



## Query-based Debugging and Visualization in JIVE

Jeffrey K. Czyz

Department of Computer Science and Engineering

University at Buffalo

The State University of New York

Adviser: Dr. Bharat Jayaraman



## Motivation

- *“Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated \$59.5 billion annually.”*

National Institute of Standards and Technology  
[http://www.nist.gov/public\\_affairs/releases/n02-10.htm](http://www.nist.gov/public_affairs/releases/n02-10.htm)

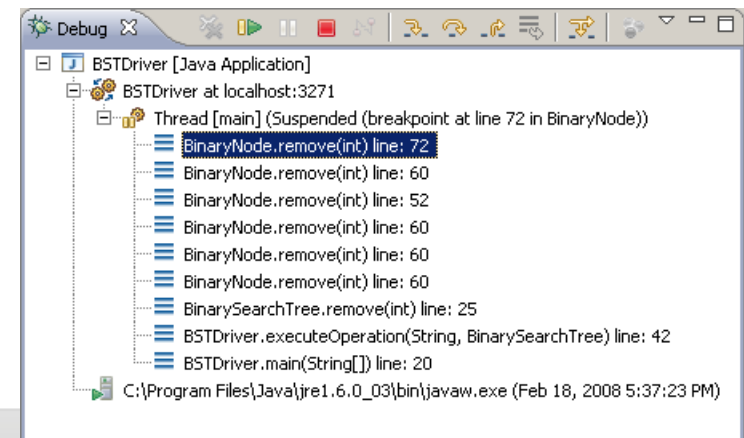
- According to the same study, \$22.2 billion can be saved through more effective identification and removal of bugs.



# JIVE: Java Interactive Visualization Environment

## Motivation

- Debugger technology has not changed much
  - Setting breakpoints
  - Spying on variables
  - Stepping forward in execution
  - Examining variables using the call stack
- Modern debuggers are still mainly *procedural* and *textual* in nature





# JIVE: Java Interactive Visualization Environment

## Declarative and Visual Approach

- Visualize execution history and state to clarify object interactions and object structure
  - Enhanced Object Diagram (execution state)
  - Sequence Diagram (execution history)
- Provide an extensible set of queries over a program's execution history and individual runtime states
  - Formulate queries using diagrams or source code
  - Report results as diagram annotations
- Allow revisiting previous runtime states



# JIVE: Java Interactive Visualization Environment

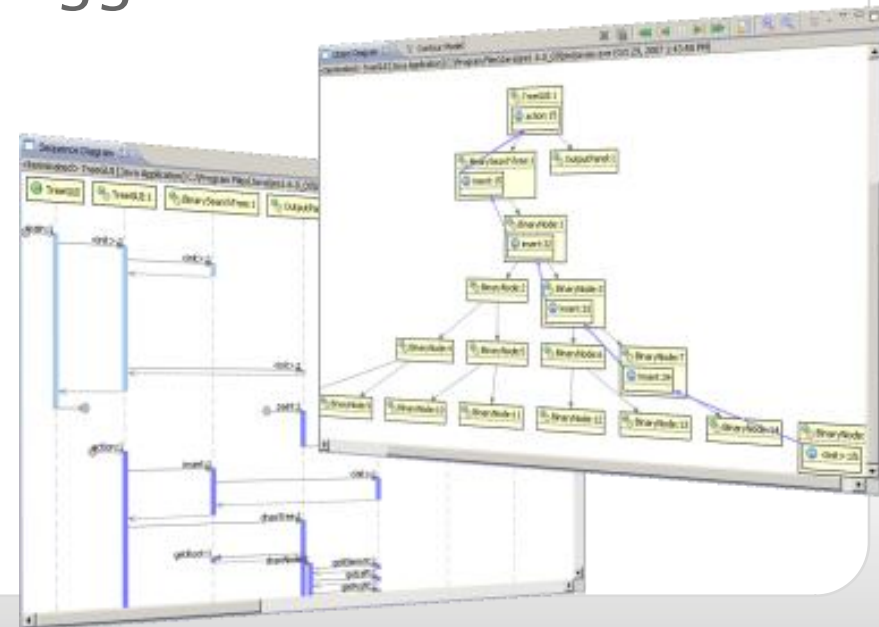
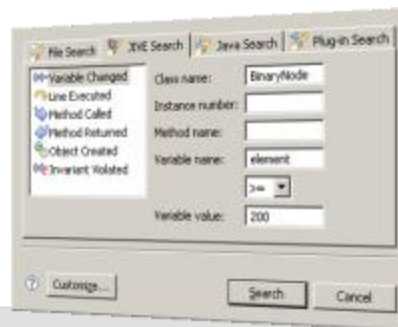
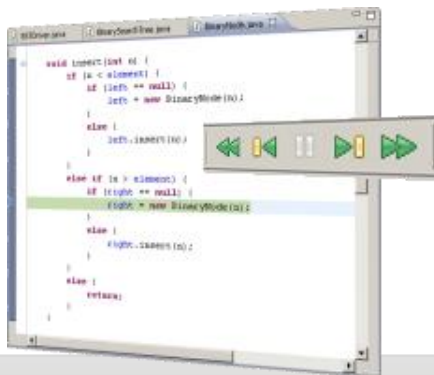
*Declarative debugging complements procedural debugging  
just as web searching complements web browsing.*



