Software Visualization

Margaret-Anne Storey
Dept. of Computer Science, UVic

Outline

- Mental imagery and software visualization
- Price et al's taxonomy on software visualization
- Cognitive questions in software visualization
- A cognitive framework for software exploration

When you are designing a (or understanding an existing) program, describe how you go about doing such a task?

Mental imagery and software visualization

- Some quotes:
  - *"The first step in programming is imagining - I like to imagine the structures that are being maintained, the structures that represent the reality I want to code ... The code for the most part writes itself, but it's the data structures I maintain that are the key. They come first and I keep them in my mind throughout the entire process." (Charles Simonyi, page 15)*
  - *"You have to simulate in your mind how the program's going to work, and you have to have a complete grasp of how the various pieces of the program work together." (Bill Gates, page 73)*
  - *"One of the earliest things is to visualize this structure in my head, a dynamic structure, so I can think about how things fit together and how they work ... and once I have the structure fairly strong and clear in my mind, I move it around and move around inside it, examining it and tweaking it ... "*
- 2 key elements:
  - Structure of information
  - How it works

Price's taxonomy on software visualization

- Scope
- Content
- Form
- Method
- Interaction
- Effectiveness

Scope

What is the range of programs that the SV system may task as input for visualization

- Two main concerns:
  - Generality
    - Hardware
    - Operating system
    - Language (Concurrency)
    - Applications
  - Scalability
    - Program
    - Data sets

Content

What subset of information about the software is visualized by the SV system?

- Program
  - Code (control flow)
  - Data (data flow)
- Algorithm
  - Instructions (control flow)
  - Data (data flow)
- Fidelity and Completeness
  - Invasiveness
- Data Gathering time
  - Temporal control mappings
  - Generation time

Form

What are the characteristics of the output of the system (the visualization)?

- Medium
- Presentation style
  - Graphical vocabulary
  - Animation
  - Sound
- Granularity
  - Elision
- Multiple views
- Program synchronization

Method

How is the visualization specified?

- Visualization specification style
  - Intelligence
  - Tailorability (customization language)
- Connection technique
  - Code ignorance allowance
  - System code coupling

Interaction

How does the user of the SV system interact with and control it?

- Style
- Navigation
  - Elision control
  - Temporal control (direction, speed)
- Scripting facilities

Effectiveness

How well does the system communicate information to the user?

- Purpose
- Appropriateness and clarity
- Empirical evaluation
- Production use

Cognitive questions in software visualization

- Isn’t enough to mimic paper based tasks….
- “Even simple tools may improve software visualization – if they are the right ones”

Cognitive questions in software visualization:

1. What is visualization useful for?
   - Presenting large data sets
   - Demonstrating the virtual machine
   - Changing the perspective
   - Display-based reasoning

Cognitive questions cont.

2. Does visualization mean pictures?
3. Is SV a way into `the expert mind' or a way out of our usual world view?
4. Why are experts often resistant to other people's visualisations?
5. Are visualizations trying to provide a representation that is more abstract, or more concrete?
6. What model are we representing?
7. What kind of tasks are we supporting?
8. What do we know about perception, anyway?
9. What if there aren't enough dimensions?
10. Are representations good for everyone? What is the importance of individual skill and variation?
11. When are two representations better than one? …
14. Can I take a version to bed?

Cognitive Design Elements to Support the Construction of a Mental Model during Software Visualization

Software Visualization research overview

How do programmers understand programs?

What tools do programmers need to understand programs?

A Cognitive Framework of Design Elements to guide tool design

Evaluate tool usefulness and usability

SHriMP Views

Terminology

- A *mental model* describes the maintainer's mental representation of the program to be understood
- A *cognitive model* describes the processes and information structures used to form the mental model

Cognitive Models of Program Comprehension

- Bottom-up comprehension
- Top-down comprehension
- Knowledge based understanding model
- Systematic and as-needed strategies
- Integrated meta-model of program comprehension

Explaining the variation in models

- Maintainer characteristics
  - application/program knowledge
  - maintainer experience, creativity
- Program characteristics
  - application/programming domain
  - size, complexity, quality, documentation
- Task characteristics
  - adaptive, perfective, corrective, reuse
  - CASE tool, time constraints

Program comprehension

- Source code is often the only source of information for understanding programs
- *Reverse engineering* describes the extraction of high-level design information from source code
- Tilley *et al.* WPC'96
  - Data gathering
  - Knowledge organization
  - *Information exploration*

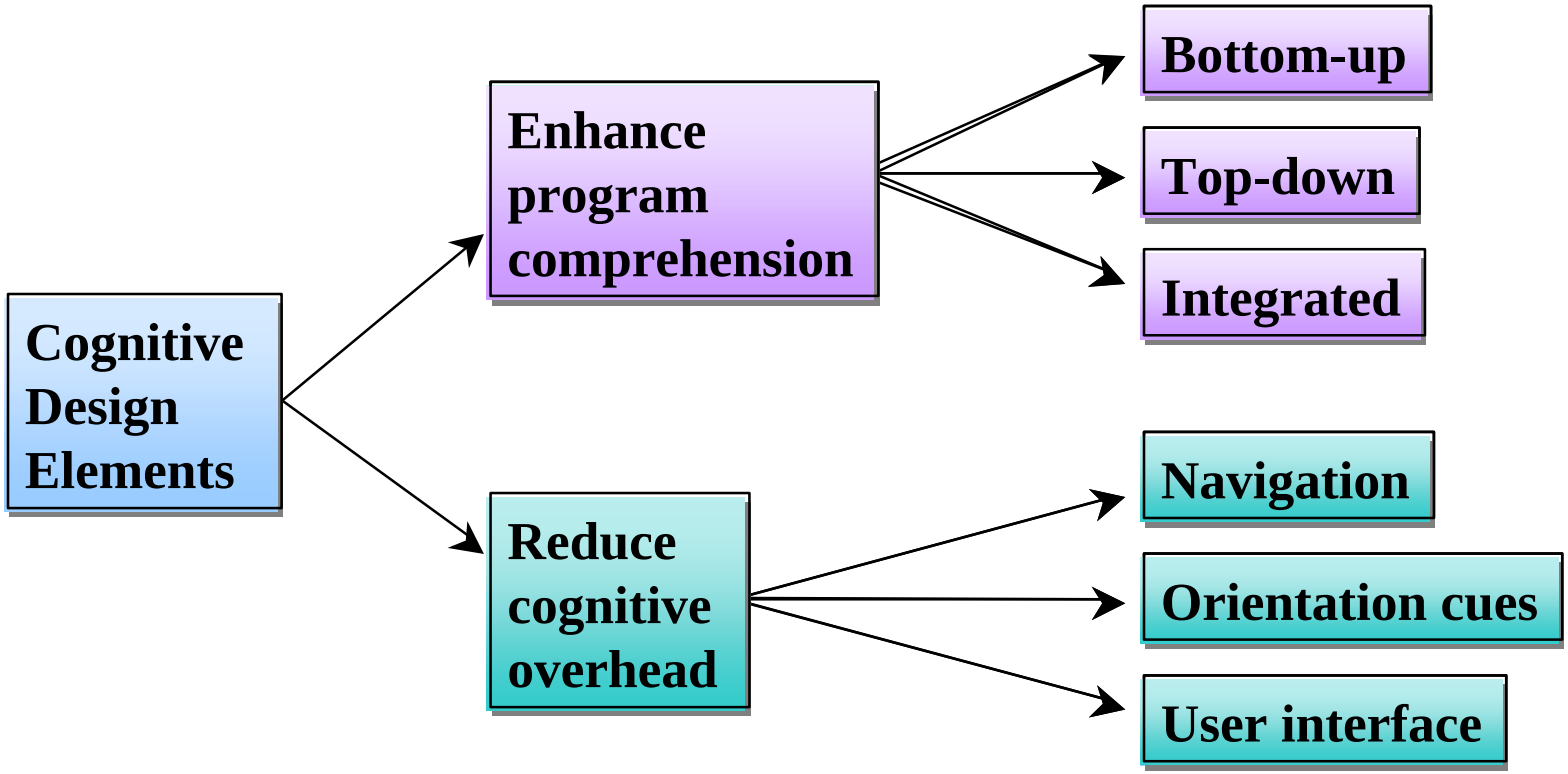
Software visualization tools

- algorithm animations
- visual debuggers
- dynamic visualizations
- pretty printers
- *exploring static software structures*

