VAST : Visualization of Attack Surfaces for Targeting

Ke-Thia Yao, kyao@isi.edu

Information Sciences Institute
Viterbi School of Engineering
University of Southern California

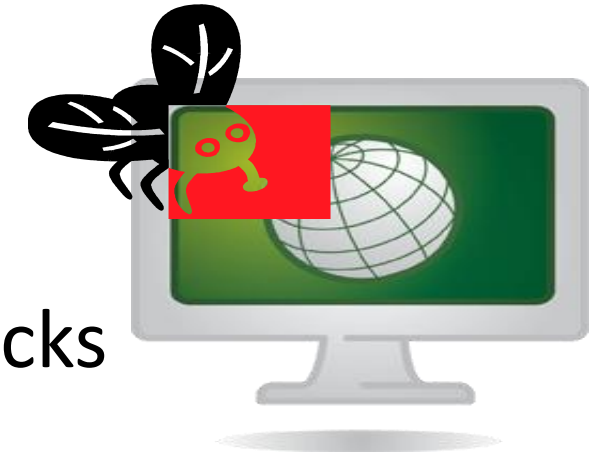
10th Annual High Confidence Software and Systems Conference

Introduction

- VAST was developed over a year at USC ISI funded by NSA SNAC
- Aid analysts in exploring and understanding large unfamiliar C codes
- Interactive Eclipsed-based
- Augmented code browser tools
- New breadcrumb tool to help internalize and learn code
- VAST/Eclipse technology useful for any code browser application

Contact: Dr. Ke-Thia Yao, University of Southern California Information Sciences Institute, kyao@isi.edu, 310-448-8297

Problem



- Software vulnerability exploits continue to plague industry
- Exponential rise in malware attacks
- Need more secure software
- Initially focus on *software vulnerability analysis problem*
  - Automated source code analyzers generate too many false positive and false negatives
  - Eyes looking at the source code is still the best way to discover vulnerability

Objectives

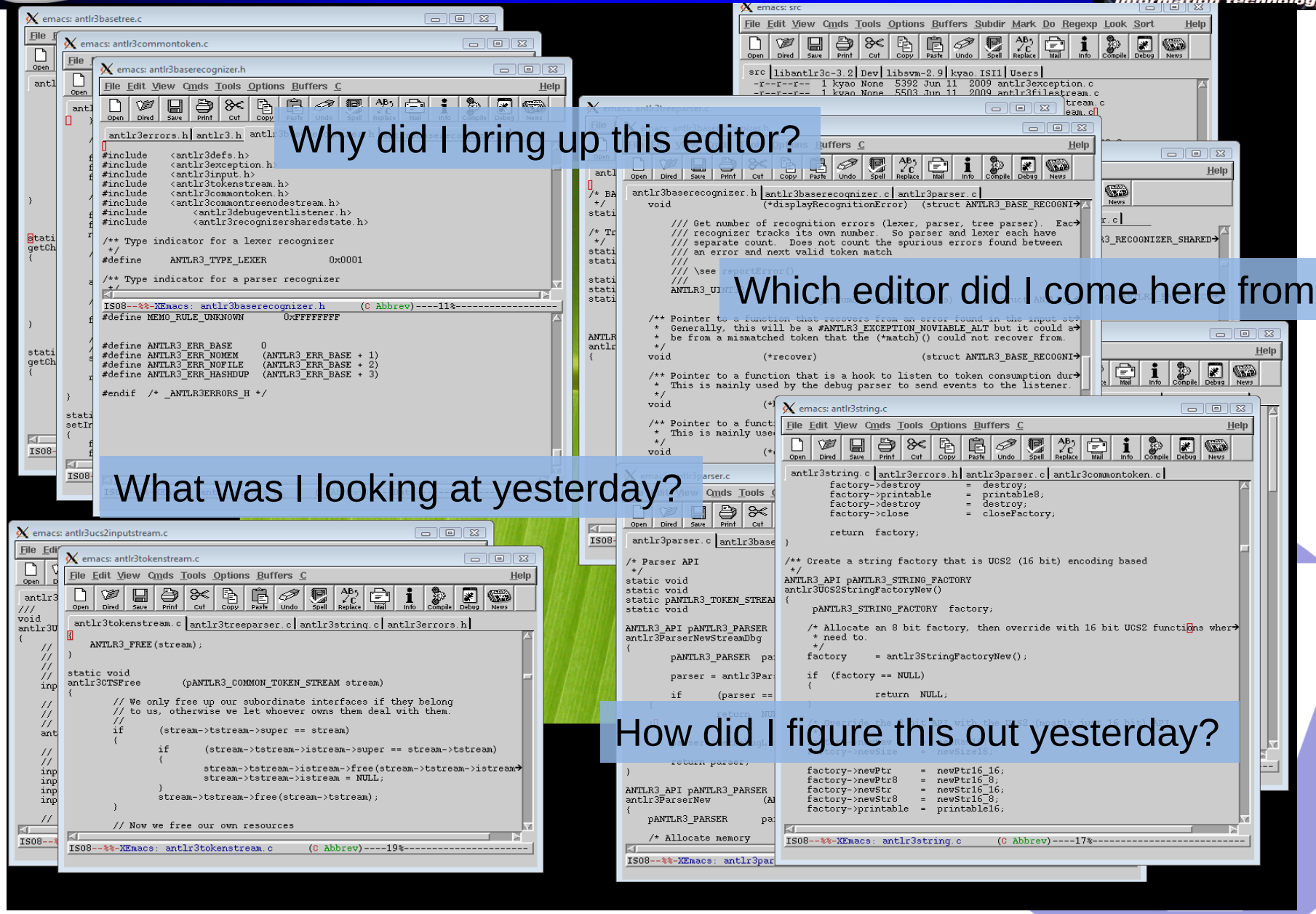
- Help human code auditor teams
  - Reduce time needed to understand large source code bases, 1 to 10 million lines
  - Reduce total time take for a team of code auditors to evaluate software
  - More thorough, and efficient analysis of software vulnerabilities
  - Increase the number of vulnerabilities uncovered during code review process

Secure software through better vulnerability assessment

Code Understanding

- Only human mind can understand code
- Achieving understanding is poorly understood
- High level –learn concepts
- Low level –what does that line of code do
- Usually code browser provides information
- How can browser make the process of understanding code easier and quicker?

