The effects of graphical and textual visualisations in multi-representational debugging environments

Pablo Romero, Benedict du Boulay, Rudi Lutz, and Richard Cox
Human Centred Technology Group,
School of Cognitive & Computing Sciences
University of Sussex, Falmer, Brighton,
East Sussex, BN1 9QH, UK.
juanr@cogs.susx.ac.uk

Abstract

The effects of graphical and textual visualisations in a multi-representational debugging environment were investigated in computing students who used a software debugging environment (SDE) that allowed them to view the execution of programs in steps and that provided them with concurrently displayed, adjacent, multiple and linked representations.

The experimental results are in agreement with research in the area that suggests that good debugging performance is associated with a balanced use of the available representations. Additionally, these results raise the issue of whether graphical visualisations promote a more judicious representation use than textual ones for program debugging in multi-representational environments.

1. Introduction

Only a limited number of studies have looked at the issue of representation coordination in multi-representational programming environments and how perceptual properties of the representations employed can affect both the way and the success with which they are used. It has been suggested, for example, that the higher the debugging ability, the less frequent the changes of focus in the sources of information provided for the task [6]; that debugging strategy choice is affected by the type of representations which are available [3] and that good debugging performance is associated with a balanced use of the representations provided by the environment [4].

Research on the use of multiple external representations in other areas has identified a set of functions that representations can play [1]. Representations can, for example, play complementary roles either because they present different information or because they support different cognitive processes. An important factor when dealing with multi-representational systems is their heterogeneity in terms of modality. Here modality is used to mean the representational form used to present or display information, rather than in the psychological sense of a sensory channel. A typical modality distinction is between propositional and diagrammatic representations. Research on this issue has suggested that representations of different modalities can activate different cognitive processes, that textual representations generally require more active search [2] and permit the expression of abstraction or indeterminacy while graphical representations compel the representation of specific information [5].

2. Method

One of the aims of the work reported here was to relate debugging behaviour, especially window switching patterns, to visualisation modality and debugging accuracy. The aspect of the experiment reported here considered one independent, within subjects variable and three dependent variables. The independent variable was visualisation modality (graphical or textual). The three dependent variables were debugging accuracy, accumulated fixation time between the available representations (total time participants spent focusing on each representation) and switching frequency between the available representations (the number of changes of focus between the windows of the debugging environment).

The debugging environment employed (SDE) presents image stimuli in a blurred form and allows visual attention to be tracked as the user moves an unblurred 'foveal' area around the screen. Use of the SDE enabled moment-by-moment representation switching between the available representations to be captured for later analysis.

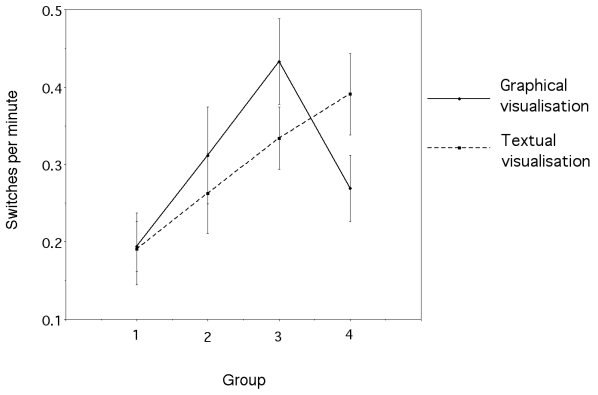

Figure 1. Window switching frequency by visualisation modality

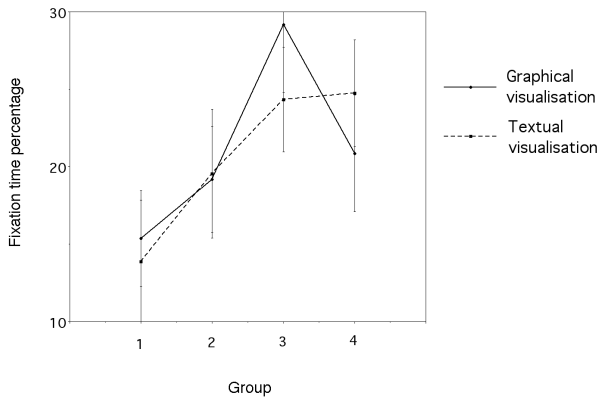

Figure 2. Window fixation for support representations by visualisation modality

The SDE enabled participants to view the execution of a Java program and presented, in addition to the code, its output and two visualisations of its execution.

The experimental participants were forty two computing undergraduate students from the School of Cognitive and Computing Sciences at Sussex University, U.K. All of the participants had taken a three month introductory course in Java, but their programming experience varied from having taken only this course to a few extra months of experience in Java and other programming languages.

Participants debugged four buggy versions of a program of medium size and complexity that simulated the behaviour of a drink dispensing machine. Each of these versions was seeded with a single error. Before starting these debugging sessions, participants spent approximately one hour studying the target program. Also, at the beginning of each debugging session, participants compared samples of both desired and actual program outputs so that they were clear about the effects of the error.

In each debugging session participants were allowed up to ten minutes to find the error. They were instructed to think aloud and to identify the error in the program reporting it verbally by stating its location, description and a proposed fix for it.

3. Results and discussion

In order to relate debugging performance to the other experimental variables, the 42 participants were divided post-hoc on the basis of quartile ranges according to their debugging accuracy level. Group 1 comprised the participants with the lowest scores while group 4 comprised those with the highest scores.

