DRESREM 2: An Analysis System for Multi-Document Software Review using Reviewers' Eye Movements

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

Abstract

To build high-reliability software in software development, software review is essential. Typically, software review requires documents from multiple phases such as requirements specification, design document and source code to reveal the inconsistencies among them and to ensure the traceability of deliverables. However, most previous studies on software review (reading) techniques focus on finding defects in a single document in their experiments. In this paper, we propose a multi-document review evaluation system, DRESREM 2. This system records reviewers' eye movements and mouse/keyboard operations for analysis. We conducted eye gaze analysis of reviewers in design document review with multiple documents (including requirements specification, design document, etc.) to confirm the usefulness of the system. For the performance analysis, we recorded defect detection ratio, detection time per defect, and fixation ratio of eye movements on each document. As a result, reviewers who concentrated their eye movements on requirements specification found more defects in the design document. We believe this result is good evidence to encourage developers to read high-level documents when reviewing low-level documents.

1. Introduction

Software review[1] is a technique to improve the quality of software documents and detect defects (i.e. bugs or faults) by reading the documents [2]. In software review, a developer reads requirements specification, design document, source code and other documents to understand systems' functions and structures, then detects defects from the documents. Defect detection by review can be performed in the early phases of software development without implementing the system, therefore, future rework costs can be reduced [5]. Especially in large scale projects, because defect detection and correction consume huge resources, defect detection by review is necessary.

Many of studies about review have been conducted, such as proposal of systematic reading techniques, experimental comparison of reading techniques, and analysis of reviewers' behavior [1, 3, 4, 5, 6, 7]. According to these experiments, PBR (Perspective-Based Reading) is relatively more effective than CBR (Checklist-Based Reading) and AHR (Ad-Hoc Reading), which are commonly used techniques in the industry [6]. Most of these studies assume only a single document is used in the review. In these studies, subjects read a target document (requirements specification, design document, source code or others) with a specific review technique, and find defects from the document.

However, software review in the industry uses not only the target document but also other relevant documents [10]. For example, in source code review, reviewers read source code as well as the requirements specification and design document to understand system structures, functions, and data structures. Also, reviewers compare source code (target document) with the requirement/design document (related document) to find inconsistencies between design and implementation. Moreover, comparison of documents is performed to confirm the traceability among different phases of documents. In such multi-document review, time spent to read each document and reading procedure should affect the defect detection performance. Hence, we believe empirical and quantitative analysis of review behavior in multi-document review is required for development of effective review techniques and/or guidelines.

In this paper, we propose a multi-document review evaluation system, DRESREM 2. This system is an enhancement of our previous system DRESREM [9],

[1]In this paper, we use the word "review" to indicate software review, inspection, walkthrough and/or other reading techniques.

which was used to record reviewers' program reading procedure (single document review). Our enhanced system DRESREM 2 has the following four characteristics to enable us to analyze multi-document review processes using reviewers' eye movements: 1) Detection of reviewers' document switching (among requirements specification, design document, source code, checklist, etc.), 2) Line-wise eye movement recording, 3) A feature enabling reviewers to take notes about detected defects during software review and 4) Data analysis support (visualizing and replaying eye movements and document switching.) These characteristics facilitate observation of multiple document review and analysis of eye movements.

In the following Sections, we explain the architecture of the system and its characteristics. Then we describe an experiment of design document review to evaluate the system's usefulness.

2. Multi-document Review

Industry developers usually review the target document (e.g. source code) with its high-level documents (requirements specification, design document) or other related documents (test specification, user manual) [10]. Figure 1 describes the relationship between source code and other documents at source code review. Source code has several blocks of functions, methods, classes, etc. In single-document review (only with source code), a reviewer reads all the blocks to understand the program wholly (e.g. through *Function a* to *Function d*) and tries to find any defect during program understanding.

In addition to this, in multi-document review (source code review with requirement/design document), reviewer reads each block of source code (e.g. *Function b*) as well as related blocks in the requirement/design document (e.g. *Design A* and *Requirement* α) to find any inconsistencies among different levels of blocks. This activity is "comparison" rather than "understanding."

