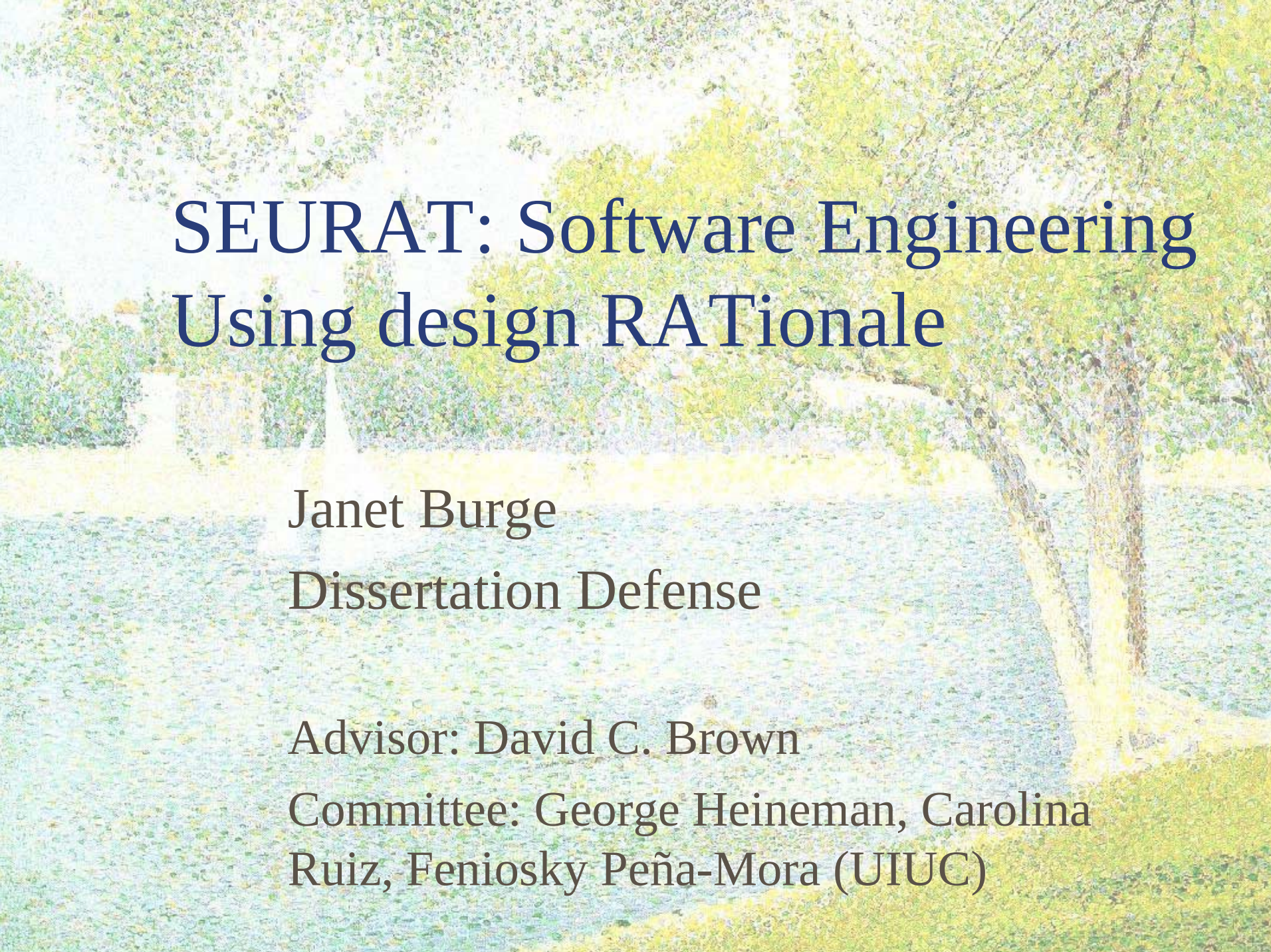


The background of the slide is an impressionistic painting of a path or road winding through a dense forest. The brushstrokes are visible and textured, with a palette of greens, yellows, and browns, suggesting a dappled light effect from the trees. The path leads from the bottom left towards the upper right, disappearing into the foliage.

SEURAT: Software Engineering Using design RATionale

Janet Burge

Dissertation Defense

Advisor: David C. Brown

Committee: George Heineman, Carolina
Ruiz, Feniosky Peña-Mora (UIUC)



What is Design Rationale?

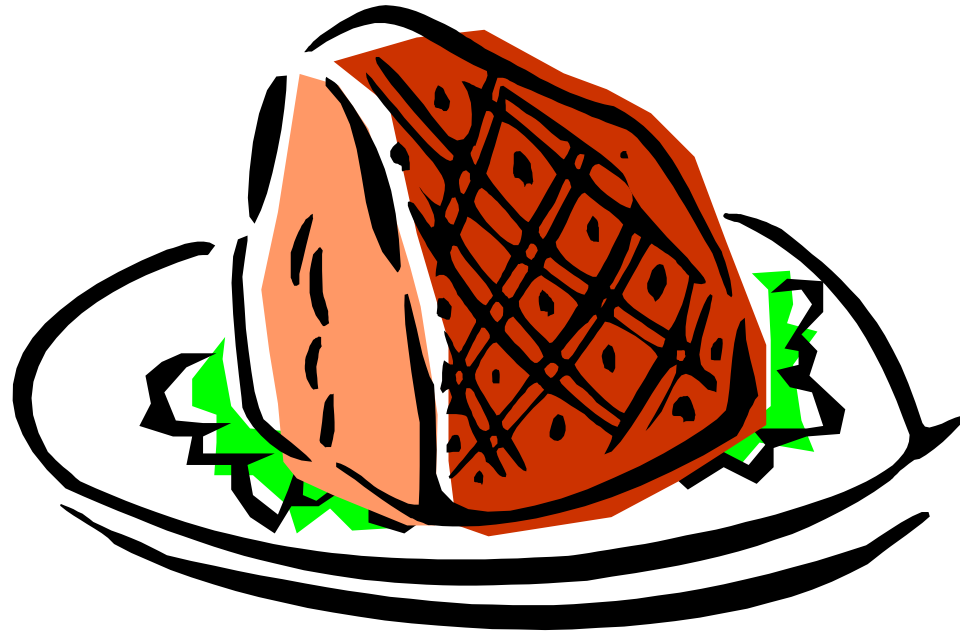
- “Design rationales include not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision”

[Lee, 1997]

The *whole* story behind the design, not just a static snapshot of the final product!



The Ham Story





Example (from a Conference Room Scheduling System)

- *Decision:* How do we represent a conference room in the system?
 - *Alternative:* store the name (location) as a string
 - *Argument for:* simple to code
 - *Argument against:* difficult to extend
 - *Alternative:* create a conference room class
 - *Argument for:* can contain information other than location



Why is DR valuable?

- Captures designer's intent
- Avoids duplicating past effort by providing alternatives already considered
- Avoids repeating past mistakes by documenting when something was tried and failed

