Abstract

Concurrency bugs are difficult to detect due to variations in thread interleavings (order of execution). Eclipticon is a tool used to expose concurrency bugs in Java programs, by inserting noise into the source code so that the threads will change their interleavings.

Problem Statement

Eclipticon is an open-source tool, implemented as an Eclipse plug-in, that allows concurrent Java code to be instrumented, and allows the tester to control the granularity of instrumentation with the final goal of exposing concurrency bugs.

Background

- Concurrency is increasing in popularity
- Java offers many concurrency features
- Concurrency bugs appear intermittently which makes detection and testing difficult
- Fine-grained control potentially helps localize concurrency bugs to speed error correction
- Eclipticon was inspired by IBM's ConTest testing tool which is a closed-source command line tool to instrument Java code

ECLIPTICON

Chris Forbes, Kevin Jalbert, Cody LeBlanc
Faculty of Engineering and Applied Science, Capstone Project, April 2010

Supervisors: Dr. Ramiro Liscano,
Dr. Jeremy S. Bradbury

UOIT
CHALLENGE INNOVATE CONNECT

Design

GUI

- Starts the plug-in
- Creates the sidebar in Eclipse
- Handles user interaction

Parsers

- Finds Interest Points
- **Pre-Parser** – finds syncronized methods
- **File Parser** – finds Interest Points
- **Annotation Parser** – creates and reads annotation comments, if comment is present: Interest Point becomes Instrumentation Point.
- **Method Call Validator** – verifies if synchronized method call is valid based on import statements

Test

- JUnit test cases for all important Eclipticon classes

Data

- Contains the data structures
- Internal data represenations
- **Source File** – each file has a name, path, import statements, and Interest Points
- **Interest Point** – a concurrency construct found in a source file
- **Instrumentation Point** – an Interest Point that will have noise added to the source code
- **Configuations** – settings used for Automatic Instrumentation

Instrumentation

- Performs instrumentation and backup files
- **Noise Maker** – creates noise (sleep, yield)
- **Instrumentor** – places noise in front of Instrumentation Points in source code