To analyze such reviewer activity in the multi-document review, we adopt eye movements of reviewers. Using the eye movements allows us to observe of reviewers' reading patterns and quantitative analysis of the relationship between the pattern and the review performance. We implemented a multi-document review evaluation system to record reviewers' eye movements in the review. To make clear the purpose of the system, we present four requirements to be satisfied by the system.

- **Requirement R1: Detection of document switching**

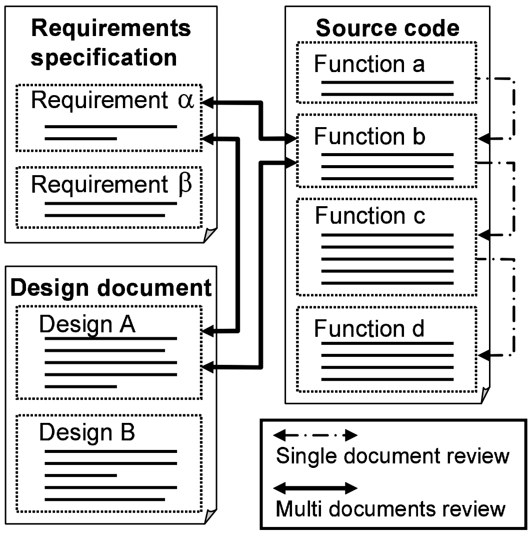


Figure 1. Source code review with multiple documents.

To observe multi-document review activities, the system is required to identify which document is read by the reviewer. Usually, multiple documents were displayed in multiple windows or a window that has a tab to switch documents displayed in the window, hence documents can be overlapped with other documents during review tasks. This means the current focus of the reviewer cannot be identified from coordination of eye movements alone. Therefore, the system should have a functionality to identify which document is currently focused on by the reviewer by recording tab switching activities and window focusing activities.

- **Requirement R2: Line-wise recording of eye movements**
A primary construct of a document is a line. In particular, most programs are written on a one-statement-per-line basis. So, it is reasonable to consider that the reviewer reads the document in units of lines. Hence, the measuring environment has to be capable of identifying which line of the document the reviewer is currently looking at. Note that the information must be stored as logical line numbers, which is independent of the font size or the absolute position where the document lines are currently displayed.

- **Requirement R3: Enable reviewers to take notes about detected defects**

To analyze the relationship between review performance and the reading procedure, details of detected defects need to be recorded. Hence, the system must enable reviewers to take notes about details of detected defects, e.g. document name, location (line number) and date.

- **Requirement R4: Data analysis support**
  Preferably, the measuring environment should provide tool support to facilitate an analysis of the recorded data. In particular, features to play back and visualize the data will contribute to the efficient analysis. Such features are also useful for the purpose of educating novice reviewers.

In Section 3, we explain how the proposed system satisfies these requirements.

3. DRESREM 2

3.1. Outline

A multi-document review evaluation system, DRESREM 2 was developed based on a single-document review evaluation system, DRESREM [9]. Figure 2 shows the architecture of DRESREM 2. This system consists of an Eye Tracking Device, Fixation Analyzer and Review Platform. We used non-contact eye mark tracker EMR-NC[2] to record subjects' eye movements. Figure 3 shows the eye tracking device used in the system. Fixation Analyzer is a software tool to calculate fixation points from sampled gaze points. Fixation is a particular coordinate at which the eye mark stays for a given moment. The fixations can be useful to distinguish an instance of reading from a glance. Review Platform is a software system to show documents for the reviewers and record their operations. This platform was implemented in Java language comprising 5700 steps and 80 classes with SWT (The Standard Widget Kit)[3]. The platform shows the documents to reviewers through **Document Viewer**. A screenshot of Document Viewer is shown in Figure 4. Reviewers select a document that they want to display on the Document pane using the Document tab located on the top of the Document viewer.

DRESREM 2 measures how a reviewer reads each line of a document on the computer display using eye movements and operation logs (e.g. document switching and window scrolling.) When a reviewer finds a defect in a document, the reviewer takes notes about the defect in the pop up window, which appears when the reviewer double-clicks a line of the document. The system records these notes with a document name, line number and date. In addition, the reviewer can take notes about anything whenever he/she wants using Memo pane. The reviewer can also search for any keyword in a document using the Search pane during the review.

