zeligsoft

The Problems and Promise of UML 2.0 Structures for SCA

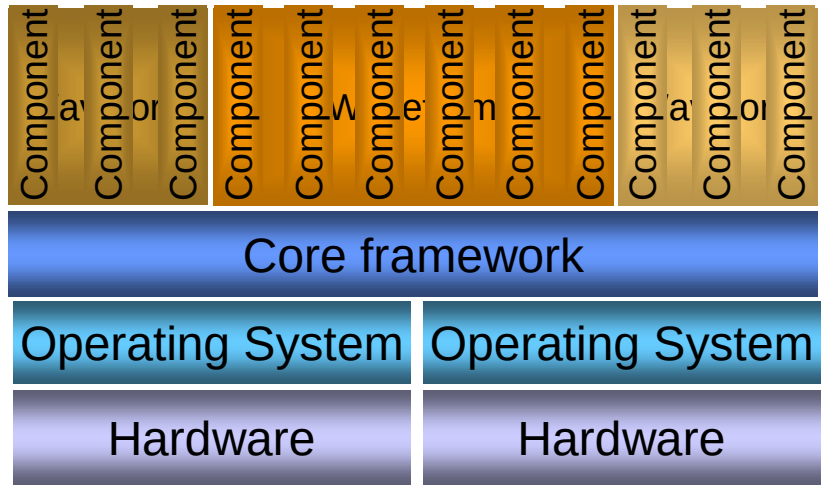
John Hogg
CTO, Zeligsoft

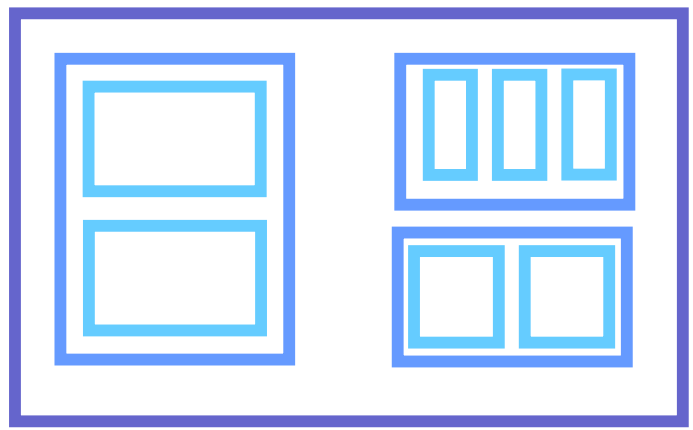
2004-09-15

Version 1.4

Scalability Through Hierarchical Decomposition

- Many large systems have flat deployments
- However, virtually every large problem is made tractable using **hierarchical decomposition**
- Some advantages:
  - Understandability, maintainability
  - Composability, reusability, reconfigurability
  - Avoidance of development conflicts/bottlenecks
- Strangely, SCA profiles aren't hierarchical—or are they?



