



USC - Center for Software Engineering

Using Patterns to Integrate Architectural Views in UML

Alexander Egyed

**Center for Software Engineering
University of Southern California**

GSAW 99

March, 1999



Outline

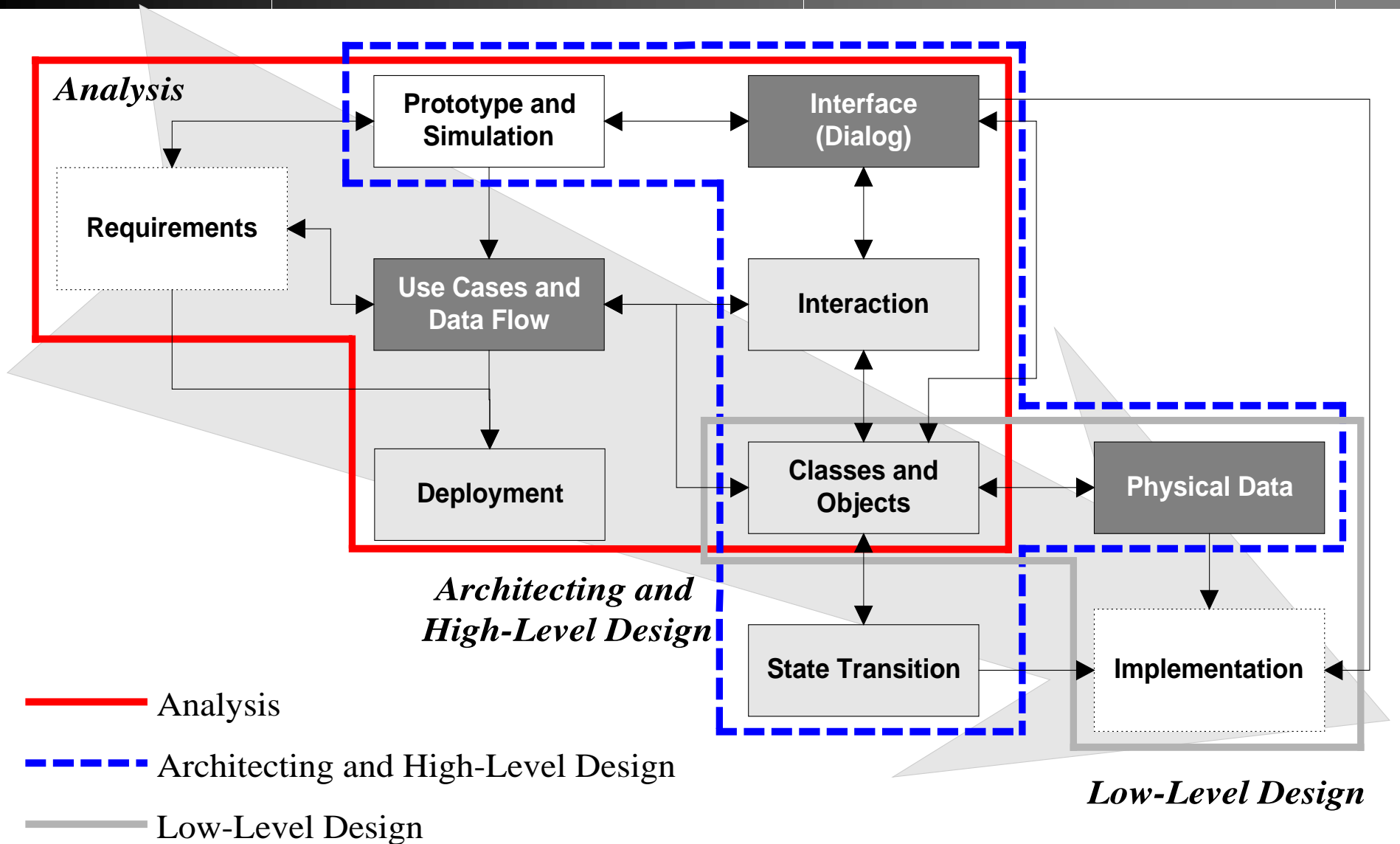
USC - Center for Software Engineering

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



Architecting and UML

USC - Center for Software Engineering





USC - Center for Software Engineering

Architecting and UML

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

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.



