Integrating Architectural Views in UML

Alexander Egyed

CS 612 Presentation

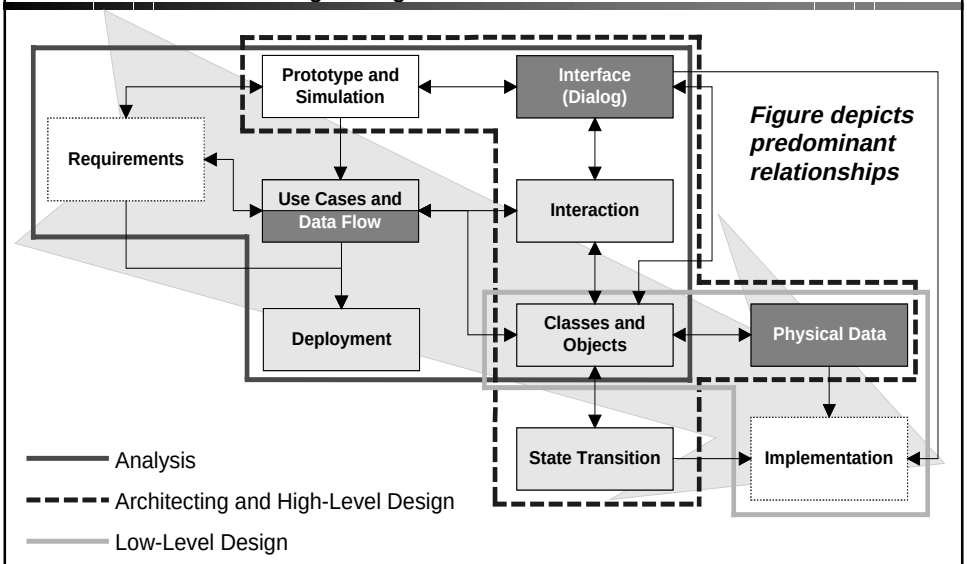
March 24, 1999

Outline

- **Motivation**
- **Examples of Mismatches**
- **Integration Framework**
- **Mismatch Identification**
- **Conclusion**

Architecting and UML

USC - Center for Software Engineering



Integrating… what and why?

USC - Center for Software Engineering

- **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.