[2]http://www.eyemark.jp/
[3]http://www.eclipse.org/swt/

3.2. System functions and procedures

The procedure of recording reviewers' eye movements and operations is as follows (Figure 2). Documents used in the review are displayed in the Document Viewer. The Eye Tracking Device outputs the reviewer's gaze points, represented as coordinates (x, y) on a display. These sampled gaze points are converted to fixation points by the Fixation Analyzer. **Window Event Capturer** observes user operations on a Document Viewer and records Window information, i.e. window position and window size, and current scroll position (line number) of the document currently focused on. This satisfies Requirement R1. **Fixation Point/Line Converter** calculates the logical line number of a document from Window information and fixation points. This satisfies Requirement R2. Reviewer operations such as defect description recording, keyword searching and taking of notes are recorded by **Operation Recorder**, then **Review Information Integrator** combines the operations and eye movements to create the time series data of the review history. This satisfies Requirement R3.

Recorded eye movements and operations are visualized in **Result Viewer**. Figure 5 shows an example of visualized eye movements and operations in a source code review. In this figure, the left side of the window shows a source code that is read in the review, and the right side of the window describes eye movement fixations and operations as a bar chart. Also, the sequence of the eye movements can be played back in this window. These features satisfy Requirement R4.

DRESREM 2 outputs review history (i.e. time series of eye movements and operations) as three type formats, document-wise, block-wise and line-wise. In each format, eye movements were recorded as series of fixations on documents, blocks or lines. These formats allow users to easily analyze the review history from different granularities.

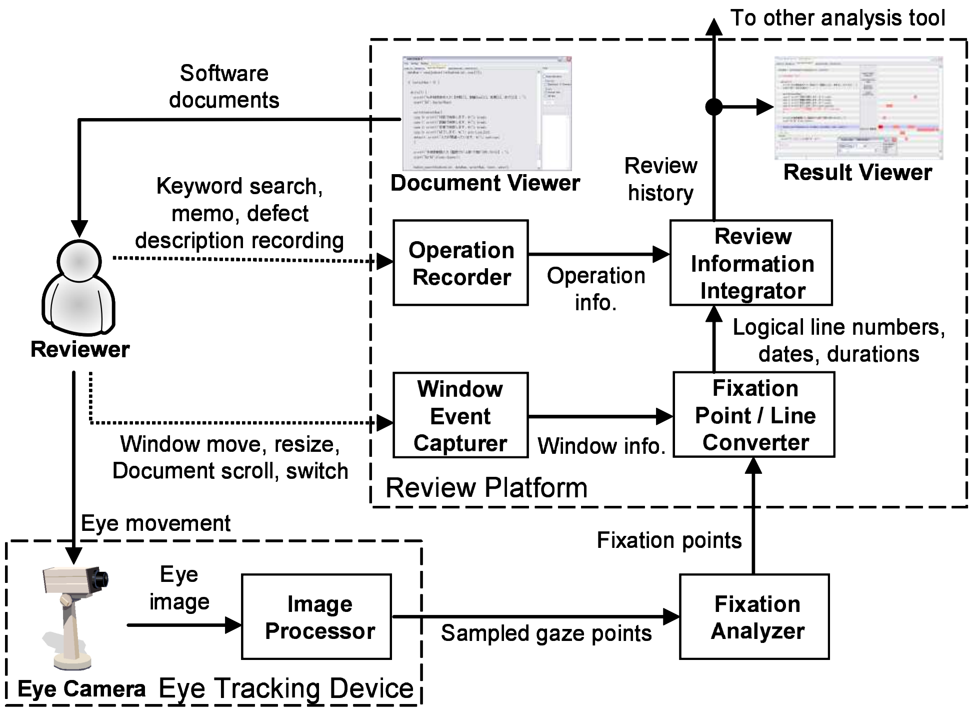


Figure 2. Architecture of DRESREM2.

