Architecture View Integration and UML

Alexander Egyed
for the Annual Research Review

March 10th, 1998

Outline

- View Integration in USC-CSE Architecture Workshops
- UML View Integration
  - Example Problem
  - Short Survey of Techniques
  - Collaboration with Rational: Rose Architect
- Future Plans
1994: An architecture should:
- Address all of the stakeholders' concerns
- Be able to abstract and relate the different architectural views
- Maintain consistency between the various views
- Include means of analysis

1997: Need for Completeness, correctness, consistency, view specific constraints, integration, automatic enforcement of views
- Particular important context is the UML view integration

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Concern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Schedule and budget estimation</td>
</tr>
<tr>
<td></td>
<td>Feasibility and risk assessment</td>
</tr>
<tr>
<td></td>
<td>Requirements traceability</td>
</tr>
<tr>
<td></td>
<td>Progress tracking</td>
</tr>
<tr>
<td>User</td>
<td>Consistency with requirements and usage scenarios</td>
</tr>
<tr>
<td></td>
<td>Future requirement growth accommodation</td>
</tr>
<tr>
<td></td>
<td>Performance, reliability, interoperability, etc.</td>
</tr>
<tr>
<td>Architect</td>
<td>Requirements traceability</td>
</tr>
<tr>
<td></td>
<td>Support of tradeoff analyses</td>
</tr>
<tr>
<td></td>
<td>Completeness, consistency of architecture</td>
</tr>
<tr>
<td>Developer</td>
<td>Sufficient detail for design</td>
</tr>
<tr>
<td></td>
<td>Reference for selecting / assembling components</td>
</tr>
<tr>
<td></td>
<td>Maintain interoperability with existing systems</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Guidance on software modifications</td>
</tr>
<tr>
<td></td>
<td>Guidance on architecture evolution</td>
</tr>
<tr>
<td></td>
<td>Maintain interoperability with existing systems</td>
</tr>
</tbody>
</table>
Example of UML View Redundancy

UML Views

- Each view tells me something new even though similar information is repeated.

Problems with View Redundancy

Redundancy is risky!
Problems with View Redundancy

View Integration Techniques

Expertise-based (ad-hoc)  
Model-based (ACME, UML)

Heuristics-based (patterns, styles)  
Process-based (WinWin, LCA)

View-based (Diagrams, ADLs)

Reduce risks early on...
There is only one model and its views are derived from this model. The stakeholders (e.g. architects) derive views from that model, fill in the missing blanks, and reconcile the changes with the model.

**Model-based Integration**

- System Model
- A View Independent Representation
- Abstract
- View
  - View Synthesis
  - Iterate
- View Analysis
- Reconcile

**View-based Integration**

- Defining views (e.g. diagrams) which are needed
- Formalizing those views maybe refine them
- Identify information in views which overlap with other views
- Define rules which define mapping between those views (e.g. constraints, etc.)

Example of such an approach: Mapping algorithm from Collaboration diagram to State diagram at the University of Montreal.
Process-based Integration

Waterfall Model:
Strongly defined borders
Usually, work is repeated for each stage for different stakeholder and views.

Spiral Model:
The difference between stages are defined through their level of detail but not through some artificial border.

Heuristics-based Integration

- Identify recurring patterns, frameworks, styles, and other forms of relationships.
- Describe characteristics of those patterns, etc.
- Define rules which describe those patterns (e.g. constraints, etc.) and their usage.

Example of such an approach:
Rose Architect - Captures transitive relations between classes
Rose Architect (RA)

RA was conceived by Philippe Kruchten:
- A step into the single model idea
- Focuses on integrating class diagrams only
- Divides classes into 'planes' of views
- Abstracts complex interrelated class diagram views into simpler views by applying heuristics - e.g. key classes
- Heuristics are described in form of rules which are used to simplify the model.

RA Demo
RA Demo

RA: What is the benefit?

- Indirect relationships between classes in different class diagrams can be depicted easily
- Dependencies which can be derived automatically don't need to be described manually
- Consistency between detailed and simplified class diagrams can be checked
Future Plans

- Evaluate Survey Results
- Analyze Library UML Projects
  - Formalizing Integration Problems and Issues

Library Projects

- Source of data for analyzing view integration techniques.
- Source of prioritizing integration issues based on what they did poor and well.
- All teams used Rational Rose to describe their architecture.
Future Plans

- Evaluate Survey Results
- Analyze Library UML Projects
  - Formalizing Integration Problems and issues
- Integrating ACME with UML
- Map various UML views
- Map to other views not supported by UML (e.g. ADL or DFD)
- Focus on strongest (combination of) resulting approaches