Software Development Tools

Saad Godil & Rashid Hameed
April 11, 2006

EE 382V – Software Architecture and Design Intent

Introduction

- Paper 1: Design Rationale (DR) Systems research paper
- Paper 2: Integrating DR with Process Model research paper
- Conclusion
- Questions

Research Paper I

1. Jintae Lee. **"Design Rationale Systems: Understanding the Issues"**. IEEE Expert, Vol. 12, No. 3, 1997, pp. 78-85

The Why

- Motivation:
  - Design rationale system can improve:
    - Management
    - Collaboration
    - Reuse
    - Maintenance
    - Learning
    - Documentation

The Why

- Objective:
  - Purpose of paper to help researchers and developers of future design rationale systems understand the options and tradeoffs

5

The How

Methodology:

- Informal Survey of major existing design rationale systems
- Discussion with workshop participants, including those in the 1992 AAAI Design Rationale Capture and Use Workshop

6

The What

Narrowed down to Seven main issues:

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

7

The What

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

8

The What: Issue 1

Common Services classified into four major groups (and who benefits):

1. Better design (designers)
2. Better Maintenance (system maintainers)
3. Learning (new trainee, students …)
4. Documentation (future designers and maintainers)

9

Issue 1: Better Design

- Dependency Management
  - Design as a process of managing dependencies
  - DR can make explicit dependency relations among:
    - Design parts
    - Decisions
    - Arguments
    - Alternatives
  - Allows all of these to work together consistently

10

Issue 1: Better Design

- Collaboration/ Project Management
  - DR provides common foundation when multiple parties are involved
  - Ex: Shared-DRIM: system checks for conflict whenever design makes recommendation, informs all relevant parties, identifies cause of change, and looks for constraint violations

11

Issue 1: Better Design

- Reuse/Redesign/Extension Support
  - Serves as indices to past knowledge
    - EX: SoftDA acquires relationship information about design and requirements and uses it to index documents and codes
  - Designers reuse the rationales themselves
    - Ex: Sibyl uses the goals from past decision rationales to suggest potentially relevant alternatives, and uses both goals & alternatives to retrieve the arguments for the alternatives

12

Issue 1: Better Maintenance

- DR explains design decisions and therefore will also be helpful for maintaining the design
- Most existing systems provide this service using “comments”
- Ex: EES extracts richer development rationales, using them to generate more sophisticated explanations for system maintenance

13

Issue 1: Learning Support

- DR helps both people and system learn mutually
- Ex: Janus has “critics” which provides designer with appropriate recommendation if it encounters a sub-optimal decision

14

Issue 1: Documentation

- Automatically generate documentation
- Helps other than designer
  - Managers can use to evaluate
  - Lawyers can use to determine if design is Intellectual Property

15

The What

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

16

Issue 2: What to Represent

There are two aspects to this:

1. Functional Dependency
2. Generic Structure
   - Decision Layer
   - Design Artifact Layer
   - Design Intent

Issue 2: Functional Dependency

Representation depends on use:

- Argumentation Perspective
  - Used by designers for problem solving
  - Logical structure → explicit
- Documentation Perspective
  - Enable outsiders to understand
  - Only results need to be captured
- Project Management
  - Metrics for project status must be captured
    - Ex: pending issues, deadlines and people responsible for them

Issue 2: Generic Structure

Design Layer
- Issue
- Argument
  - Explicit arguments underlying a decision
  - Relation: support, refutes qualifies
- Alternative
  - Explicit individual alternative
  - Relation to the argument
- Evaluation
  - Explicit evaluation measure
  - Relation: nominal, ordinal, real values
- Criteria
  - Explicit criteria
  - Relation: mutually exclusive, tradeoffs

Issue 2: Generic Structure

…Design Layer:
- Not all sub-layers always present
- gIBIS only represent issue, alternative and argument
- DRL same as gIBIS, but adds criteria layer/constructs
- DRCS provides evaluation layer
- Note: extra layers come with extra over-head

Issue 2: Generic Structure

Design Artifact Layer

- Makes explicit the decision making steps and their relation to design components
- How info relates to individual decision
- Both not always present
- Ex: FR only expresses relation between design components and requirements
  - Useful for simulation, verification….
  - Not able to show alternatives and arguments explored