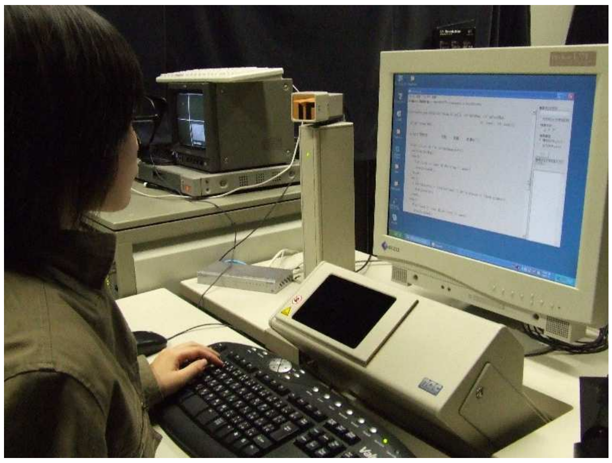


Figure 3. Eye tracking device EMR-NC.

4. Case Study

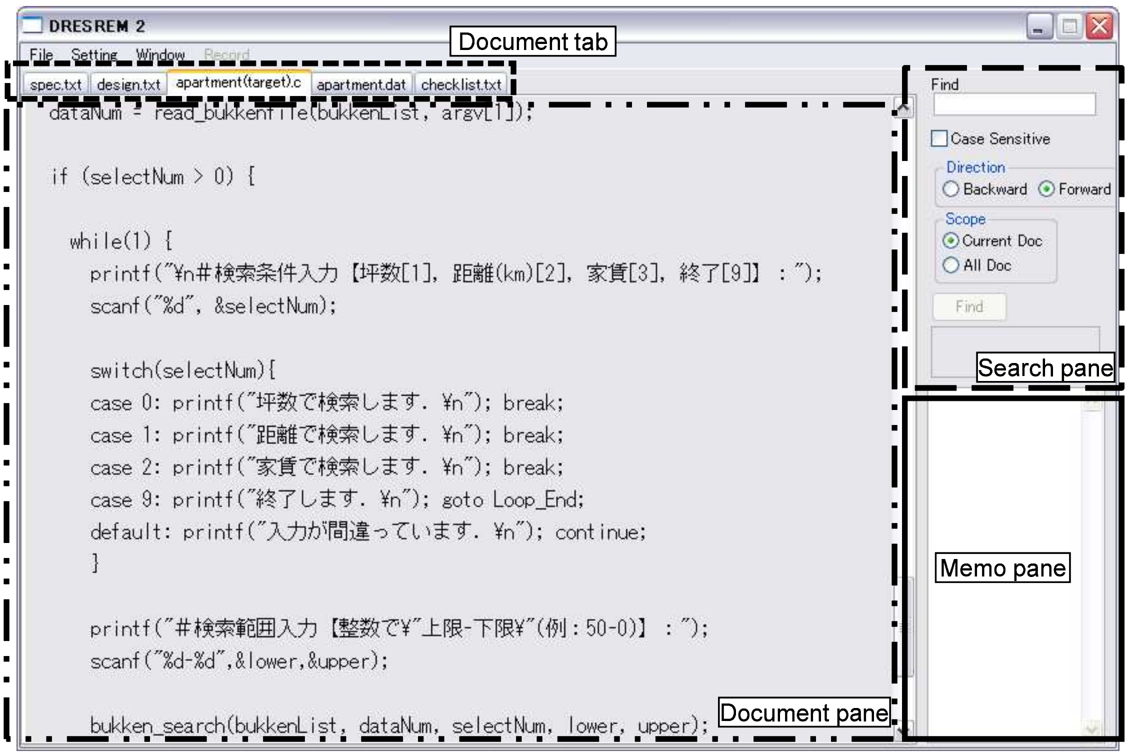
4.1. Outline

We experimentally evaluated the usefulness of the proposed system. In the experiment, subjects were asked to find defects in a design document by reading all given documents. In the review, four documents — requirements specification, design document, data file, and a checklist for design document review — were used. The original requirements specification and design document contained no defect. We injected nine defects to the design document. The review was finished when a subject (reviewer) concluded that the design document had no more defects.

