IBM Software Group

P17 System Testing
Monday, September 24, 2007

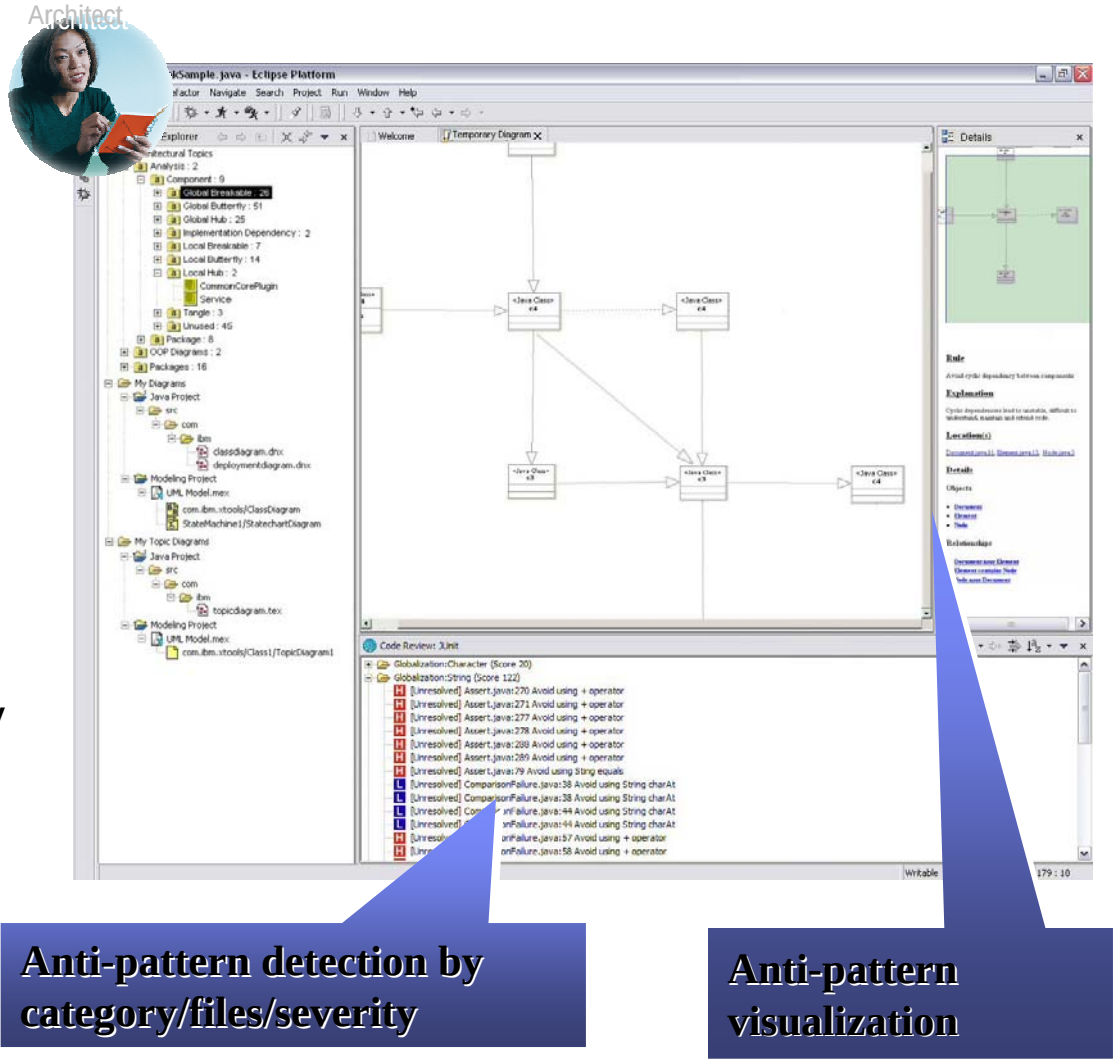
Module 8 : IBM Rational Testing Solutions

Rational. software

Marty Swafford
IBM Rational Software
IBM Certified Solution Designer - Rational Manual Tester, Rational Performance Tester, Rational Functional Tester for Java
mswaffor@us.ibm.com

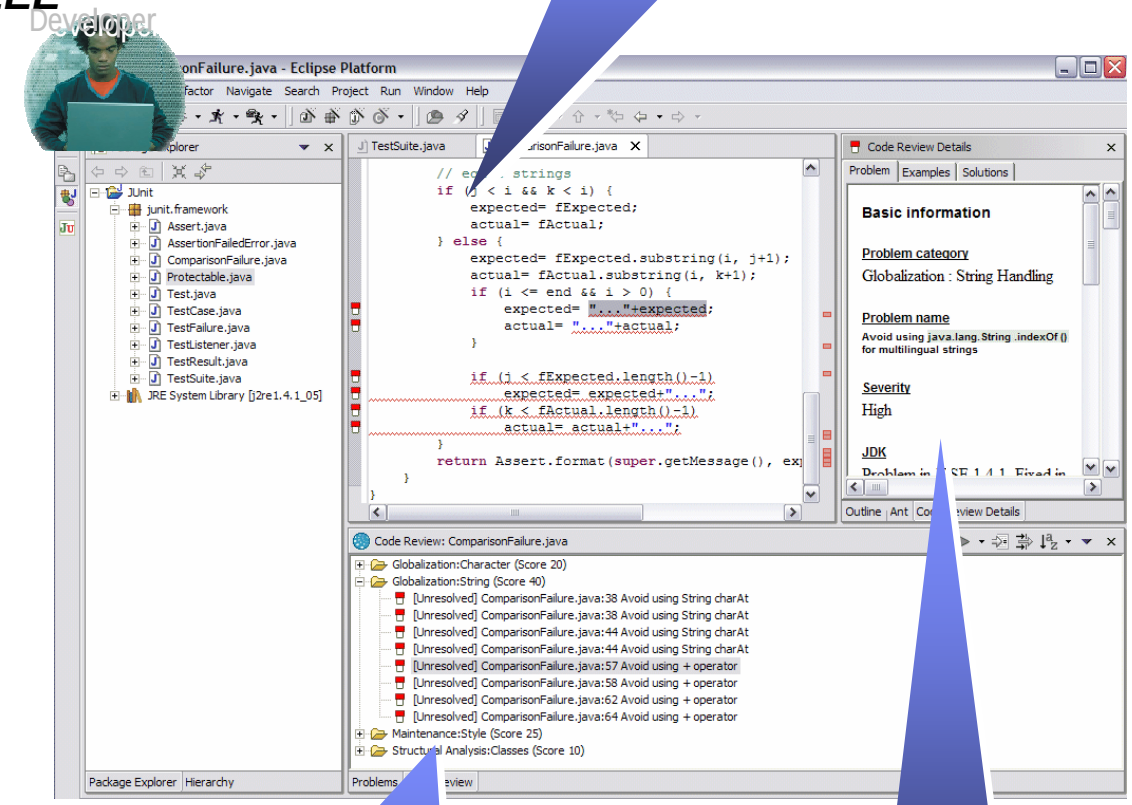
Design: Architecture Discovery, Analysis and Control

- ***Architect discovers the architecture as well as control the changes made during implementation***
- Automatic diagramming of significant structures (e.g. inheritance trees, class internals, package structure)
- Insight into the architecture through Diagram Browsing capability
- Validation of structural quality through structural antipattern rules
- Definition of custom architectural rules ensure design integrity