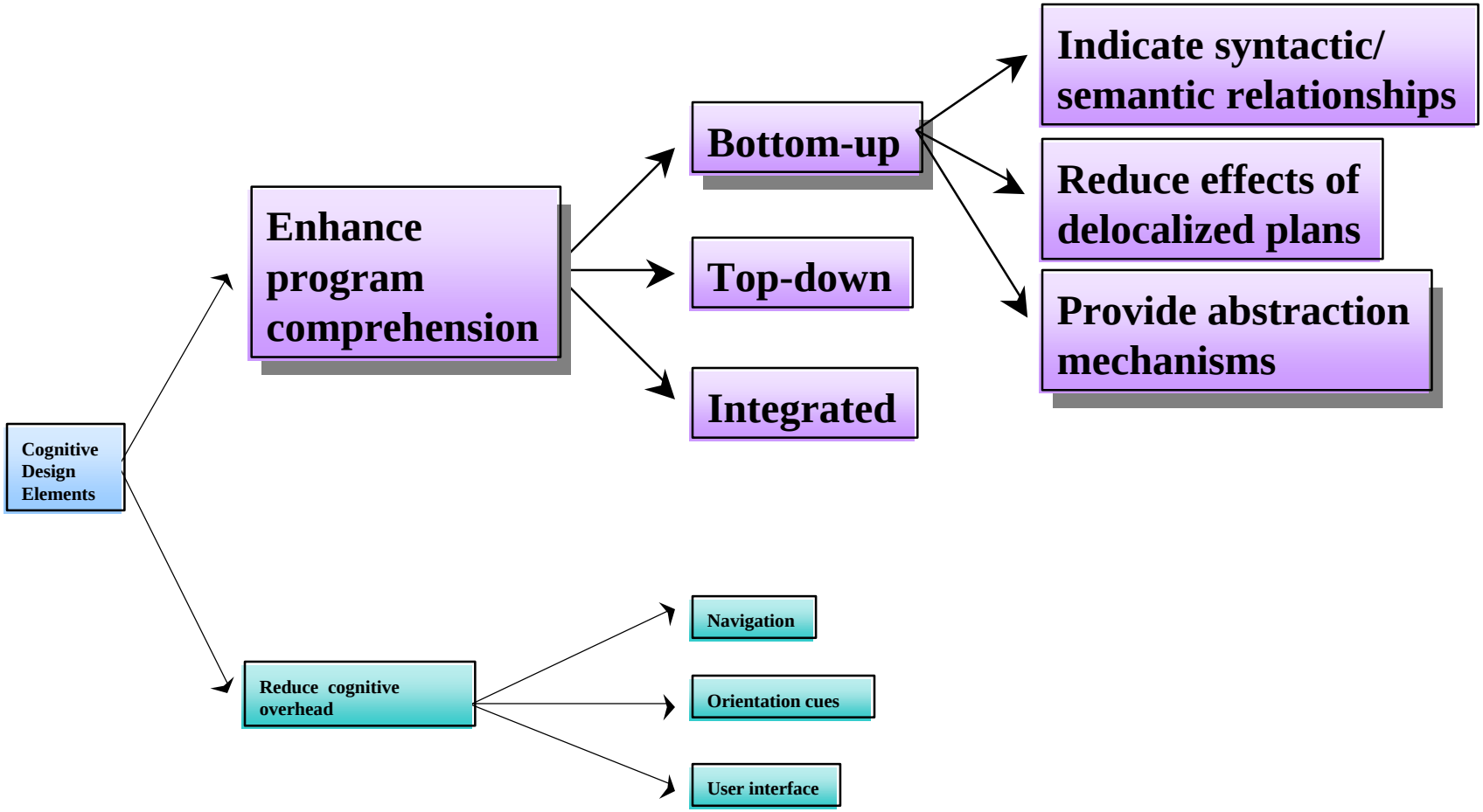
A Hierarchy of Cognitive Design Issues for Software Exploration

- Software information often has *web*-like structures
- Several hypertext browsers for source code
- Comprehension of a hyperdocument involves the *"construction of a mental model that represents the objects and semantic relations in a text"*
  - Increase coherence (local and global)
  - Reduce cognitive overhead