Architecting, Views, and UML

USC - Center for Software Engineering

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

The View Integration Problem

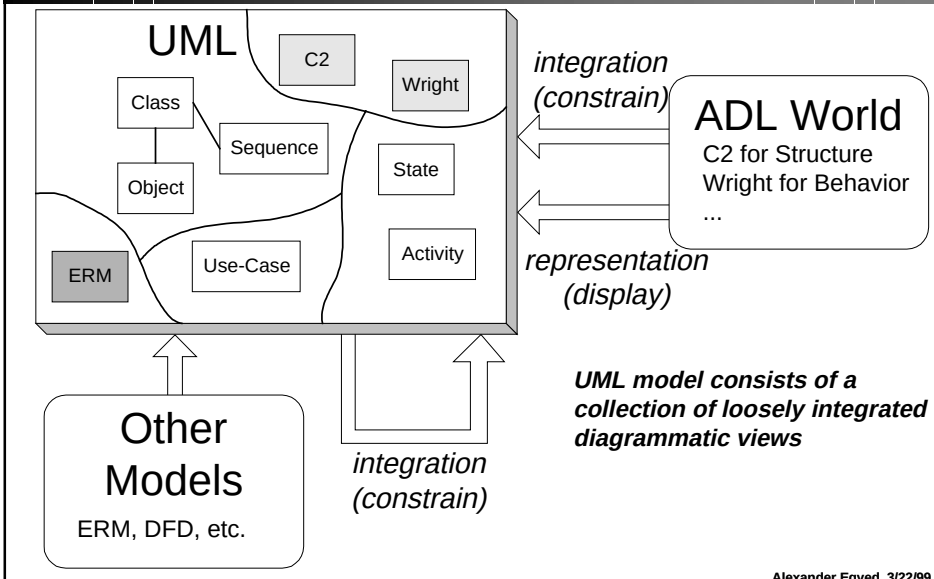
USC - Center for Software Engineering

- **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

USC - Center for Software Engineering



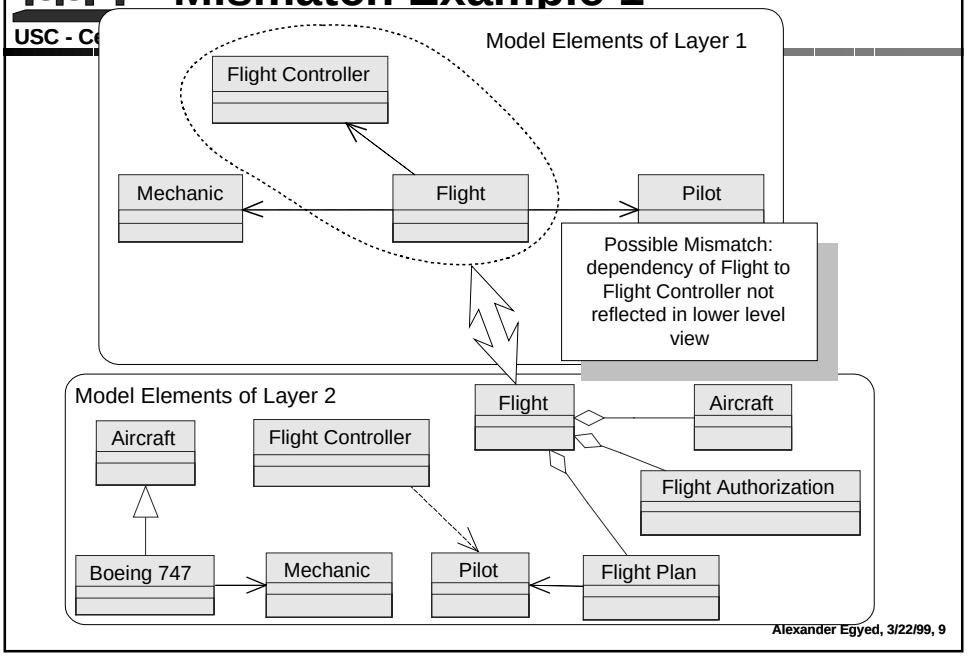
UML model consists of a collection of loosely integrated diagrammatic views

Outline

USC - Center for Software Engineering

- **Motivation**
- **Examples of Mismatches**
- **Integration Framework**
- **Mismatch Identification**
- **Conclusions**