The results for window switching frequency are illustrated in Figure 1. There were main effects for group ($F(3,38) = 3.704$, $p < .01$) but not for modality. Planned contrast comparisons indicated that switching frequency peaked for group 3 ($t(38) = 2.355$, $p < .05$). There was also an interaction effect for modality by group ($F(3,38) = 5.10$, $p < .01$). Planned contrast comparisons revealed a linear trend for the textual condition ($F(1,38) = 9.995$, $p < .01$) and a quadratic trend for the graphical condition ($F(1,38) = 8.061$, $p < .01$) which indicates a linear increase in switching frequency for the textual condition when going from less to more accurate groups and quadratic (peaking in an intermediate group) for the graphical condition. Individual t-tests also showed significant differences in terms of modality for groups 3 ($t(11) = 2.548$, $p < .05$) and 4 ($t(10) = -2.308$, $p < .05$).

The results for window fixation times reveal main effects for window ($F(3,38) = 823.36$, $p < .01$) and interaction effects for window and group ($F(9,38) = 2.431$, $p < .05$). The code window was the most frequently used (about 79% of the time against 9% for the objects and output windows and only 3% for the call sequence window). Regarding the interaction effect, planned contrast comparisons taking into account the sum of fixation times for the three supporting representations (the two visualisations plus the output window) indicated that fixation time for these three representations peaked for group 3 ($t(38) = 2.117$, $p < .05$). Figure 2 illustrates these interaction effect graphically. This figure presents the data split into graphical and textual conditions to provide a point of comparison with Figure 1; however, as there were no interactions for the combination of window, modality and group, the differences between graphical and textual support visualisations illustrated in this graph are not significant.

These results suggest that debugging accuracy is related

to a more balanced use of the available representations. While poor performers show a low degree of interaction with the support visualisations, those in the intermediate and high accuracy groups interact more with these representations (the intermediates even more than the advanced, as both switching frequency and fixation time for the support visualisations reach a peak for the third group). These results seem to suggest that participants in different accuracy groups might have chosen different debugging strategies. Although all of them focused predominantly on the code window, participants in the lowest performance group chose to use a strategy based almost exclusively on the code, while participants in other groups seemed to use the other representations to support the inferences drawn from the code.

The fact that switching frequency peaked for group 3 seems to be in agreement with the suggestion in [6] that, at least for professional programmers, debugging ability was inversely related to switches of focus between the sources of information provided for the task. These results were explained in terms of the programmers' chunking ability (the ability to detect meaningful, hierarchical units in the code during problem solving). A high chunking ability is related to a robust mental representation of the program and therefore to a low need for duplicating references to the available representations. It is possible that the debugging ability of participants in groups 3 and 4 has reached a point for which this explanation is valid.

The results also suggest different effects for the advanced group in terms of modality. Participants of this group switched more frequently in the graphical condition than in the textual one while for window fixation there were no significant differences involving modality. This suggests that participants in the highest accuracy group performed longer fixations in the graphical condition than in the textual condition. Therefore it seems that the textual visualisation condition required more active representation coordination for this group.

These results can be interpreted in several, possibly complementary, ways. One is that the specificity of graphical representations [5] helps programmers to detect the meaningful units or chunks of the code, and in this way decreases the need for switching constantly between the available representations. Other interpretation is that as textual representations generally require more active search [2], this condition might have employed a higher amount of working memory resources. This probably meant that participants were not able to hold as much information in working memory about the program as in the graphical condition. As a result, representation switching was more frequent when working with textual visualisations.

Another way to explain these results is that different modality conditions promoted the deployment of different debugging strategies for people in the best performing group. Different debugging strategies might require a different type of support from specific information sources [3], and therefore, in this case, a different frequency of switching. Further research to clarify the cause of these effects will examine the verbal protocols data.

4. Conclusions

This paper presented results of representation use in a multi-representational debugging environment and discussed the effect of modality within this environment. In general, good debugging performance is associated with a more balanced use of the available representations; however, this investigation has raised the question about whether graphical visualisations promote a more judicious use of the visualisations for successful performers. Further analysis, possibly of the verbal protocols data, is needed to clarify this issue.

Acknowledgments This work is supported by the EPSRC grant GR/N64199. The support for Richard Cox of the Leverhulme Foundation (Leverhulme Trust Fellowship G/2/RFG/2001/0117) and the British Academy is gratefully acknowledged. The authors would like to thank the participants for taking part in the study.

References

[1] S. Ainsworth. The functions of multiple representations. *Computers & Education*, 33(2-3):131–152, 1999.

[2] R. Cox. Representation construction, externalised cognition and individual differences. *Learning and Instruction*, 9:343–363, 1999.

[3] D. J. Gilmore. An investigation of the utility of flowcharts during computer program debugging. *International Journal of Man-Machine Studies*, 20(1):357–372, 1984.

[4] P. Romero, R. Lutz, R. Cox, and B. du Boulay. Co-ordination of multiple external representations during java program debugging. In S. Wiedenbeck and M. Petre, editors, *2002 IEEE Symposia on Human Centric Computing Languages and Environments*, pages 207–214. IEEE press, Airlington, Virginia, USA, 2002.

[5] K. Stenning and J. Oberlander. A cognitive theory of graphical and linguistic reasoning: logic and implementation. *Cognitive Science*, 19(1):97–140, 1995.

[6] I. Vessey. Expertise in debugging computer programs: a process analysis. *International Journal of Man-Machine Studies*, 23:459–494, 1985.