# JIVE: Java Interactive Visualization Environment

## What is JIVE?

- Interactive Execution Environment for Eclipse
  - Declarative and Visual Approach to Debugging
  - Pedagogic Tool
- Extension of the JDT Debugger
  - Interactive Visualization
  - Query-based Debugging
  - Reverse Stepping





# JIVE: Java Interactive Visualization Environment

## 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

# JIVE: Java Interactive Visualization Environment

## JIVE Perspective

Current State

The screenshot displays the JIVE Eclipse IDE interface. On the left, the Package Explorer shows the project structure. The central editor shows the source code for the `BinaryNode` class. The right-hand side features the Object Diagram, which visualizes a binary tree structure with nodes like `TreeGUI.1`, `BinarySearchTree.1`, and various `BinaryNode` instances. Below the Object Diagram is the Sequence Diagram, which tracks the execution flow between objects. The Console at the bottom shows a list of assignments for the `BinaryNode.left` property.

```
public class BinaryNode {
    private BinaryNode left;
    private BinaryNode right;
    private int element;

    BinaryNode(int n) {
        this.element = n;
        left = null;
        right = null;
    }

    int getElement() {
        return element;
    }

    BinaryNode getLeft() {
        return left;
    }

    BinaryNode getRight() {
        return right;
    }

    void insert(int n) {
        if (n < element) {
            if (left == null) {
                left = new BinaryNode(n);
            } else {
                left.insert(n);
            }
        } else {
            if (right == null) {
                right = new BinaryNode(n);
            } else {
                right.insert(n);
            }
        }
    }
}
```

| Thread           | Number | Event        | Context       | Variable | Value         |
|------------------|--------|--------------|---------------|----------|---------------|
| AWT-EventQueue-0 | 1106   | Assign Event | BinaryNode:4  | left     | BinaryNode:5  |
| AWT-EventQueue-0 | 1327   | Assign Event | BinaryNode:9  | left     | null          |
| AWT-EventQueue-0 | 1343   | Assign Event | BinaryNode:7  | left     | BinaryNode:9  |
| AWT-EventQueue-0 | 1597   | Assign Event | BinaryNode:10 | left     | null          |
| AWT-EventQueue-0 | 1578   | Assign Event | BinaryNode:11 | left     | null          |
| AWT-EventQueue-0 | 1684   | Assign Event | BinaryNode:5  | left     | BinaryNode:11 |
| AWT-EventQueue-0 | 2182   | Assign Event | BinaryNode:12 | left     | null          |
| AWT-EventQueue-0 | 2305   | Assign Event | BinaryNode:13 | left     | null          |
| AWT-EventQueue-0 | 2349   | Assign Event | BinaryNode:14 | left     | null          |
| AWT-EventQueue-0 | 2855   | Assign Event | BinaryNode:6  | left     | BinaryNode:14 |
| AWT-EventQueue-0 | 3216   | Assign Event | BinaryNode:15 | left     | null          |

