What is JIVE?

JIVE is an interactive execution environment for Eclipse that supports a declarative and visual approach to debugging object-oriented software. It can also serve as a pedagogic tool for teaching object-oriented programming. JIVE extends Eclipse’s Java debugging facilities with interactive visualization, query-based debugging, and reverse stepping.

Principles of JIVE

- Support Full Java Language
- Depict Objects as Environments
- Visualize Current State and Execution History
- Provide Multiple Views of Runtime State
- Produce Aesthetic Layouts
- Support Forward and Reverse Execution
- Support Declarative Queries on Runtime States
- Use Existing Java Technology

Interactive Visualization

JIVE depicts both the runtime state and execution history of a program in a visual manner. The runtime state is visualized as an enhanced object diagram, showing object structure as well as method activations in their proper object contexts. The execution history is depicted as a sequence diagram, clarifying the object interactions that occur at runtime. The diagrams are scalable and can be filtered to show only information pertinent to the task at hand.

Query-based Debugging

Traditional debugging is a procedural process in which a programmer must proceed step-by-step and object-by-object in order to uncover the cause of an error. In contrast, JIVE offers a declarative approach to debugging by providing an extensible set of queries over a program’s execution history. Queries are formulated using the source code or the diagrams, and the results are shown in a tabular format and also as diagram annotations. Declarative debugging complements procedural debugging just as web searching complements web browsing.

Queries can be about individual states or the execution history. Examples include:

- Object Attribute and Local Variable Changes
- Class/Object Invariant Checking
- Object Creations
- Method Activations and Statement Executions
- Exceptions Thrown and Caught

User-defined queries can be added using Eclipse’s JIVE’s extension point mechanism.

Reverse Stepping

JIVE supports both forward and reverse stepping of Java programs. Often, a programmer may discover that an error has occurred only after the errant statement has been executed. Providing the ability to step backwards saves a programmer the time and effort of re-executing the program until the point of error. JIVE also provides the ability to jump directly back to any previous point in the execution history in order to observe the object diagram at that point. Reverse stepping and jumping work closely with query-based debugging to narrow down the cause of program errors.

Future Work

- Reduce Overheads of Data Collection
- Produce Scalable Visualizations
- Runtime Flow Analysis for ‘Why’ Queries
- Database Support for Long Executions
- Efficient Jumping to Previous States
- Closer Integration with the JDT Debugger