

Graphical visualisations and debugging: a detailed process analysis

Pablo Romero, Benedict du Boulay, Richard Cox
Rudi Lutz and Sallyann Bryant

Department of Informatics, Sussex University, U.K.

Abstract. This paper investigates the question of how programmers exploit and integrate multiple sources of information. In particular it analyses how undergraduate computer science students used the multiple representations available in a software debugging environment (SDE). This environment allowed them to view the execution of a program in steps and provided them with concurrently displayed, adjacent, multiple and linked representations. These programming representations comprised the program code, two visualisations of it and its output. This investigation studied debugging strategy in terms of rich process data about the use made of the representations available in the SDE and stepping facility. These data comprised computer interaction logs, audio recordings and data about visual attention focus.

The experimental results suggest that graphical representations seemed to promote a more efficient use of the available visualisations and were therefore associated with a relatively low level of interaction. This paper discusses these results and their implications for programming instruction.

1 Introduction

Much computer programming is performed via the use of software development environments which provide a variety of external representations and other sophisticated functionality. These representations and functionality enable programmers to treat programs not just as code text, but also as a range of abstract entities which can be visualised according to different criteria or executed under a variety of conditions. These visualisations can be presented in formats that range from mostly textual to mostly graphical [1].

The debugging step facility is one of the most helpful pieces of functionality of such environments. This facility allows programmers to execute and pause the program at different points. At these points they can inspect the visualisations provided to obtain information about various aspects of the program. Such program visualisation and debugging facilities should be especially helpful for novice programmers because they have the potential to enable them see the program not as a black box but as an abstract machine containing a set of elements and states. However, their effective use requires the programmer to deploy knowledge about how to decode and coordinate the available representations as well as skill in operating the SDE itself. It is often assumed that novices possess this knowledge. Thus, novice programmers can face a double challenge. As well as trying to learn abstract concepts about programming, they have to

