Visual attention and representation switching in Java program debugging: A study using eye movement tracking

Roman Bednarik and Markku Tukiainen
Department of Computer Science
University of Joensuu
{roman.bednarik, markku.tukiainen}@cs.joensuu.fi

Keywords: POP-V.A. Attention Investment, POP-V.B. Eye Tracking, POP-II-B. Debugging

Abstract

This paper describes a study of Java program debugging using a multiple window software debugging environment (SDE). In this study we have replicated an earlier study by Romero et al. (2002a, 2002b, 2003), but with the difference of using both the Restricted Focus Viewer and the eye tracking equipment to track the visual attention of the subjects. The study involved ten subjects debugging short Java programs using the SDE. The SDE included three different representations of the Java programs, those of the program source code, a visualization of the program, and its output concurrently in three separate panels in SDE. We used the Restricted Focus Viewer (RFV) and a remote eye tracker to collect the visual attention of the subjects. A with-in subject design, similar to Romero et al., employing both RFV/no-RVF task conditions was used.

The overall results of the time distributions over three different representations of the programs agree with the study of Romero et al. But the results of visual attention switching raise some questions to be considered in later studies.

Introduction

Modern tools for software development usually include debugging environments, which typically consist of multiple mutually linked representations of the program under development. Programmers use these different representations to build up a mental model of a program. While discovering the errors in the source code, programmers have to perform a large number of mental tasks and at the same time have to coordinate different representations offered by the debugger. Typically, the program code, the debugger output, and some other kinds of representation of the program are displayed simultaneously. One of the aims of the research related to debugging is to understand, how programmers coordinate multiple representations, and what debugging strategies they exhibit.

Previous work on visual attention and representation switching in debugging

One way to investigate the coordination of multiple representations employed in debugging tasks is to use a tool to track the visual attention. For tracking visual attention eye tracking equipment has become more common in recent years. The Restricted Focus Viewer (RFV) has been developed as an alternative to eye tracking (Blackwell, 2000). One of the main advantages of RFV is that it enables automated collection of subjects' focus of attention. The RFV blurs the stimuli image and displays it on a computer screen, therefore allowing the subject to see only a limited focused region at a time. In order to get another part of the stimuli focused, subject has to move the focused region using the computer mouse. The RFV then records the moves over the stimuli, which are stored for later analysis. The tool collects the data for both mouse and keyboard events together with the timestamps, focused region's index, total durations of debugging sessions, and other events.

A special software debugging environment (SDE) on top of the RFV has been developed for the purpose of tracking the switches between the representations (Romero et al. 2002a, 2002b). For that purpose, the RFV has been modified so that the subject has to click a mouse button to set the place of the focused region. The SDE also remembers the positions of the focused regions for each of the displayed panels. When the subjects return their attention pointer to the panel with a representation,

the SDE automatically focuses the last region of interest. The SDE shows several representations of a program in the adjacent windows, namely the source code, the output, and the visualization. While working under the restricted focus condition, only a small region of the SDE window is shown in focus. Figure 1 shows a typical screen shot of the sample interaction with SDE under the restricted condition. The focused region following the mouse pointer is shown over the visualization.

Although the RFV-based debugging environment allows for easy tracking of attention, it also introduces several constrains. First, the environment is limited to the static pre-computed stimuli images, thus it does not allow us to track the visual attention in unlimited dynamic environments. Second, it requires subjects to move frequently the computer mouse to see the areas of interest focused. Finally, the temporal resolution of the RFV-based measurement is limited to the movement by-movement level of detail. To our best knowledge, eye tracking has not been applied to research related to debugging strategies of programmers.

Eye movements

While searching for a visual object people make rapid eye movements, called saccades, shifting the point of gaze. It is supposed that no information is collected during the saccade since vision is

