UML View Integration

Alexander Egyed

Annual Research Review

February 8-9, 1999

Outline

- Problem Solving in UML
- Examples of Mismatches
- Integration Activities
- 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.
Problem Solving in UML
Examples of Mismatches
Integration Activities
Goals and Limitations

Mismatch Example 1

Model Elements of Layer 1
Flight Controller
Mechanic
Flight
Pilot
Aircraft
Flight Authorization

Model Elements of Layer 2
Boeing 747
Mechanic
Pilot
Flight Plan

Possible Mismatch: dependency of Flight to Flight Controller not reflected in lower level view.
Mismatch Example 2

Outline

- Problem Solving in UML
- Examples of Mismatches
- Integration Activities
- Goals and Limitations
Transformation - SCED

Using SCED to transform Scenarios into State Diagrams.

NOTE: here patient is created by screen

Work from Koskimies, Systä, Tuami, and Männistö
University of Tampere, Finland

Transformation - SCED

Derived State Diagram using SCED
Compare derived state diagram with original one reveals potential design mismatches.

**Comparison**

**Derived State Diagram**

- `do: create patient`
- `[patient found]`
- `do: create visiting record`
- `[patient not found]`
- `do: get patient`
- `check for patient`

**Original State Diagram**

- **Input:**
  - ID invalid
  - ID valid
- **Actions:**
  - User input
  - Patient created
  - Patient not found

**Transformation - Rose/Architect**

**Using Rose/Architect to abstract class diagrams**

- **Boeing 747**
  - Mechanic
- **Flight Authorization**

**Use Rule 4**

1. Flight
2. Boeing 747
3. Mechanic

**Use Rule 67**

- Flight
- Boeing 747
- Mechanic
Comparison

**Derived (abstracted) Class Diagram**

- Mechanic
- Flight
- Pilot

Compare derived state diagram with original one reveals incompleteness.

**Original Class Diagram**

- Flight Controller
- Flight
- Pilot

View Integration Activities

**System Model**

- View Synthesis

**View Analysis**

- Mapping (Cross-Referencing)
  - named
  - triples

- Differentiation (Comparison)
  - check for consistency and completeness

- Transformation (Extraction)
  - CollD to StateD
  - RoseArchitect
  - Merging and Splitting
**Problem Solving in UML**

**Examples of Mismatches**

**Integration Activities**

**Goals and Limitations**

**Issues:**
- What other Techniques?
- State Explosion Problem?
- Mismatch Identification vs. Resolution?
- Scalability?

**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).