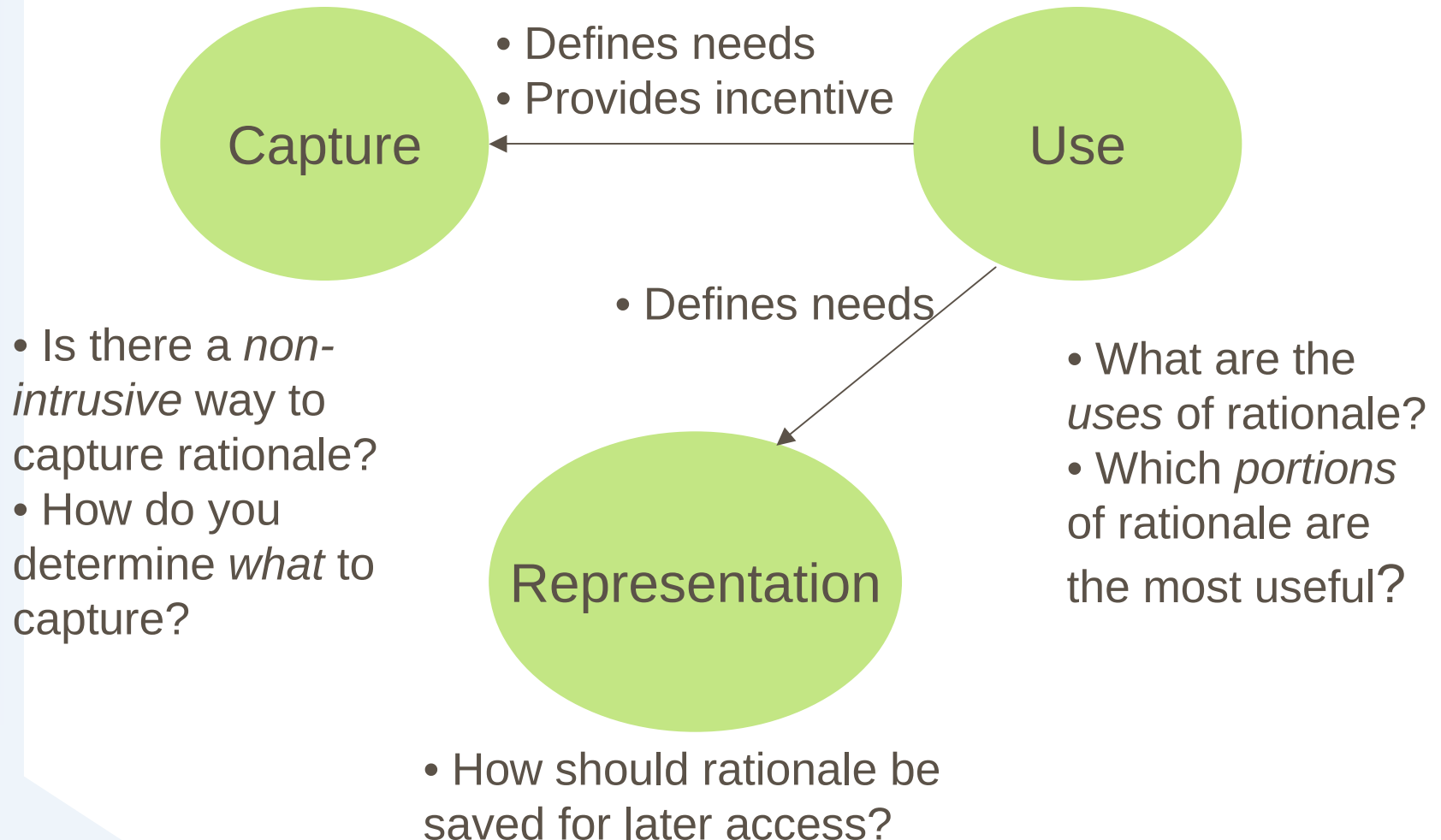


Why isn't DR used now?

- Collection can impede design process
- Collection is often tedious
- Designers are reluctant to record “mistakes”
- Collection is very costly
- Not enough examples of use to provide motivation



Issues with Design Rationale





Using Rationale to Support Software Development

- Focus on Software Maintenance:
 - Software lifecycle is very long
 - Maintenance costs are high
 - Original designers are unlikely to be available
- Rationale supports inference to help maintainers find problems, fix problems, and extend software with less risk



Our Hypothesis

- With appropriate tool support, rationale can provide useful support to the software maintainer.
 - Improved efficiency – less time required to perform maintenance tasks
 - Improved effectiveness – rationale assists maintainers in making better decisions



SEURAT: Software Engineering Using RATionale

- Using rationale to assist in software development and maintenance:
 - verifying consistency and completeness of the rationale
 - evaluating the support for design alternatives
 - ensuring that rejected alternatives are not repeated
 - presenting applicable rationale to the maintainer to assist in modification
 - maintaining rationale consistency by propagating results of rationale modifications



SEURAT Capabilities

- Tight integration with development/maintenance environment
- Allows “what if analysis” of
 - changing design priorities
 - disabling assumptions
 - disabling requirements
- Supports traceability of requirements to decisions (and then to code)



Key SEURAT Issues

- *Inference* – how can we inference over the rationale to support uses that go beyond presentation?
- *Representation* – what needs to be represented to support inference?
- *Ontology* – how do we provide a common vocabulary to support inferencing over content?
- *Integration* – how can we encourage rationale use by integrating with an existing development environment?



Inference

- Produce hypothesis, not conclusions
- Two categories:
 - Syntactic inference that uses the structure of the rationale
 - Semantic inference that uses the content
- Syntactic inference over structure:
 - Look for decisions with no selected alternative
 - Look for selected alternatives with no supporting arguments
 - Check for unanswered questions



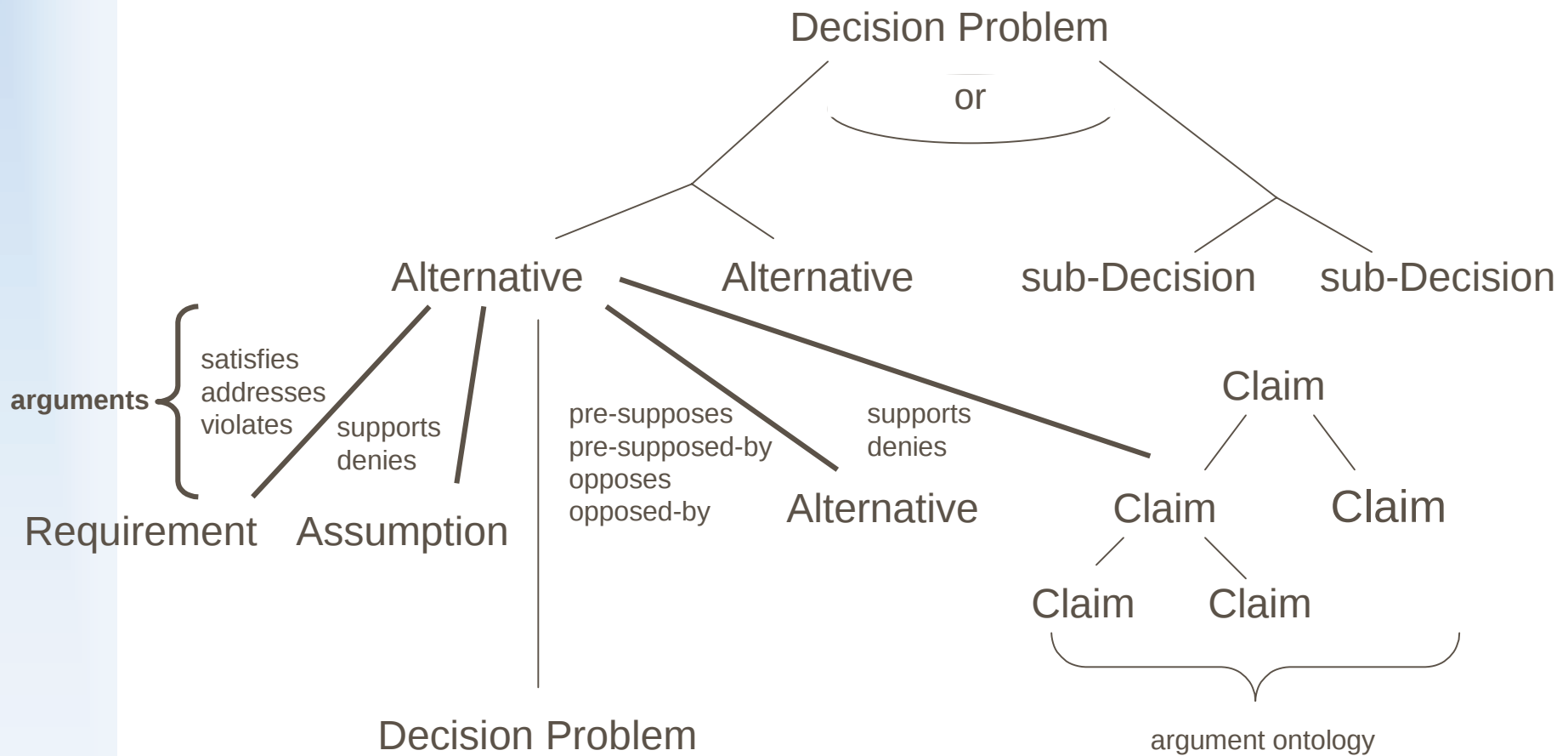
Inference (cont.)