Lost in Code


Why did I bring up this editor?
Which editor did I come here from?
What was I looking at yesterday?
How did I figure this out yesterday?

Code Spelunking

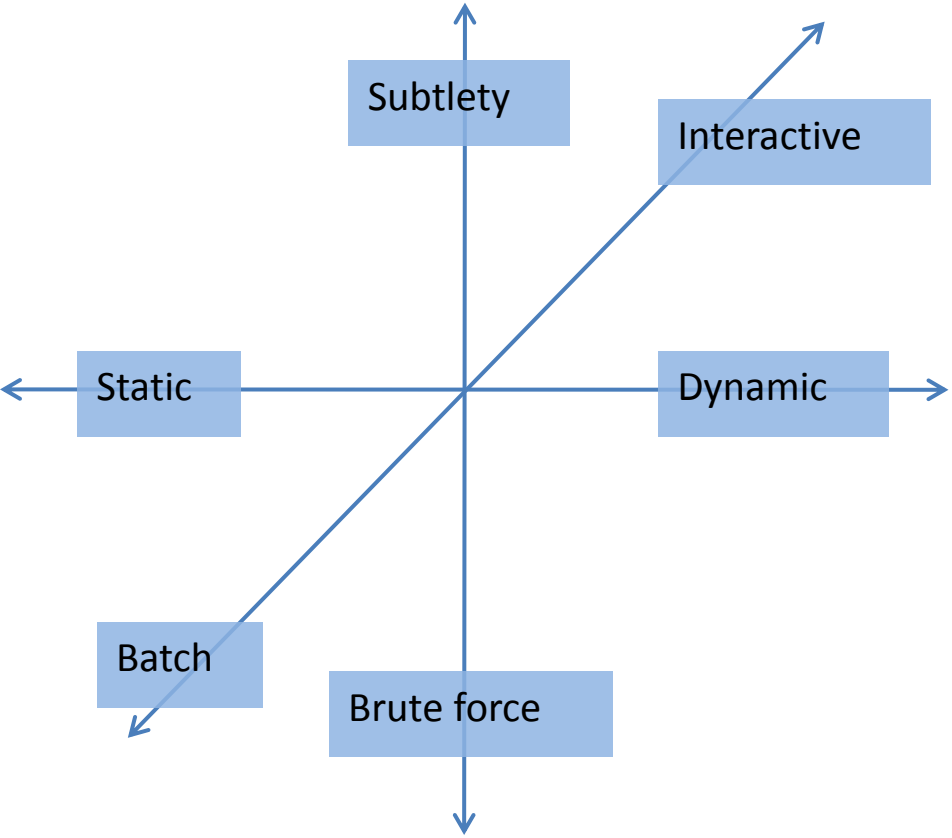
- Exploring cavernous code bases
  - Phrase coined by Neville-Neil
- Size of popular software

Program	Version	Directories	Files	Lines
Apache Web Server	1.3	28	471	158,332
DB	1.4.25	176	234	99,836
Emacs	21	43	2586	1,317,915
FreeBSD Kernel	5.1	420	4758	2,140,517
Linux Kernel	2.4.20-8	642	12,417	5,223,290
Nvi	1.81.5	29	265	77,176
Python	2.2.3	245	1158	356,314

Note: Every attempt was made to disregard documentation or other files not directly related to the source code.

- Types of Spelunking behavior
  - Debugging a program crash
  - Debugging an intermittent error
  - Looking for security vulnerability

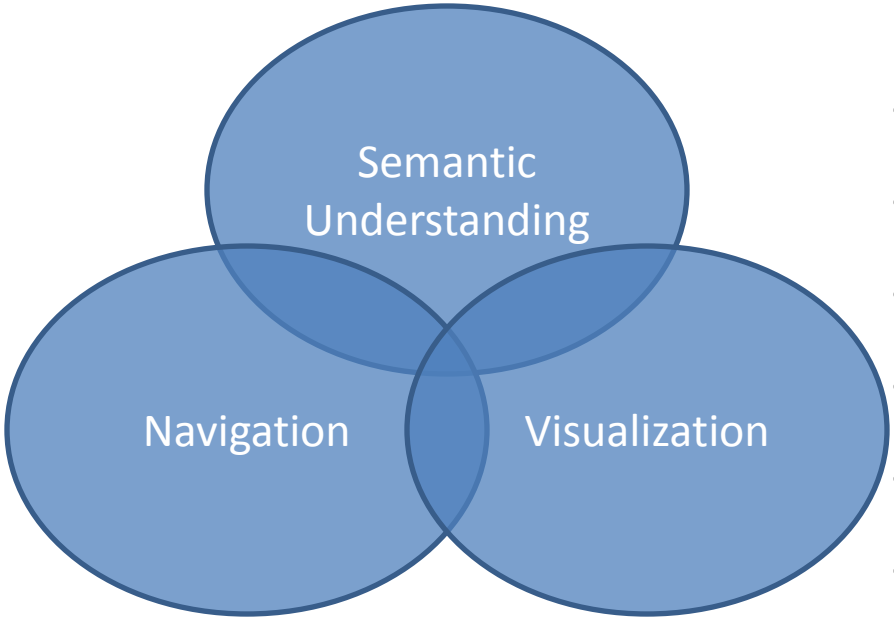
Dimensions for Tools and Techniques



- Static vs. Dynamic analysis
  - Looking at source code versus looking at the runtime behavior
- Brute force vs Subtlety
  - Using text-based tools (find, grep) vs. tools that under stand language semantics (cscope, cdt)
- Batch vs. Interactive
  - Using dumps and printfs vs. debuggers and code browsers

