Integrating Architectural Views in UML

Alexander Egyed

CS 612 Presentation

March 24, 1999

Outline

• Motivation
• Examples of Mismatches
• Integration Framework
• Mismatch Identification
• Conclusion
• **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.
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 rarely share modeling elements
=> they are for different audiences/stakeholders (different interpretations)
=> they are often used concurrently

• 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.
Views Together but still Apart

UML model consists of a collection of loosely integrated diagrammatic views.

Outline

- Motivation
- Examples of Mismatches
- Integration Framework
- Mismatch Identification
- Conclusions
Mismatch Example 1

Model Elements of Layer 1

Flight Controller
Mechanic
Flight
Pilot

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

Model Elements of Layer 2

Aircraft
Flight Controller
Mechanic
Pilot
Flight Plan

Mismatch Example 2

State Diagram for Class Screen

user input
validate input
Enter pressed
ID invalid
ID valid
check patient DB
create visiting record

Possible Mismatch: creation of Patient not visible in Screen
Mismatch Example 3

Design in UML Class Diagram

Potential Mismatch:
Link between component B and connector y not reflected in design view

Potential Mismatch:
Link between component A and component COTS violates C2 architectural constraints

Outline

- Motivation
- Examples of Mismatches
- Integration Framework
- Mismatch Identification
- Conclusions
Three Dimensions of VI

vertical integration
- layers
- packages

horizontal integration
- static
- dynamic

dynamic integration
- analysis
- architecting
- low-level design

architecting:
- expertise
- heuristics
- formalism

View A

View B

View C

View Integration Framework

System Model
e.g. UML model

View Synthesis
(graphical and textual)

Mapping
(Cross-Referencing)
- through names
- through traces
- through association

Transformation
(Extraction)
- through abstraction
- through translation
- through filter

Differentiation
(Comparison)
identify differences between model, rules, and constraints

View Analysis

Alexander Egyed, 3/22/99, 13

Alexander Egyed, 3/22/99, 14
• Motivation
• Examples of Mismatches
• Integration Framework
• Mismatch Identification
• Conclusions
Transformation - Rose/Architect

Model Elements of Layer 2

Aircraft <-> Flight Controller <-> Aircraft
Boeing 747 <-> Mechanic <-> Pilot <-> Flight Plan

Using Rose/Architect to abstract class diagrams

Flight <-> Aircraft <-> Boeing 747 <-> Aircraft
Use Rule 4
Flight <-> Boeing 747 <-> Aircraft
Use Rule 67

Comparison

Derived (abstracted) Class Diagram

Original Class Diagram

Comparing derived class diagram with original one reveals potential mismatch
Mismatch Example 2

State Diagram for Class Screen

- User input
- ID invalid
- ID valid
- Enter pressed
- Validate input
- Check patient DB

Possible Mismatch: creation of Patient not visible in Screen

Transformation - SCED

Using SCED to transform Scenarios into State Diagrams.

Two Scenarios: Only if patient is not found, a new patient object is created.

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

Derived (abstracted) Screen State Diagram

Original State Diagram of Screen

Comparing derived state diagram with original one reveals potential mismatch

Mismatch Example 3

Potential Mismatch:
- Link between component B and connector y not reflected in design view.
- Link between component A and component COTS violates C2 architectural constraints.

Interface between component A and connector x.
Interface between connector x and connector y.
Comparison Algorithm

for each C2 component and C2 connector find corresponding UML classes
for each C2 link
  find and mark C2 link as well as corresponding class links (interface)
for each unmarked C2 link raise incompleteness mismatch
for each unmarked Class link raise inconsistency mismatch
for each C2 component find at least one class call dependency between classes corresponding to that C2 component and other C2 components connected via the same connector

Differentiation

Traverses the views to identify (potential) mismatches between them

=> (Graph) Comparison Algorithms
=> Constraint/Rule Checking

* defined by

View
(or instance of view)

Constructs which describe a view (e.g., formal notation)

Problem within a single instance of a view

Problem within a set of instances of a view

Problem between a set of instances of different views
Outline

- Motivation
- Examples of Mismatches
- Integration Framework and Techniques
- Mismatch Identification
- Conclusions

Conclusions

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).