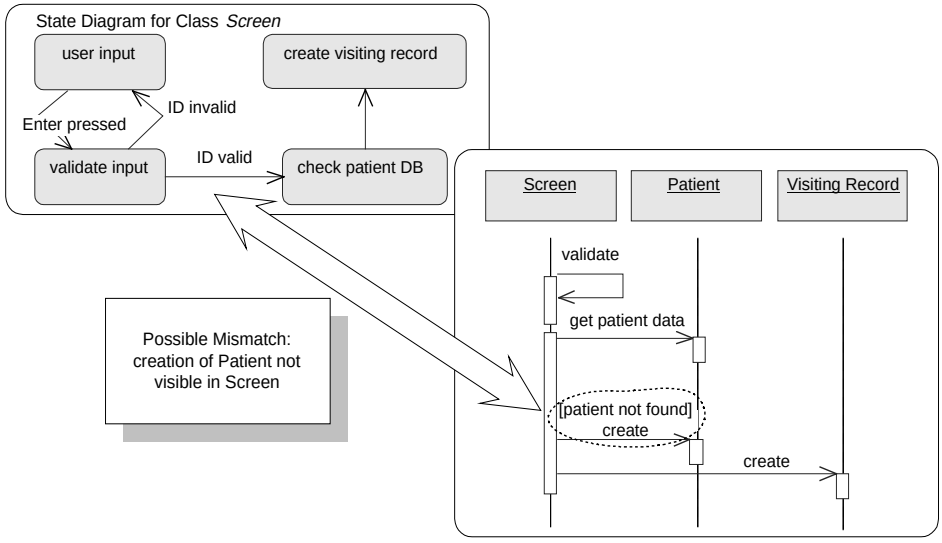
Mismatch Example 1



Alexander Egyed, 3/22/99, 9

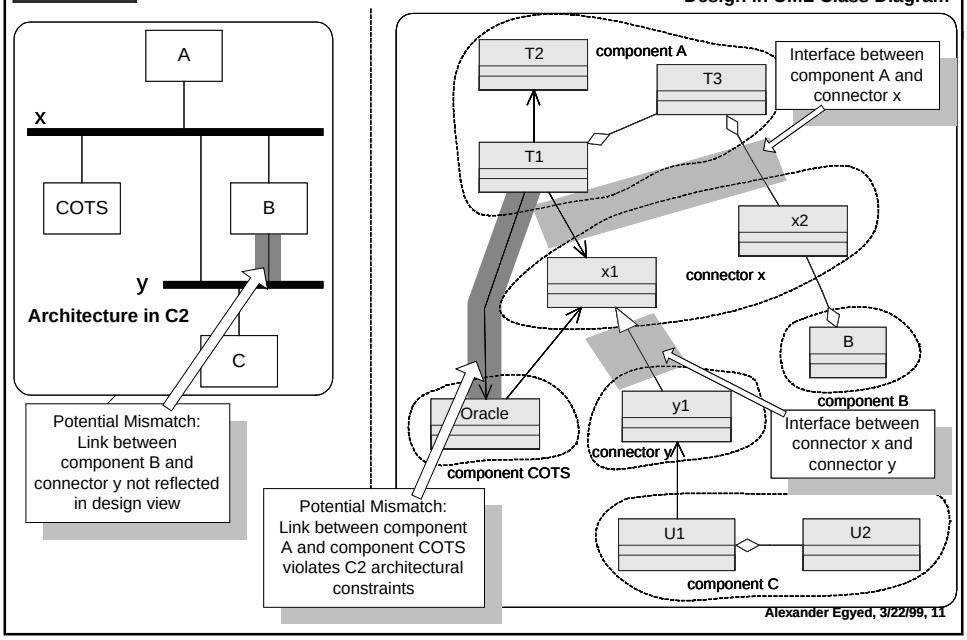
Mismatch Example 2

USC - Center for Software Engineering



Alexander Egyed, 3/22/99, 10

Mismatch Example 3



Outline

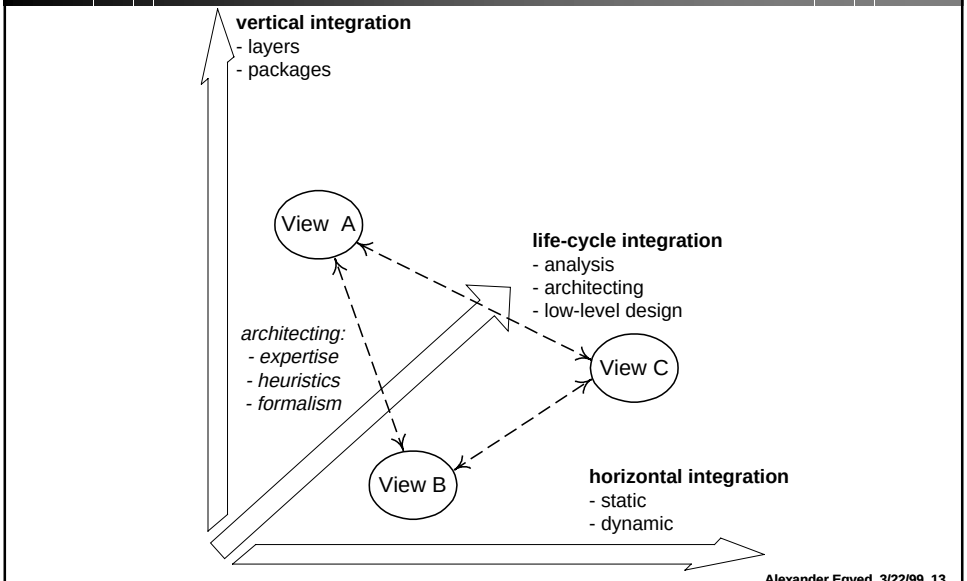
USC - Center for Software Engineering

- **Motivation**
- **Examples of Mismatches**
- **Integration Framework**
- **Mismatch Identification**
- **Conclusions**