- Semantic inference over content:
 - Evaluate alternatives and alert if weaker alternatives are selected
 - Re-evaluate decisions after an assumption is disabled
 - Re-evaluate decisions if argument priorities change
 - Check for tradeoff violations
 - Check for dependencies between alternatives
 - Check for requirement violations



Rationale Representation

- Argumentation Representation
 - Semi-structured representation that is readable by machines and people
 - Captures the arguments for and against each alternative
 - Supports arguments about requirements, assumptions, claims (non-functional requirements), and other alternatives (dependencies)

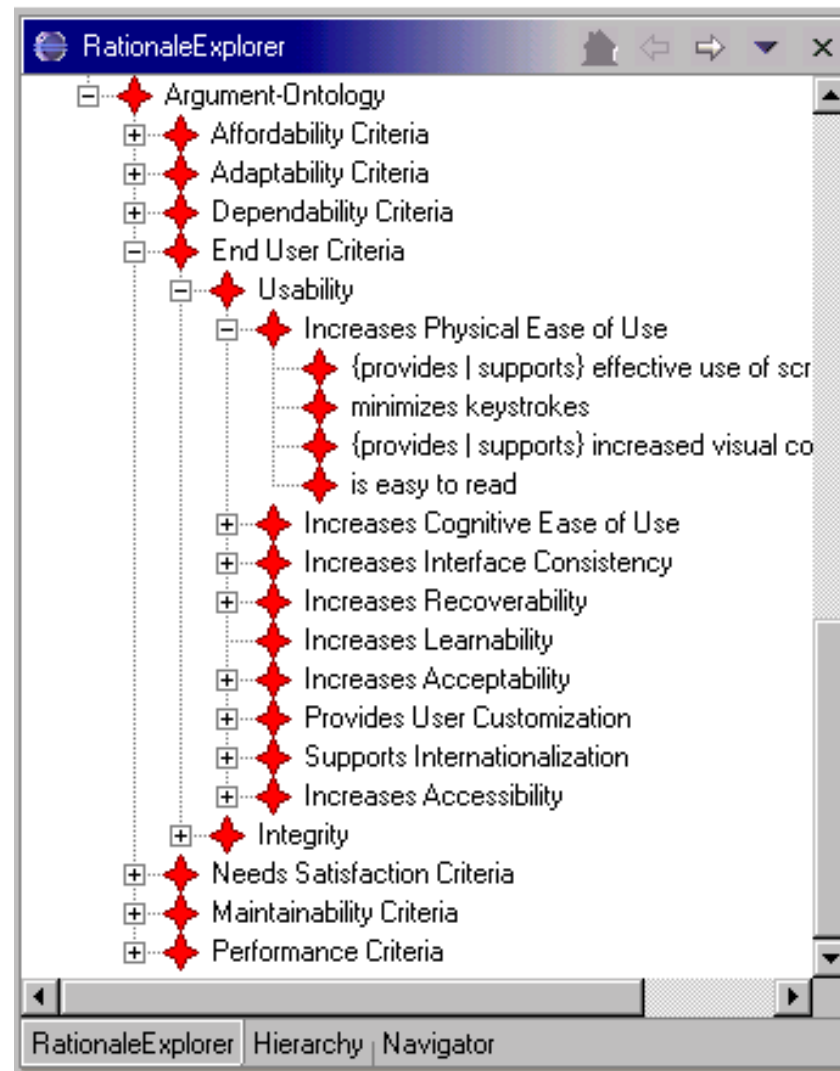




Argument Ontology

- Provides a common vocabulary used to compare alternatives and arguments during inference
- List of common arguments for software changes at varying levels of abstraction
 - Based on the “ilities” (affordability, scalability, etc.)
- Each ontology entry has an associated default importance that can be inherited by rationale that refers to it

