Capturing and Replaying Architectural Knowledge through Derivational Analogy

Ibrahim Habli
Tim Kelly
University of York, UK

Architecture Reuse

• Architecture definition is intellectually intensive, critical and costly
• Reuse of architecture is a potential means to reducing costs and improving maturity quality
  – A Motivation: Product Line Engineering

• Knowledge underlying architecture is as important as the architectural views and models
• Reuse of architecture design in isolation from architectural reasoning may violate hidden design assumptions and the integrity of the entire system
  – A particular concern in our domain
Architecture Knowledge Reuse through Analogy

- **Analogy** is a recognised approach to knowledge reuse ("This is like that")
  - Analogies can be drawn between previously designed architectures and a new one
- Analogy already used in other SE domains
  - e.g. Requirements Engineering, Frame-based reasoning
- In software architecture, design knowledge is embodied in the history of architectural **process**
  - e.g. application of tactics and design patterns, making assumptions, and dismissing alternatives

Transformational Analogy

- If a solution has successfully solved an earlier problem, it can be applied to solve a ‘similar’ problem
  - Transformation: old $\rightarrow$ new requirements, old $\rightarrow$ new solution
- In context of architecture design
  - Copying a successful architectural design solution from a past design
  - Reapplying old design solution to a new design problem
    - As long as a sufficient number of relevant goals and conditions exist between the two design problems
- ✓ Advantage: large potential payback in terms of cost and time
- ✗ Disadvantage: limited consideration of the assumptions and dependencies of design decisions and rationale behind the rejection of design alternatives
Derivational Analogy

• Instead of reusing past solutions directly, derivational analogy reuses and replays the process (derivation) leading to these past solutions
  – Particular design steps or routes are skipped if the design assumptions do not hold in the context of the new problem
  – Mismatches between earlier design decisions and the new problem’s requirements can be detected and overcome

✔️ Advantage: more confidence in the design solution
❌ Disadvantage: more effort and involvement of the architects and designers (as compared to transformational analogy)

Replaying Architectural Reasoning through Derivational Analogy

• Steps of Process
  – Capture and representation
    • Capturing the process, not just the results (time-ordered sequence important, e.g. in ADD)
  – Relevance and Retrieval – can borrow from RE work
  – Adaptation
  – Evaluation (to close the loop – important)

• Constraints on Design Replay [Mostow89]
  – Functional and quality specifications
  – Software cost
  – Context conditions – e.g. different environments, captured in domain model?
  – Documentation standards
  – Design methodology
Capturing the Architectural Reasoning

Architectural Process

- **Architecture Knowledge** = \{drivers, decisions, analysis\}
- Quality Attribute Capture
  - Scenarios
- Architecture Design
  - Attribute-Driven Design Method (ADD)
- Architecture Evaluation
  - Architecture Tradeoff Analysis Method (ATAM)
- Picture possibly a little deceptive
Drivers and Design

Quality Attribute Scenarios:
- Editing a Geographic Feature
- Retrieve Data
- Change GIS Data Format
- Add GIS Component
- Updating Real-time Data
- Interface GIS with External Software

Architectural Decisions:
- Maintain semantic coherence
- Published interfaces
- Client-Server style
- Minimize clients & servers interaction
- Data accessor design pattern
- Façade design pattern
- Concurrency
- Caching
- Increase hardware resources
- Mediator design pattern
- Geographic information standards
- Separation unit operation
- Generalization style
- Adapter (or wrapper) design

ATAM Extract

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>ATUM FAILURE: Data Source Failure Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Availability</td>
</tr>
<tr>
<td>Environment</td>
<td>Normal operation</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Data source failure</td>
</tr>
<tr>
<td>Response</td>
<td>The system detects the failure, restarts the connection to the backup data source, logs the failure, and notifies the concerned administration</td>
</tr>
<tr>
<td>Response Measure</td>
<td>Data source availability detection time/recovery time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Architectural Decision</th>
<th>Sensitivity</th>
<th>Tradeoff</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data replication</td>
<td>S3</td>
<td>R3</td>
<td>S4</td>
</tr>
<tr>
<td>Caching</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- S3: Minimum number of backup sources is one; nevertheless, adding more than one backup source improves availability
- T2: The availability improves when active data sources synchronize more frequently with passive sources. Yet, this makes the system less efficient
- R4: The amount of data loss might be large if the active data source synchronizes less frequently with the passive sources
ATAM Results

• Application of ATAM to GIS uncovered:
  – 16 sensitivity points
  – 10 tradeoff points
  – 13 risks

• Need to check (in replay) that we share the same understanding of
  the sensitivity points, tradeoffs and risks

• Priorities can change – can affect order of architectural decisions

Replaying the Architectural Reasoning
Additional Safety-Related Services

- The new architecture (largely) share the same drivers with the past architecture, e.g. modifiability concerns
- A major difference lies in the safety-related services required for the new architecture (safety 'constraint' can influence our prioritisation of other quality attributes)
- Safety is a system property and hence needs to be considered at the architectural level

<table>
<thead>
<tr>
<th>Safety Services of Ambulance Dispatch System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source:</strong></td>
</tr>
<tr>
<td><strong>Stimulus:</strong></td>
</tr>
<tr>
<td><strong>Artifact:</strong></td>
</tr>
<tr>
<td><strong>Environment:</strong></td>
</tr>
<tr>
<td><strong>Response:</strong></td>
</tr>
<tr>
<td><strong>Response Measure:</strong></td>
</tr>
</tbody>
</table>

Design Adaptation

- Recorded architectural knowledge shows that all requests to GIS services are processed through 'Mediator' Subsystem,
  - Due the application of the mediator design pattern
- Recorded architectural analysis reveals that the mediator may constitute a single point of failure
  - A risk that cannot be tolerated in safety-related services

  - To reduce impact of mediator on the safety-critical services a new explicit architectural decision is made, *freedom of interference by non-safety functions*
    - To separate safety-related services from non-safety services
Conclusions

• Systematic articulation of architectural drivers, design decisions, and analysis is the foundation for replaying the architectural process
  – e.g. ATAM helps articulates parameters needed for effective reuse of architectural design decisions
  – Process sequence is important
  – Alternatives (as with IBIS) also useful

• Our context: Capturing the design reasoning is essential in the course of justifying the dependability of critical systems
  – Certification authorities mandate transparency with respect to the carrying out of the design process

• Practicality: Process of making architectural decisions may be based on intuition and tacit knowledge
  – Difficulty of articulating such decisions, let alone reusing them!