Visual Debugger

-summary-

https://doi.org/10.1109/ICSME55016.2022.00066

Măierean Mircea - Ignat Dragoș - Ichim Vlad

Approach and Motivation

Debugging is an essential part of software maintenance and evolution since it allows software developers to analyse the program execution step by step. As such, software developers spend a large percentage of their time validating and debugging software. This results in high software maintenance and evolution cost. **The Visual Debugger Tool** was developed to address the challenges software developers face when trying to understand complex program execution states during debugging. Traditional debuggers in modern IDEs, such as IntelliJ IDEA, represent data in textual form, which can be limiting. This becomes especially inefficient during software maintenance and evolution, which already consume a significant portion of developer time. The end goal of the tool is to use graphical representation of program states using UML such that developers can explore intuitively.

Aim of the tool and innovations

As mentioned earlier, text-based debugging works fine when you just need to look at a single object and its attributes. But if you're trying to understand how a bunch of objects are related, i.e. the whole structure, it doesn't make sense. But asking the tester to install and use extra tools just to get a visual representation doesn't really make sense either. The tool's innovation lies in its seamless integration with the IDE's debugging process, its lightweight and simplified interface. Unlike existing visual debuggers, the tool focuses strictly on object diagrams and avoids clutter by showing only relevant data, shown in-scope.

Validation of the tool

To validate its effectiveness, the tool was tested in practical debugging scenarios, especially unit testing where usually structured according to the Arrange-Act-Assert (AAA). Using a test that involved objects built from other relevant objects, the tool was used to debug a failing unit test. During the Arrange phase, the relevant objects were initialized, forming the object structure to be tested. By setting a breakpoint at the beginning of the Assert phase, the visual debugger provided an immediate and clear overview of the object world, including computed values from the Act phase. This demonstrated its ability to visualize complex object relationships clearly, helping in bug identification and correction. With over 14000 downloads and positive feedback, the tool has strong potential for large scale use. It would be of great interest to extend its functionality across other IDEs, especially as it is lightweight and takes up minimal resources.