Subjects were eleven graduate students and one faculty member of Nara Institute of Science and Technology. Their average programming experience was 7.6 years. Two of them had software development experience in industry.

4.2. 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 and data file. This system reads a data file in which a set of rental houses is listed. A system user inputs a condition about rental houses (e.g. distance from the 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.



```
dataNum = read_bukkenfile(bukkenList, argv[1]);

if (selectNum > 0) {

  while(1) {
    printf("¥n#検索条件入力【坪数[1], 距離(km)[2], 家賃[3], 終了[9]】 : ");
    scanf("%d", &selectNum);

    switch(selectNum){
    case 0: printf("坪数で検索します. ¥n"); break;
    case 1: printf("距離で検索します. ¥n"); break;
    case 2: printf("家賃で検索します. ¥n"); break;
    case 9: printf("終了します. ¥n"); goto Loop_End;
    default: printf("入力が間違っています. ¥n"); continue;
    }

    printf("#検索範囲入力【整数で¥"上限-下限¥"(例:50-0)】 : ");
    scanf("%d-%d",&lower,&upper);

    bukken_search(bukkenList, dataNum, selectNum, lower, upper);
```

Figure 4. Screenshot of Document Viewer.

- **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.
- **Data File:** This file is read by the system when the system starts. The file consists of a list of rental houses.
- **Checklist:** This is a generic checklist for a design document review, written based on existing literature [7, 8].

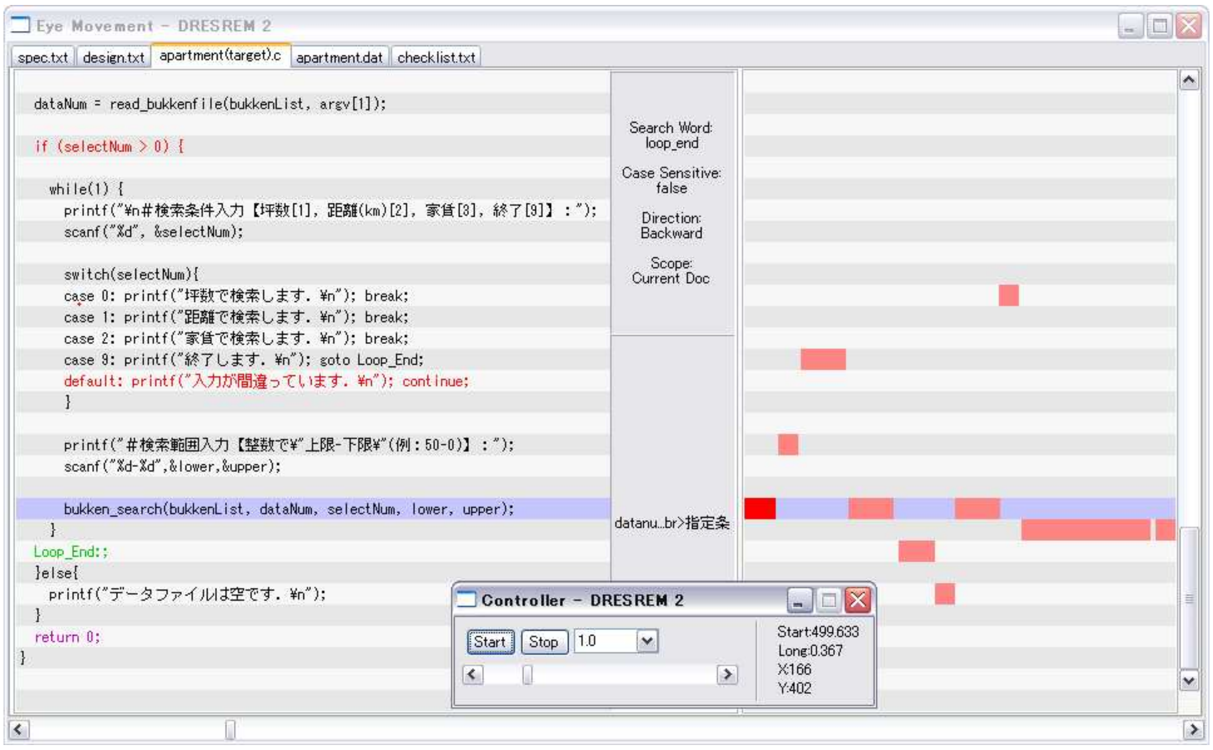
We injected nine defects in total (three defects for each of three defect types) into the design document in advance. Defect types are described as follows.