Argument Ontology





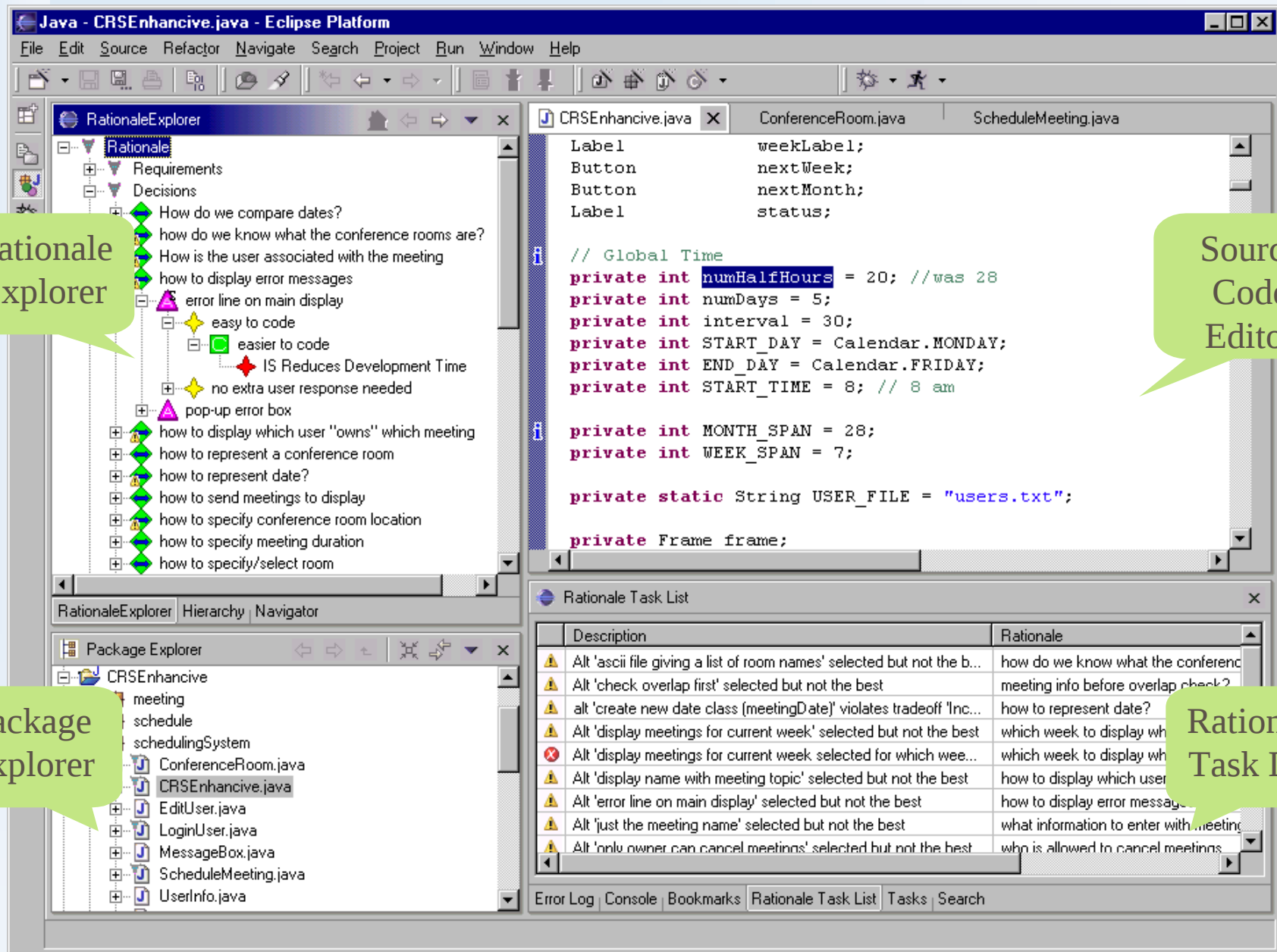
Integration

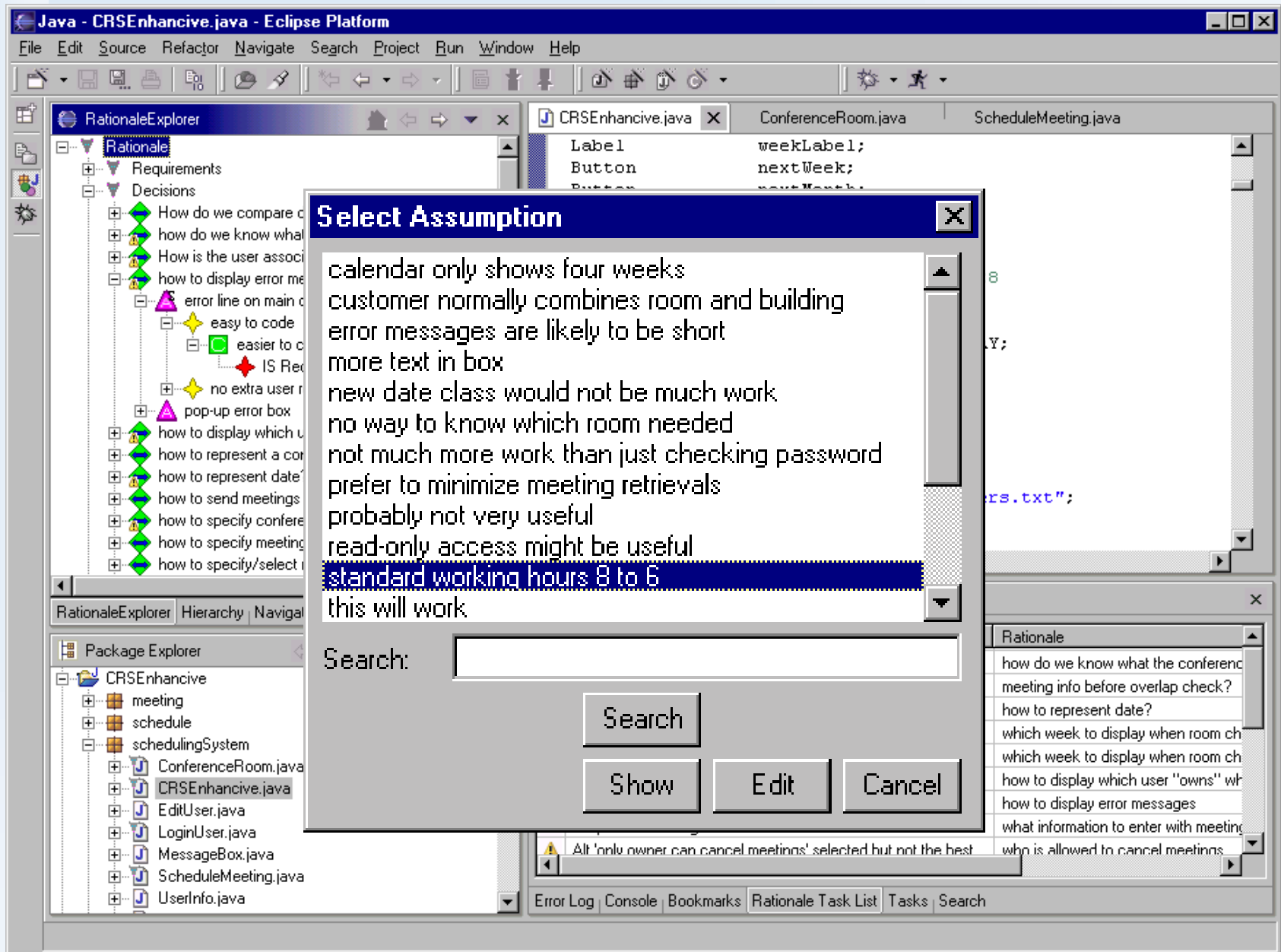
- SEURAT is implemented as an Eclipse plug-in
 - Rationale is more likely to be used if the developer does not need to switch tools
 - Rationale can be directly associated with code and its presence indicated in the editor used by the developer to write and maintain code
- Rationale is stored in a MySQL database
 - Scalable to large amounts of rationale
 - SQL queries support inference and presentation