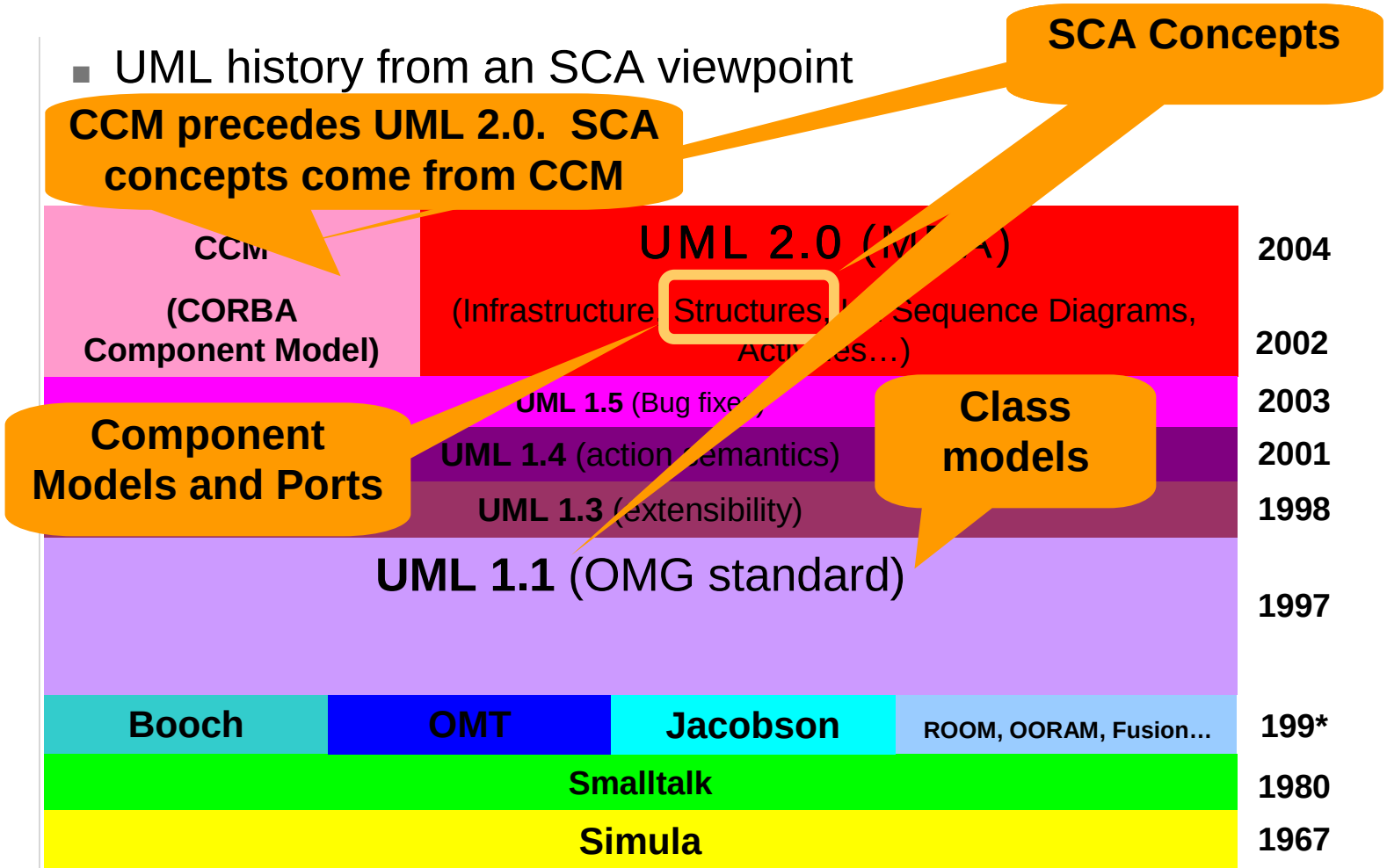


The Plot

- SCA structures are role models—not class models and not object models
- Powerful abstraction and reuse require hierarchical structures
- The SCA can support these today—if the right transformations are applied
- Work at the PIM level, not the PSM level