- **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 have not been described in the requirements specification. This defect can be also considered as insufficient description of the requirements specification.

4.3. Result

Twelve subjects' data were collected in the experiment. One of them was removed from the analysis because of insufficient data accuracy of eye movements. The average review time was 25 minutes. From the interview of reviewers conducted after the experiment, we confirmed that the motivation of subjects to find defects was kept high during the experiment. All subjects found at least three bugs (the average was 5.45).

Using the replay function of DRESREM 2 extensively, we investigated the eye movements of the individual subjects. As a result, we found that every reviewer switched documents frequently in their review. Figure 6 depicts an example of reviewers' eye movements in the experiment. This graph describes time series of eye movements on each line of documents, the horizontal axis shows fixation ID (transitions of fixa-



```
  dataNum = read_bukkenfile(bukkenList, argv[1]);

  if (selectNum > 0) {

    while(1) {
      printf("¥n#検索条件入力【坪数[1], 距離(km)[2], 家賃[3], 終了[9]】:");
      scanf("%d", &selectNum);

      switch(selectNum){
      case 0: printf("坪数で検索します. ¥n"); break;
      case 1: printf("距離で検索します. ¥n"); break;
      case 2: printf("家賃で検索します. ¥n"); break;
      case 9: printf("終了します. ¥n"); goto Loop_End;
      default: printf("入力が間違っています. ¥n"); continue;
      }

      printf("#検索範囲入力【整数で¥"上限-下限¥"(例:50-0)】:");
      scanf("%d-%d",&lower,&upper);

      bukken_search(bukkenList, dataNum, selectNum, lower, upper);
    }
  Loop_End:;
  }else{
    printf("データファイルは空です. ¥n");
  }
  return 0;
}
```

Search Word: loop_end
Case Sensitive: false
Direction: Backward
Scope: Current Doc

datanu...br>指定条

Start:499.633
Long:0.367
X:166
Y:402

Figure 5. Example of eye movement visualization using DRESREM 2.

tion among lines) and the vertical axis shows the line number of documents. In the figure, eye transitions between requirements specification and design document were observed. In the experiment, reviewers switched documents every 19.9 seconds on average.

From a quantitative analysis, we found that reviewers spent different fixation time on each documents. Table 1 shows the percentage of fixation time for each document. This result indicates that reviewers spent a fair amount of time on the design document. They spent most time on the requirements specification and design document (96.5% on average.) However, there were quite a few differences in their reviews. For example, $SubjectA$ spent only 19.8% on the requirements specification and spent most time on the design document (72.9%.) On the other hand, $SubjectB$ concentrated more on the requirements specification (40.9%) and less time on the design document (54.8%.) The result of a statistical analysis revealed a significant correlation between the defect detection ratio of "omission of requirements" and review time on requirements specification ($r = 0.593$, $p - value = 0.054$.) This suggests that to find the omission of requirements in the design document, we need to read the requirements specification. It can be said that reading the design document only yields less understanding of the system requirements.

Table 1. Fixation ratio for each document.

	Average	Minimum	Maximum
Requirements	28.5%	19.8%	40.9%
Design	68.0%	54.8%	73.3%
Data file	0.6%	0.0%	1.2%
Checklist	2.2%	0.0%	3.6%
Other	0.7%	0.0%	3.8%

5. Conclusion

In this paper, we proposed a multi-document review evaluation system, DRESREM 2. The proposed system records reviewers' eye movements and mouse/keyboard operations and visualizes them to analyze the relationship between review performance and reading procedure. The system also provides features to play back the eye movements and the operations for a qualitative analysis of software review activities.

