SEURAT: Software Engineering using RATionale

Janet Burge
PhD Proposal Presentation

Advisor: David C. Brown
Committee: George Heineman, Carolina Ruiz, Feniosky Peña-Mora (MIT)
What’s special about today?  
(besides my presentation)

20 February, 2002

20 02 2002
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!
Why is DR not used now?

- Collection can impede design process
- Collection is often tedious
- Designers are reluctant to record “mistakes”
- Collection is very costly
Issues with Design Rationale

Capture

- Is there a non-intrusive way to capture rationale?
- How do you determine what to capture?

Use

- Defines needs
- Provides incentive
- What are the uses of rationale?
- Which portions of rationale are the most useful?

Representation

- Defines needs
- How should rationale be saved for later access?
- Defines needs
- Provides incentive
- Defines needs
Uses of Design Rationale

- Communication
- Evaluation
- Reuse
- Verification
- Maintenance
- Assistance
- Teaching
- Documentation
Why is Rationale useful for Software Maintenance?

- Software lifecycle is very long
- Maintenance costs are high
- Original designers are unlikely to be available
DR in the SW Development Process

- Could be generated at any stage:
  - Requirements
  - Analysis
  - Design
  - Implementation
  - Maintenance
SOFTWARE PRODUCT

Requirements:
- what it must do (F)
- constraints on how
  - NFRs, scheduling, re-use
- User Interface

Analysis:
- Use Cases
  - Collaboration Diagrams

Design:
- Class Diagrams (S)
- Sequence Diagrams (B)
  - for each use-case (F)

Implementation:
- Code
  - what decisions are made that are not captured in the design?
    - error handling
    - persistent storage
    - logic/control/branching
    - algorithms

RATIONALE

“why” for requirements
  - application specific
  - domain specific
  - customer specific
  - alternative or rejected requirements and reasons

why these use-cases
  - alternative or rejected use-cases and reasons
  - why these classes
  - why these interactions

why these types
  - why these types
  - why this visibility
  - why these parameters
  - why these returns

why these classes
  - why these attributes
  - why these methods
  - why these relationships
  - why this order
  - why these messages
  - why these collaborators

why handle errors this way
  - why this type of storage
  - why these control structures
  - why this algorithm
Goal: Use Rationale to Support the Maintainer

- System/Methodology for:
  - Presentation of existing rationale
  - Verification and evaluation of new rationale
  - Propagation of rationale modifications
Questions to be addressed

- How can rationale be used to assist maintenance?
- What is the appropriate level of detail for the rationale?
- Does rationale (used and captured) differ for different types of software modifications?
- Does maintenance rationale differ from original rationale?
- Is rationale more useful for some portions of the design or phases of the design process?
- What is the relationship between rationale at different development phases?
Related Work

Includes:

- Lee: Decision Representation Language - basis for the InfoRat representation
- Peña-Mora: DRIM – Design Recommendation and Intent Model used to augment design patterns with rationale
- Klein: C-Re-CS - consistency checking on requirements.
- Beñares-Alcántara, King: KBDS - evaluation of alternatives and use of keywords
- Bose: Decision Ontology within the WinWin framework
Current Status

- Prototype: Inferencing over Rationale
- Study: Rationale for Software Maintenance
Inferencing over Rationale

Prototype System: InfoRat

Goal:
- Build a simple prototype that can be used to investigate what types of inferences can be made over design rationale
- Illustrate how inferencing over rationale can support validation (of the rationale) and evaluation (of the design)
InfoRat Architecture

InfoRat Engine
• Add Rationale
• Browse Rationale
• Validate & Evaluate Rationale

Background Knowledge:
• Preferences
• Tradeoffs
• Causal Relationships

Claim Vocabulary:
• Domain Vocabulary
• User Supplied Vocabulary

Requirement
Goal
Alternative
Claim
Inferencing

- Syntactic Inferencing - inferencing using the *structure* of the rationale:
  - Missing information
- Semantic Inferencing - inferencing using the *contents* of the rationale:
  - Contradictions, poorly supported choices
  - Inconsistencies
  - Violations of tradeoffs described in the background knowledge
Study: Rationale for Software Maintenance

- Example system: Meeting Scheduler
- Three types of maintenance:
  - Corrective – fixing a bug in a system
  - Perfective – correcting a design flaw
  - Enhancive – adding a new feature
- Rationale collected for original design and each modification
Study Goals

- Better understanding of what software design rationale *is*
- Investigation into how rationale is used and modified during maintenance
- Produce an agenda for further research into using DR in software maintenance
Study Results: Areas to Address

■ Additional studies
  ■ Decision analysis

■ Rationale representation
  ■ Argument ontology

■ Rationale use
  ■ Inferencing options
  ■ Inspection issues

■ Need for a system to support research effort and as a proof of concept
Additional Studies

- Additional studies are needed to examine:
  - What are the key decisions made during development and maintenance?
  - Which rationale is most likely to be used?
  - How is rationale most likely to be used?
  - How does rationale capture and use relate to different types of software maintenance?
Rationale Representation

- **Argumentation Representation**
  - Support for multiple phases of software development
  - Hierarchical rationale even within a phase
  - Support for inferencing

- **Argument Ontology**
  - Capture detailed reasons for decisions
  - Allow abstraction to support inferencing
Rationale Use

- Inferencing
  - Validating the rationale
  - Evaluating the design
  - Tracing/propagating changes to the rationale and design
  - Linking decisions to requirements

- Inspection
  - Encouraging rationale inspection
  - Presentation of applicable rationale (at varying levels of the design)
SEURAT: Software Engineering Using RATIONale

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

**PROGRAM**
- Requirements:
  - System Requirements
  - User Interface
- Analysis:
  - Use Cases
  - Collaboration Diagrams
- Design:
  - Class Diagrams
- Implementation:
  - Code

**SEURAT**

- Argument Editor and Analyzer
- Rationale Repository
- Inference Engine
- Argument Ontology

**User**
Evaluation

Use the prototype system to evaluate:

- **Proposed uses for DR:**
  - Does the inferencing and presentation of rationale improve the effectiveness and efficiency of software maintenance?
    - Verification, Evaluation, Maintenance, and Documentation

- **Representation for DR:**
  - Does it support inferencing?
  - Does it capture rationale at all phases?
Evaluation Method

- Additional Maintenance Studies
- User Studies using SEURAT to assist with maintenance
  - User performs maintenance tasks
  - SEURAT presents and evaluates rationale
  - Evaluation of study results
    - User evaluates usefulness of SEURAT and usefulness of rationale
    - Examiner evaluates maintenance results, time to complete task, and quality of rationale
Expected Results of the Research

- Categorization of uses for DR during maintenance
- Method for propagating rationale changes
- Representation for rationale at multiple levels of the development process
- Ontology representing arguments at different levels of abstraction
- Types of inferences using rationale
- Prototype system using DR to support the maintainer
Discussion