ArchiRationale: Feature-Based Rationale Management System for Supporting Software Architecture Adaptation

University of Twente
Software Engineering Group
29 January 2009

Bedir Tekinerdoğan, Hasan Sözer, Mehmet Aksit

Bilkent University
Department of Computer Engineering
06800 Bilkent Ankara
http://www.cs.bilkent.edu.tr/~bedir
bedir@cs.bilkent.edu.tr
Rationale for Software Architecture

- Improved understanding because of a higher level abstract specification
- Guides subsequent phases since it embodies earliest design decisions
- Supports stakeholder communication
- Support for large-grained reuse
- Enables to evaluate system before it is implemented
- Management of software development activities
Viewpoint:
- A pattern or template from which to construct individual views.

View
- a model of the system from the perspective of one or more concerns which are held by one or more stakeholders
An architect must consider the system in three ways:

- How is it structured as a set of implementation units?
  Module Views (Module viewtype)

- How is it structured as a set of elements that have run-time behavior and interactions?
  Component-and-connector views (C&C viewtype)

- How does it relate to non-software structures in its environment?
  Allocation views (allocation viewtype)
Views give us our first principle of architecture documentation:

*Document the relevant views, and then add information that applies to more than one view.*
Software Architecture Patterns

- express a **fundamental structural organization** schemes for software systems.
- provide a set of predefined subsystems
- specify their responsibilities
- and include rules and guidelines for organizing the relationships between them
Architecture Evaluation Techniques

- Analytic Models
- Simulations
- Question-based
  - Questionnaire-based
    - List of general open questions that apply to all software architectures.
  - Checklist-based
    - detailed set of questions
    - after experience of analyzing common set of systems
    - usually domain specific
  - Scenario-based analysis
The software architect

software architect is currently equipped with a broad set of techniques to design a software architecture:

- Architectural Modeling (Views, ADLs)
- Architectural Patterns
- Architectural Evaluation Techniques
- Architectural Methods
- ...

Using these approaches the software architect makes a wide range of design decisions that leads to the selection of a particular design alternative.

“I am the Architect. I created The Matrix. …”
Design Rationale

- The reasons behind the design decisions, the justification, the alternatives considered, the trade-offs evaluated, and the argumentation that led to the decision is defined as *design rationale*.
Design Rationale Management

- The explicit capturing, documentation and usage of the design rationale is important for many different reasons such as design communication, design evolution, design maintenance, design verification, and design reuse.

- In general practitioners recognize the importance of documenting and usage of design rationale to support the reasoning about design choices.

- Unfortunately architecture design rationale is still not being documented in a consistent manner.

- Methodology and tool support for design rationale capture and usage is necessary.
Design Rationale Management System

- Rationale management systems enable the capturing, modelling and accessing of design rationale.

- Classification
  - process-oriented rationale management system
  - feature-oriented rationale management system
Process-Oriented Rationale Management Approaches

- Often applied for dynamic design domains in which the design principles are not well-established.

- There is no well-defined set of features or options. Different new questions, arguments can be raised during the development process.

- Considers the design rationale usually as a history of the design process.

- The representation of design rationale in this approach is usually graph-based in which the nodes represent questions, positions and arguments and the links the relations among these concepts.

- Examples: Ibis, QOC, DRL, PHI etc.
Feature-based Rationale Management Approaches

- evolved from the process-based rationale approaches but differ in the rationale capturing approach.

- In a feature-based rationale approach the design rationale is **based on features** of a system rather than the arguments raised during the development process.

- Feature-based rationale approaches are typically used for **well-defined domains with established design rules**.
Example Feature Diagram for Insurance Systems

Legend:
- mandatory feature
- optional feature
- alternative feature
- or-feature
Design Rationale - Focus

- In general the overall design rationale and its context becomes very large,

- and likewise it is **impossible to capture and represent an entire design rationale** explicitly

- Even if this would be possible, it is also **not always necessary or desired** to represent the complete design rationale.
The focus of this talk

- We expect that a software architecture is provided that needs to be adapted for particular quality concerns.

- The design rationale capturing process starts after the architecture design and considers then the design decisions, the alternatives and the argumentation behind the adaptation of the architecture.

- As such we do not consider capturing design rationale from the initial requirements document to the architecture design.
Context - Trader Project

Industry-as-laboratory project

- **Television Related Architecture Design to Enhance Reliability** (Trader)
- **Context:** Embedded Systems and in particular consumer electronics (Digital TV)
- **Period:** Sept. 2004-Aug. 2009
- **10 partners (industrial and academic)**
- **22 fte/yr, 7 PhDs, 2 Postdocs,**
- **around 3-4 million euro**
- **funded by The Netherlands Ministry of Economical Affairs under the Bsik programme**
Reliability

- The probability that a system will continue to function without failure for a specified period in a specified environment.