A Cognitive Framework for Describing and Evaluating Software Exploration Tools

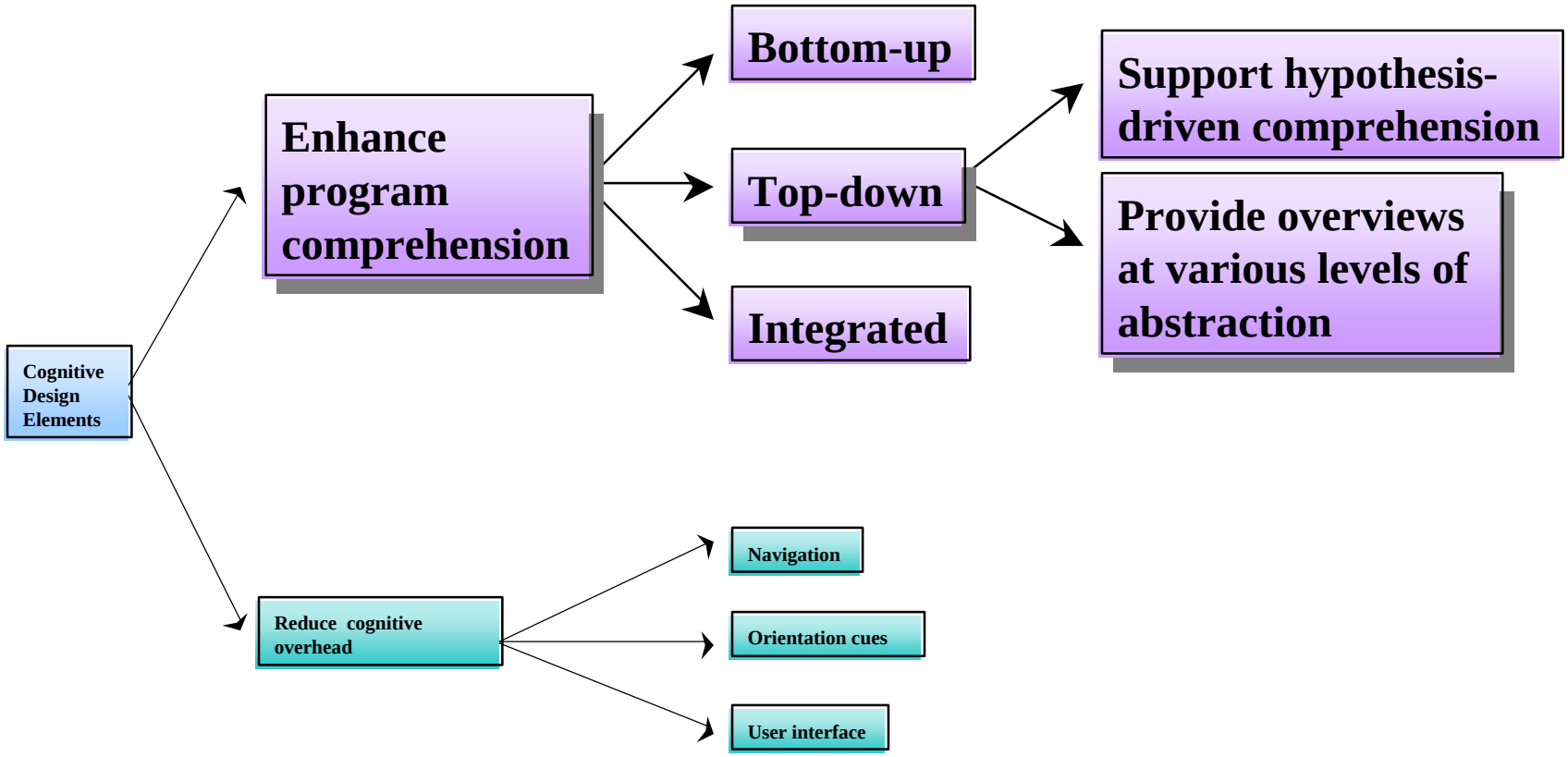


Cognitive design elements to support bottom-up comprehension



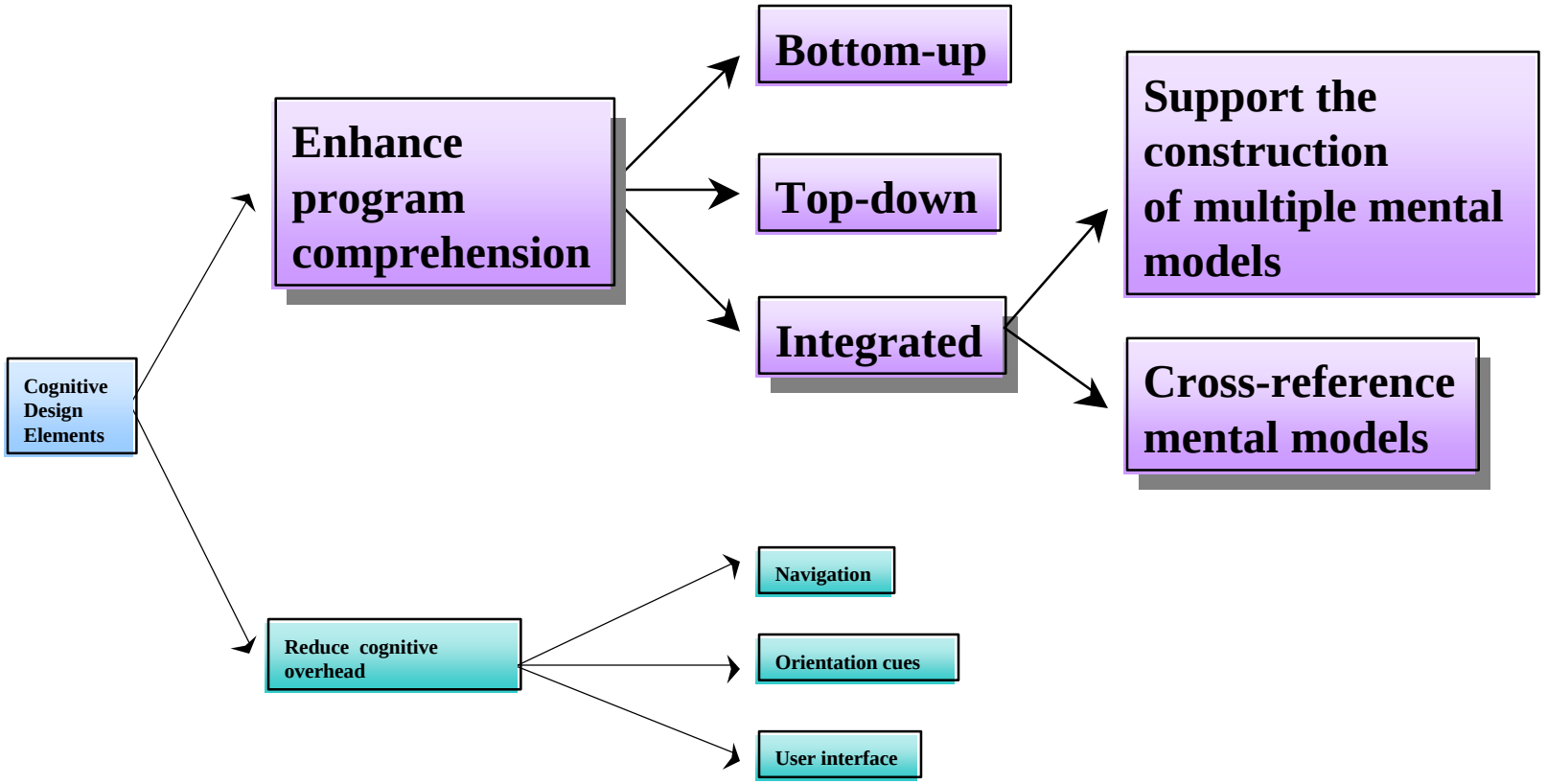
… source code listings, graphs, slicers, multiply linked views graph composition, filtering

Cognitive design elements to support top-down comprehension



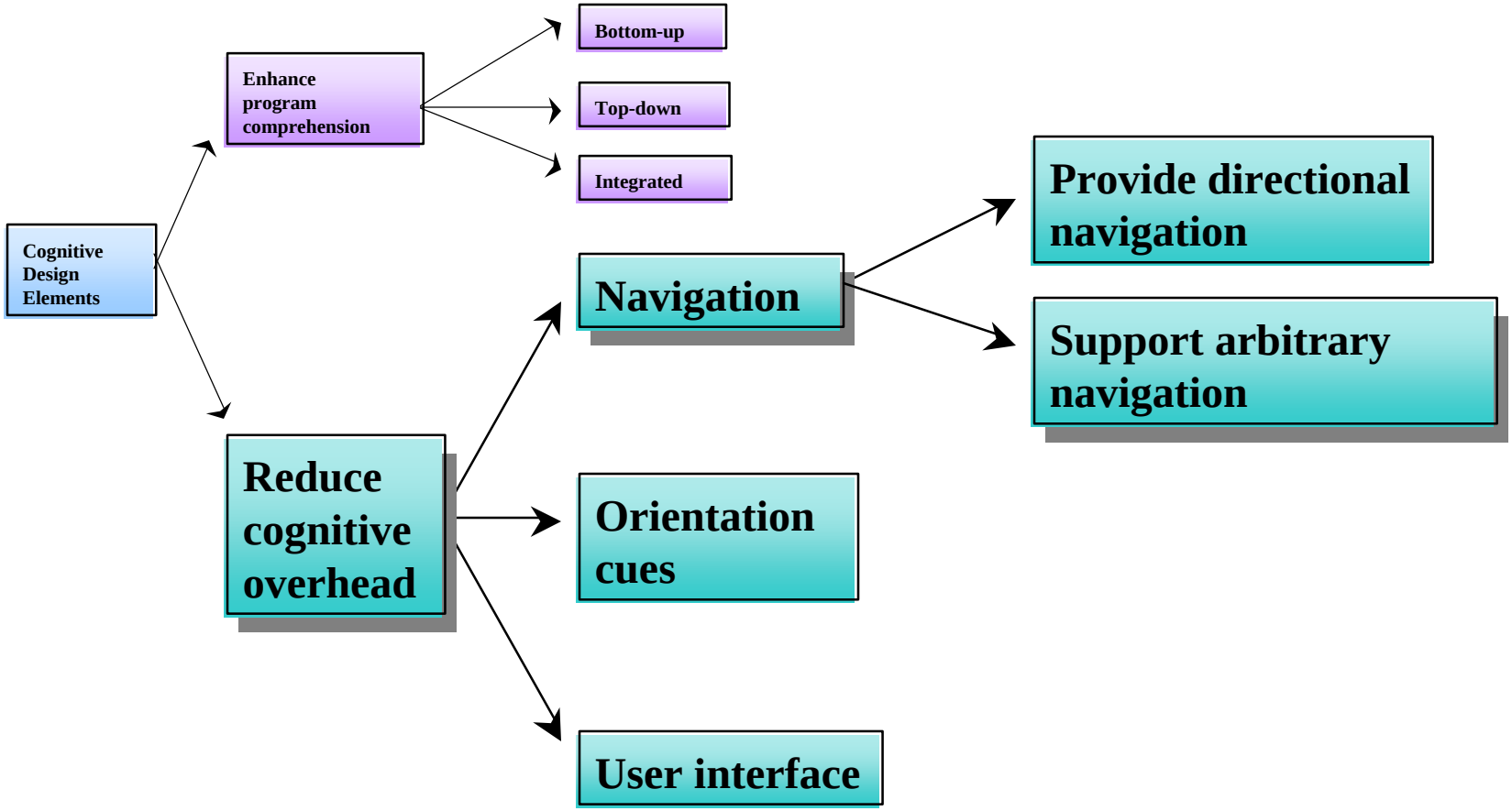
… layered annotations, design patterns, overview, maps, structure charts, nested graphs