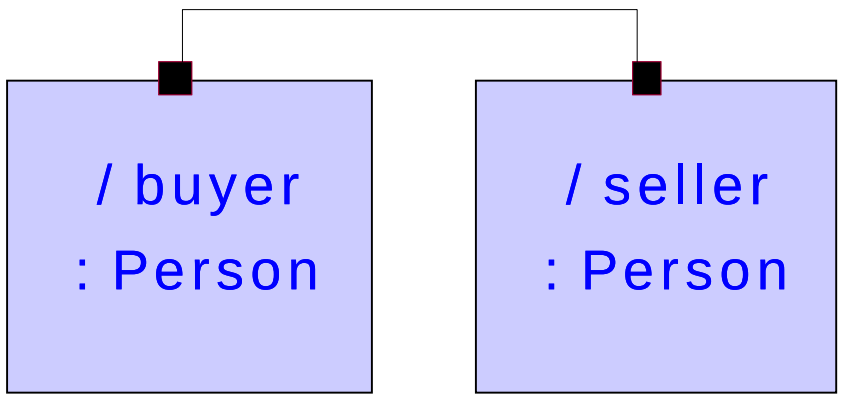
How We Got Here: A Brief Genealogy

- UML history from an SCA viewpoint

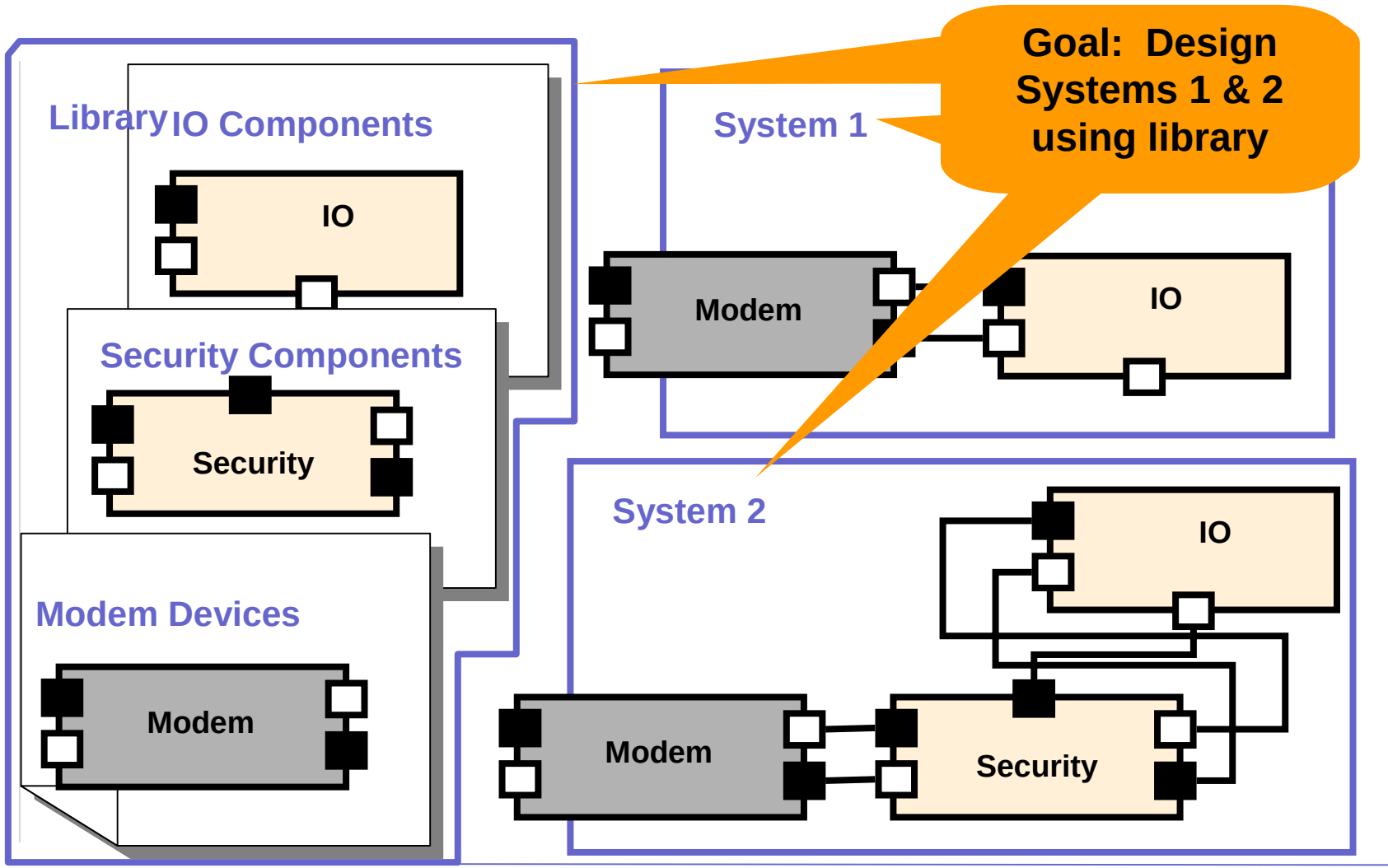


Structure Models: Representing Architecture