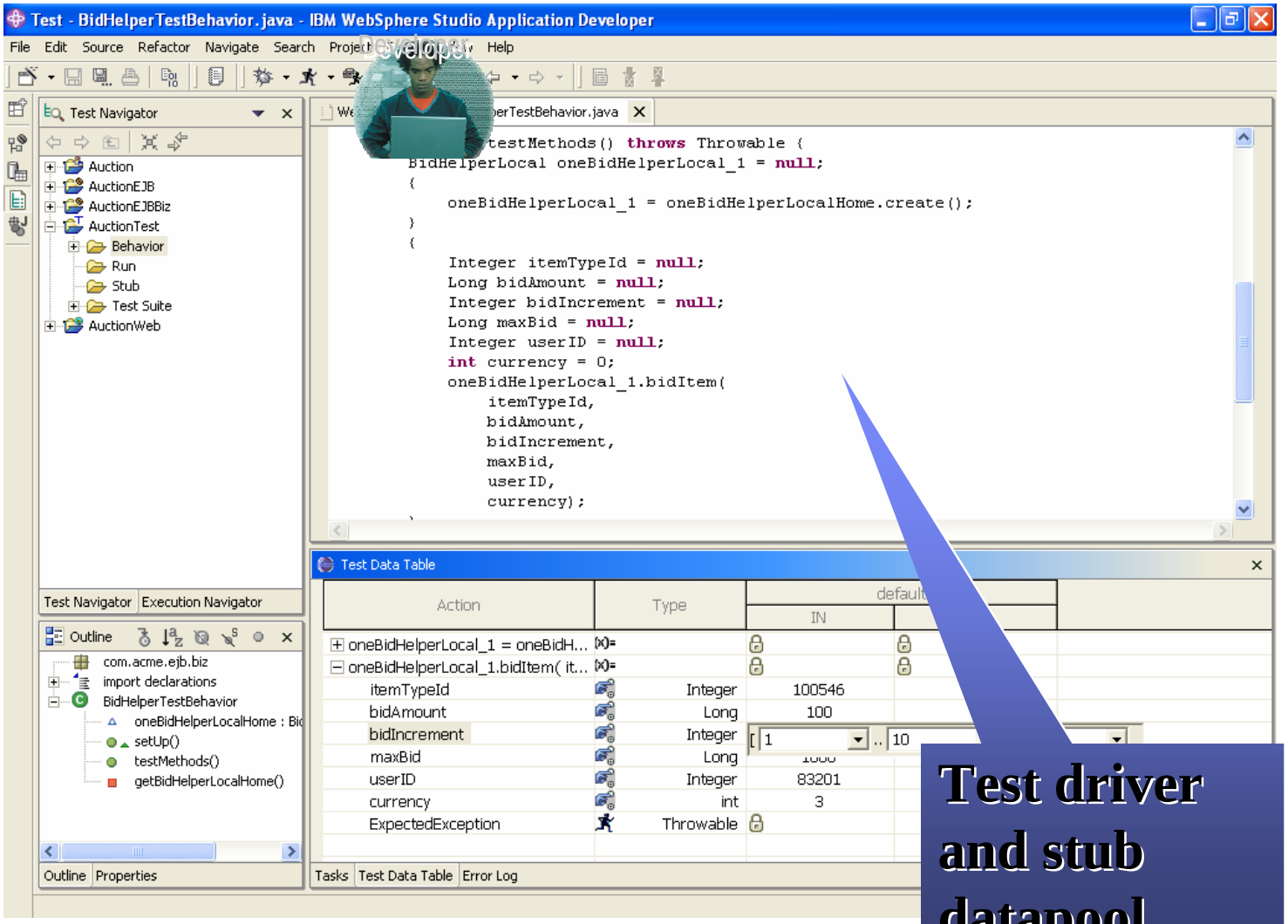
Construction: Code Review

- ***Developer implements the code and receives guidance for J2SE and J2EE best practices***
- Validation of code through sets of coding rules (200+) in various categories:
  - J2EE Best Practices
  - J2SE Best Practices
  - Performance
  - Private API
  - Coding style
  - Globalization
  - Naming conventions
  - Testability & Testing completeness
  - Design Principles
  - Structural Anti-patterns
- Users can define their own rules
- Eclipse's Quick Fix available
- Includes IBM Research IP



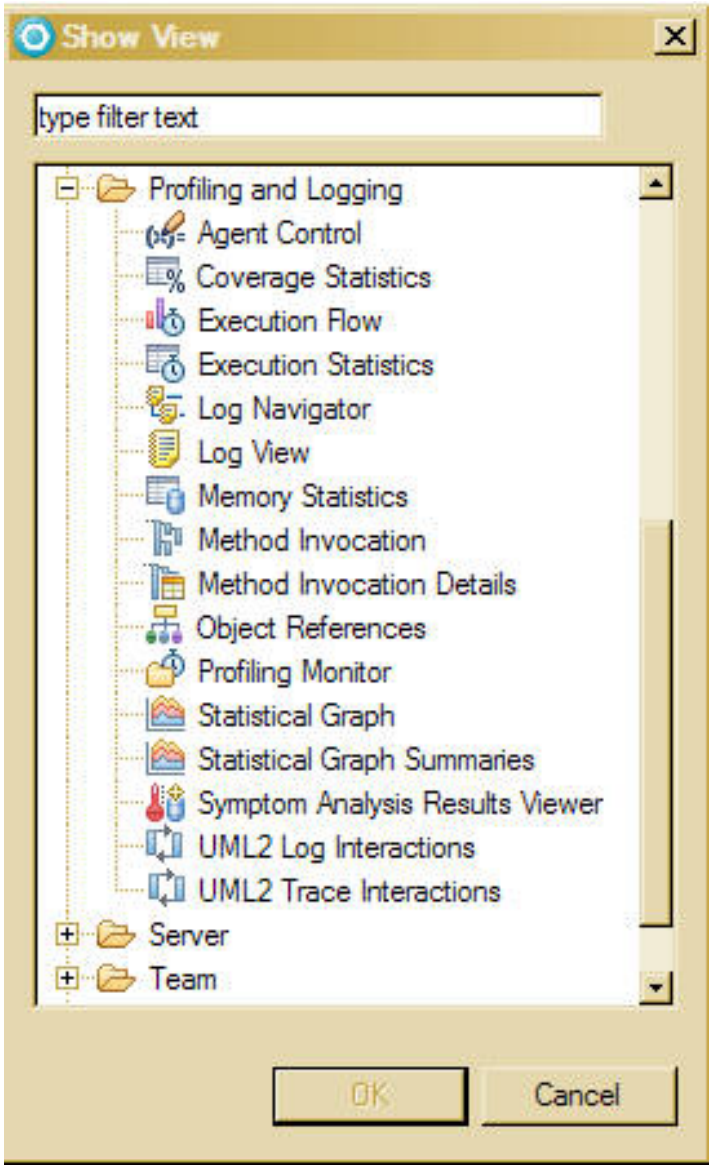
Component Test: Verify component functionality

- ***Developer automatically generates fully functional unit and API tests for Java classes, EJBs and Web Services (incl. .Net)***
- Provides testing guidance
  - What to test first based on metrics?
  - What test patterns apply to code?
- Automated test driver and stub generation
  - Includes both control flow and data
- Based on the JUnit framework
- Data-driven testing
- No need to write code!
- Integrated with debugging and line level code coverage



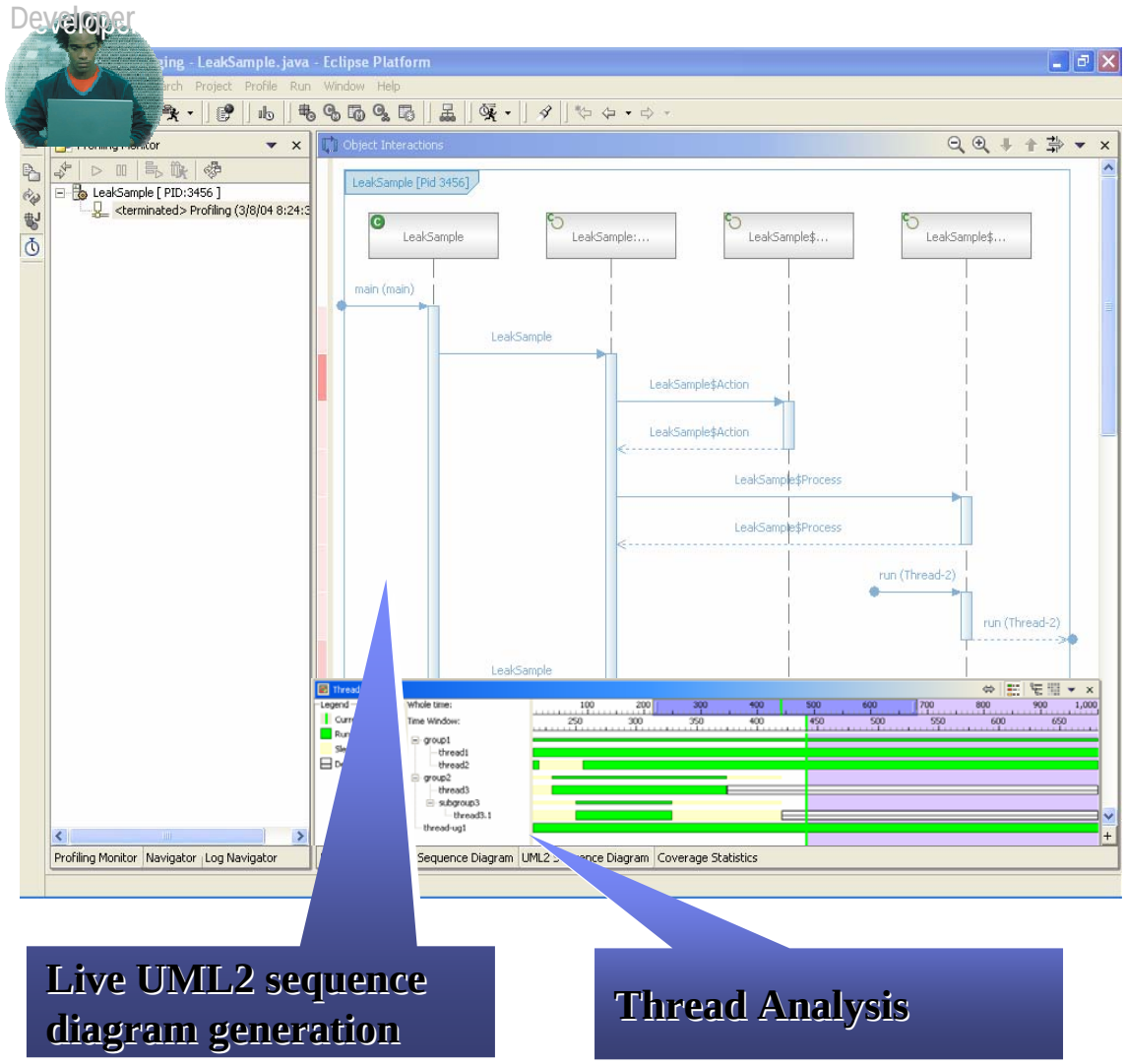
Runtime Analysis

- Built-in tools helps developer isolate and fix performance problems
- Advanced features including:
  - ✓ Execution Stats
  - ✓ Log Navigator and View
  - ✓ Memory Stats
  - ✓ Method Invocation
  - ✓ Object References
  - ✓ Profiling Monitor
  - ✓ Statistical Graph and Graph Summaries
  - ✓ Statistical Data
  - ✓ UML Sequence Diagram
  - ✓ UML Log Interactions
  - ✓ UML Trace Interactions
  - ✓ Static and Dynamic Probe Kit
  - ✓ Probe Kit Editor
  - ✓ J2EE Request Profiler
  - ✓ PI-Agent
  - ✓ RAC Security Extension
- Profiling tools can seamlessly trace across multiple servers