Approach

- Human code auditors need **navigational** aids to support exploration of large code bases
  - Code assessment oriented **user interfaces** as **cognitive aid** in semantic understanding

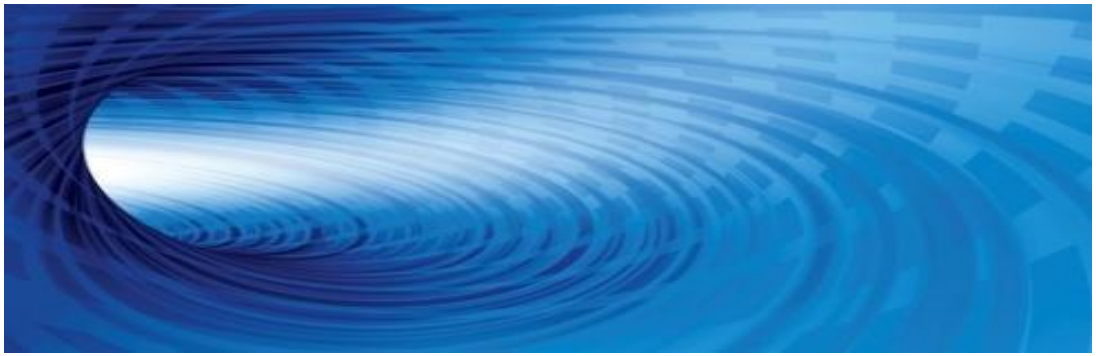


Principles/Guidelines

- Consistency of view
- Hiding irrelevant code
- Manage screen real estate
- Assist navigation of control flow
- Allow navigation by graph
- …

Approach (2)

- Need better ways to focus exploration to likely vulnerable code locations
  - **Attack Surfaces**: exploits tend to follow ***patterns*** directed at specific ***areas***
  - These areas tend to at the intersection of I/O, network, memory
  - Rank importance of attack area

