Integrating Interactive Visualizations of Automatic Debugging Techniques on an Integrated Development Environment

André Riboira, Department of Informatics Engineering, University of Porto, Porto, Portugal & HASLab / INESC TEC, Portugal

Rui Rodrigues, Department of Informatics Engineering, University of Porto, Porto, Portugal & INESC TEC, Porto, Portugal

Rui Abreu, Department of Informatics Engineering, University of Porto, Porto, Portugal & HASLab / INESC TEC, Portugal

José Campos, Department of Informatics Engineering, University of Porto, Porto, Portugal

ABSTRACT

Automated debugging techniques based on statistical analysis of historical test executions data have recently received considerable attention due to their diagnostic capabilities. However, the tools that materialize such techniques suffer from a common, rather important shortcoming: the lack of effective diagnostic reports' visualizations. This limitation prevents the wide adoption of such tools, as it is difficult to understand the diagnostic reports yielded by them. To fill this gap, the authors propose a framework for integrating interactive visualizations of automatic debugging reports in a popular development environment (namely, the Eclipse integrated development environment). The framework, coined GZoltar, provides several important features to aid the developer's efficiency to find the root cause of observed failures quickly, such as direct links to the source code editor. Furthermore, the authors report on the results of a user study conducted to assess GZoltar's effectiveness.

Keywords: *Diagnostic Reports, Information Systems, Interaction, Statistical Debugging, Visualization*

DOI: 10.4018/jcicg.2012070104

INTRODUCTION

When unexpected behavior is observed on a software system, developers need to identify the root cause that makes the system deviate from its intended behavior. This task (also known as software debugging, fault localization, or fault diagnosis) is the most time-intensive and expensive phase of the software development cycle (Hailpern & Santhanam, 2002). Regardless, it is performed since the beginning of computer history. As an indication of the downtime, debugging, and repair costs involved, a 2002 landmark study indicated that software bugs posed an annual $60 billion cost to the US economy alone (RTI, 2002). Software faults can also lead to major accidents when occurring on safety-critical systems (Dale & Anderson, 2009). Testing and debugging tasks should therefore receive considerable attention during the software development life-cycle. Hence, it is important to have powerful tools to help testers and developers on that crucial task.

Visualization is rather important for information comprehension (Van Wijk, 2005). In general, human beings find it more intuitive to understand information laid out in a logical, hierarchical way, than with a simple, rather plain and textual list of values. Nevertheless, currently available automatic debugging tools do not offer powerful visualizations of their debugging reports (Riboira, 2011).

In this paper, we present GZoltar, a visual debugger for Java programs that aims to fill the gap of currently available automatic debugging tools. The main premises for GZoltar are therefore the following:

- Implement a robust automatic debugging framework that allows different visualization techniques, and that may be easily expanded in the future;
- Help the user to find software faults faster, by aiding the understanding of debugging results;
- Be highly integrated in a multi-platform development environment to reduce the learning curve, and the time spent on swapping between faults' localization and their fixing;
- Have an easy and fast installation process to facilitate its adoption and use;

The automatic debugging tool behind GZoltar is Zoltar (Janssen, Abreu, & Van Gemund, 2009), a Spectrum-Based Fault Localization (SFL) framework whose performance is amongst the best ones for fault localization (Abreu, Zoeteweij, & Germund, 2009). GZoltar is implemented as a plug-in for the Eclipse integrated development environment (IDE) (Burnette, 2005) due to its wide adoption (Geer, 2005) and plug-in development facilities (McCullough, 2006). The interactive visualization framework uses OpenGL for graphics rendering due to (i) its flexibility to produce both 2D and 3D graphics, (ii) its performance supported by hardware acceleration, and (iii) its multi-platform availability (Shreiner & Group, 2009). As Eclipse cannot access OpenGL directly, some supporting libraries such as JOGL were used to create bindings to OpenGL native system libraries (Wolf, 2005). The interactive visualization framework can be easily extended with new visualizations, and we present two examples of possible visualizations as proof-of-concept: sunburst and treemap (Stasko, Catrambone, Guzdial, & McDonald, 2000).

The proposed interactive visualization framework eases the creation of different debugging data visualizations and allows navigation and integration with default Eclipse features, such as the code editor and the building warnings' list.

The remainder of this paper is organized as follows. In the next section we present a review of the state-of-the-art. Then, we explain automatic debugging and outline the Zoltar framework. The presentation of the GZoltar's architecture follows. Subsequently, we propose GZoltar's visualizations, followed by their interaction. Next, we report on the results of the user study. Finally, we conclude and discuss future work.