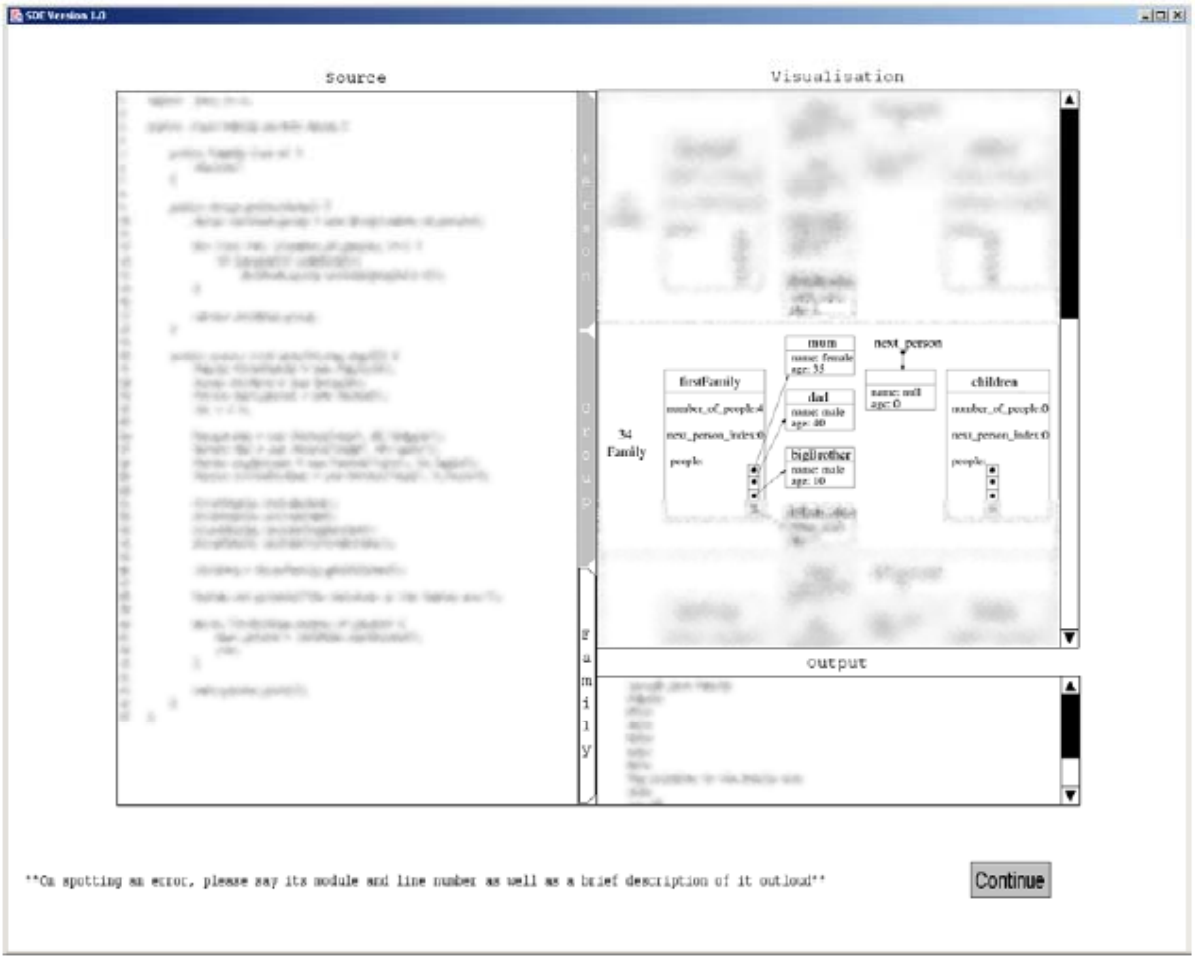

Figure 1 Software debugging environment used in the experiment

suppressed during the shifts. This phenomenon is known as saccadic suppression. Once the object is positioned on the high acuity fovea, the information from the stimuli is extracted during the fixation. Jacob and Karn (2003) define a fixation as "a relatively stable eye-in-head position within some

threshold of dispersion (typically ~2°) over some minimum duration (typically 100-200ms), and with a velocity below some threshold (typically 15-100 degrees per second)."

Eye movements and visual attention are coupled in the sense that the shifts in the foveal gaze direction are linked with the voluntary intention to change the focus of visual attention. Although the link is vivid and confirmed across the research in the recent decades, we acknowledge that it might not be always so, since we may sometimes attend to an object without moving our eyes.

While comparing the gaze and computer mouse as the input modalities, gaze is considered to be a more natural and intuitive way of input, since we move our eyes mostly unconsciously. Because the eye movements are rapid, the gaze is also significantly faster than the mouse (Sibert&Jacob, 2000); however, due to the limitation of human fovea it is less accurate, with the effective range being 1 degree of the visual field (Zhai et al., 1999).

For a comprehensive review of eye movement research see e.g. (Duchowski, 2003; Rayner, 1998; Jacob, 1995), for the overview of the research related to the relationship between eye movements and attention, see e.g. (Godijn & Theeuwes, 2003).