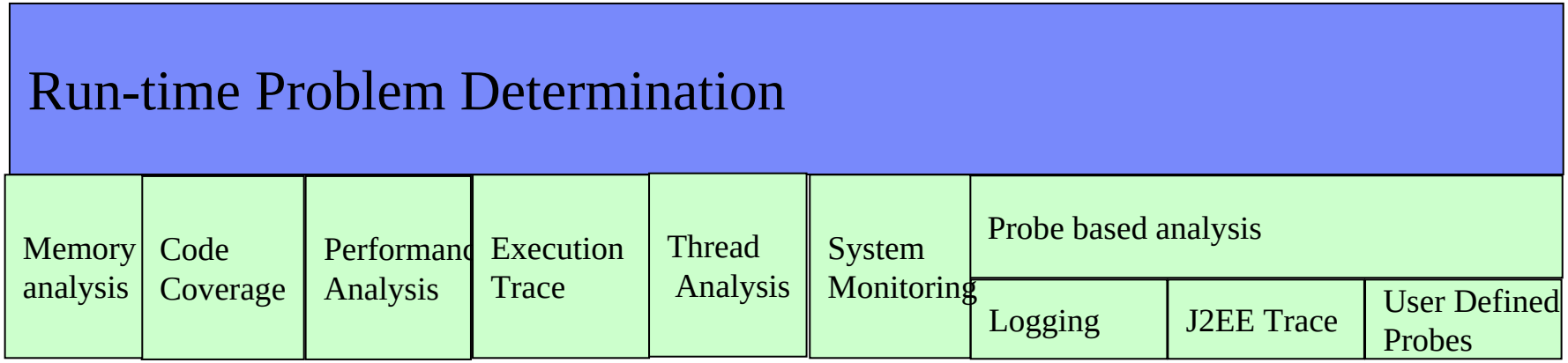


Debug: Understand behavior, performance & memory

- ***Developer analyzes application for memory, performance, functional problems***
- Application tracing
  - User defined probes for logging
  - Thread analysis
  - Multi-tier UML 2.0 sequence diagrams
  - Automated log analysis with symptom databases
- Performance analysis
  - Call graph
  - Thread analysis
- Memory analysis
  - Local and remote memory leak analysis down to the line level
  - Includes IBM Research IP
- Code coverage