**How to prevent failures?**
- fault prevention
- fault removal
- fault tolerance

Fault

Error

Failure

system state that may cause a failure

delivered service deviates from the correct service

the cause of an error

Fault-Tolerant Design

- designing a system so it will continue to operate,
- possibly at a reduced level,
- rather than failing completely, when some part of the system has an error.

Fault Tolerance:
- Error detection
- Diagnosis
- Recovery
**Example - Mediaplayer**

- *Stream* reads the input media by bytes/blocks and provides buffering, seek and skip functions.
- *Demuxer* demultiplexes (separates) the input to audio and video channels, and reads them from buffered packages.
- *Mplayer* connects the other modules, and maintains the synchronization of audio and video.
- *Libmpcodecs* embodies the set of available codecs.
- *Libvo* displays video frames.
- *Libao* controls the playing of audio.
- *Gui* provides the graphical user interface of MPlayer.

*enhance the architecture for fault tolerance*
Adapting Architecture

- each fault-tolerant architecture will be the result of a set of design decisions
- and there is a rationale behind these decisions (e.g. to reduce cost, to achieve high availability/performance).
Alternative Recovery Designs
Analysis vs. Design Rationale

- How to select and analyze a particular design alternative (alternative space analysis)

- What decisions are taken during the selection of the design alternative? (design rationale management)
**Feature Modelling** defines the metamodel for defining feature models.

**Feature Decisions** part includes the concepts for selecting features and capturing the rationale behind these features.

**Design Alternatives** part represents the approaches for realizing the selected feature decisions in the *Feature Decision Rationale*.

**Architecture Design** part of the metamodel includes the concept for modelling architectures.
Architectural Rationale: Integrated Tool

- Feature-based rationale management tool
- built in the Eclipse Platform
- To implement the rationale management approach
- customizes and integrates several open-source tools that are provided as Eclipse plug-ins
- and all based on the XML technology.

<table>
<thead>
<tr>
<th>The activities of the approach</th>
<th>Tool support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture Design</td>
<td>ArchStudio</td>
</tr>
<tr>
<td>Modeling Architectural Tactic Space of Quality</td>
<td>XFeature</td>
</tr>
<tr>
<td>Capturing Design Rationale</td>
<td>XFeature, Recovery Designer</td>
</tr>
<tr>
<td>Design Alternative Application</td>
<td>ArchStudio</td>
</tr>
<tr>
<td>Accessing Design Rationale</td>
<td>XQuery</td>
</tr>
</tbody>
</table>
The Approach

**Modeling/Preparation**
- Architecture Design
- Modeling Architectural tactic space (family feature model)

**Capturing Design Rationale**
- Instantiate family feature model (selected features) representing fault tolerant view
- For each feature selection define the rationale
- Set of feature selections define fault tolerance view
- Fault tolerance view is related to corresponding alternative design template
ArchStudio

- open-source software and systems architecture development environment
- uses an XML-based architecture description language, xADL
- integrates several tools for modeling, visualizing, analyzing and implementing architectures that are expressed in xADL.
- we model the existing architecture design and design alternatives with the Archipelago tool of ArchStudio.
- Archipelago is a front-end tool that provides a graphical user interface to create, view and modify an architecture description specified in xADL.
Defining Feature Models

XFeature:

- XFeature is a **feature modeling tool** which supports the modeling of product families and of the applications instantiated from them.
- It employs an **XML-based approach** to express the feature models and it uses XML schemas to express the meta-models.
- It is possible to **customize the XFeature tool** with respect to the type of feature diagrams that can be specified and the graphical notations that are used for representing feature models.
- We have customized the XFeature tool for **ArchiRationale** by **specifying the meta-model of design rationale**.
Architectural tactic is defined as **characterization of architectural decisions** that are needed to achieve a desired **quality attribute** response (e.g. Fault tolerance).

Architectural tactics are derived using a domain analysis process and represented in a **family feature model**.

Unlike existing feature-based rationale management approaches we focus on modeling quality features instead of product features.
Use meta-metamodel of X-Feature to define design rationale metamodel

Based on design rationale metamodel define a Architectural Tactic Space (Family Model)

Family Model is input to XSL program that generates a metamodel based on a provided family model. (e.g. Fault Tolerance Family Feature model).

Fault tolerance views are defined based on Architectural Tactic Space Meta model.
- **Fault Tolerance Family Model**, which defines the technical space of fault tolerance.

- Several **application fault tolerance models** by selecting features from the family feature model.
Every feature diagram that is specified with respect to the Fault Tolerance architectural tactic space is a *Fault Tolerance View*.