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

USC - Center for Software Engineering

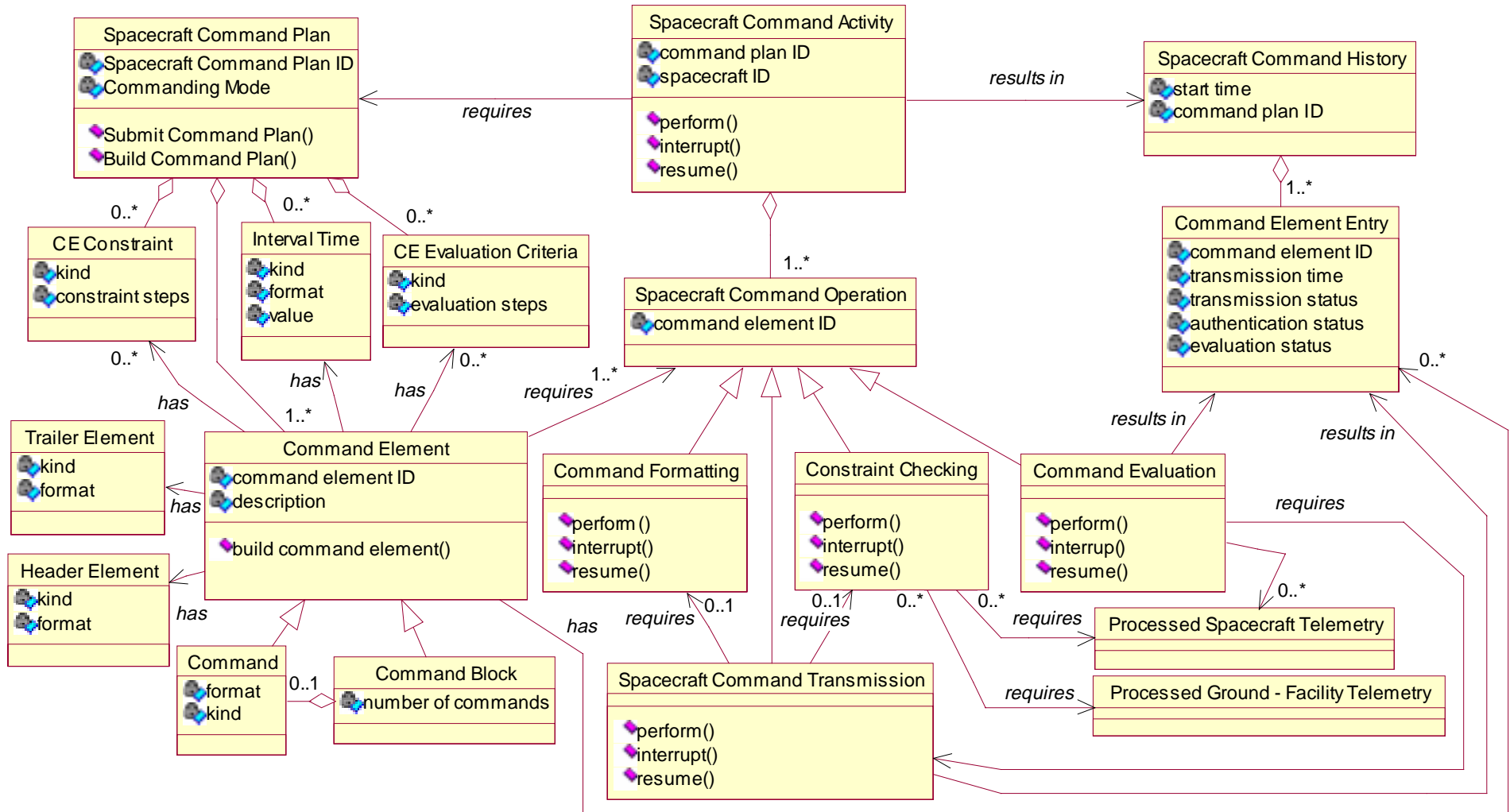
- **Need for Architectural Evaluation**
- ➔ • **Mismatches in a TT&C Example**
- **Using Rose/Architect to identify Mismatches**
- **Goals and Limitations**