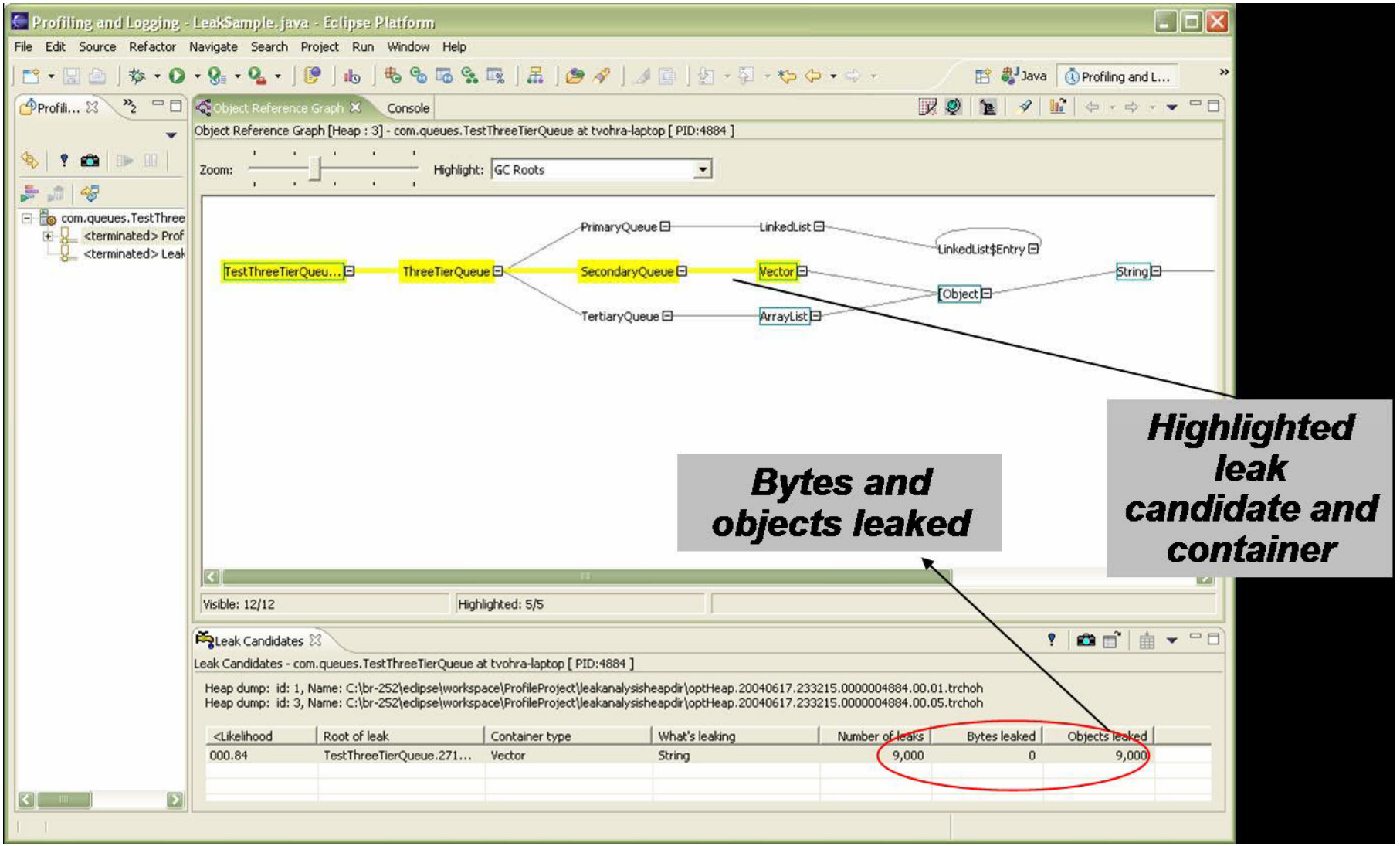


Examples of Run-Time Problem Determination

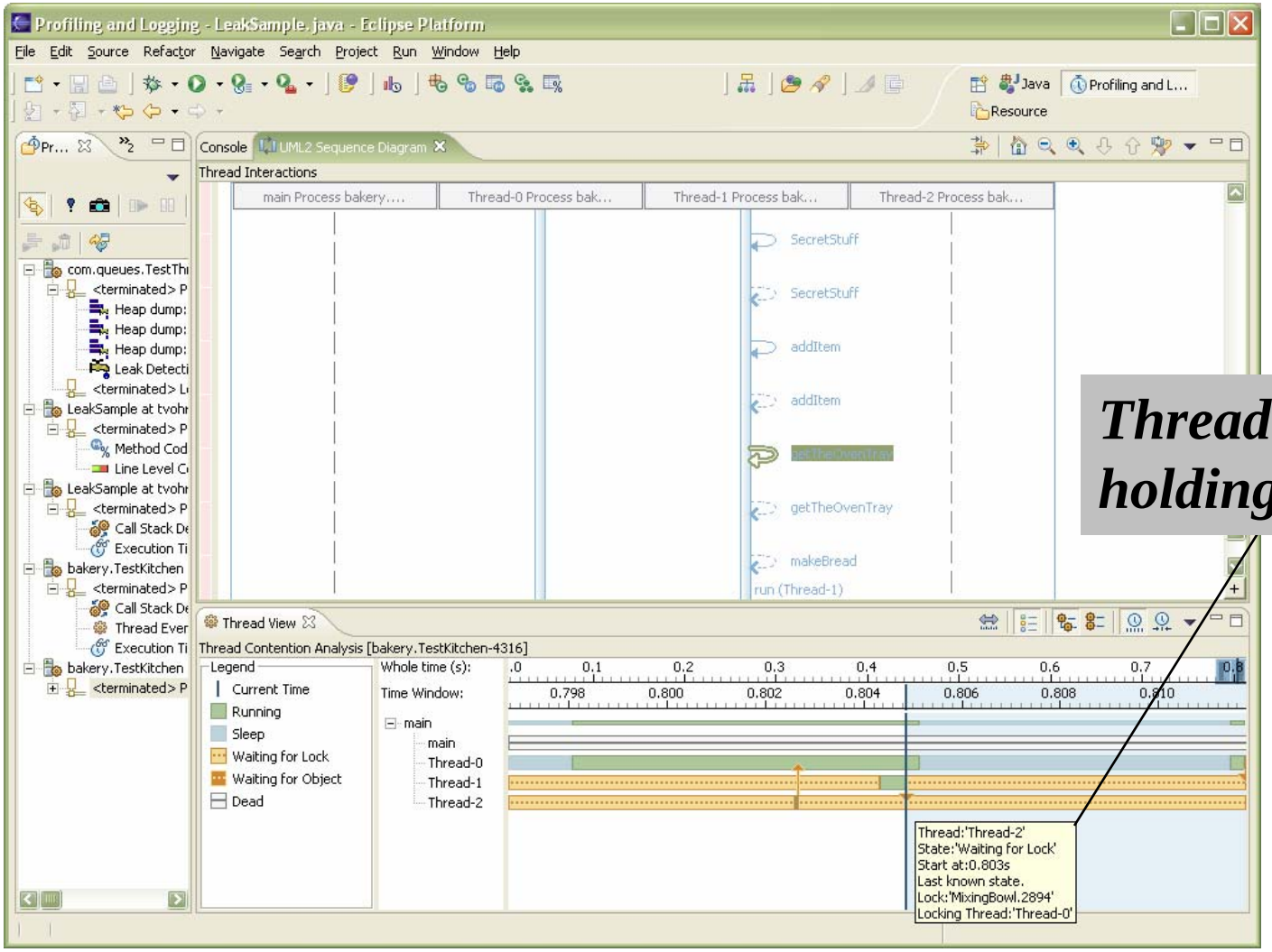


- Includes:
  - Memory profiling and leak detection
  - Performance profiling
  - Thread analysis and deadlock detection
  - Code coverage analysis
  - Runtime log and trace analysis

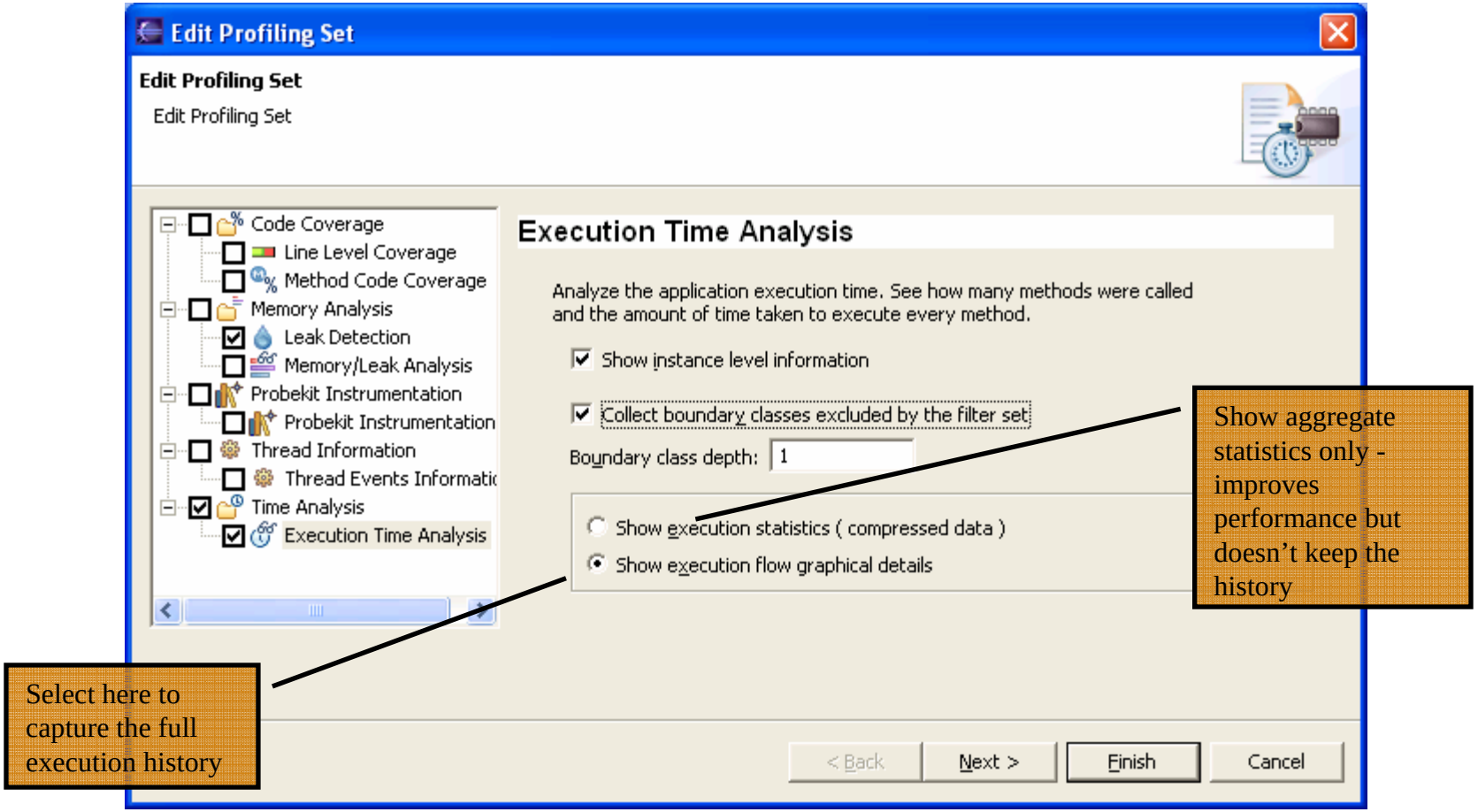
Memory Analysis



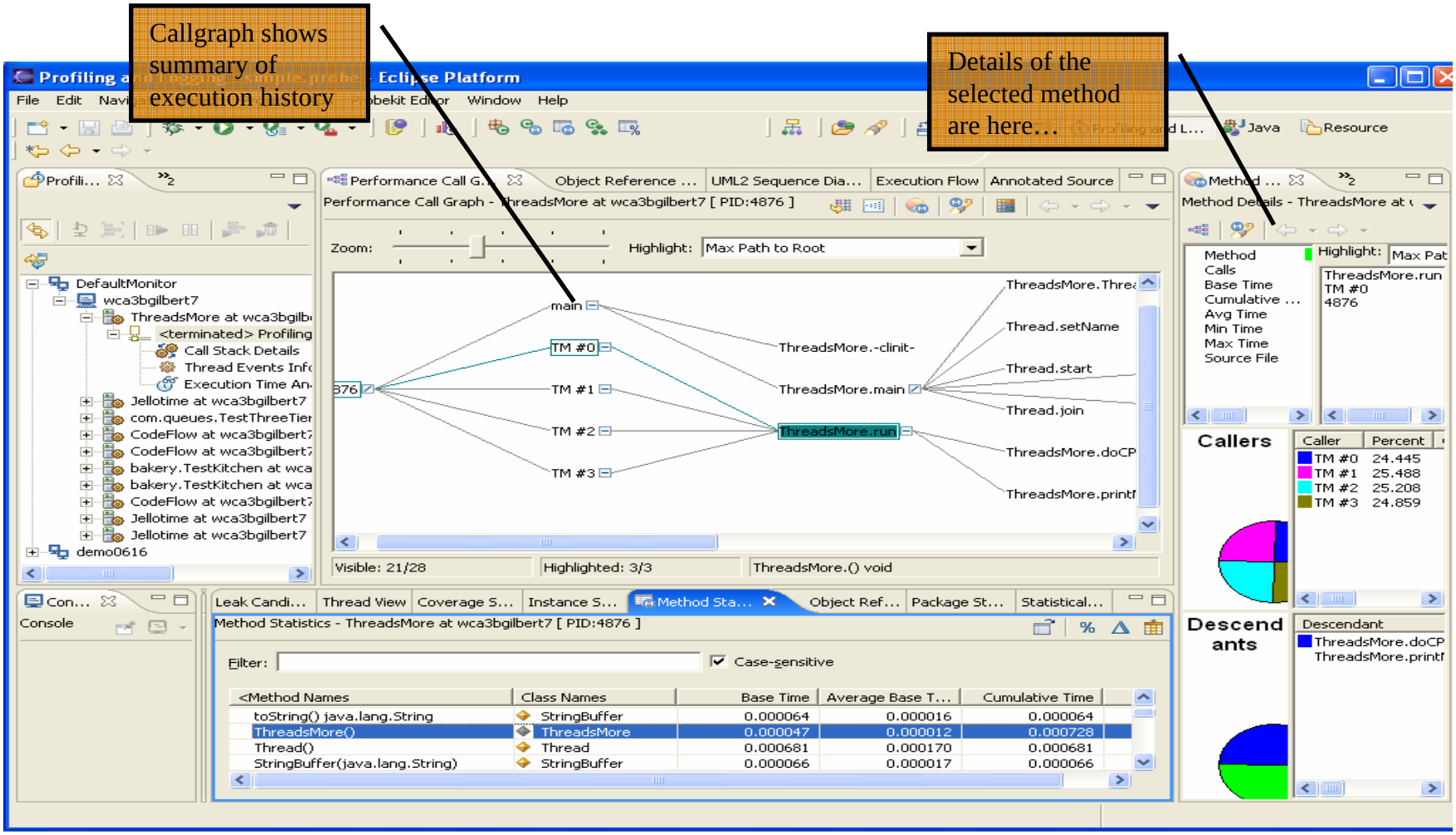
Threads view and UML2 thread interactions