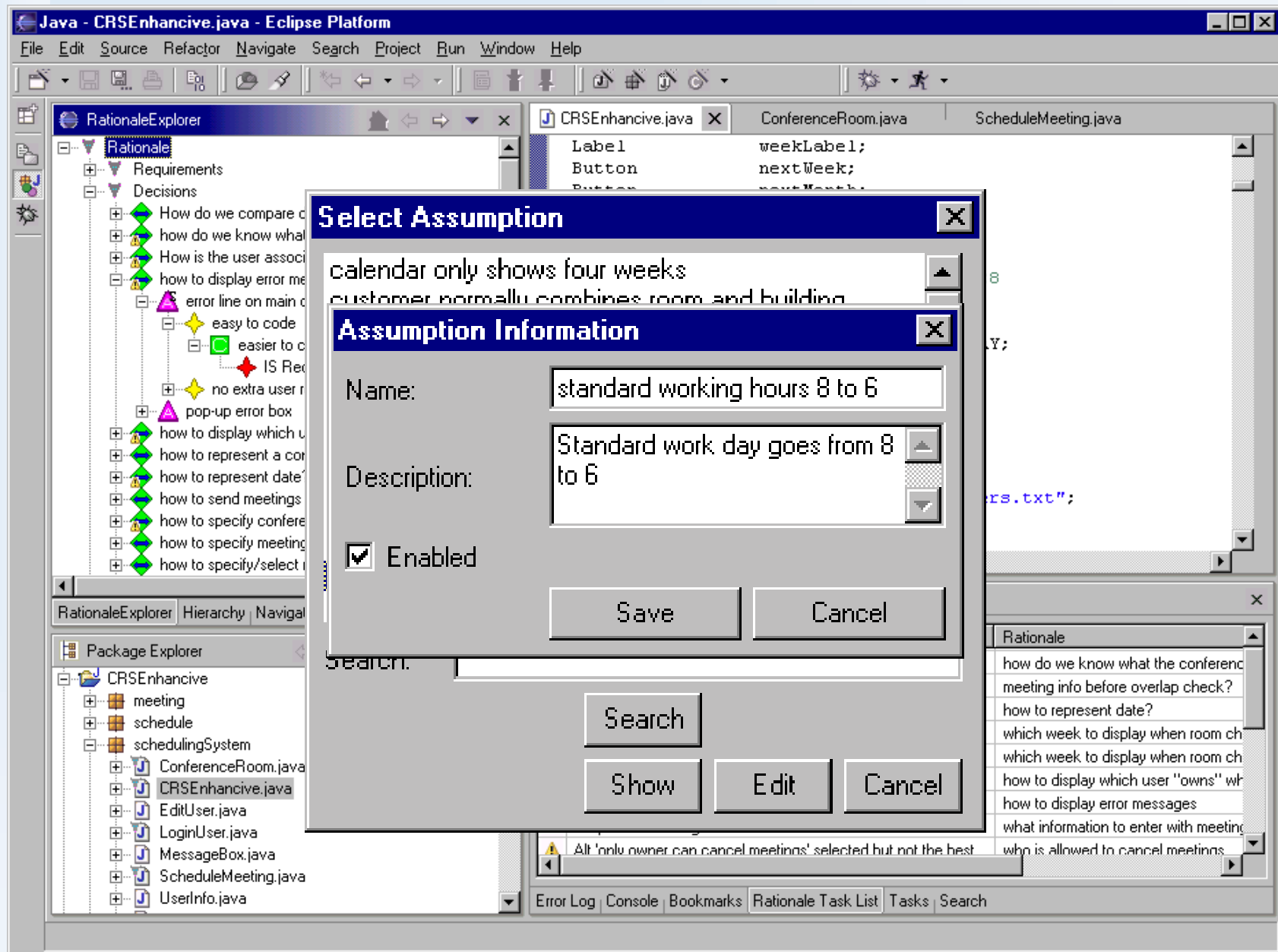


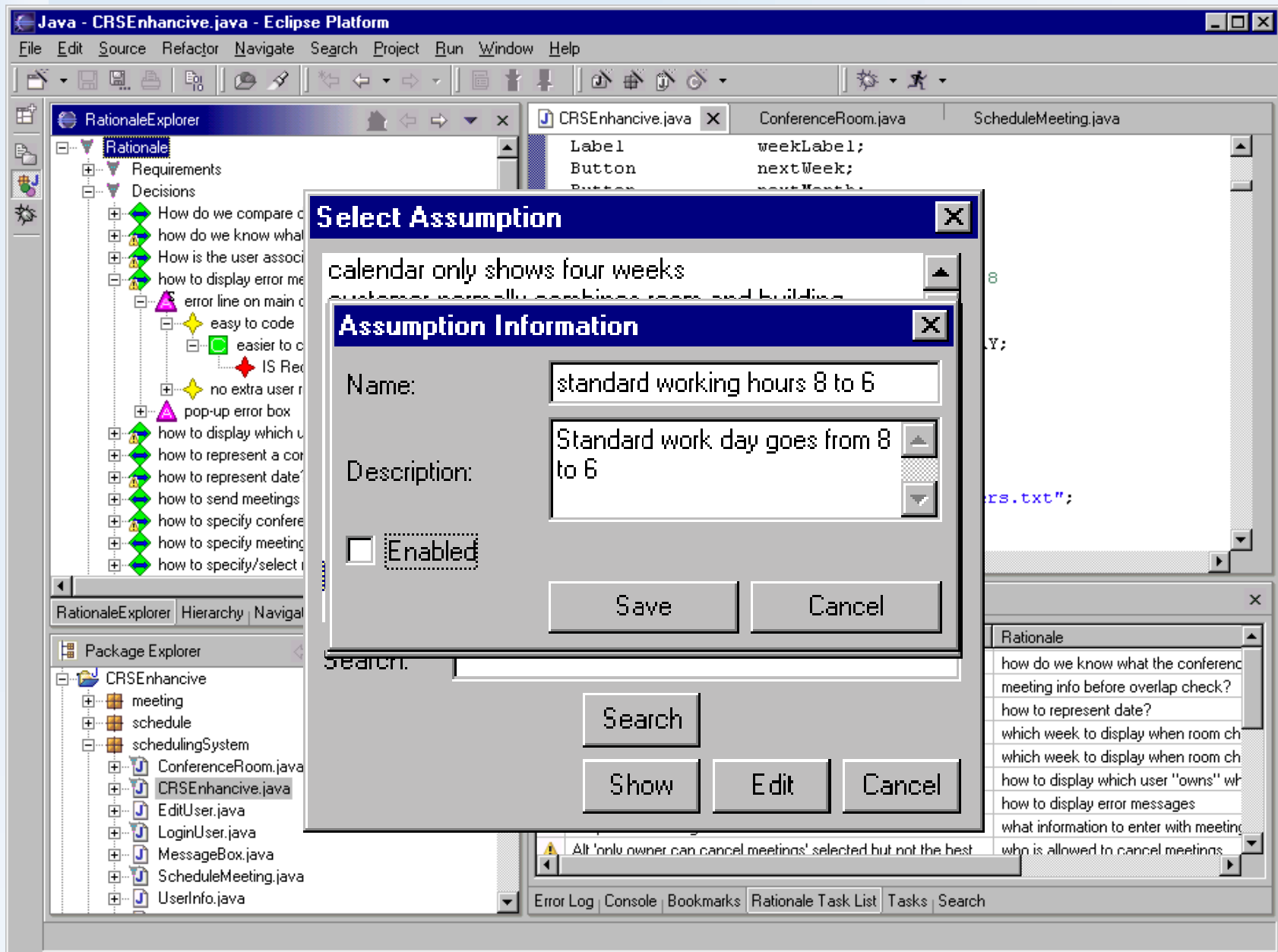
Using SEURAT in Software Maintenance

- Reasons for Maintenance
 - Corrective maintenance (fixing errors)
 - Enhanceive maintenance (new functionality)
 - Adaptive maintenance (non-functional enhancements)
- Sources of error
 - Requirement violations
 - Defects (bugs)
 - Changed or incorrect assumptions
 -