21

Issue 2: Generic Structure

Design Intent Layer

- Represents info behind design decisions:
  - Intents, strategies, goals and requirements
- Allows system to reason about the goal or intent
  - Ex: using the goals, system can derive criteria for evaluating alternatives

22

The What

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

23

Issue 3: How to Represent

- Informal
  - Unstructured form
  - Ex: audio/video recordings, raw drawings
  - Easy to create but ill-suited for computational services
- Semiformal
  - Parts are usable by computer, rest is informal
  - Ex: Sibyl: users fill put templates and forms. Some fields are natural language, others are selected from a menu of options
- Formal
  - Object and relation defined as formal objects
  - Costly but provides many more computational services

24

Issue 3: How to Represent

- Incremental Formalization
  - Transforms semi-formal to formal
  - Reduces costs
  - Less overhead
    - Captured in semi-formal way
    - Increased benefits when formalized
  - Ex: uiSibyl:
    - Starts with informal requirement descriptions
    - Tries to map keywords to existing formal objects

25

The What

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

26

Issue 4: How to Produce

In order of minimal to maximal system participation

- Reconstruction
- Record-and-replay
- Methodological byproduct
- Apprentice
- Automatic generation

27

Issue 4: How to Produce

- Reconstruction
  - Produce DR without system
  - Allows more careful reflection on representation
  - Has very high cost to produce

28

Issue 4: How to Produce

- Record-and-replay
  - DR captured as they unfold
    - Ex: share database to raise issues
  - Representation is usually informal or semi-formal
  - Formal representation would have very high overhead cost and would disrupt design flow

29

Issue 4: How to Produce

- Methodological byproduct
  - Naturally emerges from the design process method
  - The steps of the method help capture the rationale
  - Ex: EES : developers follow a certain series of steps that EES supports

30

Issue 4: How to Produce

- Apprentice
  - “Looks over developers shoulder”
  - Asks questions when it does not understand
  - Ex: ADD will ask the designer to explain his/her decision if it contradicts its prediction
  - Both benefit:
    - User benefits if system is right
    - System learns something new if system is wrong

31

Issue 4: How to Produce

- Automatic generation
  - Ex: Expert system uses trace of its rule invocations to explain why/how of its action
  - Very appealing: little effort on users part
  - High setup/initial cost
  - Several issues still need to be worked out
    - What part to capture?
    - How to infer?
    - How to assess relevance?
    - How to adapt to current situation

32

The What

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

33

Issue 5: How to Access

- User-initiative system
  - User decides what parts of DR to look at and when and how.
- System-initiative system
  - System decide what parts and when and how
  - Must have knowledge to make intelligent decisions
  - Must present in an unobtrusive method
  - Ex: Janus critic

34

The What

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

35

Issue 6: How to Integrate

- Among users
- Among multimedia objects
- With Design Modules

36

Issue 6: How to Integrate

- Among users
  - How to exchange DR info
  - Possible Sol'n: blackboard
  - Requires common protocol and language

37

Issue 6: How to Integrate

- Among multimedia objects
  - How to integrate different multimedia artifacts: notebooks, sketchbooks, phone conversations, email…
  - Ex: Phidias addresses this problem by having hypertext links to a hypermedia database

38

Issue 6: How to Integrate

- With Design Modules
  - Must integrate with different design components
  - Ex: CAD module databases and simulation packages

39

The What

1. What Services to Provide
2. What to Represent Explicitly
3. How to Represent Rationales
4. How to Produce Rationales
5. How to Access Rationales
6. How to Integrate the system
7. How to Manage Rationales Cost-Effectively

40

How to Manage Rationales Cost-Effectively (Issue 6)

Cost *vs.* Benefits

41

Issue 7: Cost *vs.* Benefit

- DR system not productive if cost outweighs benefits
- Benefits: Services the system provides
- Cost: Resources used in producing/capturing rationales
- Fixed cost: Cost incurred when building a new system or initial knowledge base
  - OK if cumulative benefits from system's use outweigh it.
- Bigger Question: Who will bear the cost for producing DR for a particular artifact and WHY?

42

Issue 7: Cost Bearer?

- Ideal: Cost bearer and beneficiary are the same
  - Uses DR system and gets enough benefit to compensate for the cost
  - Example: **Interactive** acquisition of rationales in which user and system mutually benefit
  - Constraint: Extra time/attention spent interpreting and answering questions must be minimal
- Concern: When the cost bearer is not the same as the beneficiary?