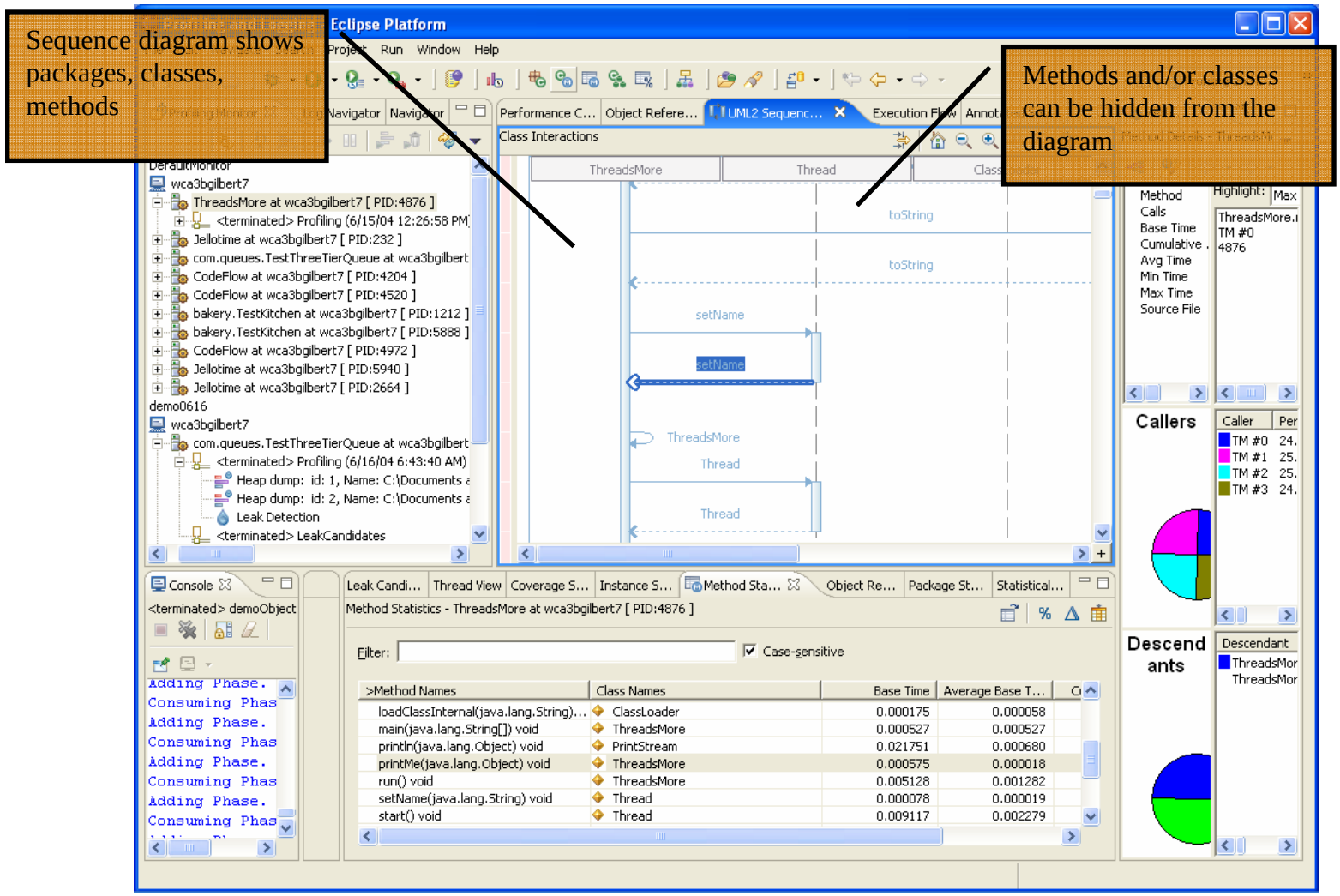
Performance analysis instrumentation



Execution history: performance callgraph

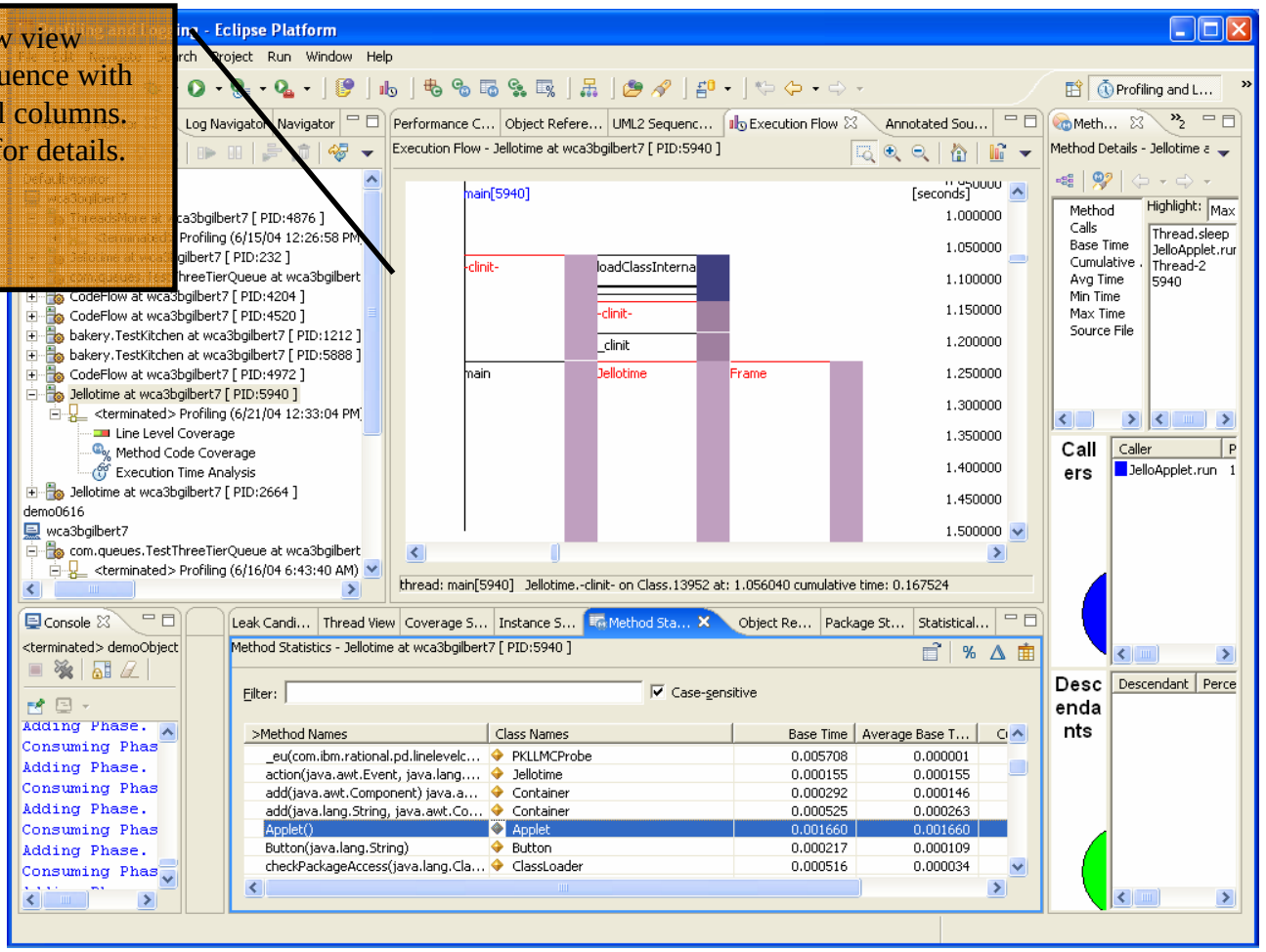


Execution History – Sequence Diagrams



Execution History: Execution Flow

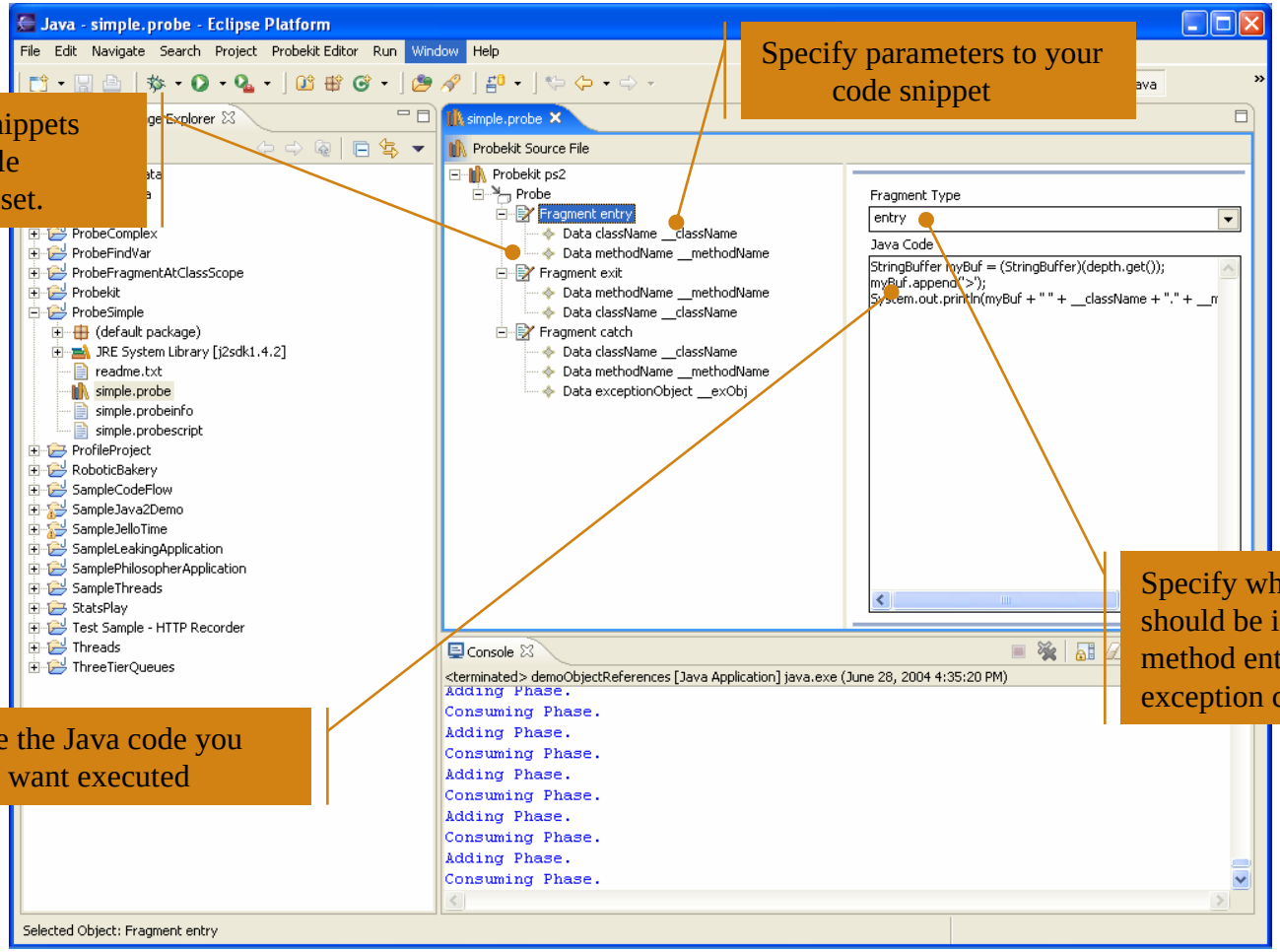
The execution flow view shows the call sequence with Threads as vertical columns. You can zoom in for details.



Probe-Based Analysis

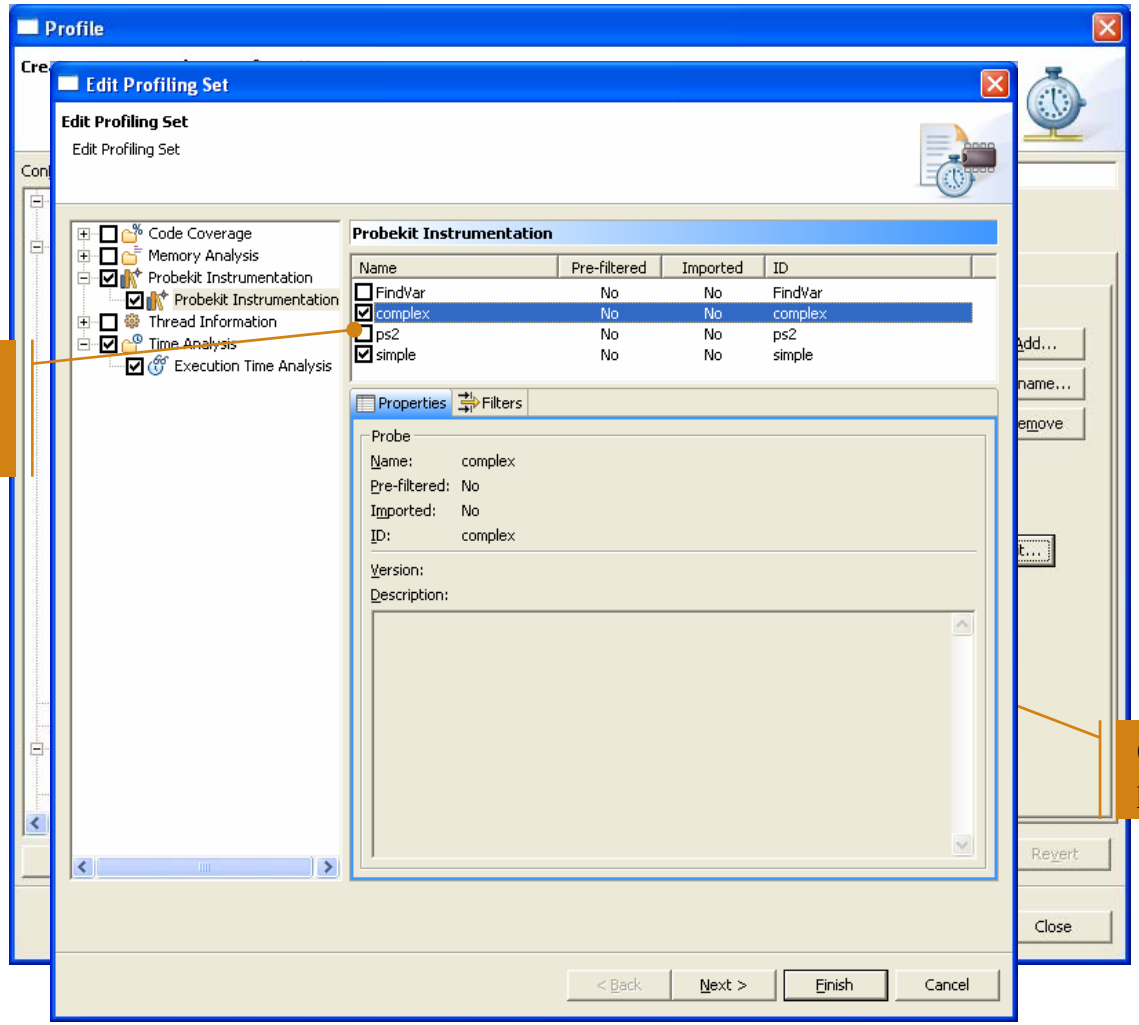