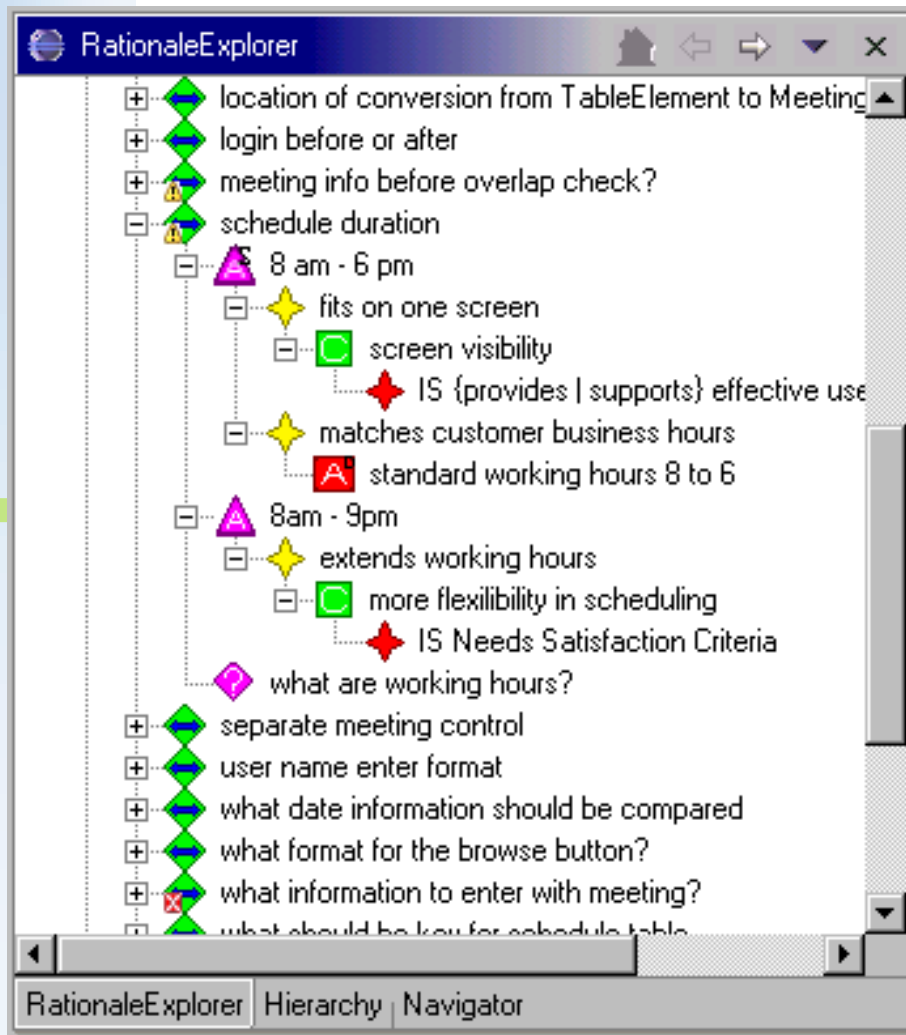








SEURAT Results

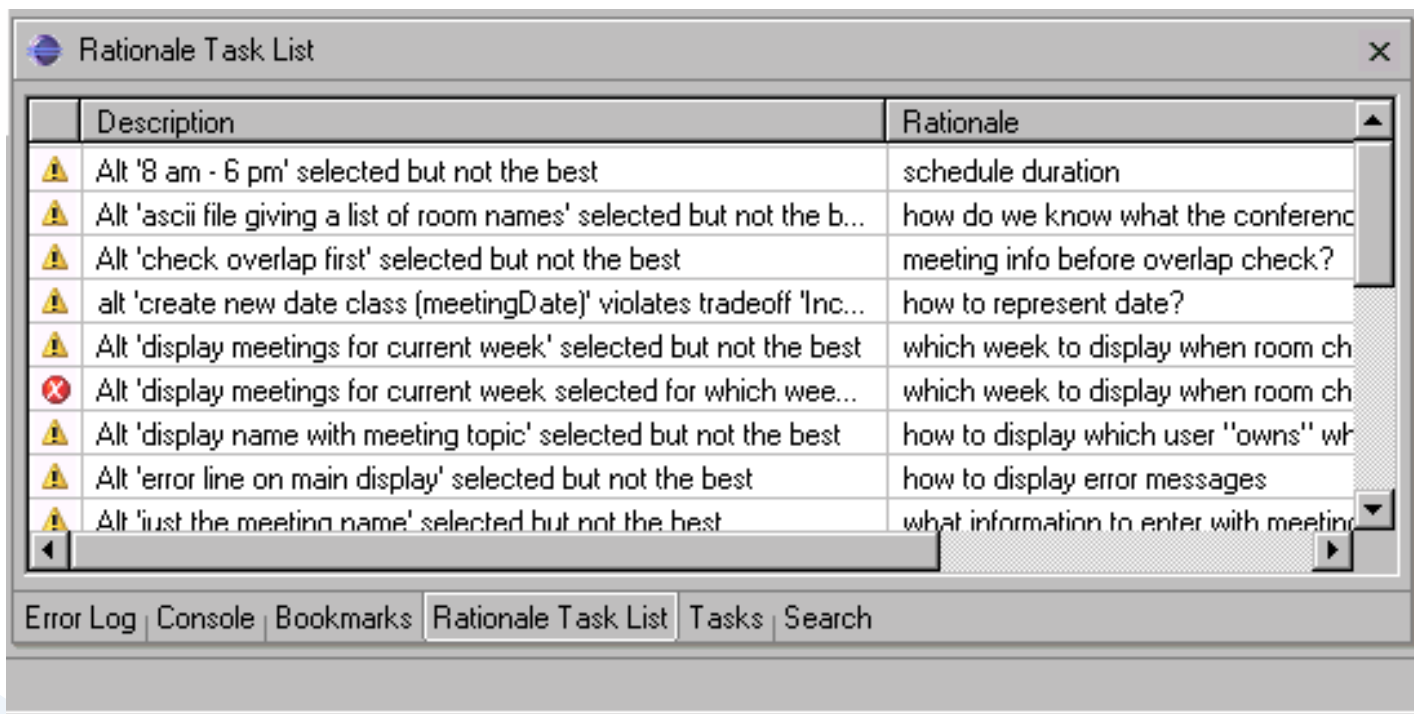


- Rationale Explorer
 - Indicates disabled assumption
 - Displays a warning for the decision that relies on the assumption



SEURAT Results

- Rationale Task List
 - Describes the warning
 - Allows the decision to be viewed in an editor





Finding the Implementation

Alternative Information

Name: 8 am - 6 pm

Description: The schedule should go from 8 am to 6 pm

Status: Adopted

Artifact: numHalfHours

Arguments For
fits on one screen
matches customer business hours

Arguments Against

Relationships

Save Cancel

- Alternative Editor
- Specifies the class, method, or instance variable

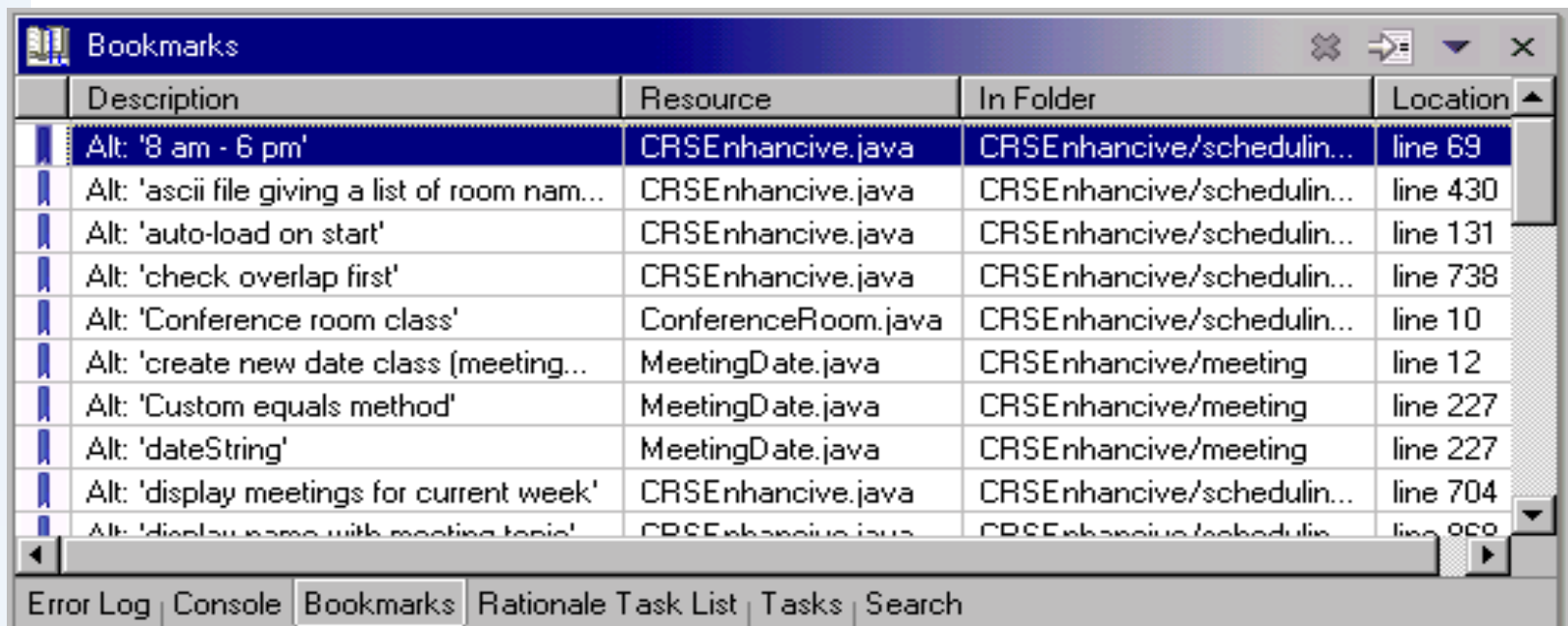




Finding the Implementation

■ Bookmark List

- Lists the file, folder, and line number
- Double-clicking brings up the code in the editor





Finding the Implementation

```
Player... Meeti... Clientl... Confir... Meeti... Meeti... Colum... J M... X
Choice      confRoom;
Label       weekLabel;
Button      nextWeek;
Button      nextMonth;
Label       status;

// Global Time
private int numHalfHours = 20; //was 28
private int numDays = 5;
private int interval = 30;
private int START_DAY = Calendar.MONDAY;
private int END_DAY = Calendar.FRIDAY;
private int START_TIME = 8; // 8 am

private int MONTH_SPAN = 28;
private int WEEK_SPAN = 7;

private Frame frame;

//R3
public MeetingApp(String user) {
```



Evaluation

- Three maintenance tasks:
 - adaptive maintenance (a non-functional change)
 - corrective maintenance (fixing a “bug”)
 - enhancive maintenance (extending functionality)
- Twenty subjects in two groups: experimental and control
- Measures:
 - time required to find the location in the code that needed changing and the time needed to complete the task (all subjects)
 - usability survey (SEURAT group only)
 - usefulness survey (SEURAT group only)