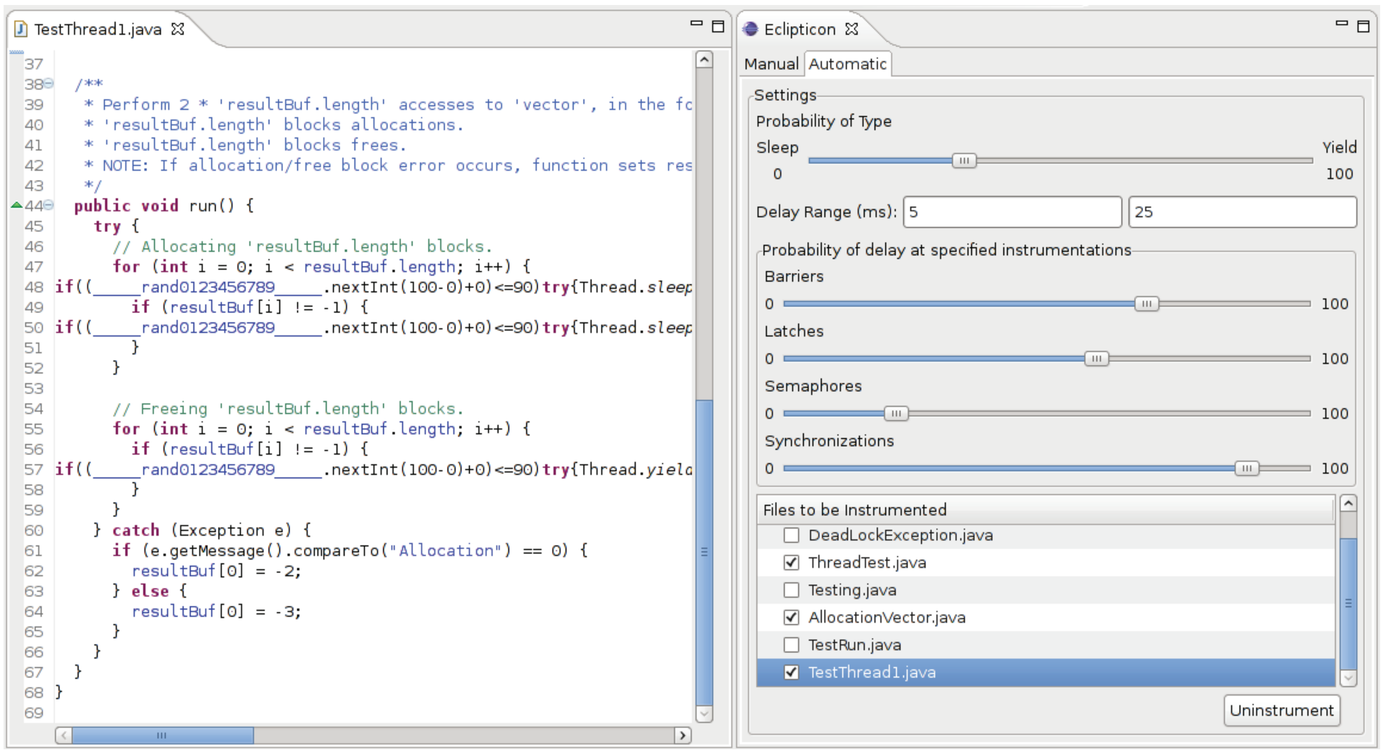

Figure 1: Eclipticon's Automatic Instrumention with Noise Inserted

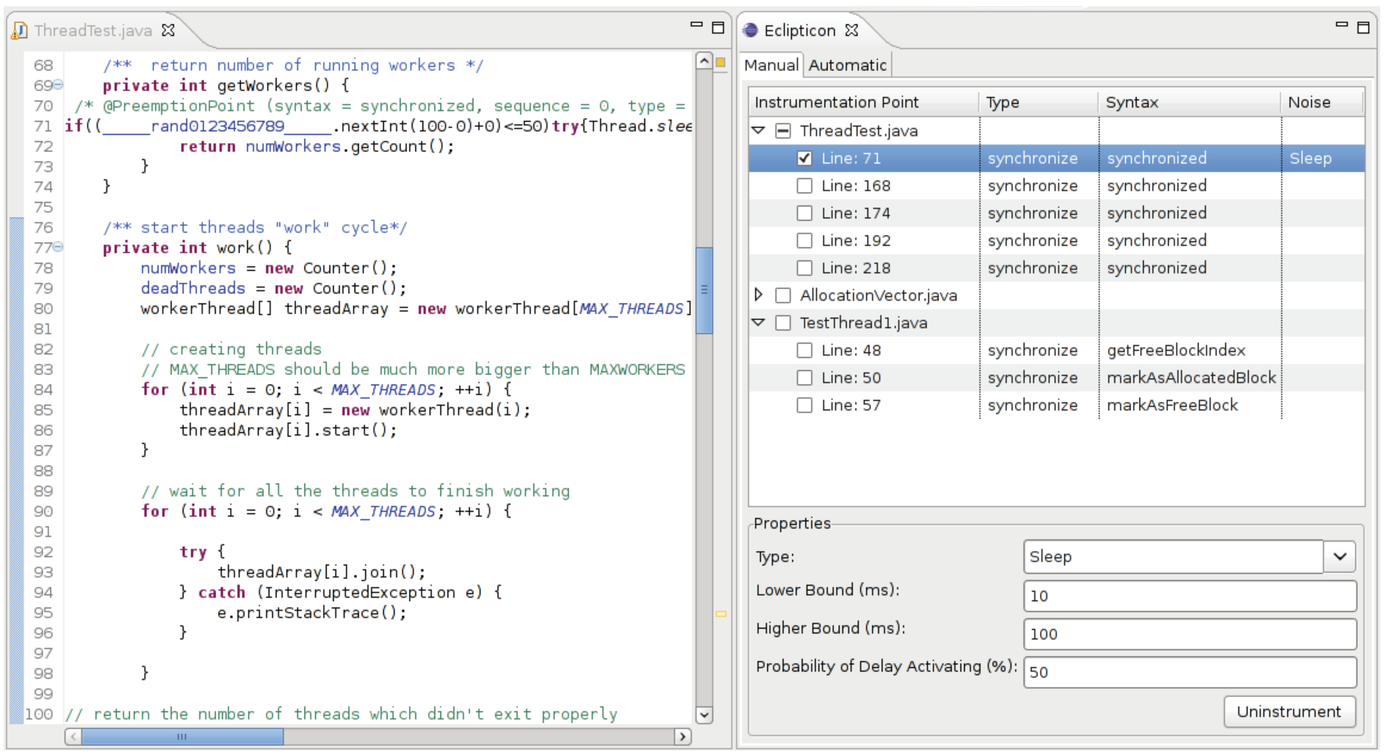

Figure 2: Eclipticon's Manual Instrumention with Annotation Comment Inserted

Analysis

- Preliminary tests using Allocation Vector program from IBM's Concurrency Benchmark
  - No instrumentation: 234 bugs / 10 000 executions
  - Yield noise: 530 bugs / 10 000 executions
  - Sleep noise: 1036 bugs / 10 000 executions

Future Enhancements

- Utilize object types to improve accuracy of Interest Point detection
- Parse source code using logical instead of physical lines
- Integrate automated JUnit testing of instrumented code