Level 2 Architectural View

USC - Center for Software Engineering

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

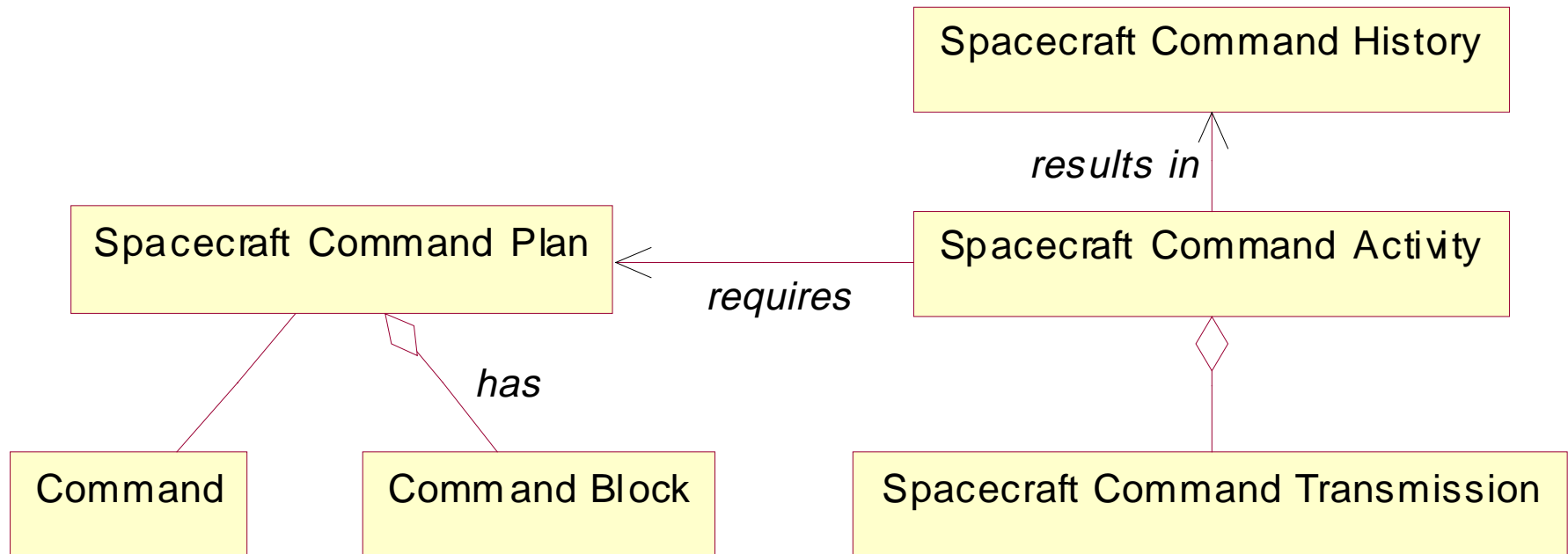




Level 1 Architectural View

USC - Center for Software Engineering

My Interpretation of previous TT&C Architecture Excerpt





USC - Center for Software Engineering

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

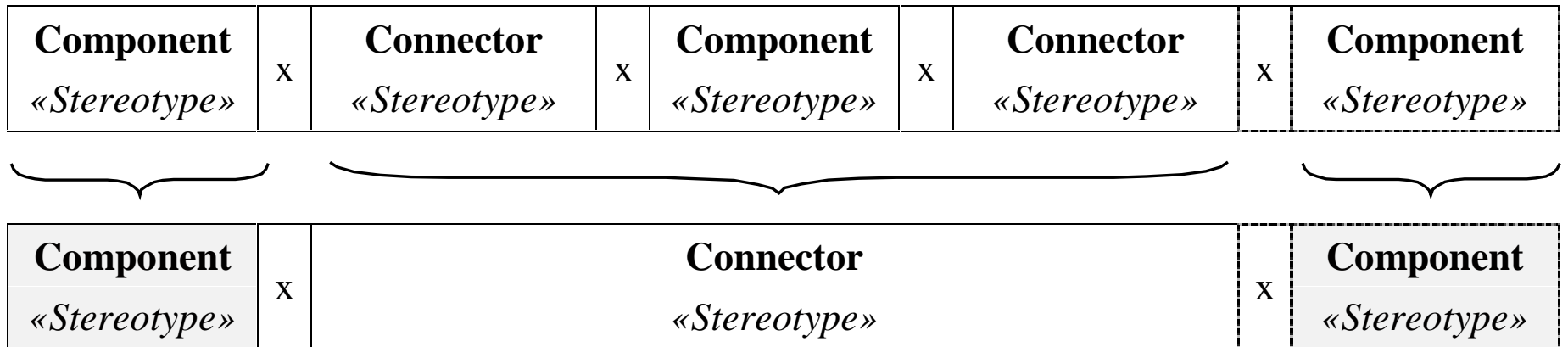
USC - Center for Software Engineering

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



Simple Rule

USC - Center for Software Engineering



Resulting patterns must be legal according to UML rules



Some Rule Examples

USC - Center for Software Engineering

Rule	Component	Connector →	Component	Connector →	Component
1	Class	Generalization →	Class	Generalization →	Class
		Generalization →			
2	Class	Generalization →	Class	Dependency →	Class
		Dependency →			
3	Class	Generalization →	Class	Association →	Class
		Association →			
4	Class	Generalization →	Class	Aggregation →	Class
		Aggregation →			
[...]					
9	Class	Dependency →	Class	Aggregation →	Class
		Dependency →			
[...]					
66	Class	← Aggregation	Class	Generalization →	Class
		← Aggregation			
67	Class	← Aggregation	Class	Dependency →	Class
		Dependency →			
68	Class	← Aggregation	Class	Association →	Class
		Association →			