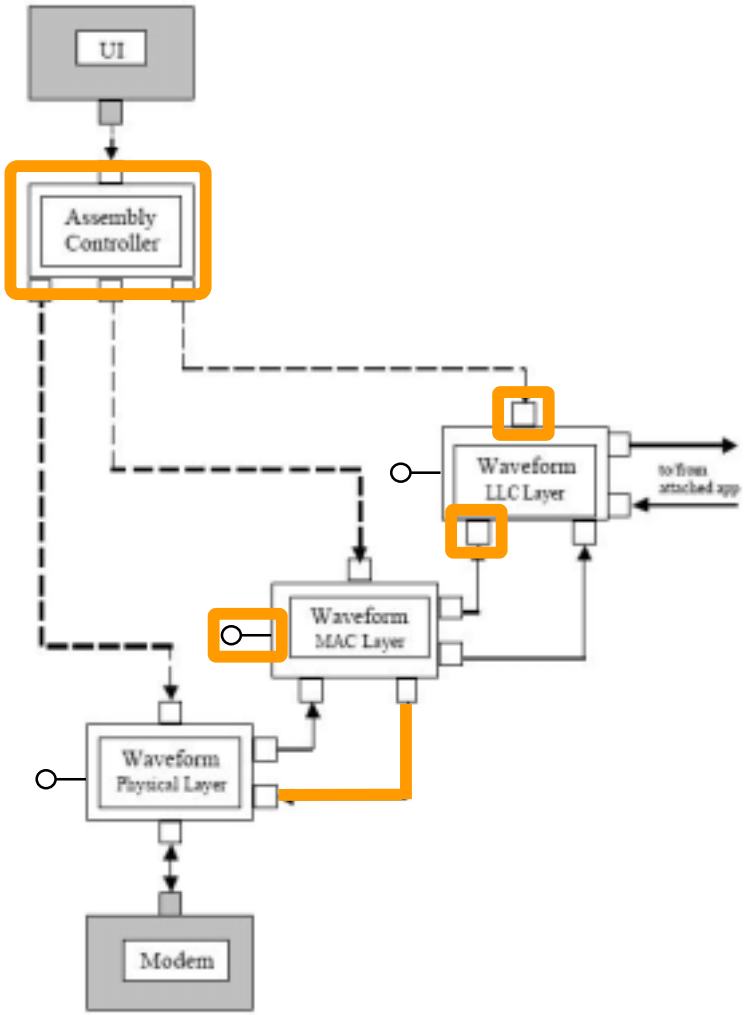
- “Architecture” is who speaks to whom and what they say
- Basis of SCA assembly diagrams
- Concepts from UML 2.0 “classes with structure”
  - Based on “role modelling” of ROOM, OORAM, UML-RT