Cognitive design elements to support the integrated meta-model



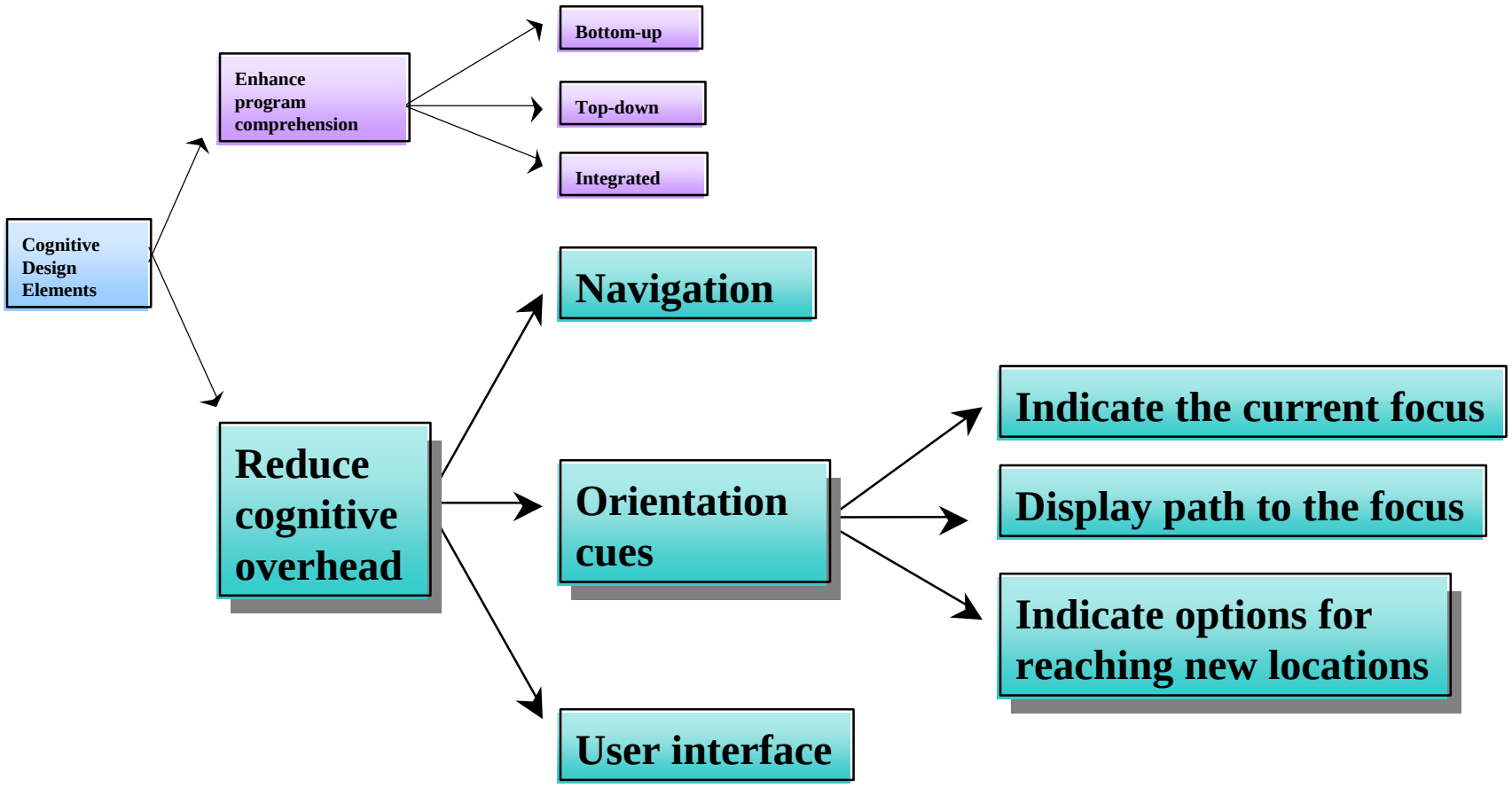
… multiple views, 3D graphics, synchronized views, hooked line diagrams, hyperlinks

Design elements to support navigation



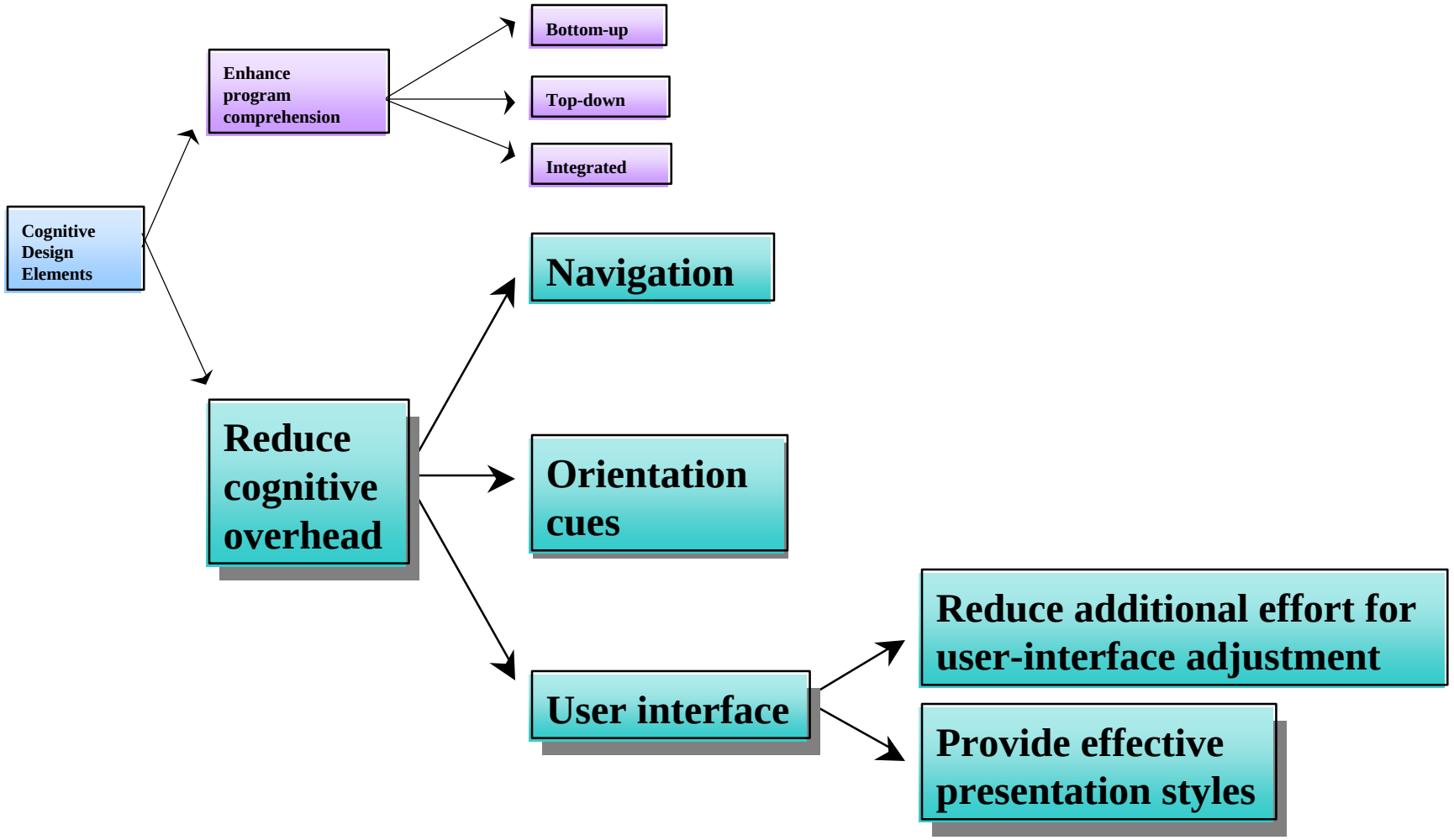
… editors, browsers, graphs, subsystem hierarchies, bookmarks, search engines

Design elements to provide orientation cues



… highlighting, colour, fisheye-views, histories, overview windows, trails, hyperlinks

Design elements to improve user-interface adjustment



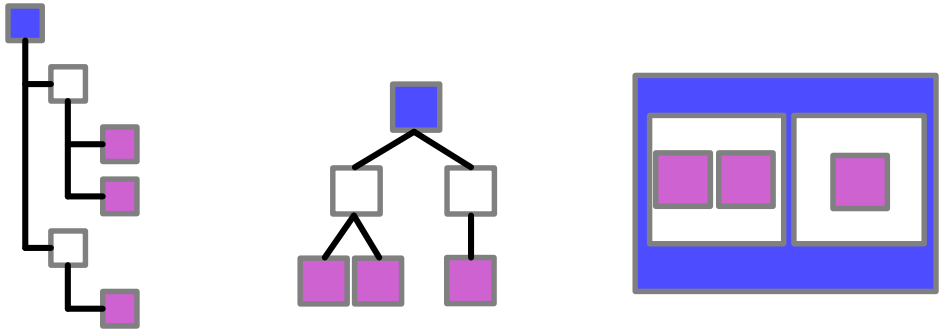
…visibility, feedback, natural mappings, graph layouts, 3D graphs, animation

Challenges – software visualization tools

- Many software exploration tools have not been adopted in industry
- Large software systems generate large graphs
- Insufficient support for switching between systematic browsing and opportunistic approaches, and for switching between bottom-up and top-down comprehension strategies
- Source code is often too far removed from the graphical views
- Multiple windows are confusing and difficult to manage
- Usefulness versus usability
- Lots of tools, not obvious which problems they are trying to solve…

SHriMP Views

- **S**imple **H**ierarchical **M**ulti-**P**erspective Views
- Prototype interface for exploring software structures and browsing code
- Makes better use of limited screen area
- Integrates code browsing using hypertext (HTML objects) embedded in the graph
- Sophisticated animation and zooming (Jazz)
- Uses nested graphs (containment)