Eye tracking

Up to date, eye tracking techniques have been involved in many different kinds of studies. Their wider application, however, has been limited by many factors. The earlier eye trackers were expensive, required frequent recalibration, were awkward for subjects due to having mechanical contact with their eyes, and the data analysis was time consuming and tedious. Recently, the price of the technology dropped and enabled more research laboratories to acquire the eye tracking system. The ability to track subjects' eyes has also been significantly improved in comparison with the systems from the recent decades. Modern, commercially available eye trackers are usually based on video images of the eye (Jacob&Karn, 2003; Duchowski, 2003). A ray of light, usually from an invisible infrared source of illumination, is shone at the subjects' eye, and multiple reflections from the eye are captured together with the geometrical properties of the eye. Multiple reflections are used to dissociate minor head movements from the rotations of the eyes. Subjects are thus allowed to move their head freely, only with minor spatial restrictions. The table-mounted, remote eye trackers, do not make any contact with subjects, some even do not introduce any visible interference with the working environment. While using a remote eye tracker the subjects' head movements are restricted to approximate spatial cube of edge about 25cm. The head-mounted version of eye tracking optics, however, poses a need for a headband firmly mounted on the subject's head. The disadvantage is counterbalanced by the fact that the head movements are not limited in space or speed.

The spatial accuracy of the modern eye trackers ranges around the half of visual degree, and the temporal resolution is often above 50 Hz, which allow tracking the eyes with the high spatial and temporal precision. In their recent study, Nevalainen and Sajaniemi (2004) confirm that eye tracking can be used in the research of psychology of programming, with a relatively high accuracy of point of gaze recording. Most of the current eye-tracking systems include a software package, which makes the collection and analysis of the data easy and fast.

Aims of the study

The goal of the current study was to validate the previous results in representation switching obtained using the RFV against results measured by the remote-mounted eye tracker. We especially aimed to set up an environment and stimuli programs as close as to those reported in (Romero et al. 2002a, 2002b; Romero et al 2003), but in addition, we used eye-movement tracking. Therefore, for the same settings we could compare the focus of attention measured by the RFV to that measured by the eye tracker. The RFV reports the focus of attention in form of the index of stimuli image and the position of mouse, the eye tracker reports the position of the gaze and corresponding area of interest index. As a part of the study, the investigation also aimed at validation of visual attention focus as measured by RFV and by eye tracking

Method

The experimental setting

For eye-movement tracking we used the remote Tobii ET-1750 eye tracker with the sampling-rate set to 30Hz. Participants were seated comfortably in an ordinary office chair, facing the seventeen inch TFT computer screen from a distance of about one meter.

The software debugging environment (SDE) originally utilized e.g. in (Romero et al. 2002a, 2002b; Romero et al 2003) was used in the experiment. In these studies and also in the current experiment, the representations, i.e. the program code, visualization, and output of the program, were precomputed and static. In the current study, all of the debugging sessions employed only graphical functional visualization.

The eye-tracking data and the audio protocol were collected throughout the whole experiment. Thus, for the same instant the three sources of data could be analyzed: the focus of attention measured by the RFV, the focus of attention measured by the eye tracker, and the audio protocol. The eye tracking protocol was processed by the automatic fixation detection algorithm, with the thresholds set as follows: the minimal fixation duration was 10 ms and the fixation radius limited to 50 pixels. The coordinates of the areas of interest corresponded to the corners of the windows presented by SDE.

Participants and procedure

The subjects in the experiment were teachers and students from the Department of Computer Science at the University of Joensuu. All subjects were volunteers and they received a lunch ticket for their participation. In the study were the total of ten participants, one female and nine males, all with perfect or corrected-to-perfect vision. None of the subjects had previously participated in eye tracking study. The level of programming experience varied, ranging from undergraduate students who had just passed an introductory course in Java, through postgraduate students with substantial programming experience, to experienced programmers with a long history of using Java. Some of the participants were Java teachers or even worked as professional programmers. Table 1 gives the background information about the participants' programming experience.

Participant	1	2	3	4	5	6	7	8	9	10
Java experience	24	4	5	6	3	10	60	12	5	6
Programming exp.	36	120	108	36	24	96	60	120	84	36
Professional exp.	No	No	No	Yes	No	Yes	Yes	Yes	Yes	No

Table 1. Programming experience of participants in months

Prior to the debugging sessions, each participant had to pass an automatic eye-tracking calibration routine, which consisted of tracking the eyes as they followed sixteen shrinking points across the computer screen. This procedure ensured the accuracy of the eye tracking and accommodation of the eye tracker's parameters to the personal characteristics. After the calibration, participants read the description of the experiment, specifying the environment used, its control and the setting of the study. Then they performed three debugging sessions. The first one was performed under the restricted focus condition and served as a warm-up, however, the participants were not aware of that fact. After the warm-up, the two main debugging sessions were performed. One of these was under the restricted focus condition, the other without the restriction. The order of the target programs and restricting conditions were counterbalanced.

Each debugging session consisted of two phases. In the first, the participants were presented with the short specification of the target program, which included a description of the problem, the approach to the solution, and two sample outputs of the interaction with the program. One output presented the

actual, erroneous behaviour of the program, and the other one presented the desired, correct behaviour of the program.