43

Issue 7: Cost Bearer? (Cont)

- Jonathan Grudin of the University of California, Irvine points out that many groupware systems fail exactly because of this mismatch
- Example: Most online meeting schedulers fail because
  - Cost: It requires ALL people to maintain their local calendars online
  - Beneficiaries: Only those who schedule the meetings
- Solution: Grudin suggests a process along with the technology that delivers some benefit to the contributor (system allows designer to send compliments to contributor)

44

Author's Opinions:

- Not all the Issues have been explored at a sufficient depth
- Designing Cost effective system is one of the most urgent issues design rationale researchers face
- If not cost-effective → may not be used or be counter-productive
- Management not beyond research concern
- Eventual goal should be to have a practical system
- Urgent Need for methods to produce formal design rationales at less cost
- Integration is an important issue
- These neglected areas, once addressed, will enable design rationale systems to contribute more to design research.

45

Research Paper II

J.E. Burge, D.C. Brown **"Integrating Design Rationale with a Process Model"**. Workshop on Design Process Modeling, Artificial Intelligence in Design '02, 2002

46

Integrating Design Rationale

- What is a Process Model?
- Types of Process Models
- Impact of Process Model Integration with DR
- Hierarchical Plan Selection and Refinement (HPS&R) Process Model
- Unified Process Model
- Model Summary

47

What is a Process Model?

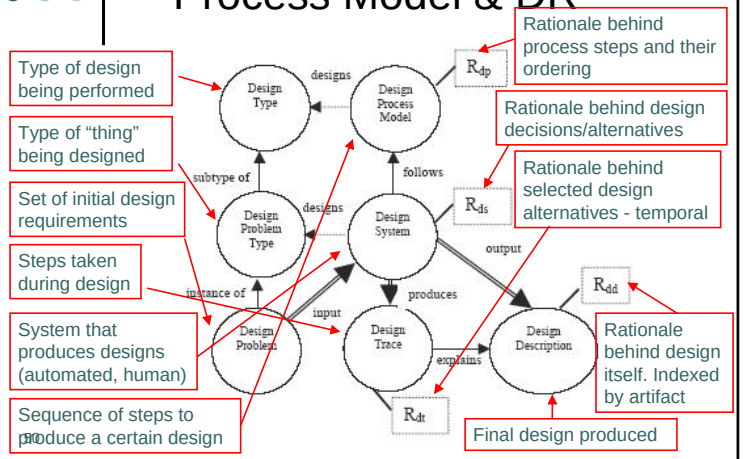
- A model that encapsulates the set of steps and activities that take place in achieving design goals/objectives
- Different process models for different types of design domains
- Serves as a **prescription** of how the design should be done
- Captures design process alternatives and their rationale explicitly
- Not only guides decisions, but provides design knowledge to help in those decisions

48

Types of Process Models

- Some are for specific designs e.g. HPS&R for AIR-CYL
- Others like Tate and Nordlund's design process roadmap is intended for use as a general model
- Some models specifically capture design rationale along with the design
  - Blessing's Design Matrix (includes alternatives)
  - Ganeshan's model

Process Model & DR



Rationale Definitions

R_{dp}	Rationale for Design ***Process***
R_{ds}	Rationale for Design ***System***
R_{dt}	Rationale for Design ***Trace***
R_{dd}	Rationale for Design ***Description***

Impact of Process Model on DR

- DR with a process model is richer in content
  - Rationale for both design decisions **AND** process model
- DR follows process model structure since design is based on process model
- Process model makes design modification easy to comprehend
  - Rationale and process model coherent
- Disadvantage: If process model produces DR, queries for DR may not be in natural language

HPS&R Model (AIR-CYL)

- Design Structures and Plans Language (DSPL) used in AIR-CYL systems
- HPS&R is specific for a well-studied design problem
  - Supports selection from only pre-determined sets of design alternatives/plans
  - “Specialists” assigned to a specific task
  - Specialists select design plans or call more specialists (hierarchical refinement)
- DSPL Selector selects appropriate plans

53

How Does Rationale Fit in HPS&R?