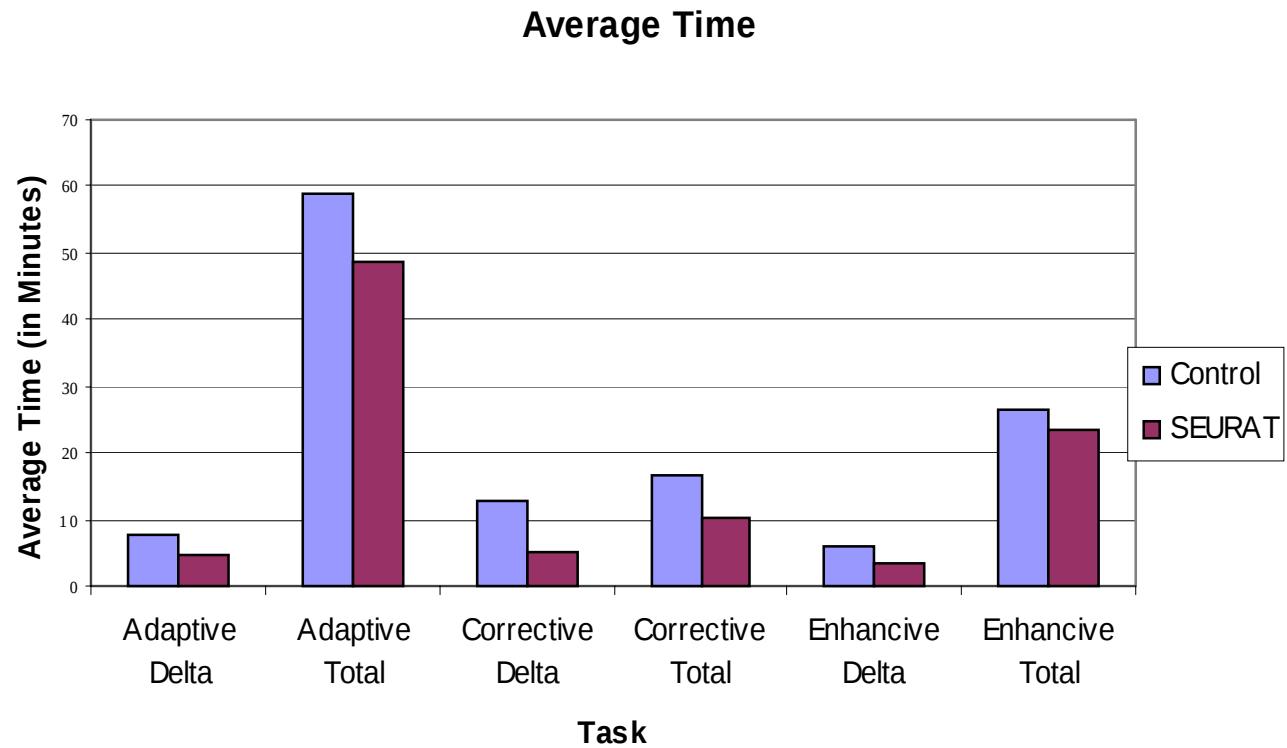


Subject Distribution

	SEURAT	Control
Java Experience (Expert/Moderate/ Some)	3/4/3 people, respectively	3/4/3 people, respectively
Average Work Experience	6.85 years	5.65 years
Eclipse Experience	60%	60%



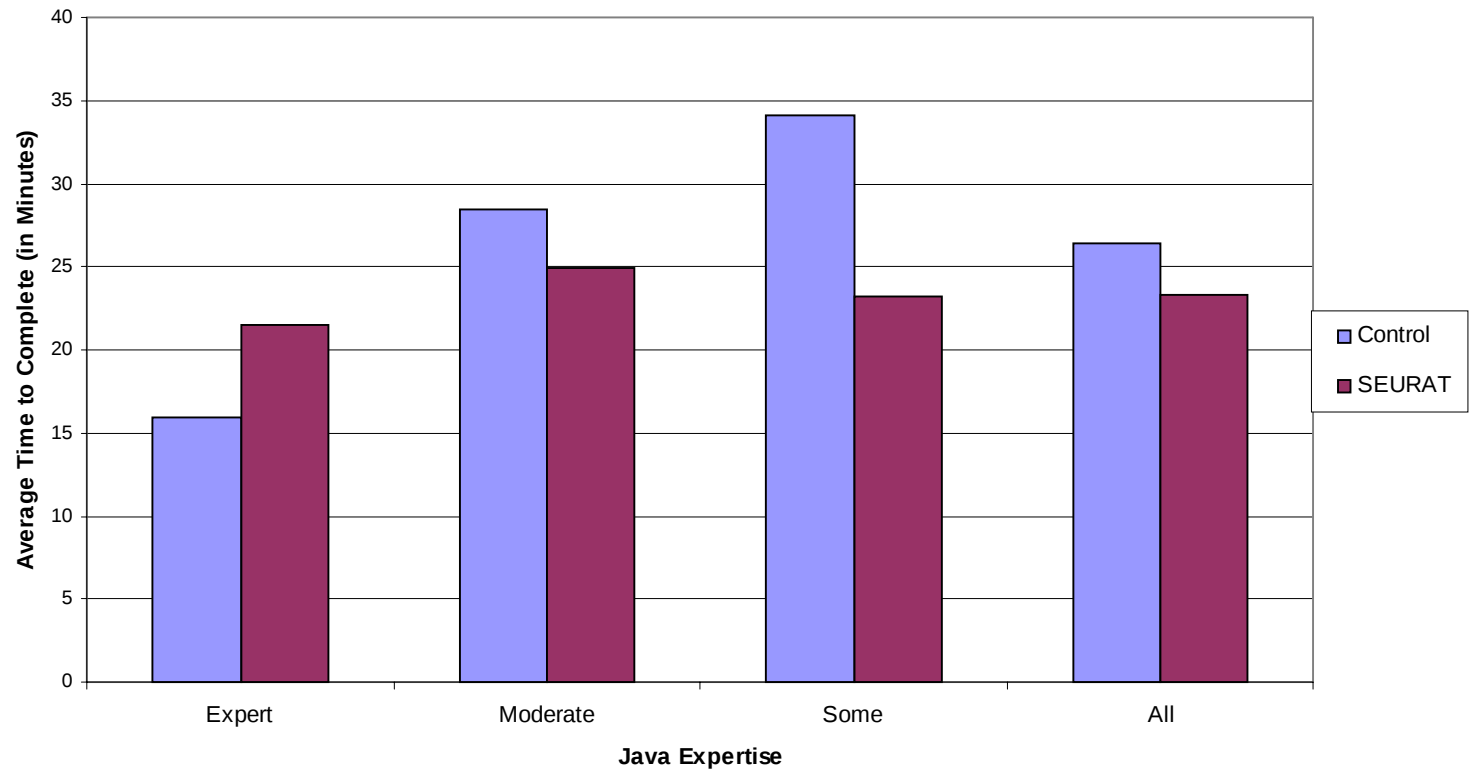
Results (Time)