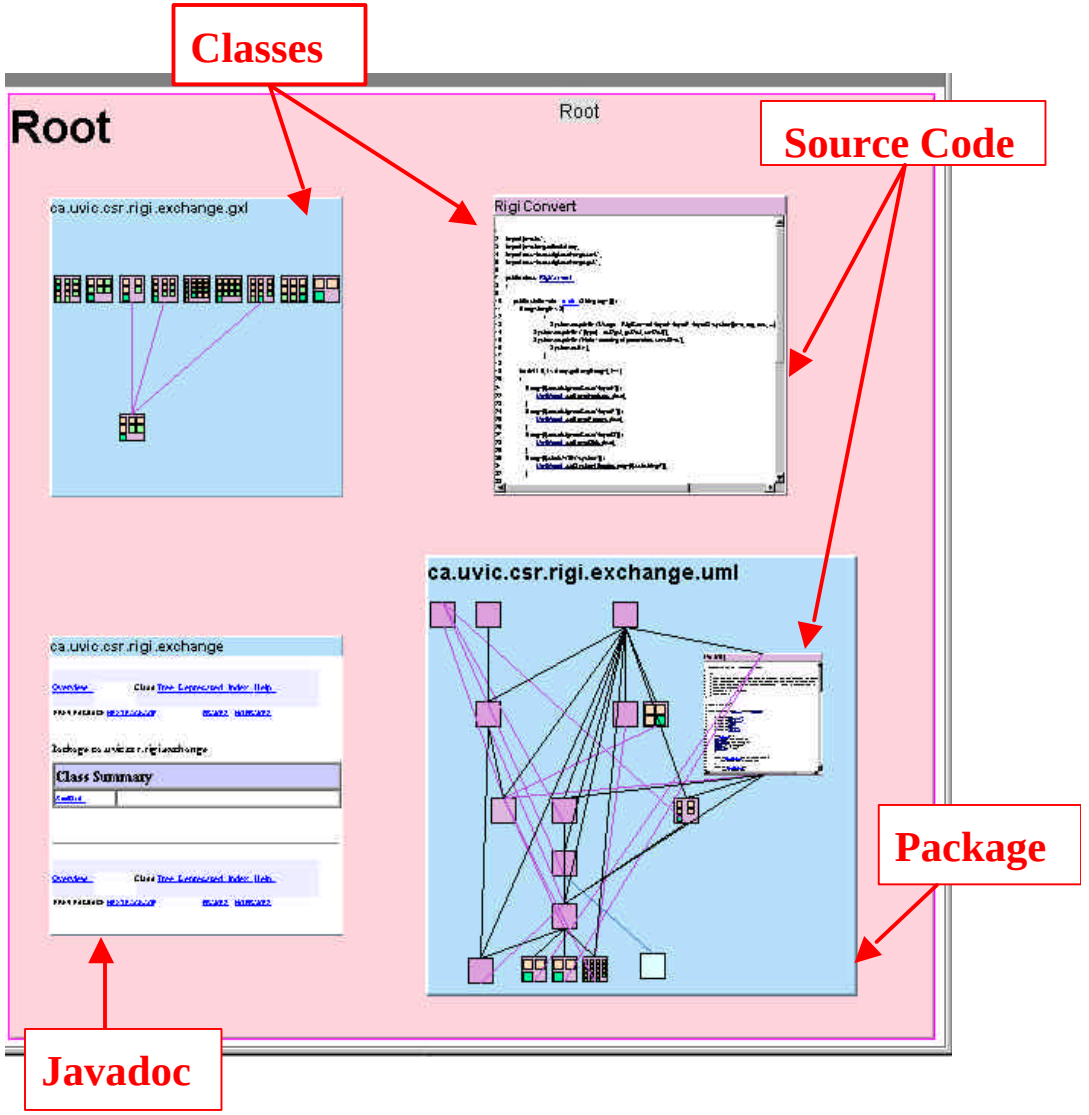


Visualizing Java programs in SHriMP

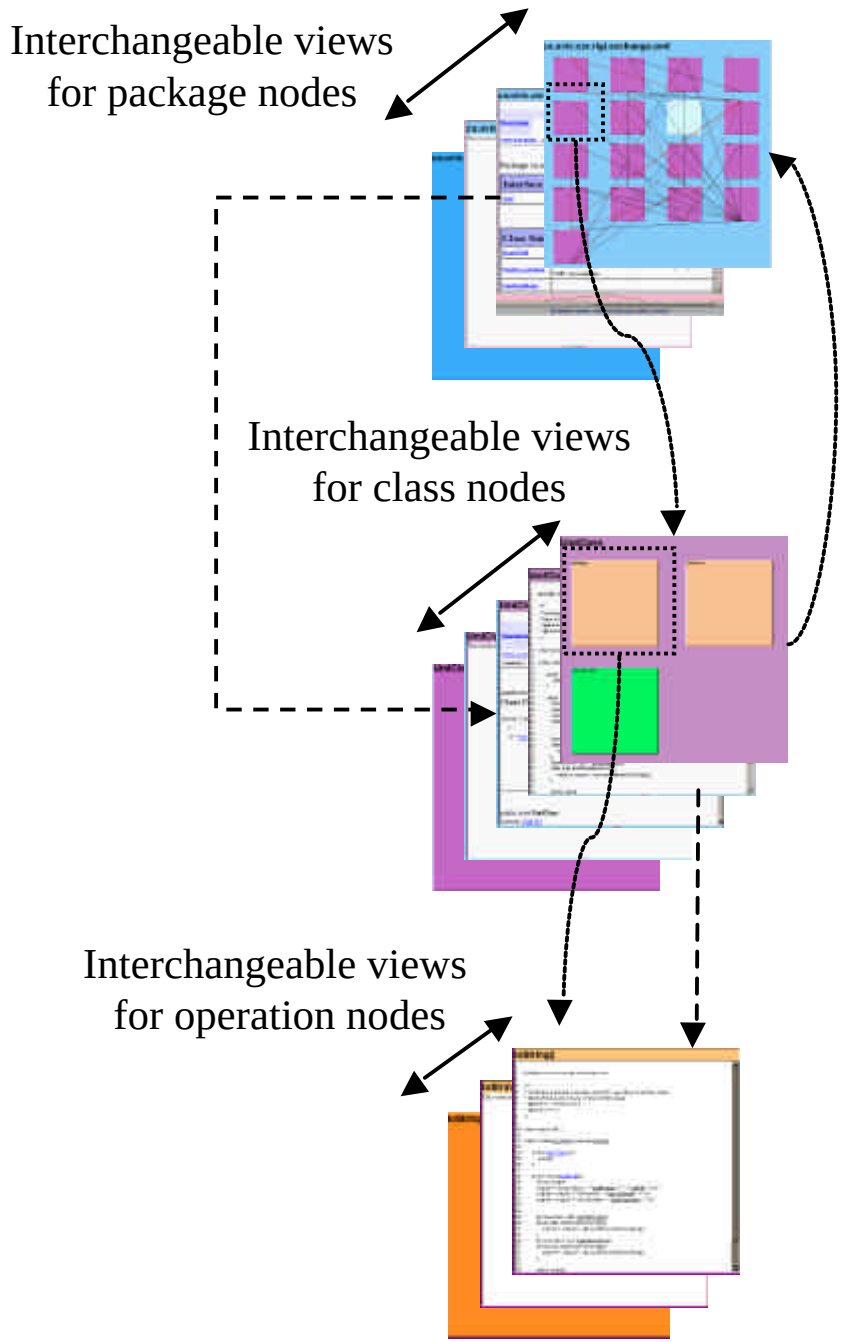
- Possible sources (layers) of information for the Java domain:
  - Analysis, metrics
  - Documentation
  - Source code
  - Architecture
  - Requirements
  - Version control and management….

Integrating multiple information views

- We use *an* architecture to provide placeholders for different views
- Which architectural view you choose depends on the stakeholder and the purpose of the visualization
- Various views are accessible at any level of granularity (method, class, package etc)
- Hypertext and zooming features support navigation
- Component design supports adding new “views”



SHriMP Demo – Visualizing a Java Program



Nested Interchangeable Views for Java Visualization

Operations for switching views:

Zoom in/out
(default action is configurable)

Semantic zoom
(e.g. following hyperlinks)

