and Software*, 50:5–31, 2000.

[5] F. Lanubile, T. Mallardo, and F. Calefato Tool support for geographically dispersed inspection teams. *Software Process: Improvement and Practice*, 8:217–231, 2003.

[6] F. Macdonald, J. Miller, A. Brooks, M. Roper, M. Wood Automating the software inspection process. *Automated Software Engineering*, 3:193–218, 1996.

[7] D. Nelson, V. Reed, R. John Pictorial superiority effect. *Journal of Experimental Psychology: Human Learning and Memory*, 2:523–528, 1976.

[8] A. Porter, L. Votta, V. Basili Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on Software Engineering*, 21:563–575, 1995.

[9] P. Runeson, C. Wohlin An experimental evaluation of an experience-based capture-recapturemethod in software code inspections. *Empirical Software Engineering*, 3:381–406, 1998.

[10] T. Thelin, P. Runeson, B. Regnell Usage-based reading-an experiment to guide reviewers with use cases. *Information and Software Technology*, 43:925–938, 2001.

[11] S. Wiel, L. Votta Assessing software designs using capture-recapture methods. *IEEE Trans. on Software Engineering*, 19(11):1045–1054, 1993.