- Probes – snippets of code inserted in your application
  - Extend runtime analysis to fit your needs
  - Access to input/return arguments
  - Define static and thread local objects
  - Ultimate debugger breakpoint
- How?
  - Author a probe
  - Create a Profile Set that includes your newly created probe
  - Profile using that Profiling Set

Author a Probe



Use a Probe

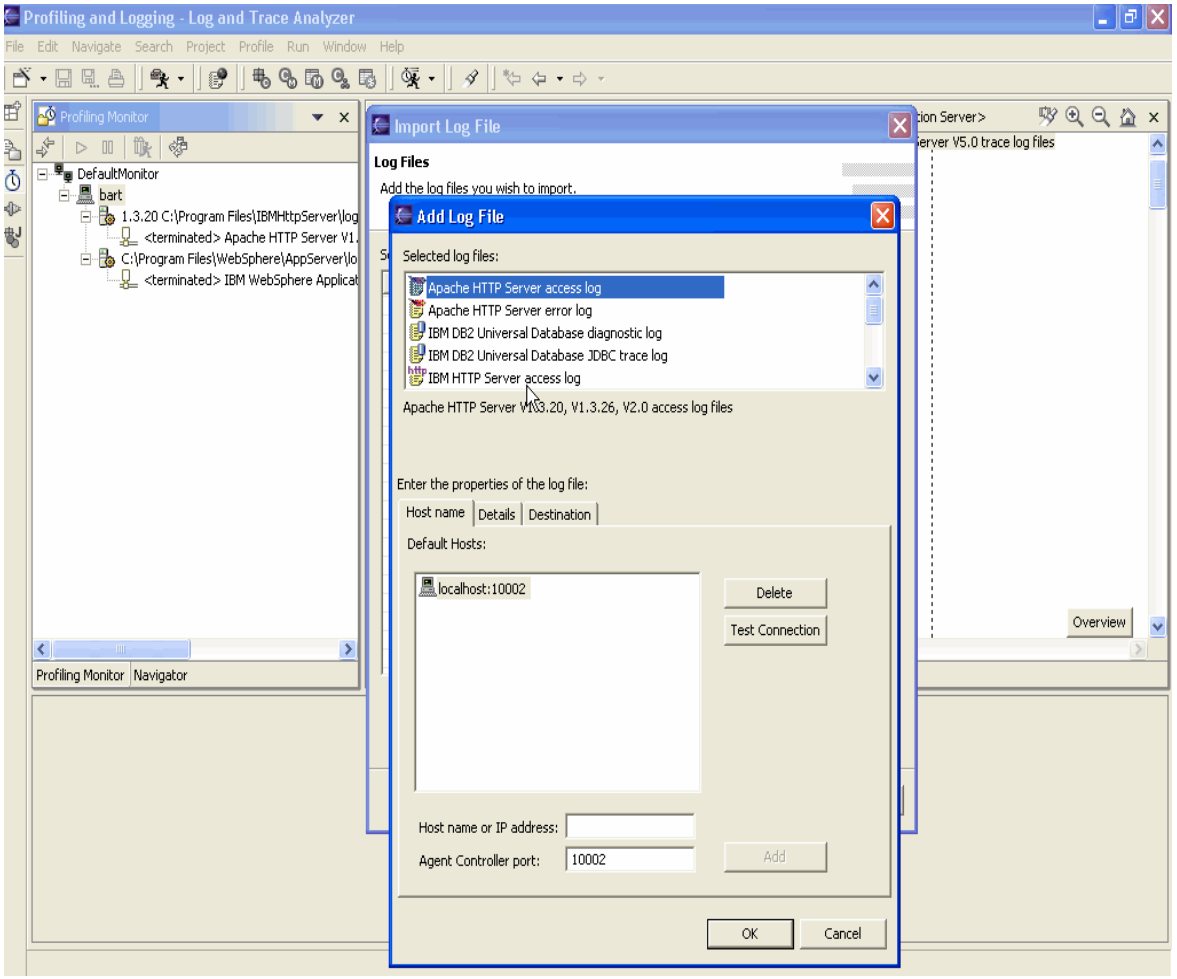
Specify one or more probes as part of your profiling set