In the second phase, participants were given ten minutes to debug each program; a sound signalled two minutes till the end of each session. Participants were instructed to find as many errors as possible and to report them aloud by stating the error itself, the class and line number in which it occurred, and how the error could be corrected. They were also encouraged, but not forced, to think aloud while debugging.

The target programs used in this study were identical to those used in (Romero et al. 2002b). The warm-up program inspected whether a point was inside a rectangle. The first program ('Family' program) prints out the names of the children of a sample family and the second program ('Till' program) counts the cash in a cash-register till, giving subtotals for the different denominations of coins. In their study (2002b), Romero et al. had two versions of the target programs; the main difference between these versions was that the second one was a more sophisticated version of the first one. We used the less sophisticated versions of the programs and the graphical functional representations in visualizations. Altogether with and without the restricted focus imposed by the RFV, there were four different experimental conditions.

The two main experimental programs contained four errors each; the warm-up program was seeded with two errors. Following the classification of the errors established in (Romero et al. 2002a, 2002b; Romero et al 2003) the errors in the target programs can be classified as functional, control-flow, and data-structure. There was no syntactical error in the programs (all programs could be compiled) and participants were notified of this.

In order to collect the additional subjective experiences, each participant was debriefed after the debugging session.

Results

Debugging performance

The results related to the debugging performance are presented in Table 2, which summarizes the number of errors spotted for the two main sessions. There were four errors in each of the target source codes. The results show that the most successful participants were number 9, who discovered all errors, then number 6 who spotted seven errors, and number 2, who located six errors. Participants 3, 8, 10 and 4, 7 achieved average scores of five and four, respectively. Participant 1 located three bugs, and participant 5 discovered one error.

The total number of errors found for all participants in restricted view condition was 22 and in unrestricted view condition was 26 out of a maximum of 40 errors. There seems to be no difference in distribution of errors spotted between the conditions of restricted and unrestricted view. In general, more experienced programmers found more errors (participants 2, 3, 6, 8 and 9, general programming experience of 6 years or more), U-value $U(5,5) = 24$, $p<0.05$ for the effect of experience on errors found.

	Participant	1	2	3	4	5	6	7	8	9	10
Errors found	RFV on	2	3	3	3	0	3	2	3	4	3
	RFV off	1	3	2	1	1	4	2	2	4	2
	Total	3	6	5	4	1	7	4	5	8	5

Table 2. Number of errors found (each condition contained 4 errors)

Eye tracking results and debugging behaviour

The global experimental results for debugging behaviour are shown in Tables 3 and 4. Table 3 presents the percentage of time that participants spent looking at each representation obtained by the

eye tracker. The results for time distribution between the windows in the SDE agree with the results obtained by Romero et al. (2002a, 2002b, 2003); most of the time is spent on the code window, then on the visualisation window, and least time is spent on the output window. In our experiment, there was nearly significant effect ($t(9)=1.895$, $p<0.05$) of proportional time spent on the visualization between the restricted view condition (RFV on) and the unrestricted view condition (RFV off). There were no significant effects for proportional time spent on the code and the output for the conditions.

	Participant	1	2	3	4	5	6	7	8	9	10
RFV on	Code	92.7	86.8	87.2	91.4	71.8	89.8	75.1	92.7	93.1	73.0
	Visualization	4.1	4.9	9.8	4.7	27.6	4.6	22.8	6.4	5.8	16.0
	Output	3.2	8.3	3.0	3.9	0.6	5.6	2.1	0.9	1.1	11.0
RFV off	Code	93.6	91.5	79.2	73.6	62.9	88.3	81.0	85.9	84.4	75.1
	Visualization	5.7	2.8	17.1	17.5	33.0	7.0	15.3	9.6	9.0	22.5
	Output	0.7	5.7	3.7	8.9	4.1	4.7	3.7	4.5	5.6	2.4

Table 3. Percentage of time spend in each representation measured by the eye tracker

Table 4 presents the average number of switches per minute measured by both the RFV tool and the eye tracker. By the term "switch" we mean the change of focus between the areas of interest, here between the code window, visualization window and output. The results obtained from the RFV tool protocol in restricted view condition (RFV on) are slightly higher than those in previous works (total average of 2.56 switches per minute for nine subjects, Romero et al. (2002a) total average of 1.73 switches per minute for five subjects), but in line with results from Romero et al (2002a). The results under the restricted condition obtained from the eye tracker, however, differ significantly from those obtained by the RFV ($t(9) = 3.49$, $p<0.01$). The participants switch their visual attention more often than they change the focused area with the mouse pointer in the RFV. This is also evident from looking into the eye movement protocols. From the eye movement protocols, we can see that the participants look at the blurred areas quite often and switch their gaze point even between the totally blurred windows in SDE. There is also a significant difference in switches per minute between the RFV on and RFV off conditions measured by the eye tracker ($t(14)=2.32$, $p<0.05$).