Component Re-Use in Different Systems



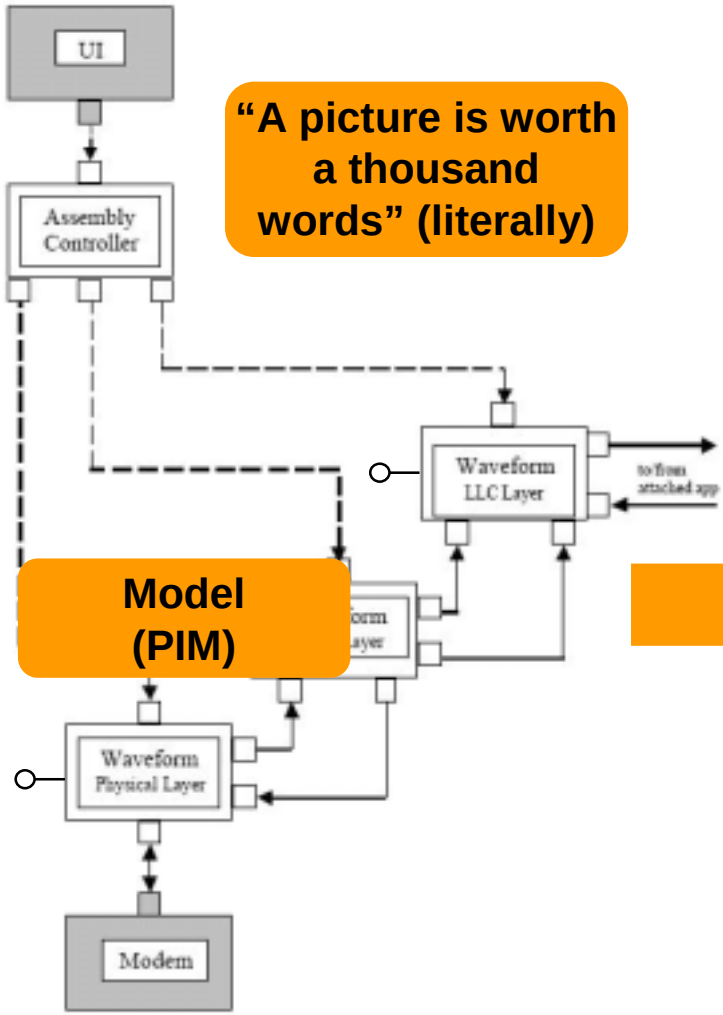
SCA Structures



- The structure is a basic concept in SCA
- Components…
- …have ports and interfaces…
- …that may be connected by connectors

- Non-normative notation
- So why are structure diagrams needed?

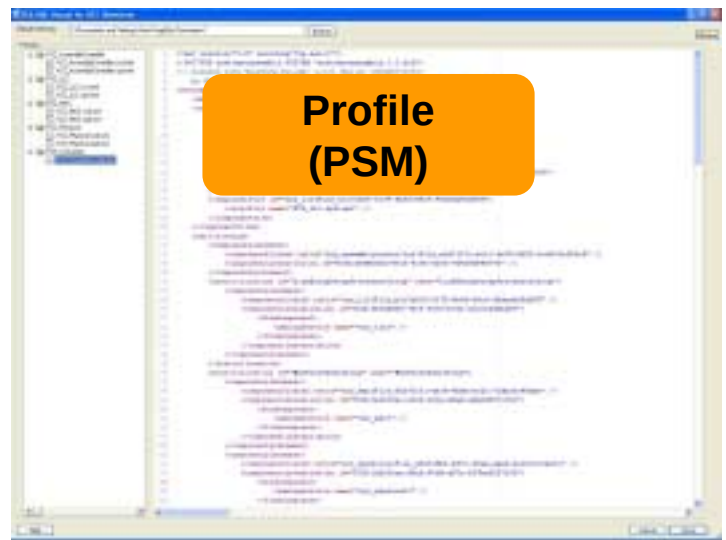
Why SCA Structures?



"A picture is worth a thousand words" (literally)

Model (PIM)