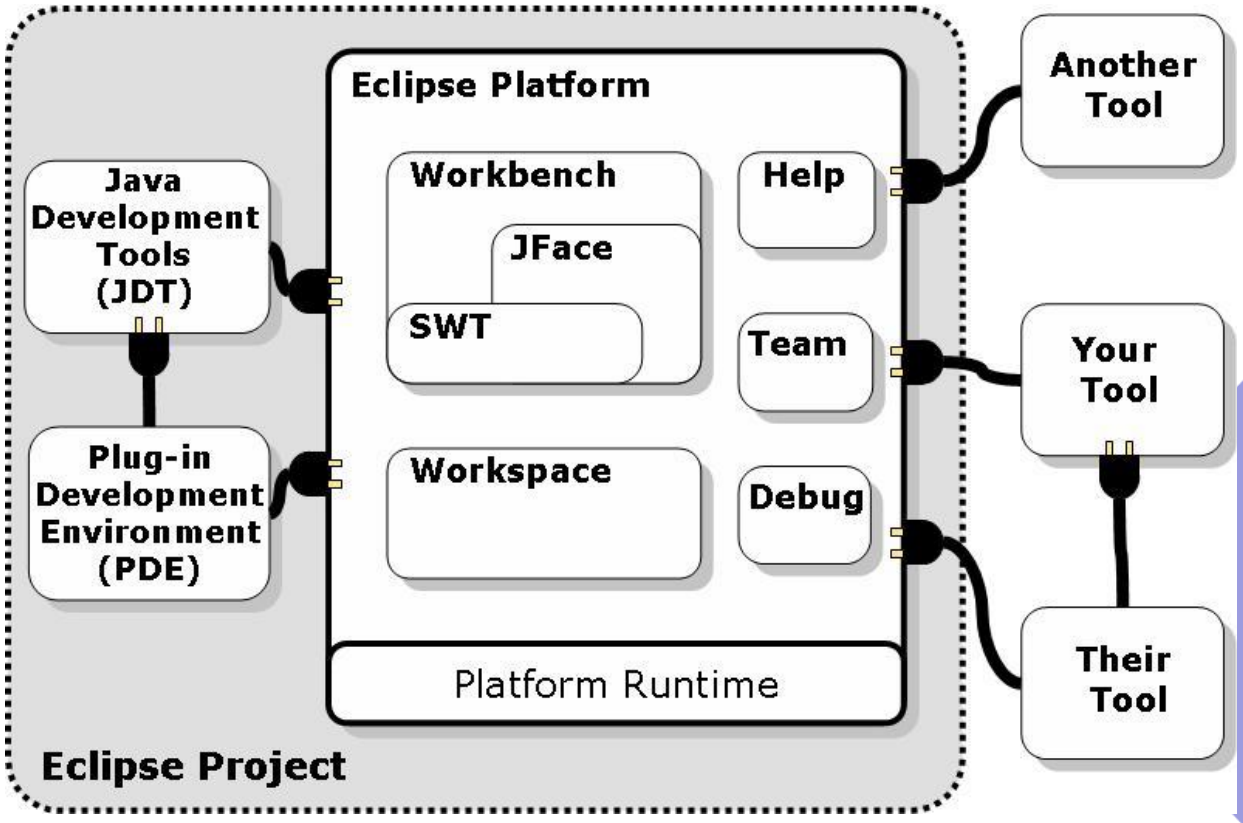


Implementation Criteria

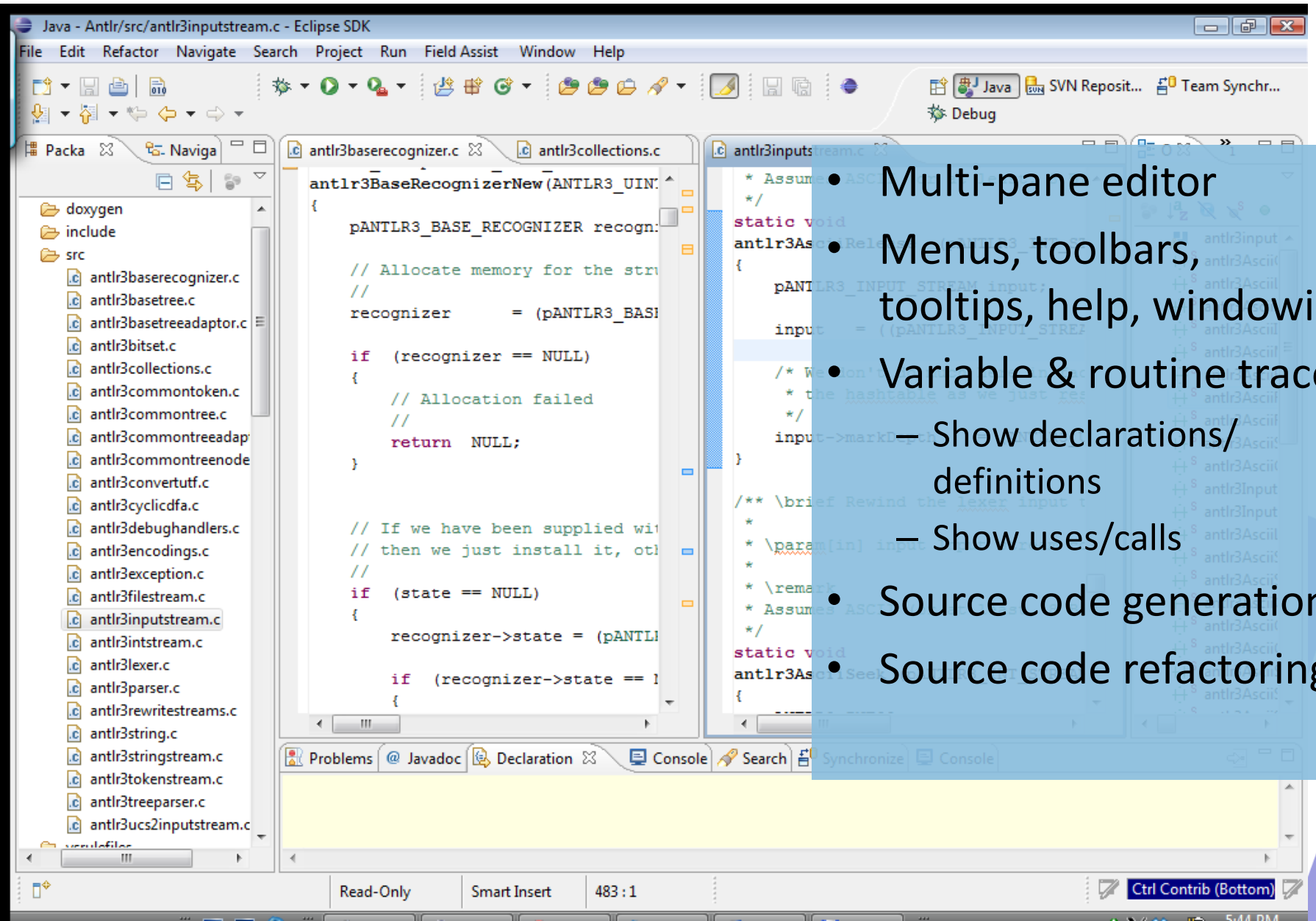
- Open extensible framework
- Develop VAST as an Eclipse plug-in
- Eclipse is free and open source software development environment
- Advantages
  - Leverage existing popular integrated development environment (IDE)
  - Multi-language: Java, C/C++, Python, Fortran
  - Plug-in system that allows additional capabilities to be added in a modular fashion
- But, significant learning curve

Eclipse Plugin Framework

- Lightweight software component framework
- Allows components to be easily customized
- Allows alternative component implementation