Alexander Egyed, 3/22/99, 12

Three Dimensions of VI

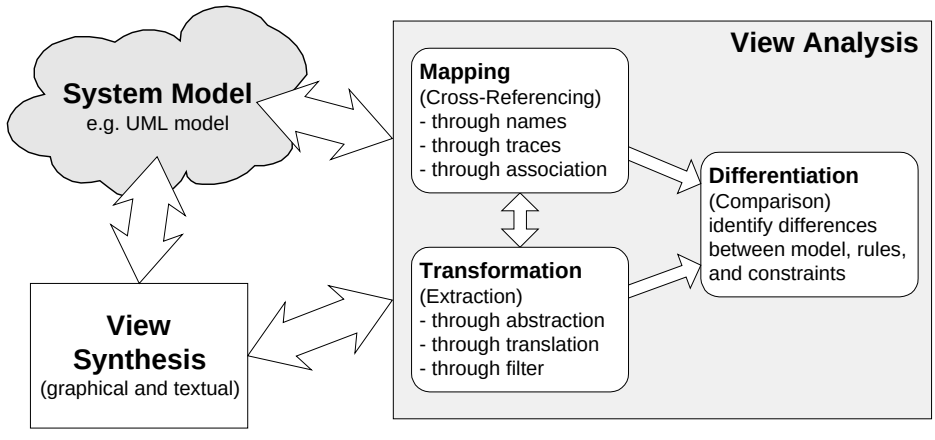
USC - Center for Software Engineering



Alexander Egyed, 3/22/99, 13

View Integration Framework

USC - Center for Software Engineering



Alexander Egyed, 3/22/99, 14

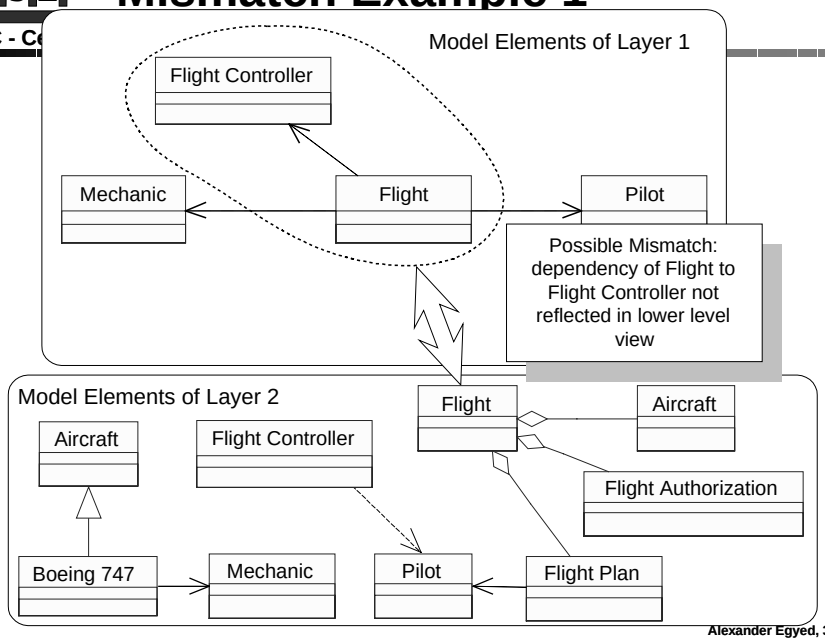
Outline

USC - Center for Software Engineering

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

Alexander Egyed, 3/22/99, 15

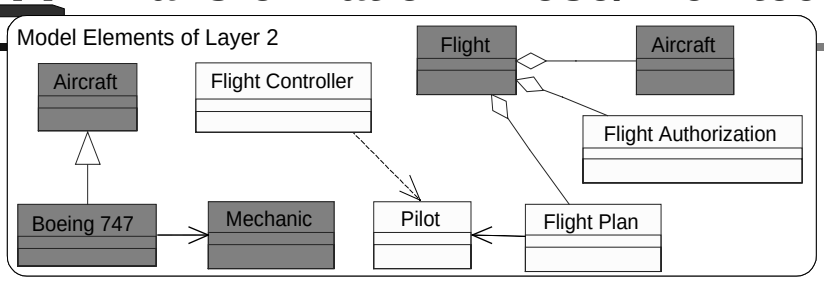
Mismatch Example 1



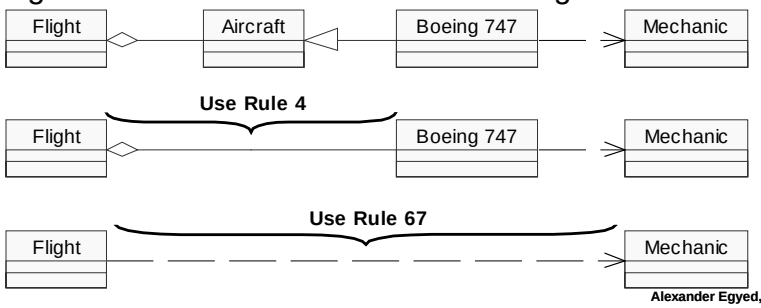
Alexander Egyed, 3/22/99, 16

Transformation - Rose/Architect

USC - Center for Software Engineering



Using Rose/Architect to abstract class diagrams

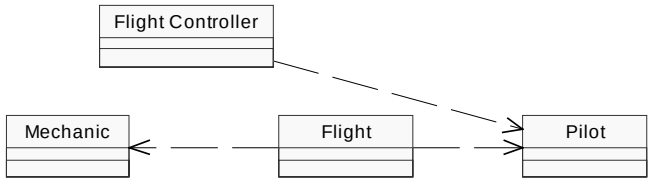