Create or edit a profiling set

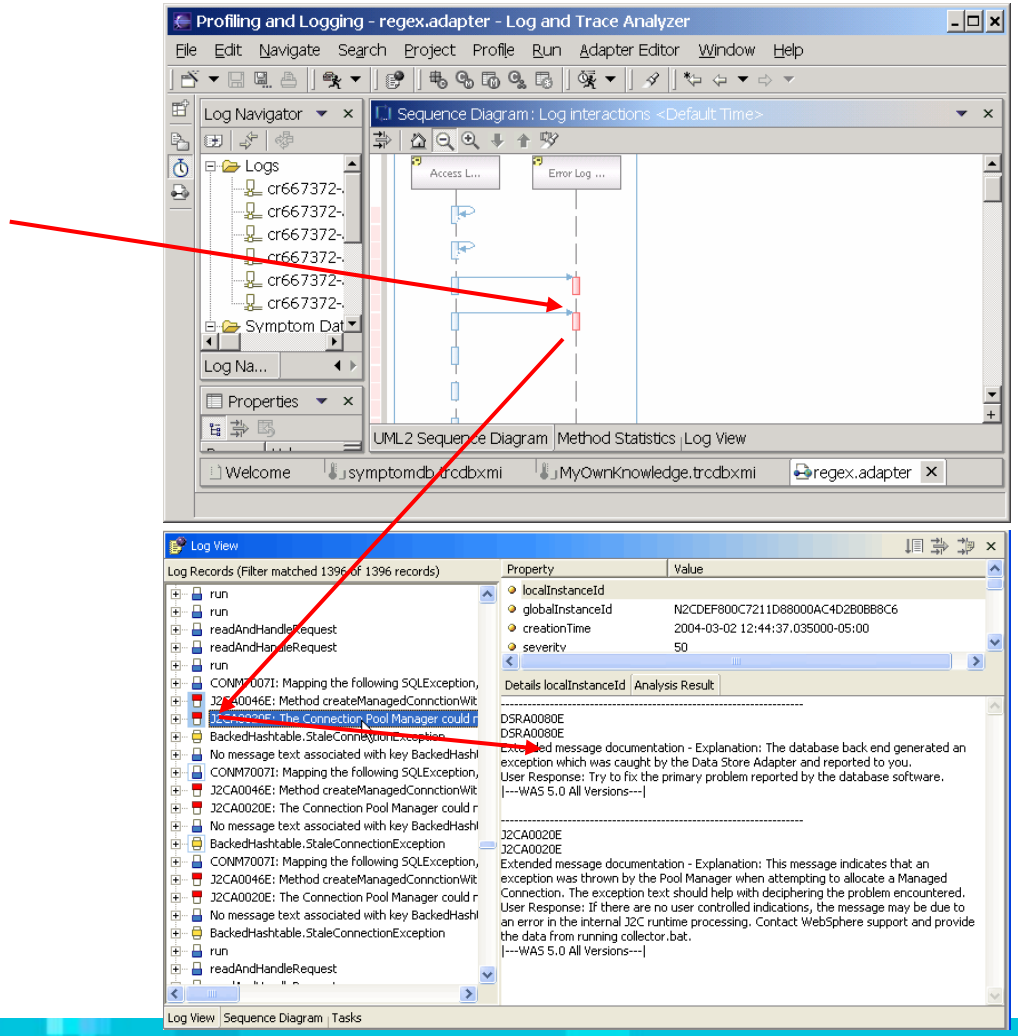
Log and Trace Analysis

- **Generic log adapter to converts existing log files into Common Base Events (CBE) format**
- **Built in parsers: Imports existing log files and converts to CBE format on the fly using the Generic Log Adapter technology**
- **Built in correlation engines: Visually displays the correlation between log records using a number of factors**
  - **Sequential Correlation**
  - **Associative Correlation**

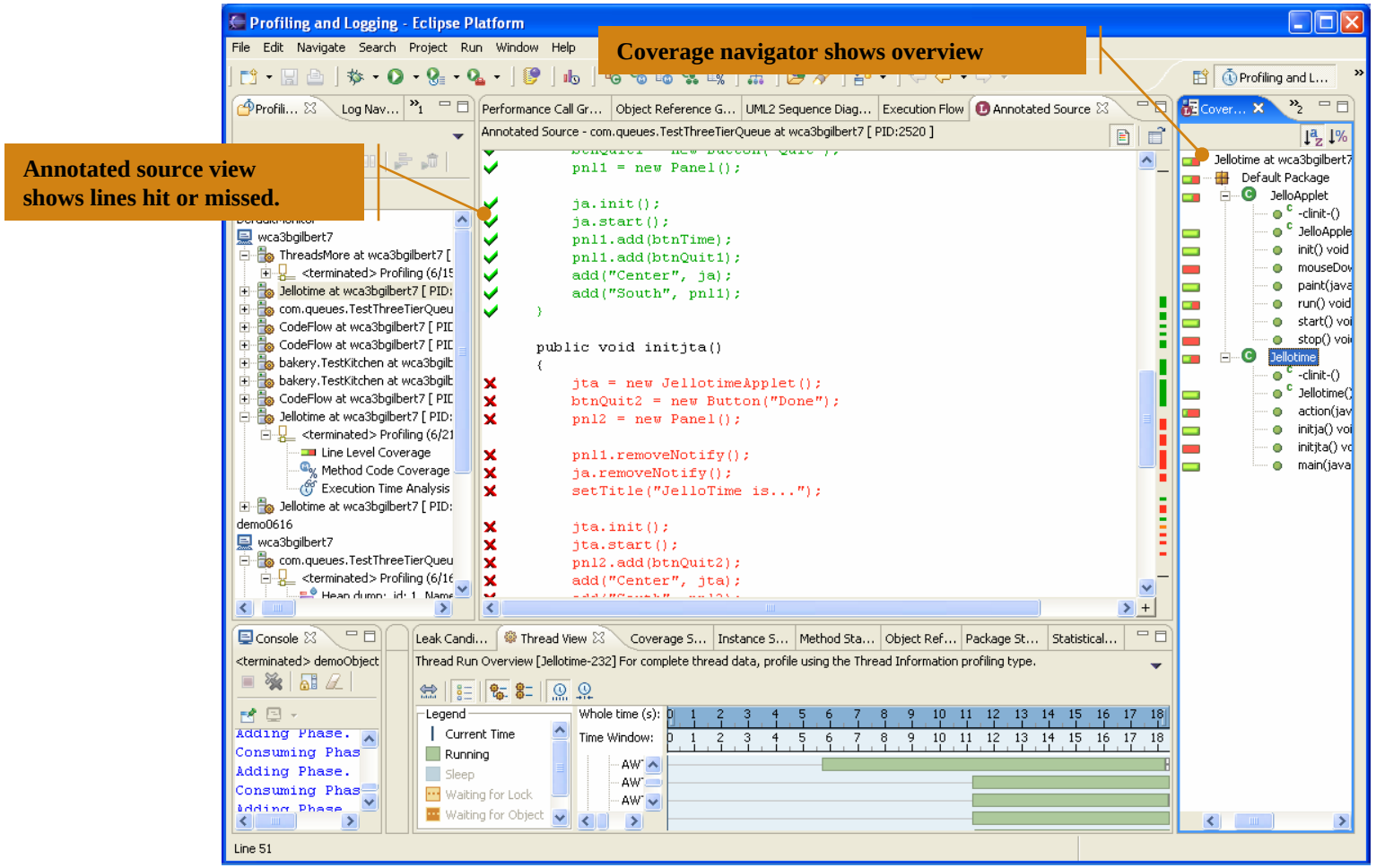


Log and Trace Analysis – Symptom Database

- **Used in the analysis of events and error messages that may occur in a log.**
- **XML file of symptoms, string match patterns, associated solutions, and directives.**