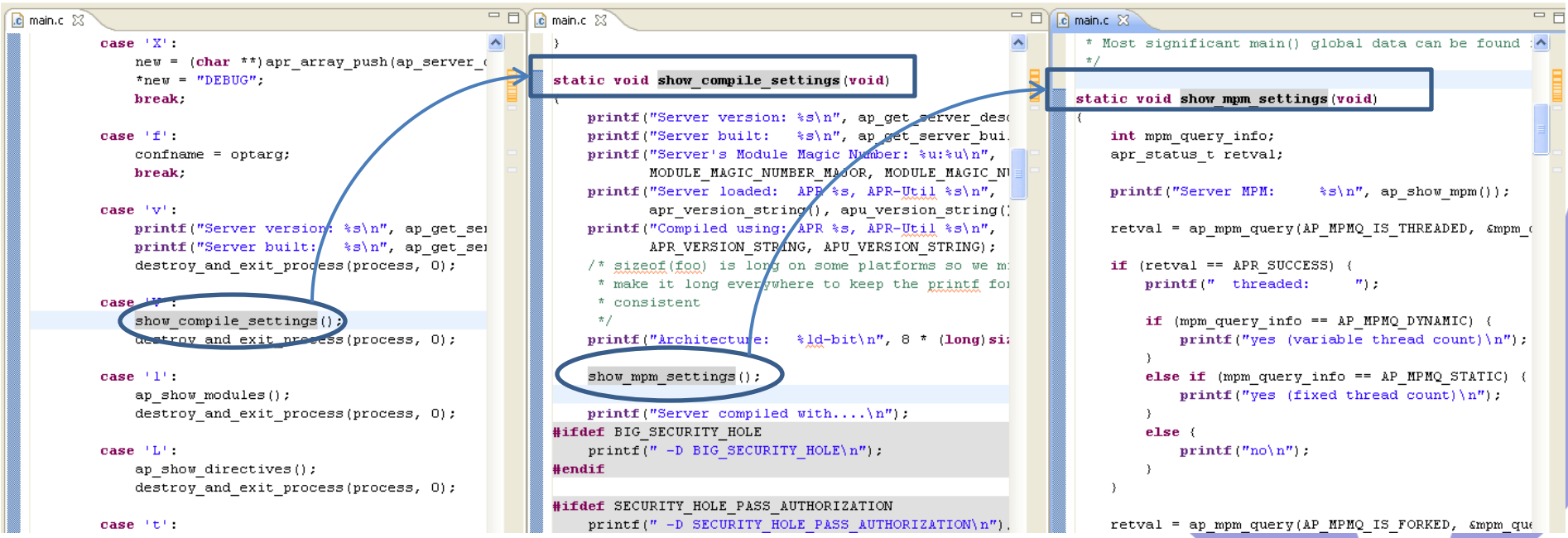
Eclipse Standard Tools



- Multi-pane editor
- Menus, toolbars, tooltips, help, windowing
- Variable & routine trace
  - Show declarations/ definitions
  - Show uses/calls
- Source code generation
- Source code refactoring

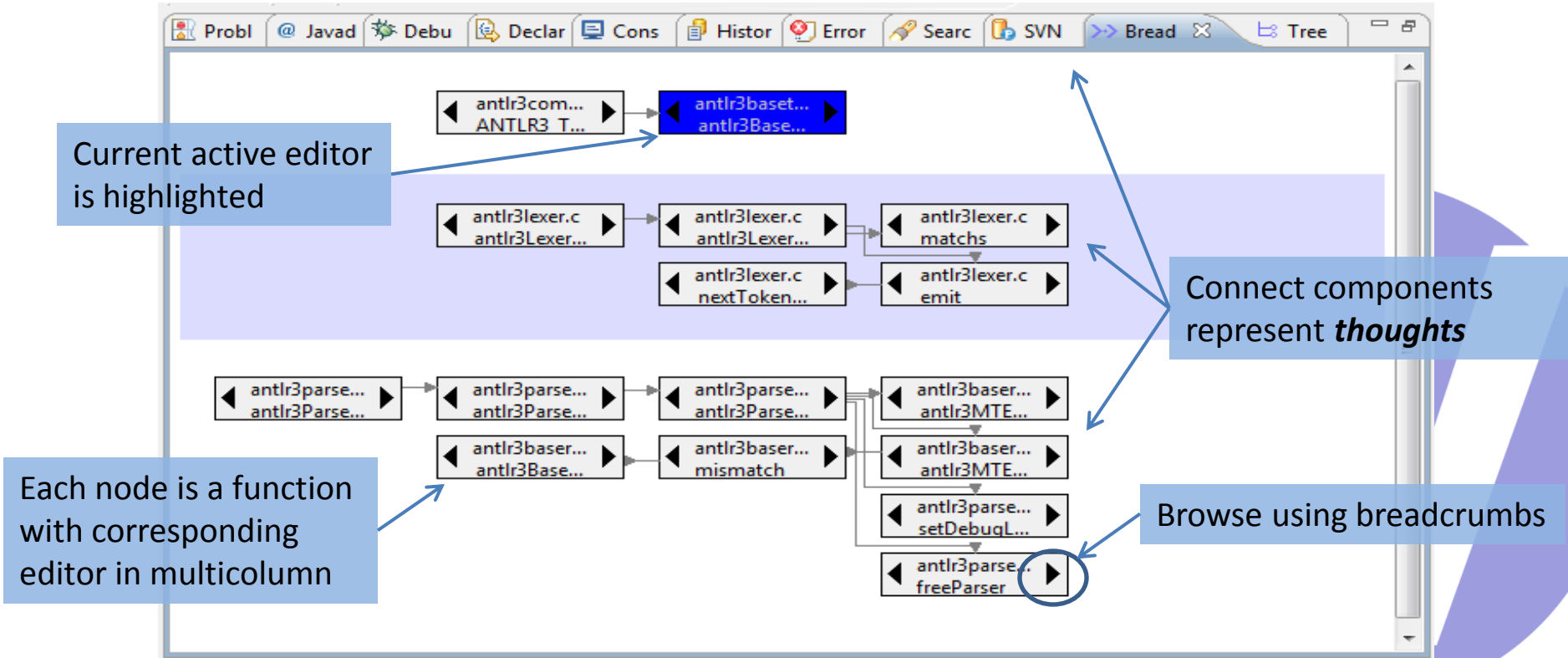
Multicolumn Editor

- Enables call trace oriented code browsing
- Opens editor from left to right
- Each *open declaration* creates a new column
- Allows the same file to opened multiple times
- Extends Eclipse's StackedPresentation class