Results by Expertise

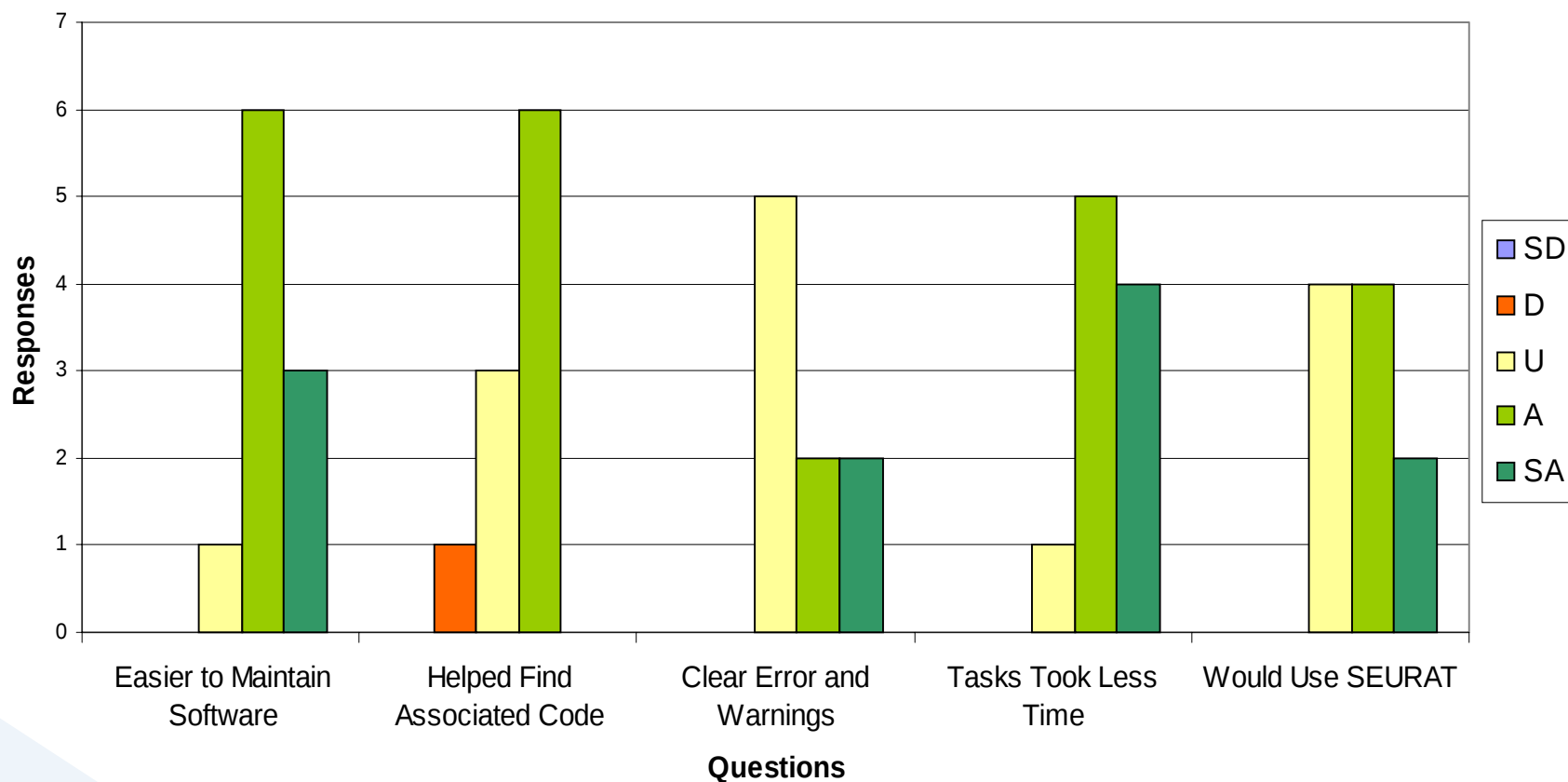
Enhance Maintenance





Results (Usefulness Survey)

Usefulness Assessment





Experiment Summary

- On average, SEURAT users outperformed the control group *except* for the “Expert” users
- The learning curve was a factor
- Variances in the control group were typically twice that in the SEURAT group
- More experiments are needed!



Contributions

- Tightly integrated usable environment that supports rationale capture and use
- Argument ontology that contains common arguments for making software design decisions
- Rationale representation tailored to software engineering and maintenance
- Uses of rationale that go beyond presentation:
 - support for “what-if” inferencing
 - checking for rationale consistency and completeness



Related Work

■ Includes:

- Lee: Decision Representation Language
- Peña-Mora: DRIM – Design Recommendation and Intent Model
- Klein: C-Re-CS
- Beñares-Alcántara, King: KBDS
- Bose: Decision Ontology within the WinWin framework
- Chung, et. al.: NFR-Framework



Summary and Conclusions

- Targeted software maintenance as an area that can best utilize rationale
- Demonstrated that with appropriate tool support, rationale can provide useful support to the software maintainer:
 - Demonstrated uses of DR that go beyond browsing and presentation
 - Integrated DR support with a standard software development environment



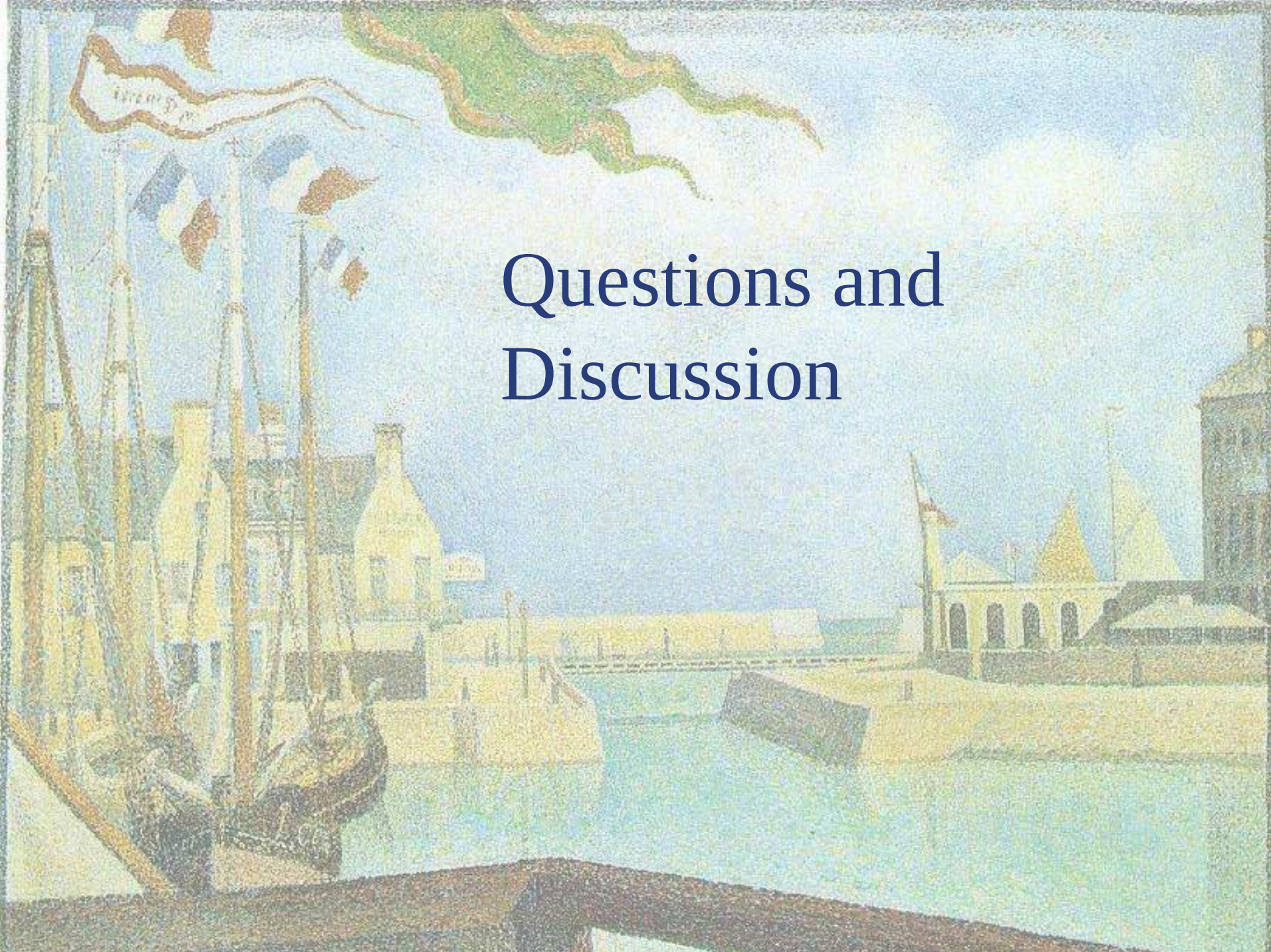
Future Work

- Expansion to additional design phases (such as associating rationale with design artifacts, e.g. UML diagrams)
- Enhanced support for rationale capture by integrating with other tools (such as Configuration Management systems)
- Study of multi-user rationale
- Additional experimentation with longer-term use, more subjects, larger projects



Acknowledgements

- My advisor: Dave Brown
- My committee: George Heineman, Carolina Ruiz, Feniosky Peña-Mora
- My experiment subjects
- My parents and my friends
- My co-workers at CRA
- My professors and colleagues at WPI

A painting of a harbor scene. In the foreground, the masts and rigging of several sailing ships are visible on the left. The water is a pale blue-green. In the background, there are buildings along the waterfront, including a prominent white building with a red roof and a tall chimney. The sky is a mix of blue and white, with a large, stylized green and yellow cloud or smoke plume on the left. The overall style is impressionistic with visible brushstrokes.

Questions and Discussion