69	Class	← Aggregation	Class	Aggregation →	Class
		Aggregation →			

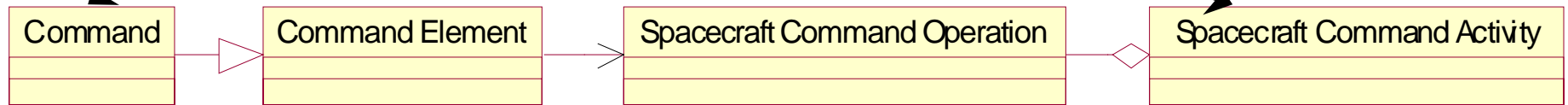
There are currently about 80 rules.



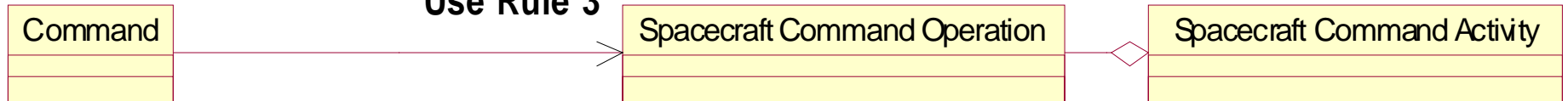
Transformation - Rose/Architect

Layer 1 Diagram Classes

One possible path using the helper classes in layer 2



Use Rule 3



Use Rule 9





Rose/Architect Tool

USC - (RoseArchitect - [Membership])

File Window Help

Belongs to: All Classes, ATCS UI, Logical Stage, domain specific

Does not belong to: ATCS Components, ATC, Physical Stage, Con

Rational Rose - c&c.mdl - [Class Diagram: Logical View / detail]

File Edit View Browse Report Query Tools Add-Ins Window Help

RA

- Spacecraft Command
- Command
- Command Block
- Spacecraft Command
- Spacecraft
- Spacecraft Command
- Spacecraft Command
- CE Constraint
- Trailer Element
- Header Element
- Command Element
- Interval Time
- CE Evaluation Criteria
- Spacecraft Command
- Command Formatting
- Constraint Checking
- Processed Spacecraft
- Processed Ground - F
- Command Evaluation
- Command Element Ent
- Component View

Spacecraft

- Spacecraft Command Plan ID
- Commanding Mode
- Submit Command Plan()
- Build Command Plan()

CE

- kind
- constraint steps

Interv

- kind
- format
- value

Trailer

- kind
- format

Command

- command element ID
- description
- build command element()