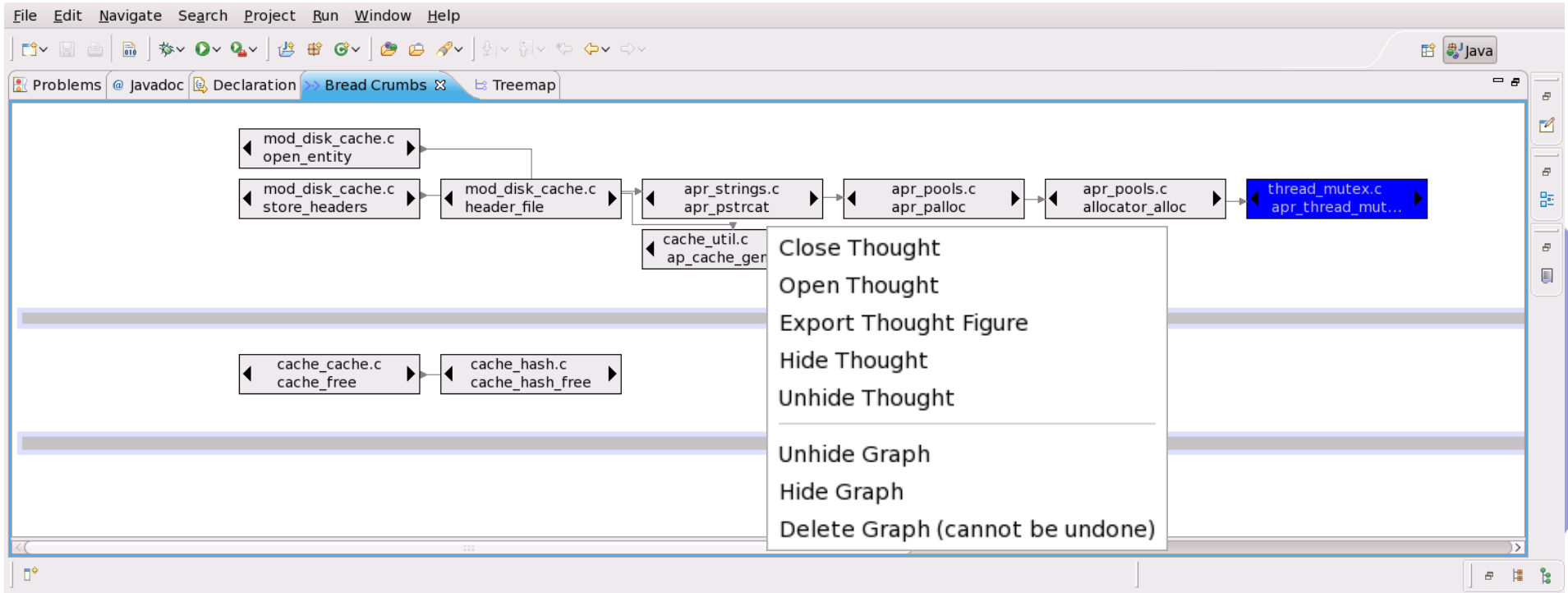
Breadcrumb View

- Helps users remember code exploration path
- Augments user's mental model, *thoughts*
- Graph layout aligns with multicolumn editor
- Implements Eclipse View using Draw2D API



Breadcrumbs (2)

- Higher level features to manage breadcrumb trails
  - Open/close code browser associated with thought
  - Export thought a picture figure
  - Hide/Unhide thought in breadcrumb view