A fault tolerance view captures and stores the design rationale related to selections and choices made with respect to the fault tolerance architectural tactic space.

A fault tolerance view relates to design template (modeled with the tools of ArchStudio, and stored as separate XML file).
Two different layers of decisions

- **decide on the features** for recovery that need to be realized by the enhanced architecture.

- Decide on the decomposition of the architecture based on design template

**Example**
- Select local recovery
- Decide which components need to be made recoverable
Alternative Recovery Designs

- Recovery
  - Error Type
  - Granularity
  - Technique
    - Persistence
      - Transient
      - Permanent
    - Global
    - Local
  - Total Compensation
  - Backward Recovery
  - Forward Recovery
  - Replication
  - Graceful Degradation
  - Stable Storage
  - Check-Pointing
  - Log-Based
  - Uncoordinated
  - Coordinated
  - Communication Induced
  - Non-Blocking
  - Blocking
  - Pessimistic
  - Optimistic
  - Causal

- Key
  - Feature
  - Optional Sub-Feature
  - Mandatory Sub-Feature
  - Alternative Sub-Features
Adapting Architecture

- After a design template is selected, the next step is to adapt the architecture accordingly.
- Several additional design alternatives can exist for adapting the architecture.
- These alternatives are specific to the selected design template.
- It requires the evaluation of several criteria and dedicated analysis techniques.
- The design rationale at this step is basically formed by the criteria set being evaluated, the evaluation method (e.g. based on simulation or analytic models), types of formal models employed, their parameters, assumptions and the analysis results.

- We use **RecoveryDesigner** tool
Querying Design Rationale

- XQuery is a language to query XML data sources.

- XQuery Development Tools provides support of XQuery inside the Eclipse platform including code templates, semantic analysis, query validation and execution.

- Using the XQuery Development Tools, we have created and validated parameterized query functions that are defined to query XML data conforming to the ArchiRationale XML schemas created with the XFeature Tool.
Artifacts delivered through the rationale process are stored in a repository and the necessary information can be queried.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getDesignDecisions()</td>
<td>Retrieve all the design decisions and their properties</td>
</tr>
<tr>
<td>getDesignOptions()</td>
<td>Retrieve all the design options and their properties</td>
</tr>
<tr>
<td>getDecisionProperties(p)</td>
<td>Retrieve a particular property regarding the design decisions</td>
</tr>
<tr>
<td>getOptionProperties(p)</td>
<td>Retrieve a particular property regarding the design options</td>
</tr>
<tr>
<td>queryDecisionProperties(p,q)</td>
<td>Retrieve the design decisions that contain the query text in the specified property</td>
</tr>
<tr>
<td>queryOptionProperties(p,q)</td>
<td>Retrieve the design options that contain the query text in the specified property</td>
</tr>
</tbody>
</table>
Characterization of ArchiRationale

- Focus on adaptation of architecture
- Defining rationale based on quality concern
- Focus on well-defined domain

<table>
<thead>
<tr>
<th>Goal</th>
<th>Support for Design Adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Type</td>
<td>Feature-Based; Features are techniques to realize quality</td>
</tr>
<tr>
<td>Services</td>
<td>Design Documentation, Constraint Checking, Design Adaptation</td>
</tr>
<tr>
<td>Represented Information</td>
<td>Feature Decisions, Design Options, Design Alternatives (three layers)</td>
</tr>
<tr>
<td>Representation Method</td>
<td>Formal (features) and semi-formal (decisions)</td>
</tr>
<tr>
<td>Capture Method</td>
<td>Methodological by-product after the initial architecture has been designed</td>
</tr>
<tr>
<td>Access Method</td>
<td>User-Initiated</td>
</tr>
<tr>
<td>Domain</td>
<td>Techniques to implement Quality Concerns (Architectural Tactic Space)</td>
</tr>
<tr>
<td>Design Type</td>
<td>Adaptation</td>
</tr>
<tr>
<td>Design Phase</td>
<td>Post-Architecture Design, Architectural Maintenance</td>
</tr>
<tr>
<td>Number of Designers</td>
<td>Any</td>
</tr>
<tr>
<td>Notation</td>
<td>Feature Modeling; Architecture description, XML-based</td>
</tr>
</tbody>
</table>
Conclusion

- A **meta-model** that defines the concepts and the relations among the architecture and the *rationale management system*.
- A systematic **rationale management approach** for documenting and accessing the rationale for architecture design alternatives.
- Because the rationale system is focusing after the architecture design process, the approach is **complementary and agnostic to the applied architecture design method**.
- An integrated tool environment **ArchiRationale** that supports the capturing and access of design rationale and the related artefacts.