- Each plan step defines a value for a design attribute

R_{ds}	Alternatives and reasons for rejecting them are both encoded
R_{dt}	In case of system failure - rationale behind steps taken to make the correction in temporal order
R_{dd}	In case of system failure - reasons for failure, cause of failure and rationale behind process taken to find the cause

54

How Does Rationale Fit in HPS&R? (cont)

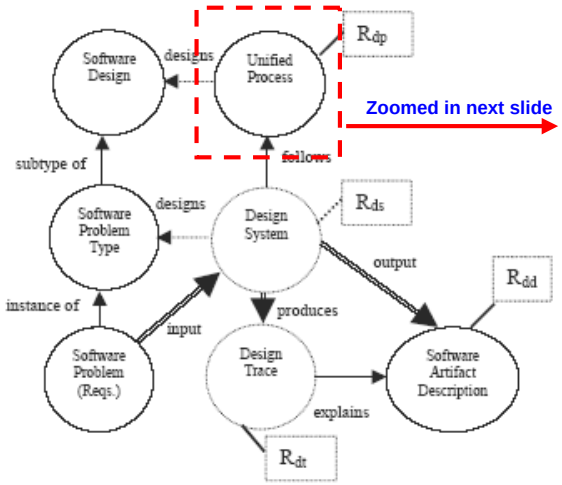
- Rationale helps HPS&R in
  - Failure handling by providing a list of alternatives
  - Detecting if the process model itself needs modification (through circular feedback of failures over time)
  - Replaying design on a different set of requirements
  - Assessing the impact of changing a parameter value

55

Unified Process Model

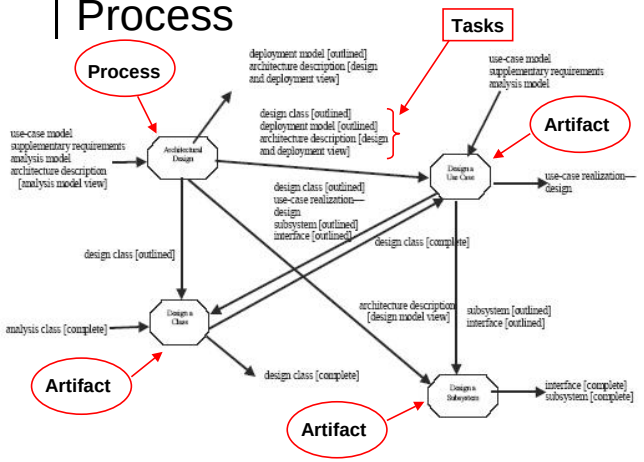
- HPS&R is specific for a well-studied design problem
- Unified Process model is used for generic software design (not automated)
- Large and complex
  - Not all of it is appropriate for every design task
  - Developers can use R_{dp} to hand pick parts of the process specific to their design needs
- R_{ds} contains many options/alternatives
  - Gives developers access to readily available information and avoids fixating on first option chosen

56



57

Design Workflow for Unified Process



58

How Does Rationale Fit in Unified Process

- Each process in the workflow has certain tasks it must carry out to produce the artifacts
- Architectural Design Process
  1. Identifying network nodes and their configuration
  2. Identifying subsystems and their interfaces
  3. Identifying architecturally significant design classes
  4. Identifying generic design mechanisms
- Not all tasks need to be performed. For example, item 1 needed only in distributed systems

59

How Does Rationale Fit in Unified Process (cont)

R_{dp}	Since some tasks might not be needed, rationale can be generated for which tasks are necessary and what order should they be performed.
R_{ds}	Rationale for what steps and information needed to complete the task. Also for frequently considered alternatives (different network configurations in distributed systems)

60

Model Summary

- Each model is useful for different design domains
- HPS&R (**specific**): Rationale is generated automatically for a well-studied parametric design
- Unified process model (**generic**): Generic and especially used for software development
- Author mentions a lot more research is required toward Unified process model
  - Determine where each type of rationale fit to assist in software development and process definition

61

Conclusions

- Issue 1: Provided services to everyone
- Issue 2: What to represent was left open to implementation
- Issue 3: How to represent rationale was open to implementation
  - Author seems to indicate that it would have to be formal
- Issue 4: How to produce rationale was left open to implementation
  - One example of complete automation

62

Conclusions (cont)

- Issue 5: How to access rationale was left open to implementation
  - Author has given examples of both system driven and user-driven
- Issue 6: Indirectly addressed the issue of cost-effectiveness
  - Indexing and organization of rationale increases the benefit
- Issue 7: How to integrate the rationale into the process model was addressed. However, did not address any of the issues

63

Questions

64