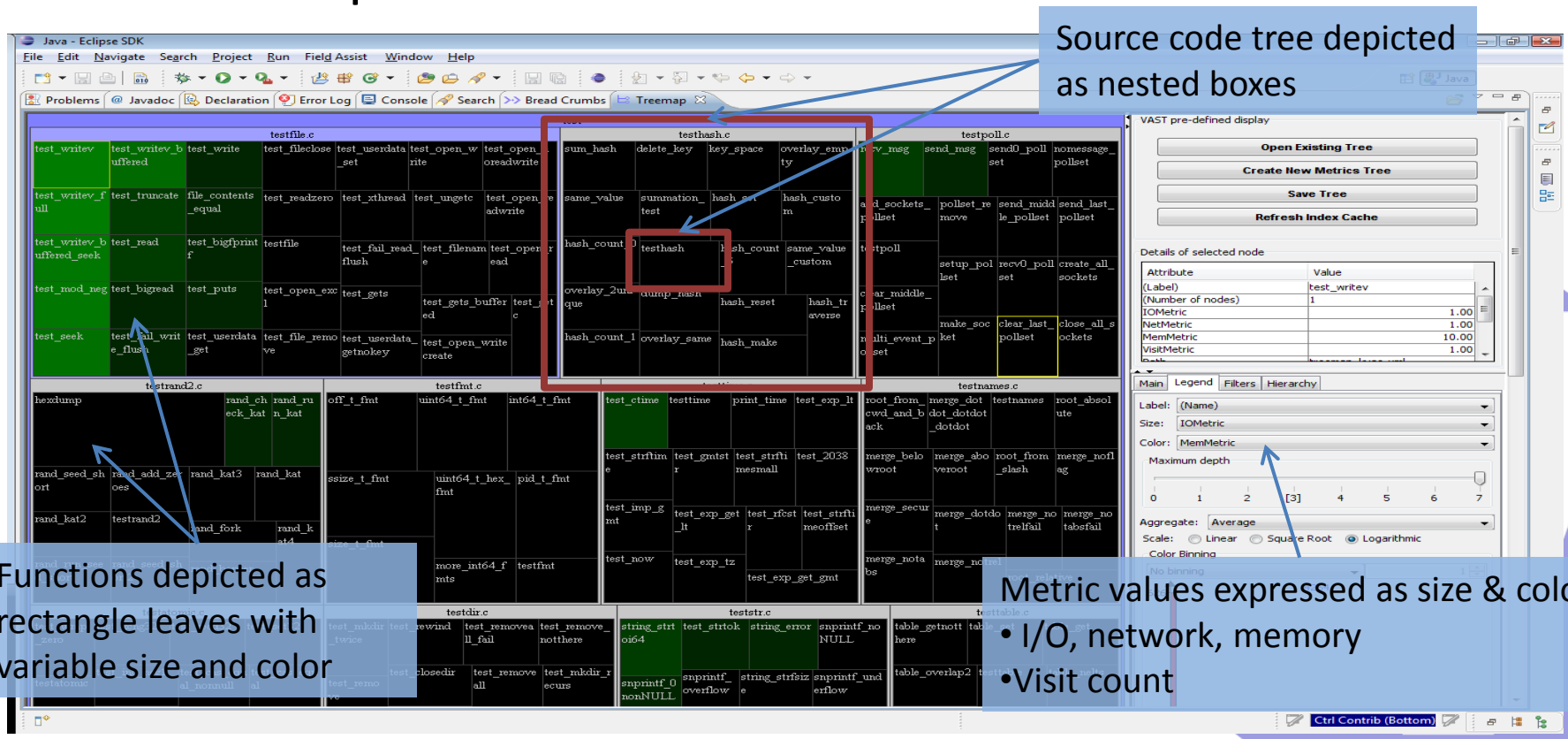


System Attack Surface

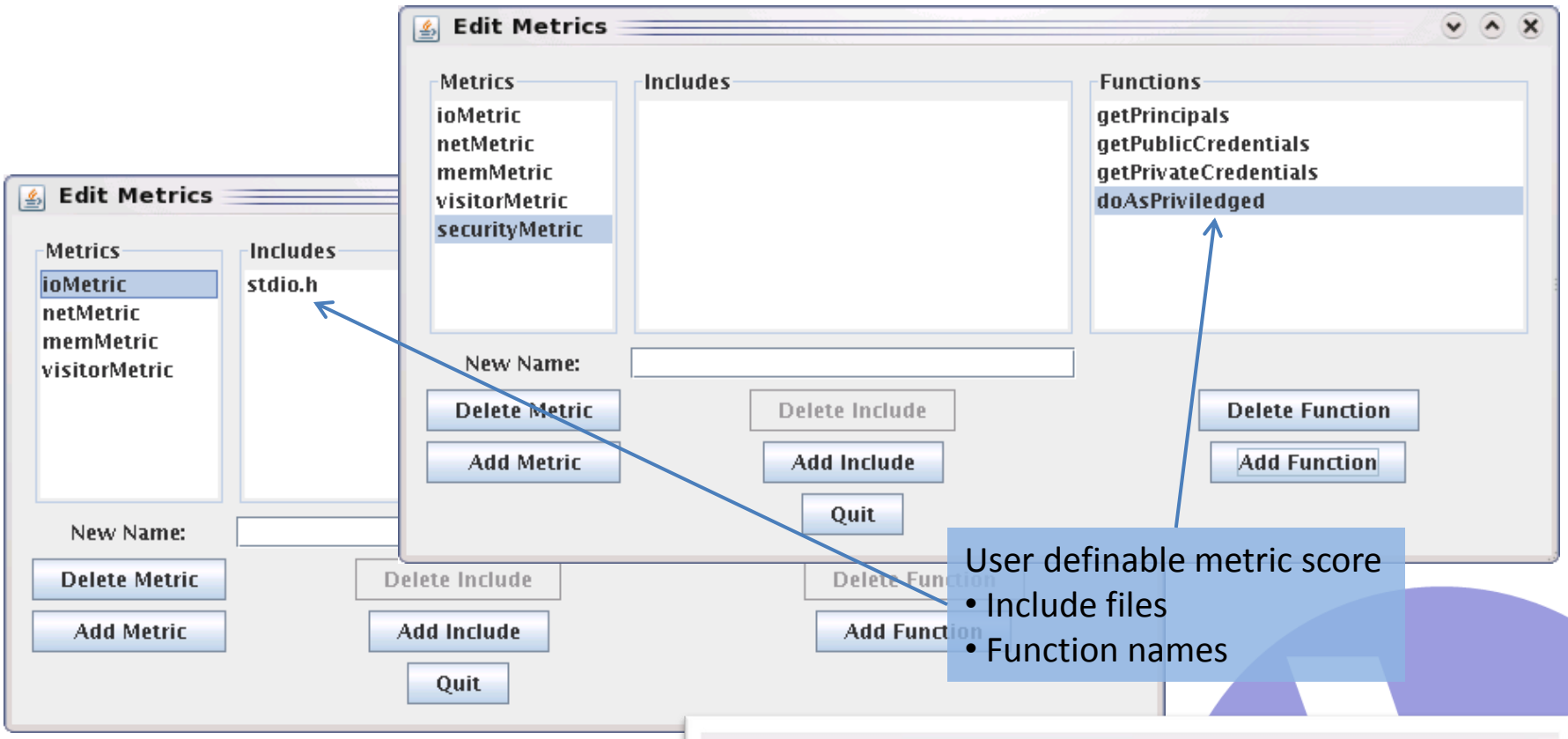
- Indicator of system vulnerability
  - Large attack surface, then more vulnerability
- Attack surface metric
  - Entry/exit points, such as API calls
  - Channels, such as sockets and RPC
  - Unsecured data, such as user input and files
  - Damage potential, such privileged and security calls
- Measures the entire system
  - E.g, numeric sum of damage over all points, channels and unsecured data