Execution History





# JIVE: Java Interactive Visualization Environment

## Interactive Execution

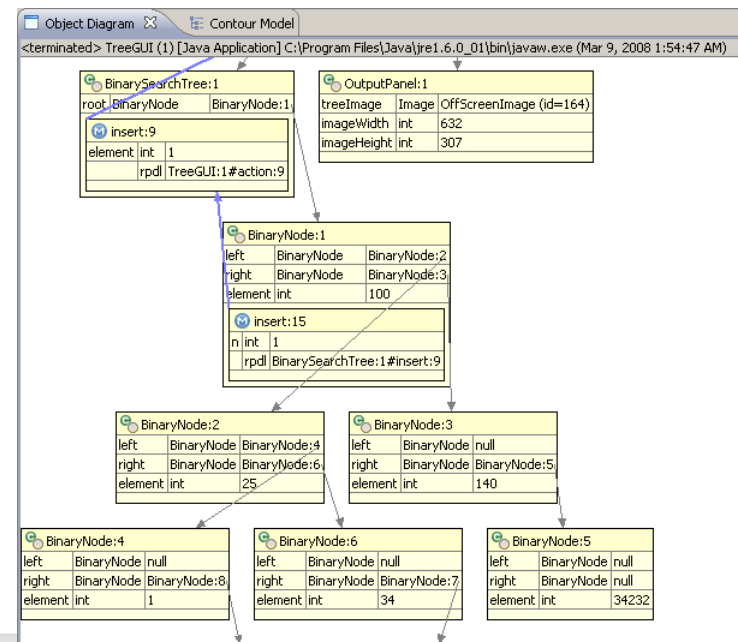
- JIVE depicts the current runtime state and execution history of a program in a visual manner
  - Enhanced Object Diagram
  - Sequence Diagram
- Diagrams are formed dynamically at runtime
  - Object Diagram changes as methods are called and returned, objects are created, variables assigned, etc.
  - Sequence Diagram grows as the program executes
- Each point on the Sequence Diagram corresponds to a runtime state (depicted by the Object Diagram)



# JIVE: Java Interactive Visualization Environment

## Object Diagram

- Depicts the current runtime state
- Method activations shown within their proper object contexts
- Call path shown
- Clarifies object structure
- Expand/Collapse objects
- Show/Hide member tables

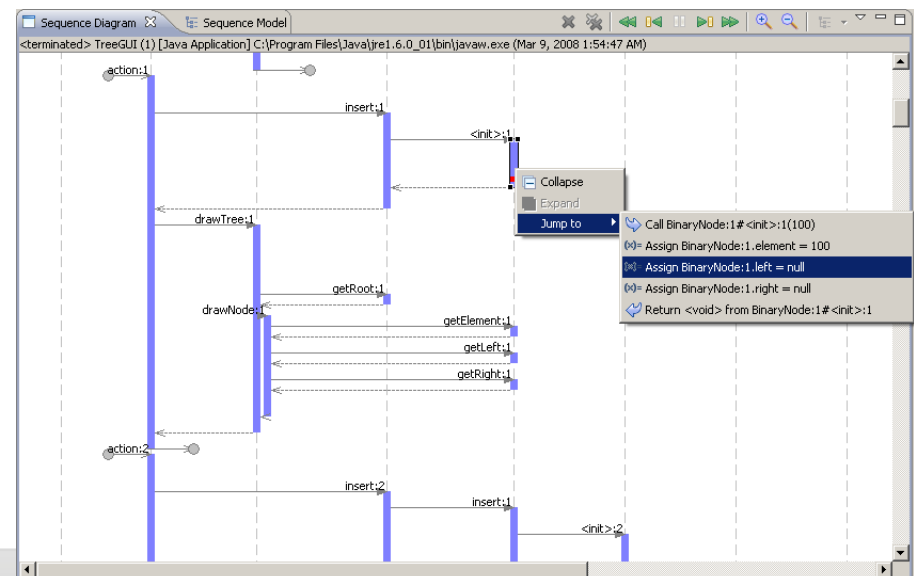




# JIVE: Java Interactive Visualization Environment

## Sequence Diagram

- Depicts history of execution
- Method activations shown along object lifelines
- Expand/Collapse method activations
- Jump back to previous runtime states
- View search results
- Clarifies object interactions