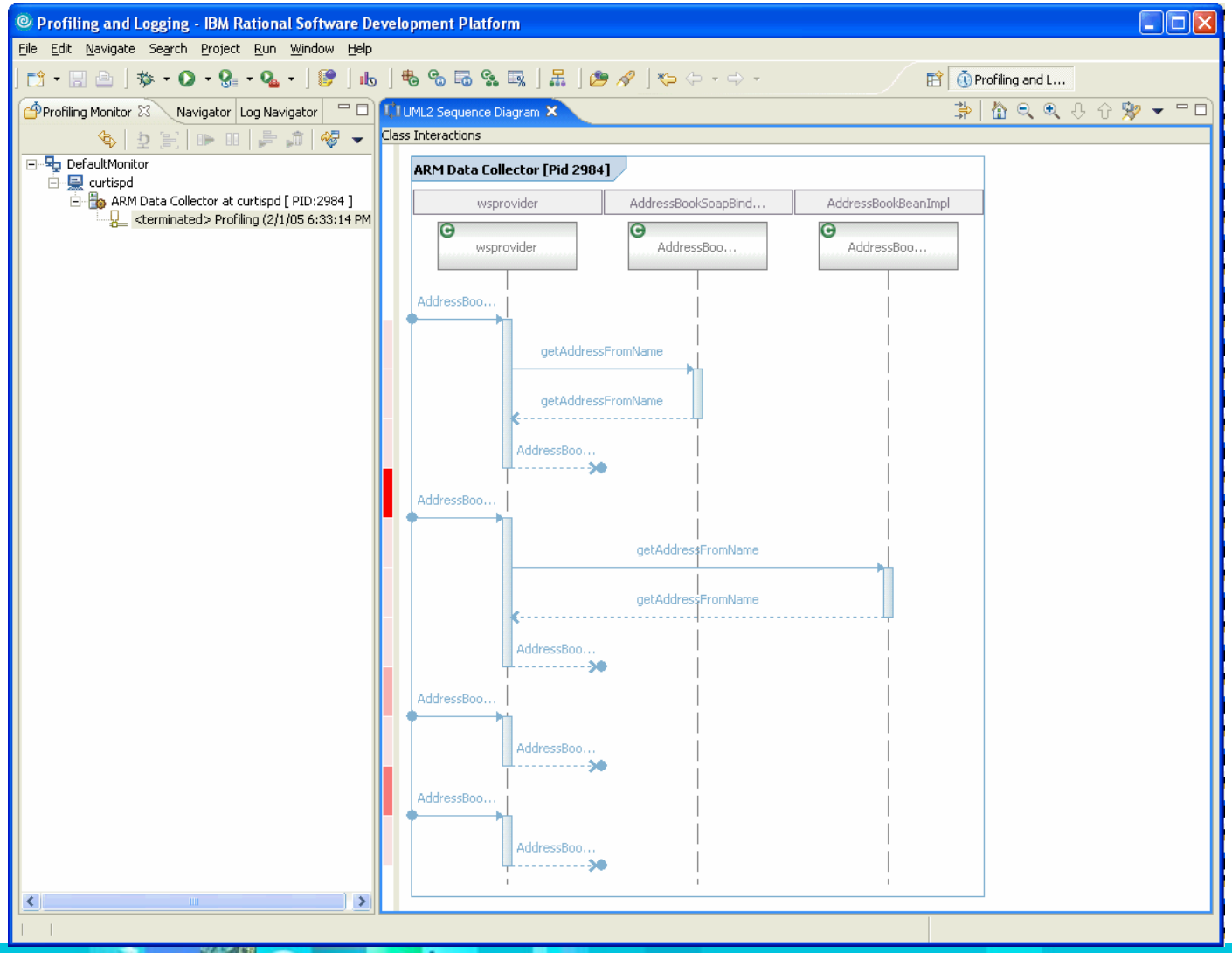
Code Coverage Analysis



Distributed Transaction Analysis

- Import application response measurement (ARM) based data collected in production systems or in test labs while during performance testing
- Analyze distributed transactions
- View them as UML2 class interactions between multiple processes or hosts
- Correlate to log views and system performance

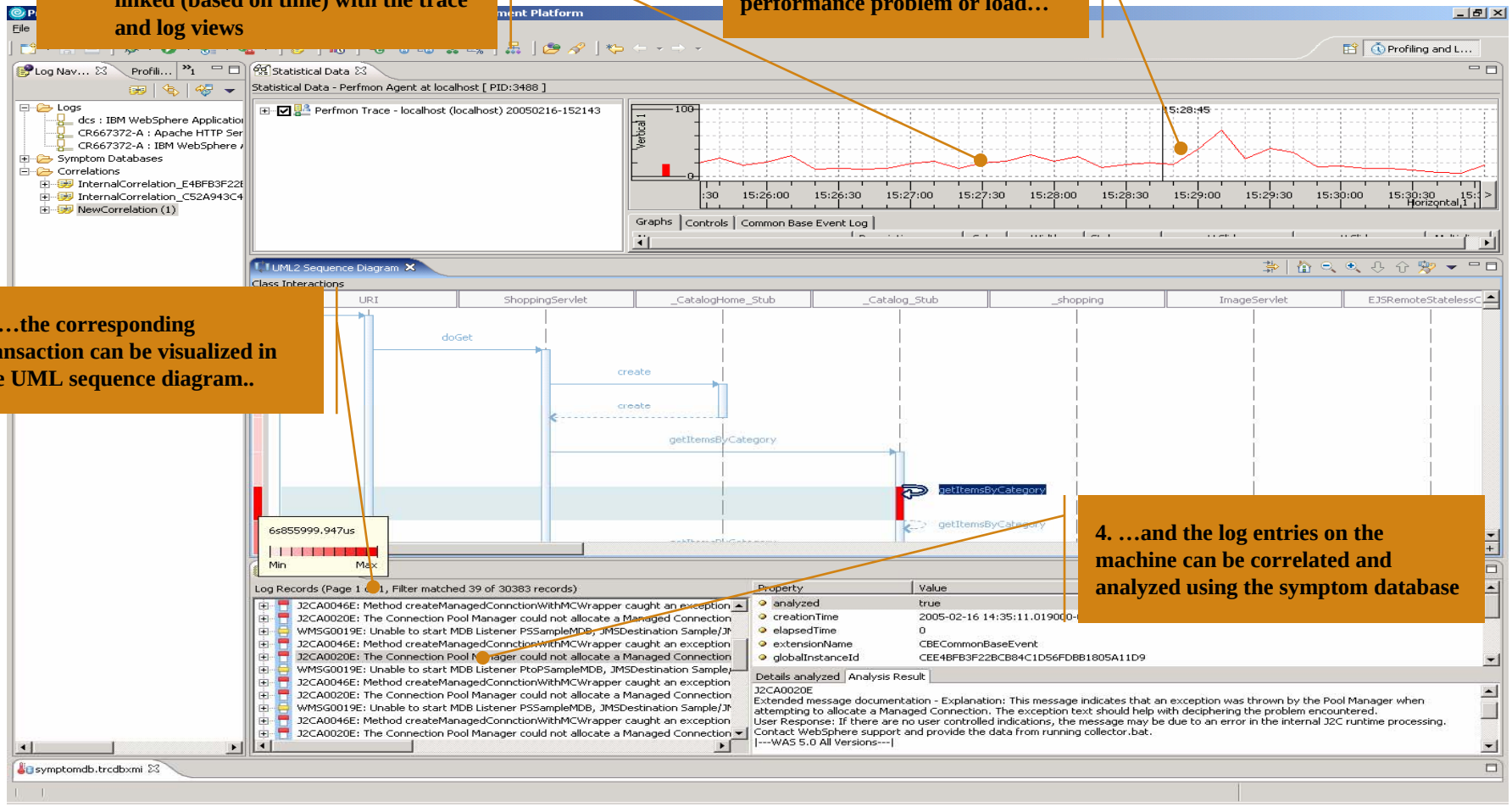
Distributed Transaction Analysis – class interaction between web services



System Analysis and Correlation

1. **The CPU monitoring on the machine where the transaction is running is linked (based on time) with the trace and log views**

2. At the point a user notices a performance problem or load…



3. …the corresponding transaction can be visualized in the UML sequence diagram..

4. …and the log entries on the machine can be correlated and analyzed using the symptom database

Summary

- Testing solutions must provide:
  - Architecture discovery, analysis and control
  - Code review
  - Component Test
  - memory analysis : leak candidates identified
  - threads analysis: thread and lock interactions
  - performance analysis: call graph, execution flow, uml2 trace
  - Probe based analysis: write and deploy multiple probes
  - Log and trace analysis: generic log adapter, symptom database
  - Code coverage analysis: summary view and annotated source
  - Distributed transaction analysis: production or test environments
  - System analysis and correlation: across multiple hosts or processes