	Participant	1	2	3	4	5	6	7	8	9	10
RFV on	RFV	1.2	1.69	3.01	7.2	2.0	4.11	3.91	2.0	1.1	4.01
	EyeTracker	5.31	6.55	8.81	4.4	5.61	6.34	6.71	4.01	3.01	8.03
RFV off	EyeTracker	4.11	6.08	4.92	7.81	6.51	13.09	9.35	10.74	12.14	10.13

Table 4 Average numbers of switches per minute measured by RFV and eye tracking

The debugging behaviour measured by the eye tracker of all the participants in terms of switches per minute between each of the representations is shown in Table 5. For both of the conditions there are six types of switches and a corresponding number of switches per minute: a switch from code to visualization (the upper number) and a switch from visualization to code (the number at the bottom), a switch from code to output and an inverse switch from output to the code, and a switch from visualization to output and from output to visualization.

The effect of the restricted view condition on switching between the code and visualization representations and between the visualization and output was not significant. The effect of the restricted view on the total number of switches per minute from the code to output and from output to code was significant ($t(9) = 2.39$, $p<0.05$).

Participant		1	2	3	4	5	6	7	8	9	10
RFV on	code ⇔ visual	2.0	1.58	2.9	1.1	2.5	2.23	1.9	1.2	1.1	2.51
		1.9	1.48	3.11	0.9	2.5	1.71	2.0	1.4	0.9	2.51
	code ⇔ output	0.3	1.37	1.2	0.7	0	0.51	0.4	0.5	0.2	1.2
		0.4	1.48	1.1	1.0	0.1	0.86	0.3	0.4	0.3	1.2
	visual ⇔ output	0.4	0.42	0.2	0.5	0.3	0.69	1.0	0.2	0.3	0.3
		0.3	0.21	0.3	0.2	0.2	0.34	1.1	0.3	0.2	0.3
RFV off	code ⇔ visual	1.5	0.9	1.01	2.2	1.6	3.83	2.11	3.51	2.91	4.41
		1.4	1.01	0.9	1.8	1.7	4.03	2.21	3.61	2.61	4.31
	code ⇔ output	0.2	1.58	1.11	1.1	0	2.01	1.51	1.1	1.4	0.2
		0.4	1.58	1.21	1.5	0	1.81	1.41	1.1	1.81	0.3
	visual ⇔ output	0.4	0.45	0.4	0.8	1.6	0.6	1.01	0.7	1.91	0.5
		0.2	0.56	0.3	0.4	1.6	0.81	1.11	0.7	1.51	0.4

Table 5 Number of switches per minute for each switch type. Upper number corresponds with the rate from code or visualization; bottom corresponds with the rate from visualization or output.

Figure 2 and Figure 3 summarize the data from Table 4 and Table 5 in the form of graphs. The debugging behavior of participants 10 (less experienced) and 9 (most successful), respectively, is shown. The diagrams on the left describe the behavior under the restricted condition; the diagrams on the right describe the behavior under the unrestricted setting. The direction of arrows follows the types of switches introduced above.

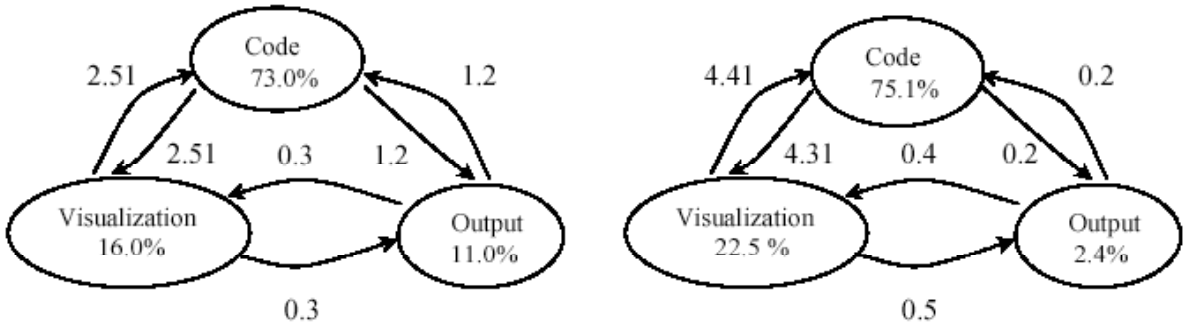

Figure 2 Number of switches per minute and percentage of time spent on each of the representation for participant 10