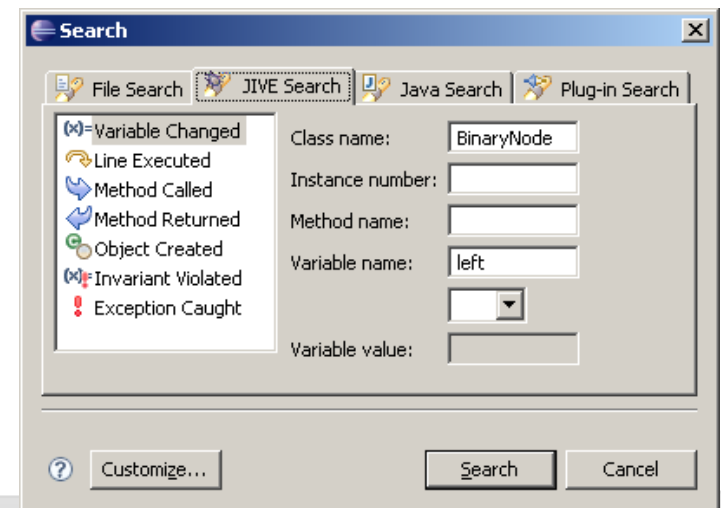




# JIVE: Java Interactive Visualization Environment

## Query-based Debugging

- Search individual states or execution history
  - Object Attribute and Local Variable Changes
  - Class/Object Invariant Checking
  - Object Creations
  - Method Activations
  - Statement Executions
  - Exceptions Thrown/Caught
- Result reporting
  - Diagram annotations
  - Tabular form
  - Call path to event



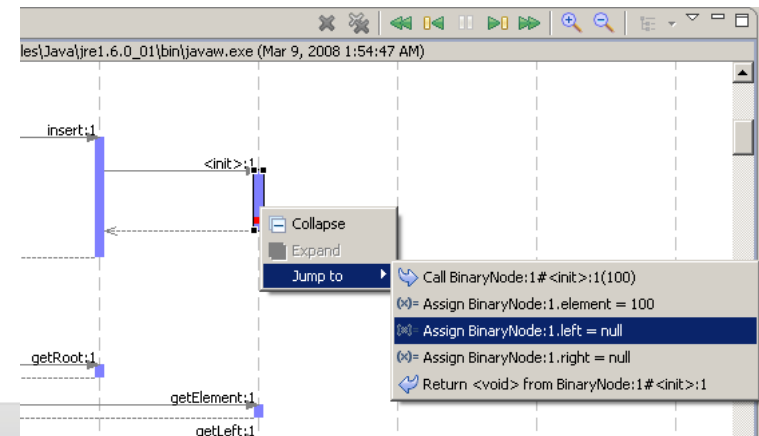


# JIVE: Java Interactive Visualization Environment

## Reverse Stepping

- Often bugs are not discovered until after the errant statement executes
- JIVE supports stepping forward or backward through execution to review past states
- Can jump immediately back to a point of interest using the diagram or query results

| Thread           | Number | Event        | Context      | Variable | Value        |
|------------------|--------|--------------|--------------|----------|--------------|
| AWT-EventQueue-0 | 275    | Assign Event | BinaryNode:3 | left     | null         |
| AWT-EventQueue-0 | 395    | Assign Event | BinaryNode:4 | left     | null         |
| AWT-EventQueue-0 | 537    | Assign Event | BinaryNode:5 | left     | null         |
| AWT-EventQueue-0 | 543    | Assign Event | BinaryNode:2 | left     | BinaryNode:5 |
| AWT-EventQueue-0 | 702    | Assign Event | BinaryNode:6 | left     | null         |
| AWT-EventQueue-0 | 888    | Assign Event | BinaryNode:7 | left     | null         |
| AWT-EventQueue-0 | 894    | Assign Event | BinaryNode:3 | left     | BinaryNode:7 |
| AWT-EventQueue-0 | 1114   | Assign Event | BinaryNode:1 | left     | BinaryNode:2 |
| AWT-EventQueue-0 | 1300   | Assign Event | BinaryNode:8 | left     | null         |
| AWT-EventQueue-0 | 1512   | Assign Event | BinaryNode:9 | left     | null         |
| AWT-EventQueue-0 | 1518   | Assign Event | BinaryNode:8 | left     | BinaryNode:9 |





# JIVE: Java Interactive Visualization Environment

## 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



# JIVE: Java Interactive Visualization Environment

## JIVE Web Site

- <http://www.cse.buffalo.edu/jive/>
- Download using update manager
- Subscribe to RSS feed for latest news and releases
- Comments and suggestions welcome

Thank you!