Alexander Egyed, 3/22/99, 17

Comparison

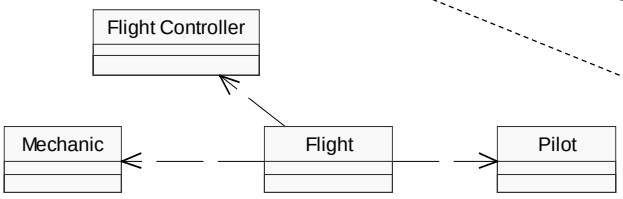
USC - Center for Software Engineering

Derived (abstracted) Class Diagram



Comparing derived class diagram with original one reveals potential mismatch

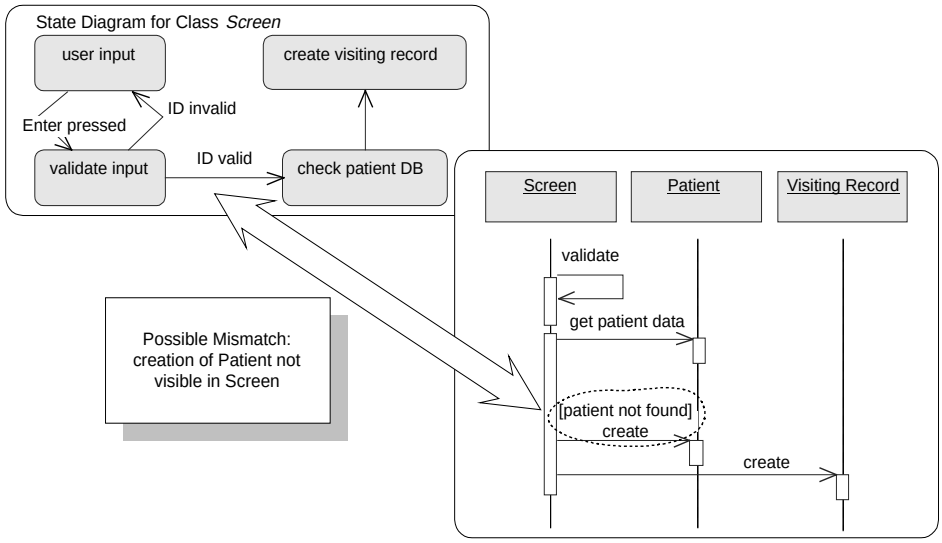
Original Class Diagram



Alexander Egyed, 3/22/99, 18

Mismatch Example 2

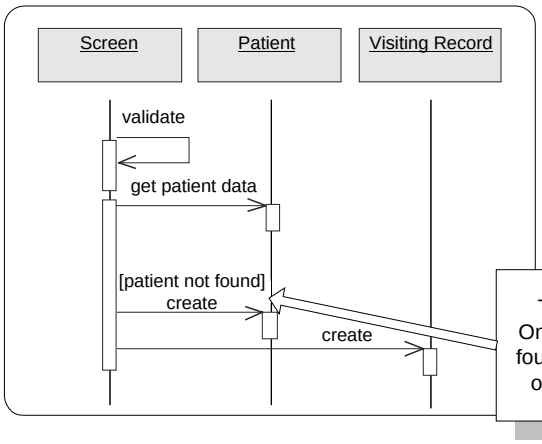
USC - Center for Software Engineering



Alexander Egyed, 3/22/99, 19

Transformation - SCED

USC - Center for Software Engineering



Using SCED to transform *Scenarios* into *State Diagrams.*

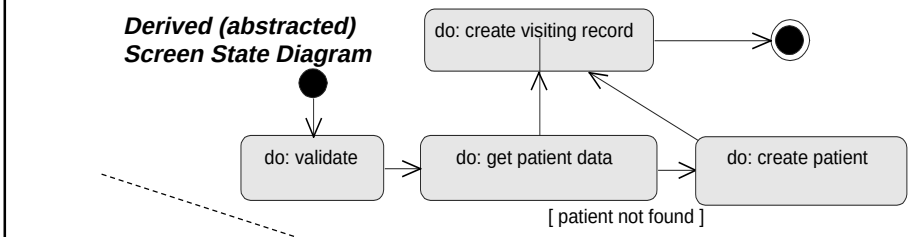
Two Scenarios:
Only if patient is not found, a new patient object is created.