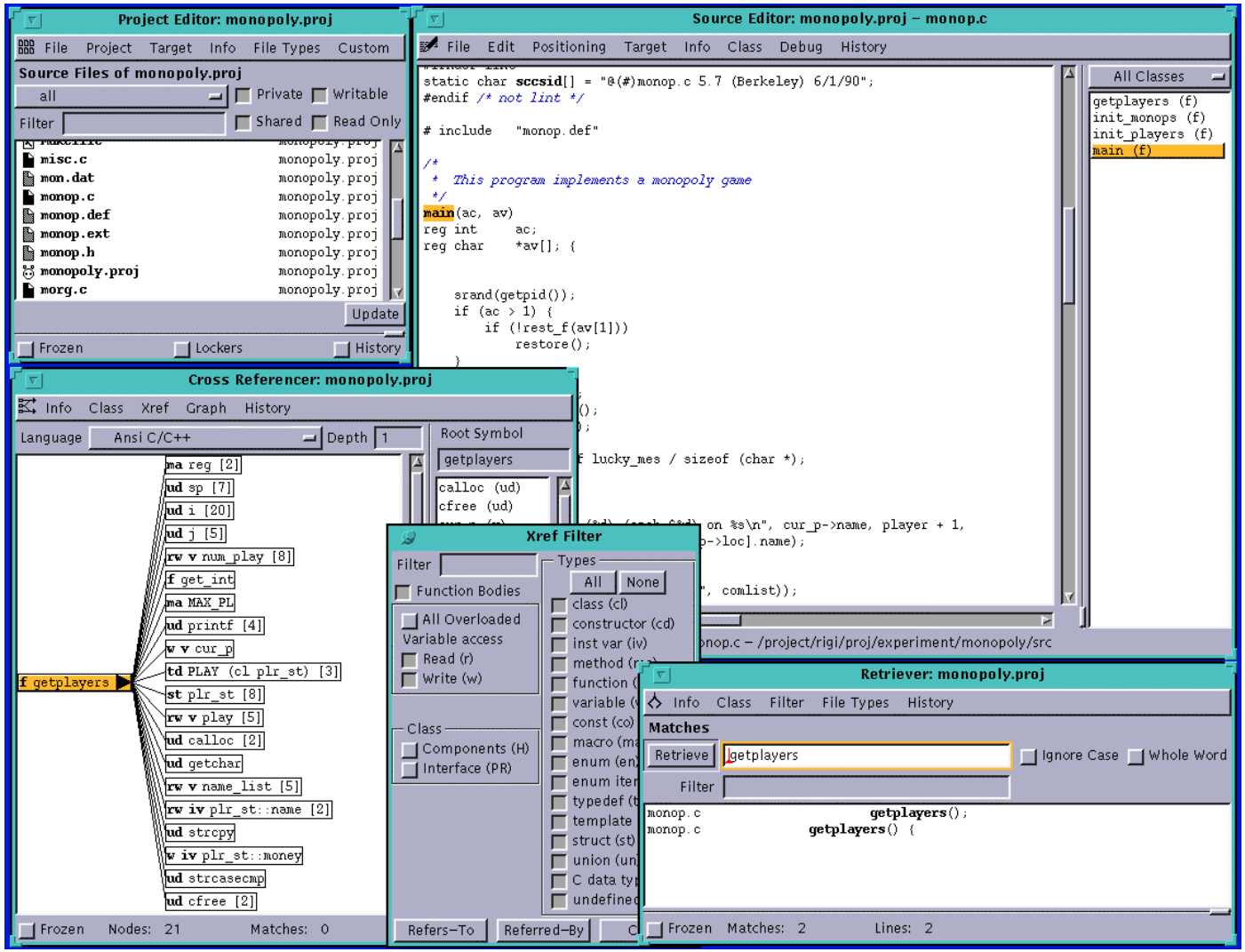
Switching between
interchangeable
views using the hotbox

Empirical evaluations of SHriMP

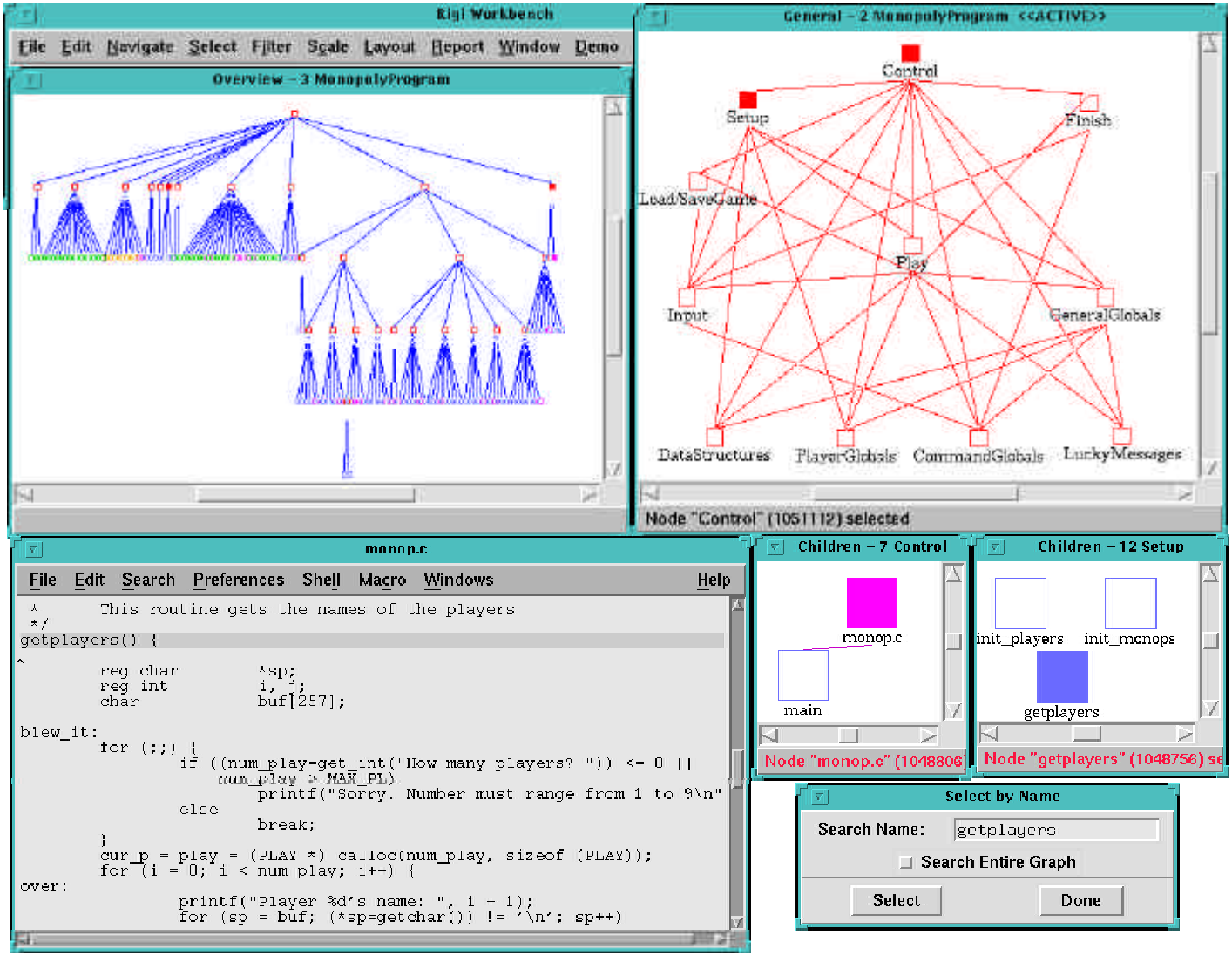
Studying the effectiveness of software exploration tools

- Pilot study, Spring 1996
- Second study, Spring 1997
- In progress, Summer 2001 and Summer 2002