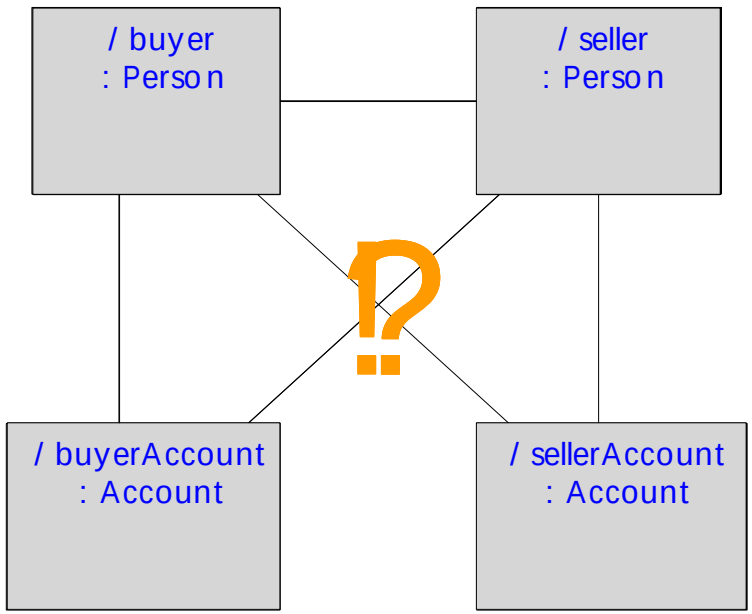
- SCA structure models are human-readable architectures
- Draw structure diagrams…
- …then manually write or automatically generate SCA profiles (XML descriptor file sets)

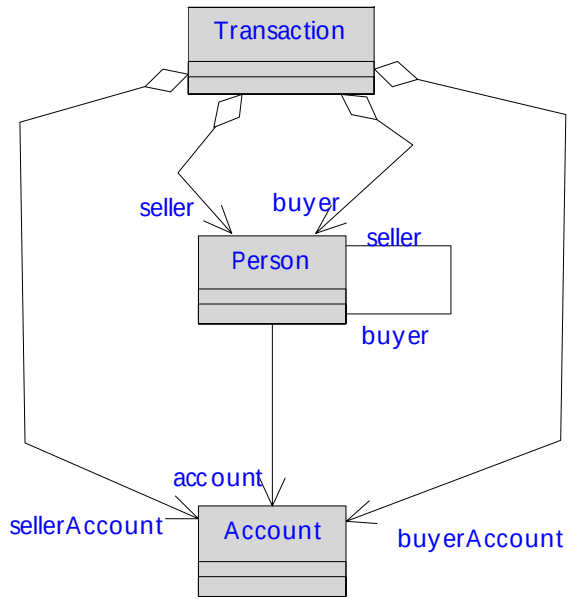


Profile (PSM)

Structure Modelling Is Not Class Modelling

- Transaction example:
  - Two Persons
  - Each Person has an Account
  - Who owns which Account?