We experimentally evaluated the usefulness of the proposed system. In the experiment, a design docu-

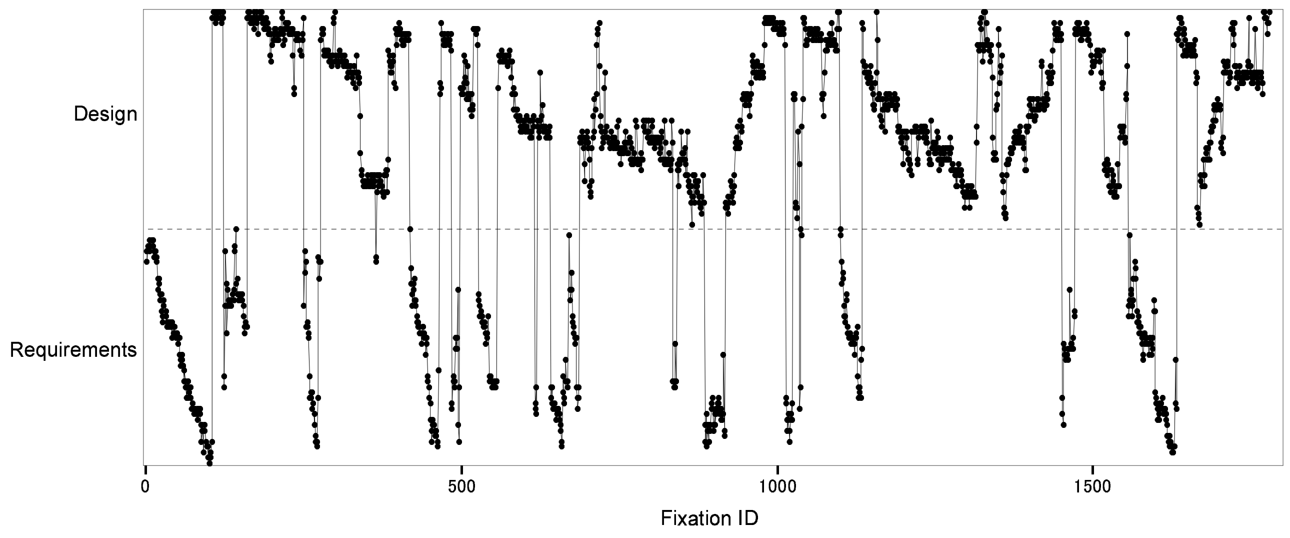


Figure 6. An example of eye movements.

ment review using multiple documents (requirements specification, design document and others) was performed. As a result, the system contributed to revealing the reading process that affected the review performance. The result of a statistical analysis revealed a significant correlation between the defect detection ratio of "omission of requirements" and review time on requirements specification. This suggests that to find the omission of requirements in the design document, we need to read the requirements specification.

The major limitation of our experiment is that the sample size was small (twelve subjects). Also, we used just one software system to be reviewed. In the future, we will increase the sample size with different software systems. More detailed analysis of eye movements such as function-wise (block-wise) analysis and time series analysis are also important future work.

6. Acknowledgment

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

References

[1] V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, and M. V. Zelkowitz. The empirical investigation of perspective-based reading. *An International Journal of Empirical Software Engineering*, 1(2):133–163, 1996.

[2] B. W. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

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

[4] M. Halling, S. Biffl, T. Grechenig, and M. Köhle. Using reading techniques to focus inspection performance. In *Proceedings of 27th Euromicro Workshop Software Process and Product Improvement*, pages 248–257, 2001.

[5] 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.

[6] O. Laitenberger and J. DeBaud. An encompassing life cycle centric survey of software inspection. *Journal of Systems and Software*, 50(1):5–31, 2000.

[7] 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.

[8] 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.

[9] H. Uwano, M. Nakamura, A. Monden, and K. Matsumoto. Exploiting eye movements for evaluating reviewer's performance in software review. *IEICE Transactions on Fundamentals*, E90-A(10):317–328, October 2007.

[10] K. Wiegers. *Peer Reviews in Software - A Practical Guide*. Addison-Wesley, 2002.