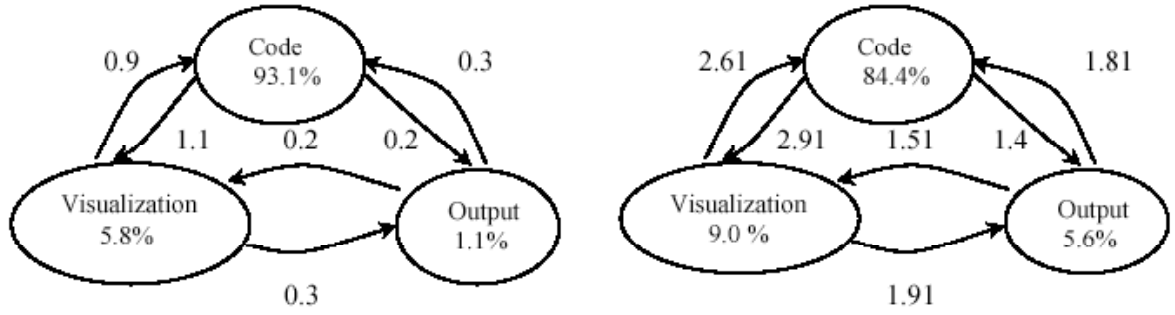

Figure 3 Number of switches per minute and percentage of time spent on each of the representation for participant 9.

From Figures 2 and 3 it can be observed that the more experienced participant (9) exhibited more balanced behavior in terms of switches per minute between all the representations than the less experienced participant (10). These two participants were selected also as the representatives of their experience groups.

Subjective experiences reported by participants

Each subject was debriefed after the experiment. The majority of the experienced participants have stated that the restrictive condition imposed by the RFV does influence their ability to extract all the information needed. For instance, the most successful participant (9) hypothesized that he "was not good with the blur" although he spotted all the errors under both conditions. On the other hand, the majority of novices expressed the feeling that the restricted focus allows them to concentrate more on the particular piece of the code, while they are not influenced by the surrounding lines. Novices often expressed the opinion that using the RFV's restricting condition might bring efficiency into their debugging performance.

Almost all of the participants asked during the warm-up session whether it is possible to modify the program and to correct the error they found. Some of the participants asked whether they could take the notes using pen and paper.

Discussion

This experiment was done in order to verify the results obtained from earlier studies (Romero et al. 2002a, 2002b; Romero et al 2003) using two different tools. The Restricted Focus Viewer is a tool which relates the focus of visual attention to the location of a fully focused area within the blurred stimuli images. The eye tracker measures the location of gaze, which is thought to be tightly connected with the locus of visual attention.

Ten volunteers participated in the experiment, half of them with the experience outreaching six years of active programming. The four experimental conditions used in the study consisted of two target Java programs and two settings of RFV (restricted focus on and off). Each of the programs was seeded with four errors. Participants debugged two different programs, being allowed to debug each program for ten minutes.

The results of debugging performance agree with the findings from earlier studies, the error-finding performance is related to programmers' experience with programming. The more experienced programmers were more successful than the less experienced programmers in finding errors in the programs. The general programming experience determined the expertise of the participants better than their experience with Java programming.

Similarly to earlier studies (Romero et al. 2002a, 2002b; Romero et al 2003), the results for visual attention show that participants spent most of the time looking at the code window. This is not surprising, because the code contains most of the information and is clearly the primary representation of the program. However, our results differed from the earlier results of Romero et al. concerning the distributions of the times between the windows of the SDE. Our participants spent more time on the visualisation window (on average 12.5 % of the total time compared to an average of 8 % of the total time for the data structure visualisation, estimated from Romero et al. 2002a and 2002b). This could be explained by at least two causes. The first is that our results are for a small number of subjects (only 10 subjects compared to 49 subjects in Romero et al. 2002b). We are going to run the experiment with more subjects to find out if this is the case. The second explanation is that the RF tool itself might affect the behaviour of the subjects. In our study we were able to measure the time distributions using the actual eye movements from the participants, the results showed that our participants spent significantly more time on the visualisation window without using the RFV, than using RFV.

The experimental results for switching behaviour agree with the findings of the earlier studies in principle, that the most common type of switch is between the code window and the visualisation window, the second most common between the code window and the output window, and the least common between the visualisation window and the output window. However, when we study the switches at the level of the visual attention-tracking tool used, we see first a significant difference in the average number of switches per minute and second, a significant increase in the number of switches between the visualisation window and the output window in the unrestricted view condition (RFV off) compared to the restricted view condition (RFV on).