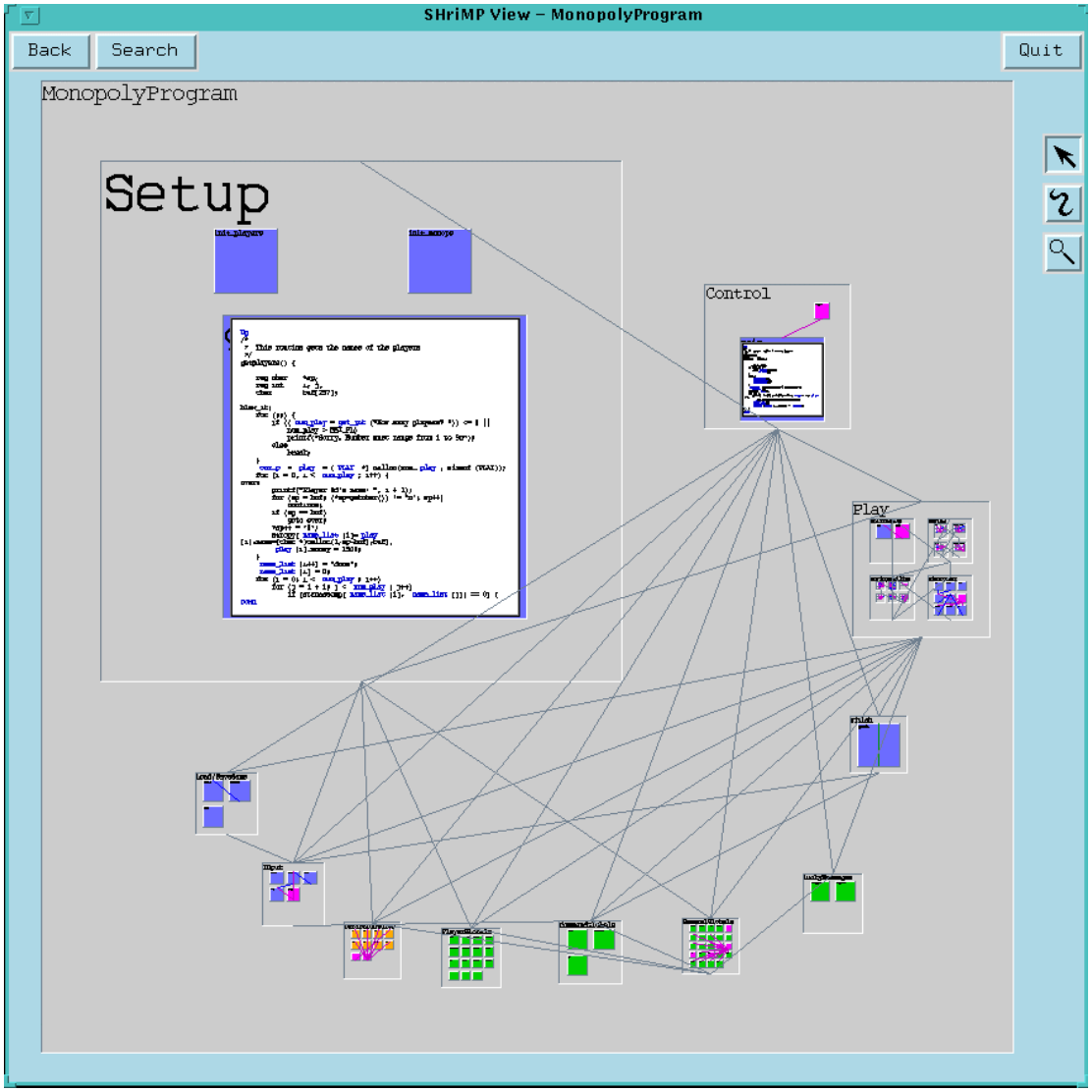
A SNiFF+ view of monopoly



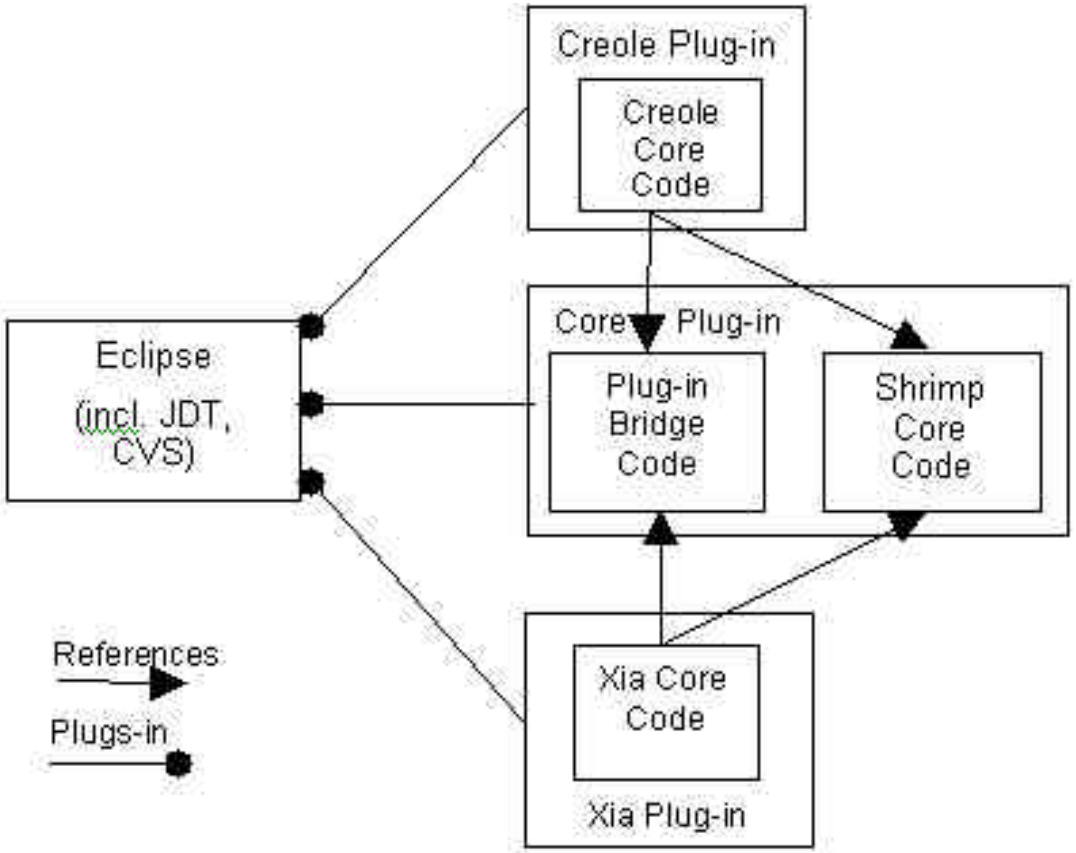
A Rigi view of monopoly



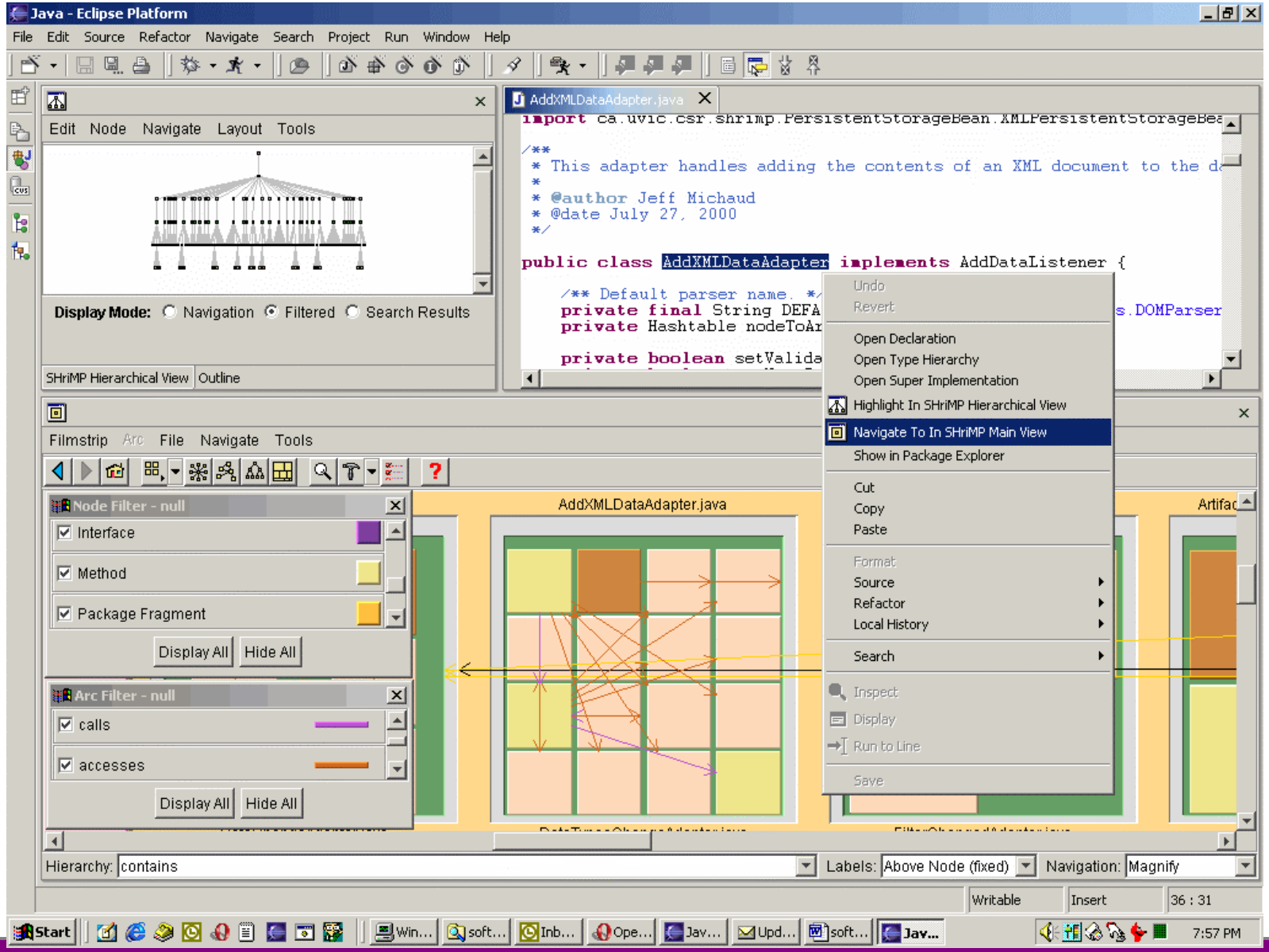
A SHriMP view of monopoly



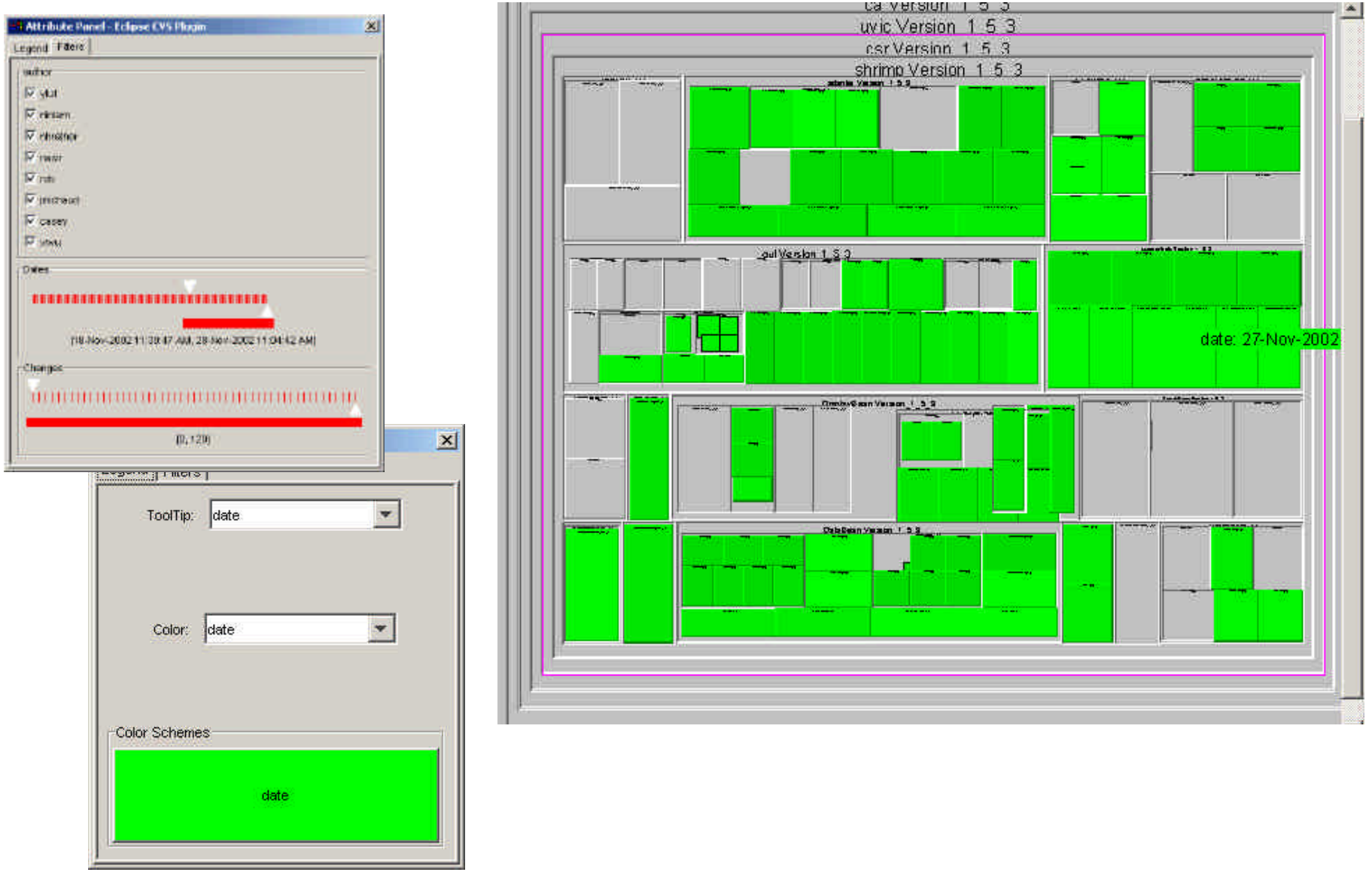
Overview of the SHriMP-Eclipse plug-in architecture



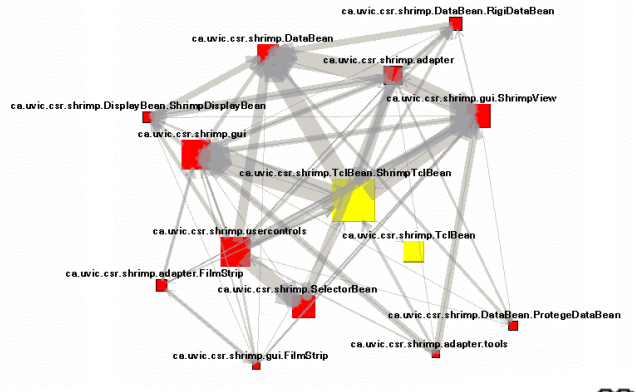
SHriMP plugged into the Eclipse platform

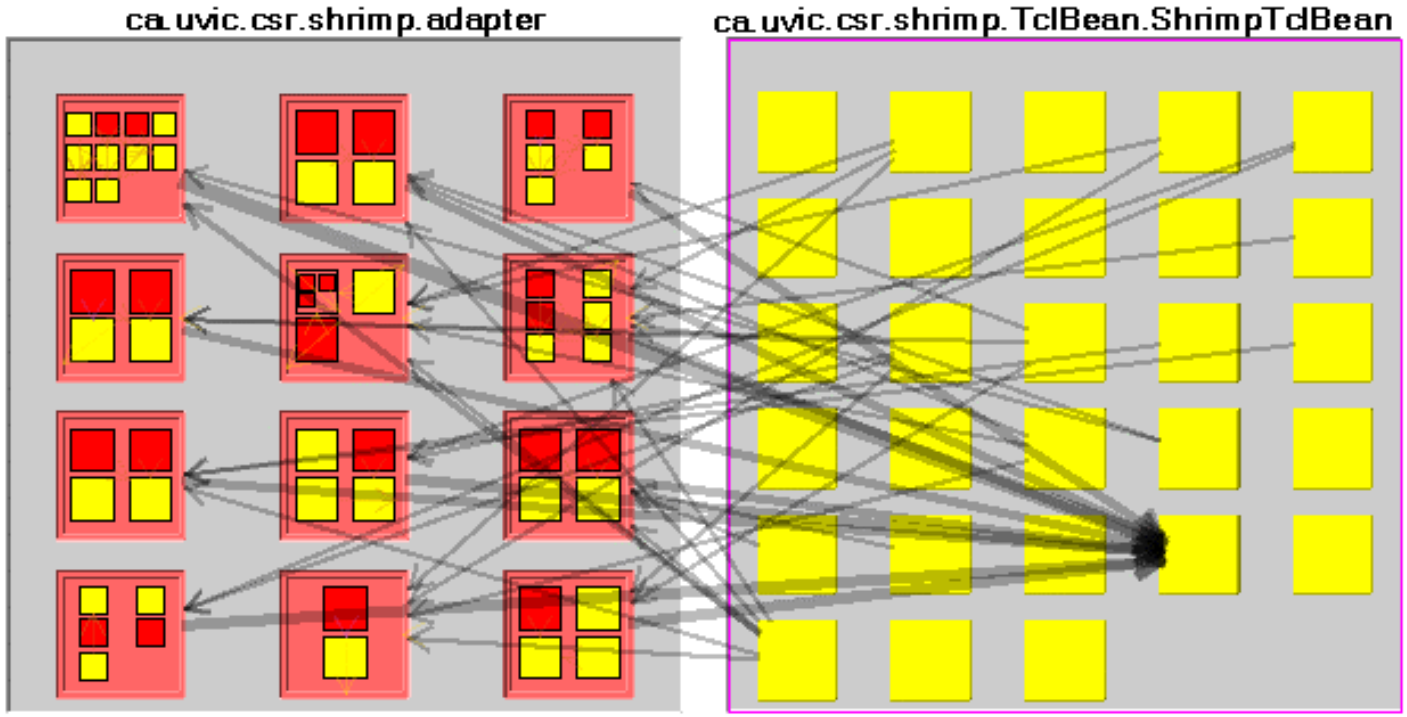


Visualizing CVS information



Integrating CVS and JDT information





Current work in software visualization

- Integration with Eclipse
  - IBM Websphere Studio Workbench www.eclipse.org
  - Eclipse is open framework for building integrated development environments
- SHriMP has been customized and retargeted for visualizing “flow diagrams”
  - Flow diagrams are used in an eBusiness project to model the dynamic aspects of a system, such as the main activities and the movement of information in a business process
  - Flow diagrams can be hierarchically composed
- User studies -- long way to go….

References

- Mental imagery in Program Design and Visual Programming, by M. Petre and A. Blackwell, http://www.cl.cam.ac.uk/~afb21/publications/IJHCS.html
- Price et al's principled taxonomy of software visualization: http://www.dgp.toronto.edu/people/RMB/papers/p20.pdf
- Cognitive questions in software visualization: http://www.cl.cam.ac.uk/~afb21/publications/book-chapter.html
- A Cognitive Framework by Storey *et al*: http://www.cs.uvic.ca/~mstorey/papers/jss.pdf