Using Patterns to Integrate Architectural Views in UML

Alexander Egyed
Center for Software Engineering
University of Southern California

GSAW 99
March, 1999
Outline

• Need for Architectural Evaluation
• Mismatches in a TT&C Example
• Using Rose/Architect to identify Mismatches
• Goals and Limitations
Software Development seems to have a diagram (view) centric problem solving approach.

Although, these views are very useful on their own, there is only little which keeps them together. This is a problem because:

  => they are standalone/independent
  => they hardly share modeling elements
  => they are for different audiences/stakeholders (different interpretations)
  => they are often used concurrently
The View Integration Problem

• That means that...
  => Same/similar information is entered multiple times
  => Related information must be kept consistent manually

• Problem is that ...
  => often not apparent what information is same/similar
  => information often cannot easily be ‘translated’

This work is about integrating architectural views in UML so that it provides more than just structural assistance and allows model information to be shared among views.
Integrating... what and why?

• **Why Architecture?**
  => Still ‘high-level’ enough for defects to be less ‘catastrophic’.
  => Already ‘low-level’ enough to be less ambiguous.

• **Why OO/UML?**
  => Because both dominate the market/standardized.
  => UML is used even beyond OO.
  => Because their views are commonly understood and used.
  => UML Notation is extensible.
  => Some progress made by others.
Outline

• Need for Architectural Evaluation
• Mismatches in a TT&C Example
• Using Rose/Architect to identify Mismatches
• Goals and Limitations
Level 2 Architectural View

Excerpt of a Satellite Telemetry Processing, Tracking, and Commanding System (TT&C) [Alvarado GSAW 98; translated from OMT]
My Interpretation of previous TT&C Architecture Excerpt

- Spacecraft Command Plan
  - Command
  - Command Block

- Spacecraft Command Activity

- Spacecraft Command History
  - results in

- Spacecraft Command Transmission
  - requires

(has, requires, results in)
Where are the Mismatches?

Even in an excerpt like this one, inconsistencies between those views cannot be seen easily. Just imagine a more complete TT&C project with thousands of classes. You still want to compare it manually?

Currently, software developers spend a lot of time, comparing various views (diagrams, source code, requirements) to ensure their conceptual integrity.

Whatever automated assistance we can give to support mismatch identification between and within views will not only reduce frustration but will also reduce cycle time.
Outline

• Need for Architectural Evaluation
• Mismatches in a TT&C Example
• Using Rose/Architect to identify Mismatches
• Goals and Limitations
Simple Rule

Resulting patterns must be legal according to UML rules
### Some Rule Examples

There are currently about 80 rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Component</th>
<th>Connector →</th>
<th>Component</th>
<th>Connector →</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Class</td>
<td>Generalization →</td>
<td>Class</td>
<td>Generalization →</td>
<td>Class</td>
</tr>
<tr>
<td>2</td>
<td>Class</td>
<td>Generalization →</td>
<td>Class</td>
<td>Dependency →</td>
<td>Class</td>
</tr>
<tr>
<td>3</td>
<td>Class</td>
<td>Generalization →</td>
<td>Class</td>
<td>Association →</td>
<td>Class</td>
</tr>
<tr>
<td>4</td>
<td>Class</td>
<td>Generalization →</td>
<td>Class</td>
<td>Aggregation →</td>
<td>Class</td>
</tr>
</tbody>
</table>

[...]

| 9    | Class     | Dependency → | Class    | Aggregation → | Class    |

[...]

| 66   | Class     | ← Aggregation | Class    | Generalization → | Class    |
| 67   | Class     | ← Aggregation | Class    | Dependency → | Class    |
| 68   | Class     | ← Aggregation | Class    | Association → | Class    |
| 69   | Class     | ← Aggregation | Class    | Aggregation → | Class    |
Transformation - Rose/Architect

Layer 1 Diagram Classes

One possible path using the helper classes in layer 2

Use Rule 3

Use Rule 9
Rose/Architect Tool

![Rose/Architect Tool screenshot](image)
Mismatch Comparison

Abstracted Level 2 TT&C Architecture using Rose/Architect

Difference between both view may indicate mismatches

Absence of differences must not imply correctness.

My TT&C Overview Architecture
• Need for Architectural Evaluation
• Mismatches in a TT&C Example
• Using Rose/Architect to identify Mismatches
• Goals and Limitations
View Integration Activities

Previous Example showed only one technique!
Useful View Integration Needs to Incorporate Many More ...

System Model

View Synthesis

View Analysis

Mapping
(Cross-Referencing)
- names
- traces

Differentiation
(Comparison)
check for consistency
and completeness

Transformation
(Extraction)
- CollD to StateD
- RoseArchitect
- Merging and Splitting
Goals and Limitations

Some Issues:

- What other Techniques?
- State Explosion Problem?
- Mismatch Identification vs. Resolution?
- Scalability?
- Syntactic Integration vs. Semantic Integration

Great Benefits:

=> There ARE automated ways of identifying mismatches between views.
=> Computer is more efficient in comparing views.
=> Mismatches may be identified as early on as they are created (e.g. agents).