The first difference is due to fact that the RFV tool and an eye tracker measure the visual attention, and thus the attention switching, differently. This is illustrated in Figure 4. The blurred image of the visualisation of the data structure used, or the output of the program, could serve as a memory aid for the programmer to recognize the specific working of the program once he/she comprehends it in the unblurred mode. This could explain why the programmer does not have to shift the unblurred spot to the window but can just look at the blurred image.

The second difference could have at least two possible explanations. The first one is that the switching has a lower cognitive cost in unblurred mode. The fact that the average number of switches per minute increases in the unrestricted view condition compared to restricted view condition could be seen as support for this. The second explanation is that the RFV tool has an effect on debugging strategy. For some reason, the unrestricted view allures the programmer to test an error hypothesis between the output representation and the visualisation representation. However, the results do not show better debugging performance for the unrestricted view condition than for the restricted view condition.

Several methodological reasons can be found for the differences between the RFV-based results and eye tracking results. The temporal resolution of RFV-based measurement is limited to the movements of the computer mouse. In our experiment, during the ten minutes of debugging a program the RFV collected about 400 data points, while the number of fixations was around 7000. Thus, the eyetracking equipment provides a finer level of temporal details.

Furthermore, the spatial resolution of fixated areas as measured by RFV can be discussed. As seen from Figure 1 and Figure 4, the size of the focused region ranges far beyond the two degrees of visual angle as limited by the fovea. Therefore, it allows a subject to move his/her eyes within the focused region of RFV without moving the mouse. These changes in focus location are not recorded by RFV.

It has often been the case that the participant working under the restricted focus view has set the focused region onto the visualization and then returns the attention to the code or output without moving the mouse. Figure 4 shows the screenshot of such a situation: the trace of the gaze corresponds roughly to an interval of one second and the diameter of the fixation matches with about 50ms and increases. This behavior consisting of two or more switches between representations is not registered by the RFV.

We also observed that subjects could be divided into two groups when use of the mouse during debugging is considered. Some participants used the mouse as a pointer while reading the source code, while the others had some problems coordinating the hand (mouse-controlled focused spot) and gaze as required by the restricting view. In these subjects the gaze replay reveals that the destinations of the mouse cursor and the gaze are almost always exclusive. This is especially true for large eye movements, when the saccade precedes the movement of the hand. Several observations from the eye movement protocol recorded during the debugging show that mouse position can also precede the gaze. We maintain that the requirement of coordinating hand and gaze during debugging might be one of the reasons why the RFV-on condition affected switching behavior.