Heade

- kind
- format

Com

- format

Comman

- number of commands

Space

0..* 0..* 0..* 0..* 0..* 1..* 0..1

has has has has requires

RoseArchitect - [Rulebook]

File Window Help

Name: Rule 28

Comments Manual Auto

From:

Class Class Class

To:

Class Class

Name	From Node	From Connector	Middle Node	To Con
Rule 28	class	aggregation	class	general
Rule 26	class	association d..	class	general
Rule 25	class	dependency	class	general

Each Delete Rulebook... Close

Now editing Rule 28.

interrupt() resume()

0..1

For Help, press F1



Mismatch Comparison

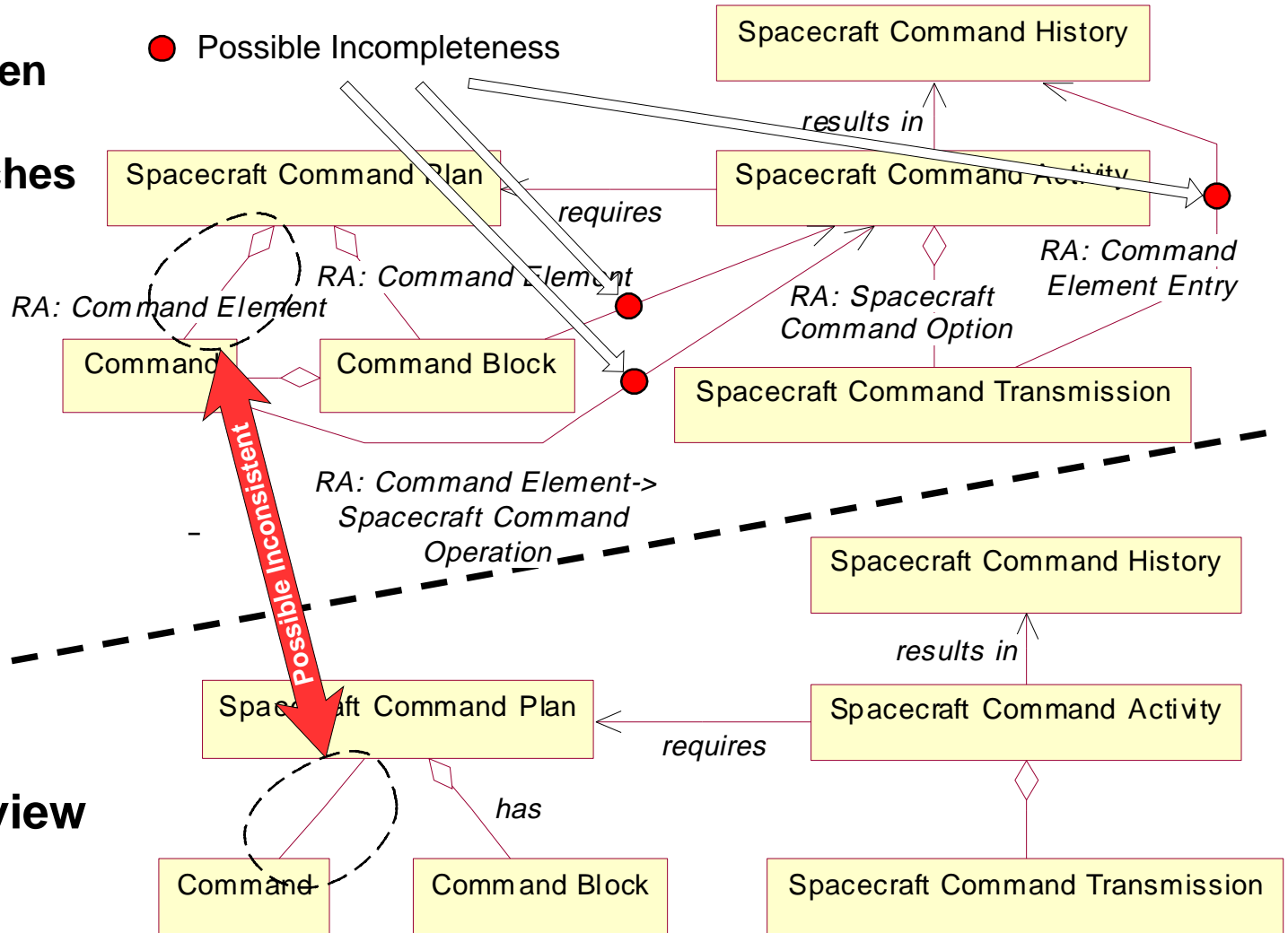
USC - Center for Software Engineering

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





Outline

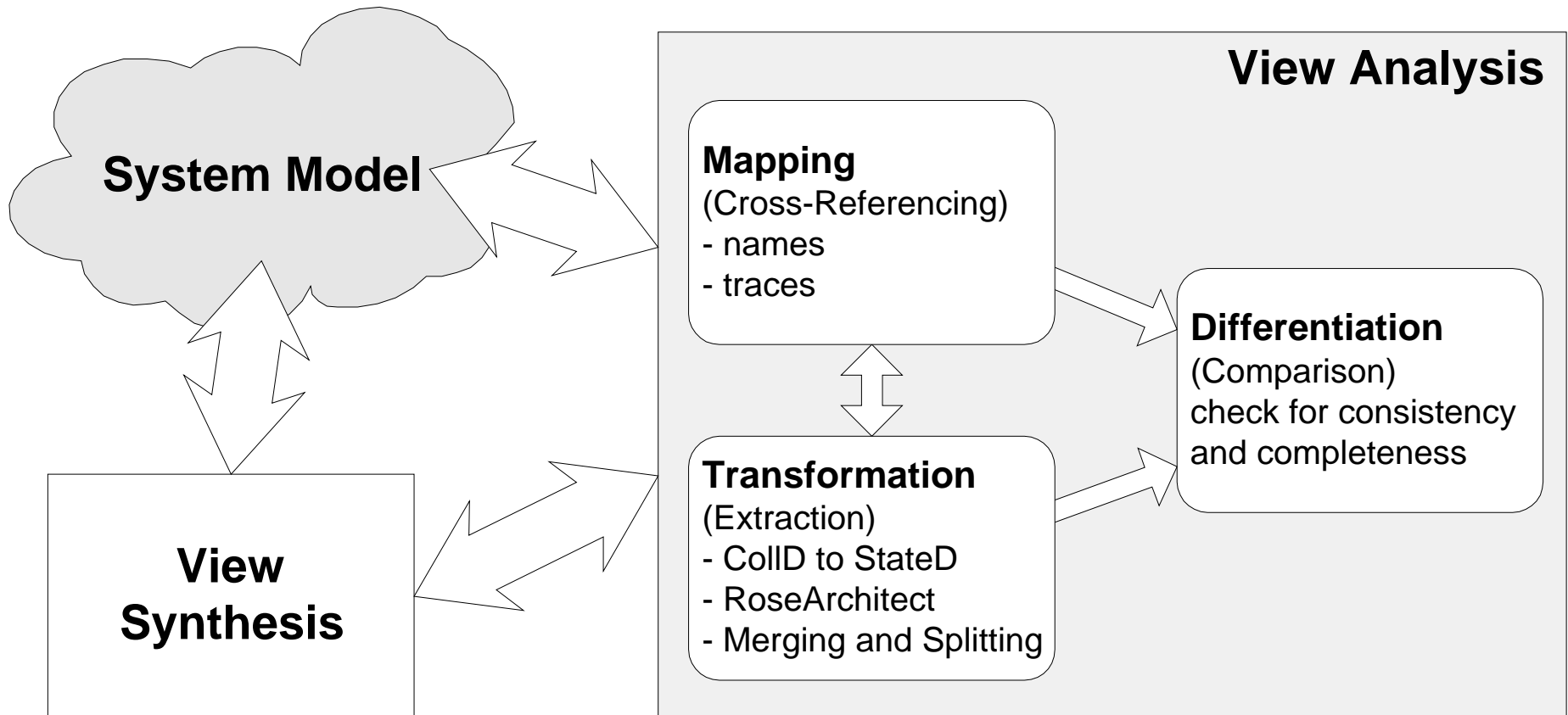
USC - Center for Software Engineering

- **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 ...





Goals and Limitations

USC - Center for Software Engineering

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).**