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

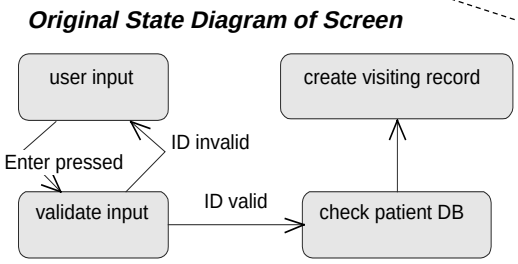
Alexander Egyed, 3/22/99, 20

Transformation - SCED

USC - Center for Software Engineering



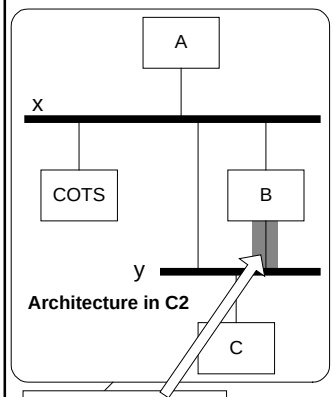
Comparing derived state diagram with original one reveals potential mismatch

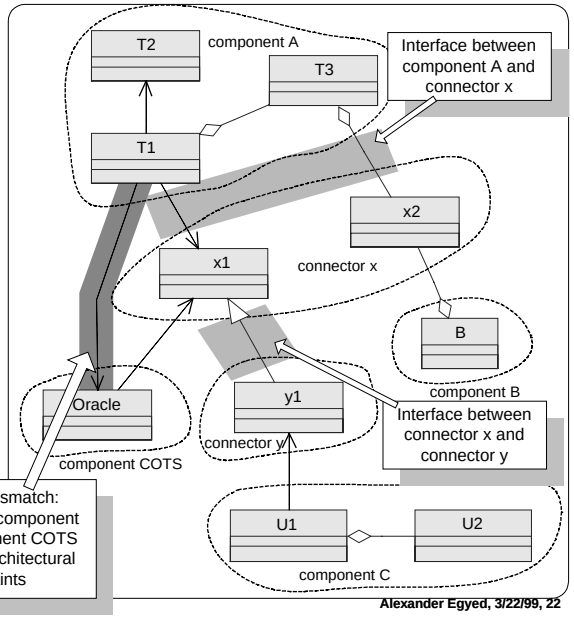


Alexander Egyed, 3/22/99, 21

Mismatch Example 3

Design in UML Class Diagram





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

Alexander Egyed, 3/22/99, 22

Comparison Algorithm

USC - Center for Software Engineering

```
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
```

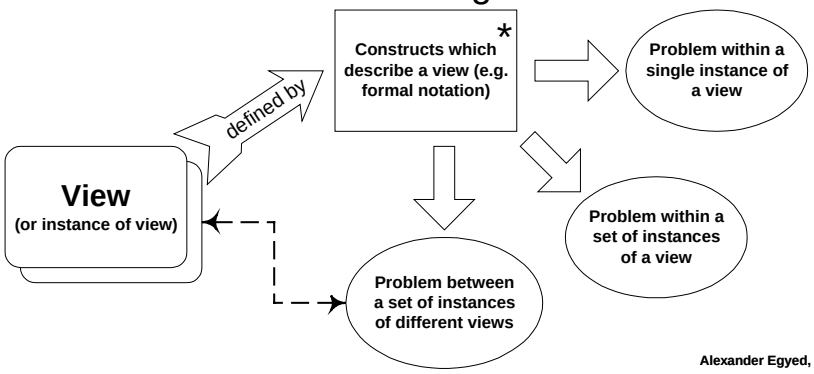

Differentiation

USC - Center for Software Engineering

Traverses the views to identify (potential) mismatches between them

=> (Graph) Comparison Algorithms

=> Constraint/Rule Checking



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).