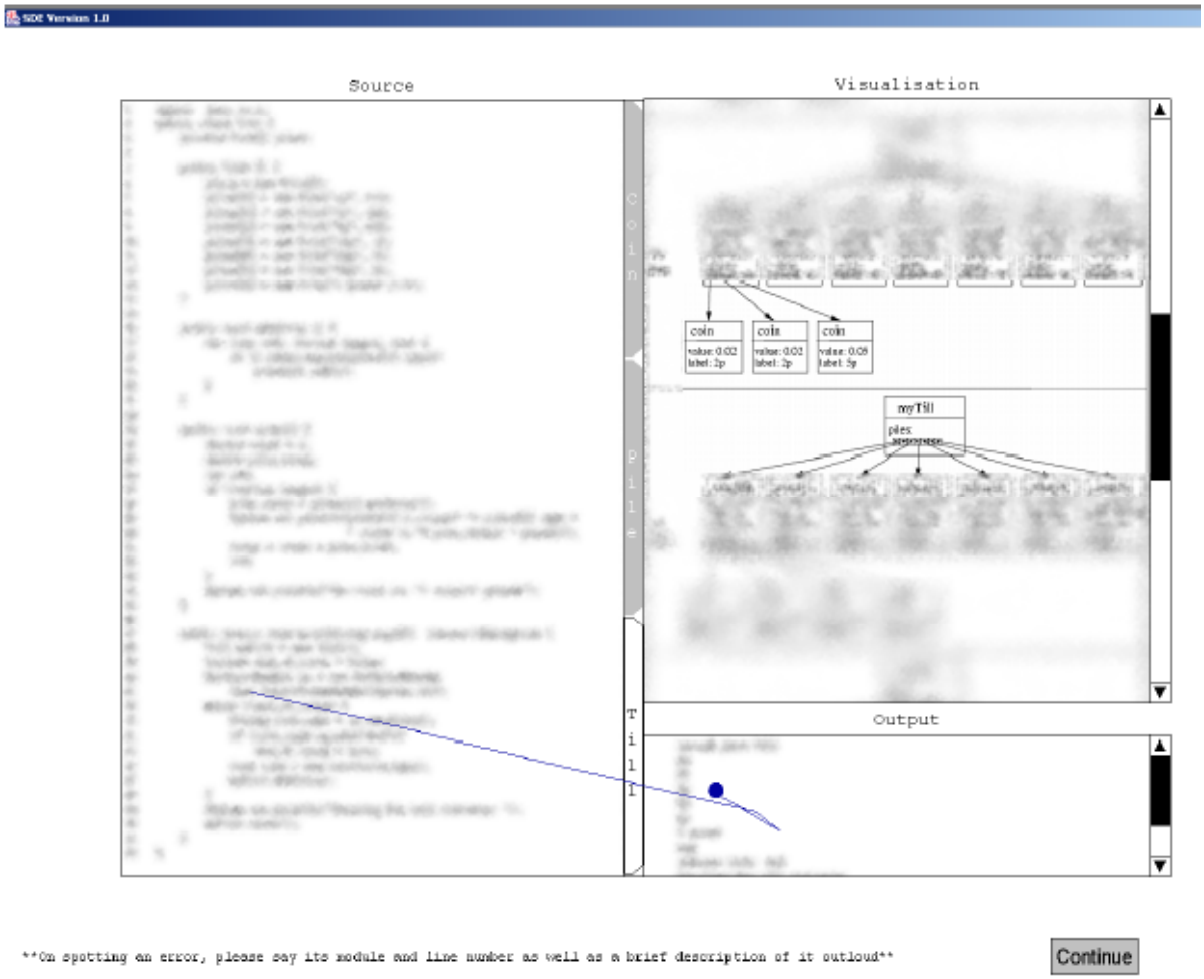

Figure 4 The eye gaze trace and a fixation superimposed over the SDE with the restricting focus on

Acknowledgments

The authors would like to thank Jorma Sajaniemi for his deep insight into programmer studies and all kinds of help with this experiment.

References

Blackwell, A. F., Jansen, A.R. and Marriott, K. (2000): Restricted Focus Viewer: A tool for tracking visual attention. In M. Anderson, P. Cheng & V. Haarslev (Eds.), *Theory and Applications of Diagrams. Lecture Notes in Artificial Intelligence* 1889, pp. 162-177. Berlin, Springer Verlag.

Duchowski, A. T. (2003): *Eye Tracking Methodology: Theory & Practice*. Springer-Verlag, London, UK.

Godijn, R. & Theeuwes, J.(2003): The relationship between exogenous and endogenous saccades and attention. In J. Hyönä, R. Radach, & H. Deubel (Eds), *The Mind's Eyes: Cognitive and Applied Aspects of Eye Movements*. Elsevier Science.

Jacob R.J.K. (1995): Eye tracking in advanced interface design. In W. Barfield and T.A. Furness (Eds.), *Virtual Environments and Advanced Interface Design*, pp. 258-288, Oxford University Press, New York, (1995).

Jacob, R. J. K., & Karn, K. S. (2003): Eye tracking in human-computer interaction and usability research: ready to deliver the promises (Section commentary). In J. Hyona, R. Radach, & H. Deubel (Eds.), *The Mind's Eyes: Cognitive and Applied Aspects of Eye Movements*. Elsevier Science.

Nevalainen S., Sajaniemi J. (2004): Comparison of three eye tracking devices in psychology of programming research. Accepted to the *16th Annual Psychology of Programming Interest Group Workshop (PPIG'04*).

Rayner, K.(1998): Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124, 3, pp. 372-422.

Romero, P., Cox, R., du Boulay, B. and Lutz, R. (2002a): Visual attention and representation switching during Java program debugging: a study using the Restricted Focus Viewer. *Diagrammatic Representation and Inference : Second International Conference, Diagrams 2002* Callaway Gardens, GA, USA, April 18-20, 2002. *Lecture Notes in Artificial Intelligence*, 2317, pp. 221-235. Berlin, Springer Verlag.

Romero, P., Lutz, R., Cox, R. & du Boulay, B.(2002b): Co-ordination of multiple external representations during Java program debugging. *Empirical Studies of Programmers symposium of the IEEE Human Centric Computing Languages and Environments Symposia*, Arlington, VA.

Romero, P., du Boulay, B., Cox, R. & Lutz, R.(2003): Java debugging strategies in multirepresentational environments. *15th Annual Workshop of the Psychology of Programming Interest Group (PPIG*), Keele University, UK.

Sibert L.E. and Jacob R.J.K. (2000): Evaluation of eye gaze interaction. *Proceedings of ACM CHI 2000 Human Factors in Computing Systems Conference*, pp. 281-288, Addison-Wesley/ACM Press.

Zhai, S., C. Morimoto & S. Ihde (1999): Manual and gaze input cascaded (MAGIC) pointing. *Proceedings of ACM CHI'99 Conference on Human Factors in Computing Systems*, pp. 246-253.