Code Surveyor View

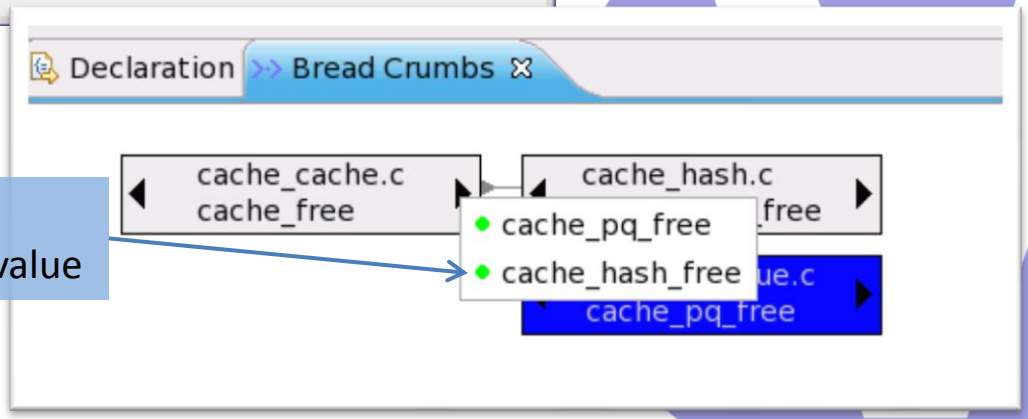
- Provides source code overview in context of software vulnerability assessment
  - Where are the attack surfaces
  - Which part of the code has been audited



Code Surveyor (2)

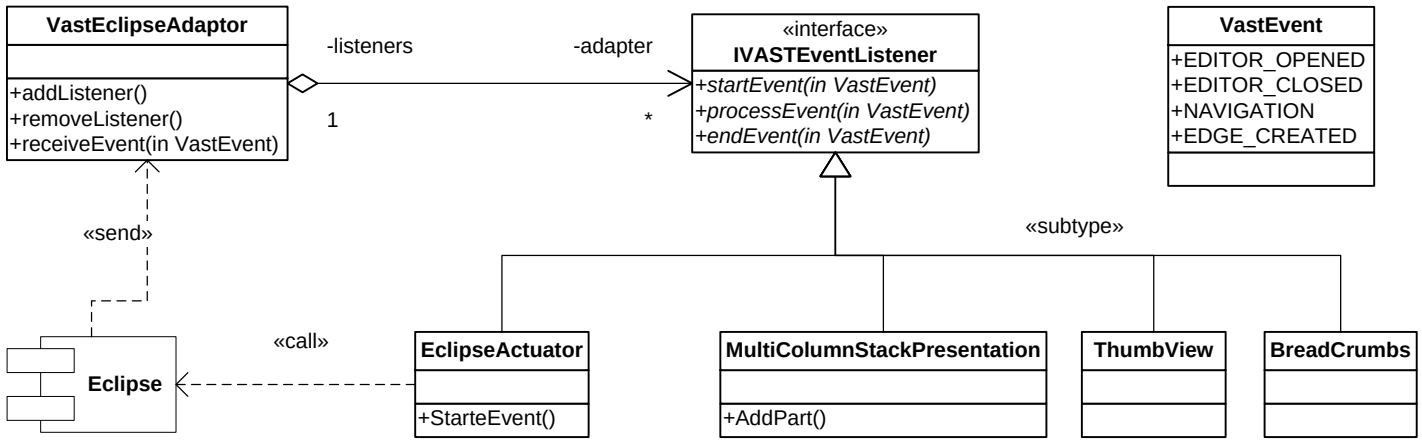


User definable metric score
- Include files
- Function names



Attack surface driven browsing
- Function color-coded by metric value

VAST Architecture



- VastEclipseAdaptor
  - Listens to Eclipse system events
  - Sends VastEvents to registered IVastListeners
  - Each VastEvent triggers 3 calls
    - startEvent, processEvent, endEvent
- EclipseActuator
  - Responsible for calling Eclipse functions

- IVASTEventListener
  - All VAST View should implement this interface to receive VastEvents
  - VAST View should NOT have
    - To listen to Eclipse system directly
    - To call to Eclipse system directly

Extending to Other Languages

- Developed for C, but was able to adapt to VAST to Java quickly
- Next step is FORTRAN
- *Photran* plugin available for Eclipse
- Potential attacks on Common block and integer overflow

Parallelization

- Often requires extensive code inspection
- Often requires extensive code modification
- Best done at a high level
- Largely unsolved except by *heroic programmer*
- Modern parallel hardware wasted without better *mass parallelization* tool

Better Toolbox

- Develop sandbox framework/server with analysis tool ensemble to help organizations quickly ingest new software
  - Need a variety of tools for breadth and depth
  - Provide all three spelunking tools dimensions
  - Integrate source code and runtime analysis tools
  - Environment to run and test new software

Contact: Dr. Ke-Thia Yao, University of Southern California Information Sciences Institute, kyao@isi.edu, 310-448-8297

Deeper Attack Surface Analysis

- Currently, attack surface metrics are based on syntactic API-level analysis of source code
- Take advantage of catalog of software exploits and attack patterns
  - *Exploiting Software: How to Break Code* by Hoglund & McGraw
  - OWASP lists 160 Web-related vulnerabilities
  - Error prune constructs, coding standards/styles
- Apply data mining and approximate pattern matching to find vulnerable regions

Contact: Dr. Ke-Thia Yao, University of Southern California Information Sciences Institute, kyao@isi.edu, 310-448-8297