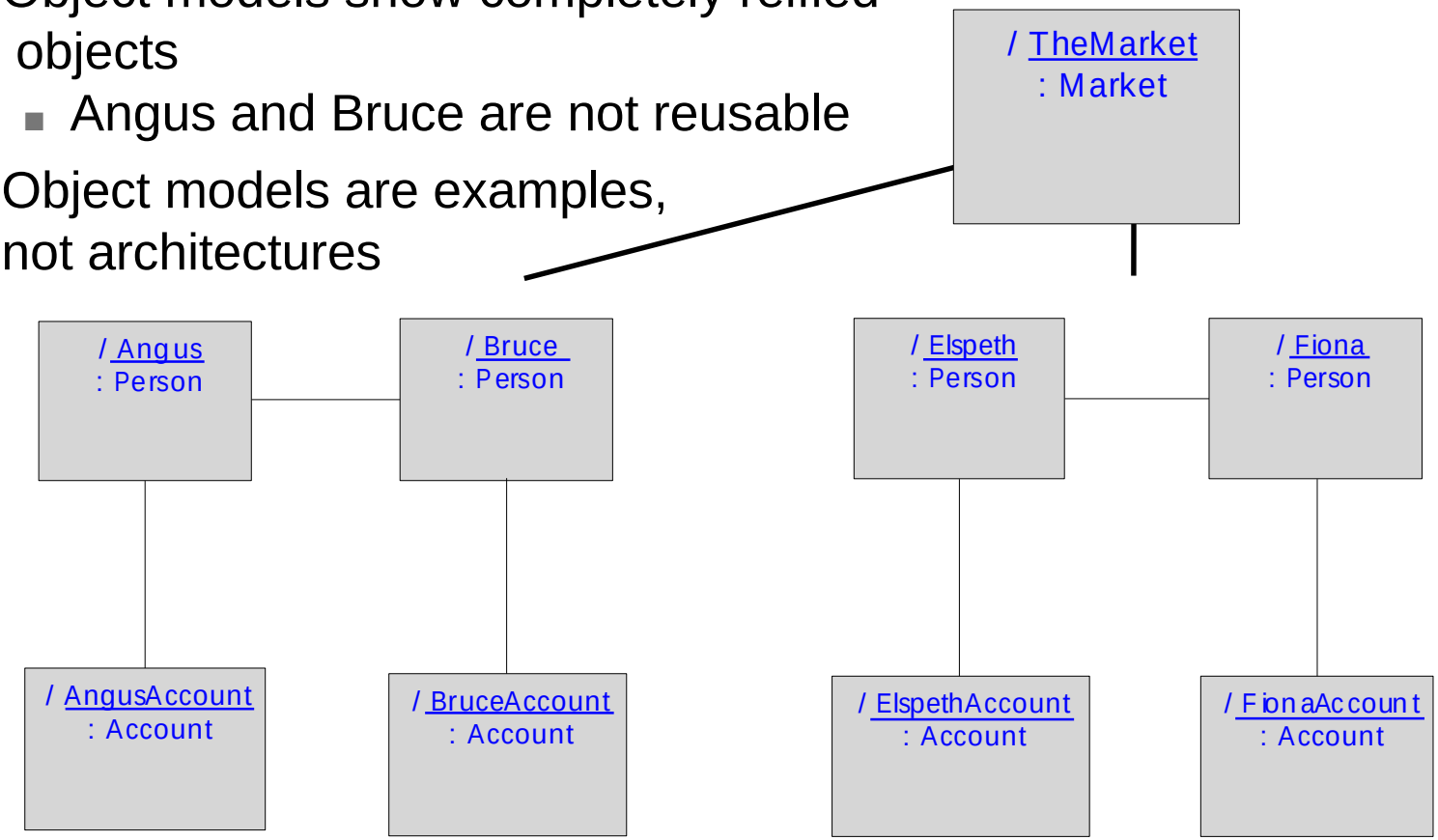




- Class models show “for all” properties
- Structure models show architectural relationships

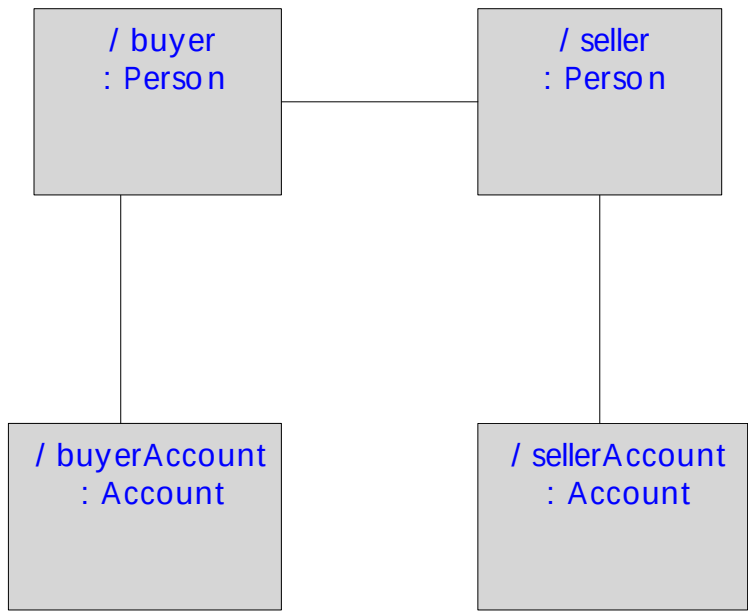
Structure Modelling Is Not Object Modelling

- Object models show completely reified objects
  - Angus and Bruce are not reusable
- Object models are examples, not architectures



Role Modelling

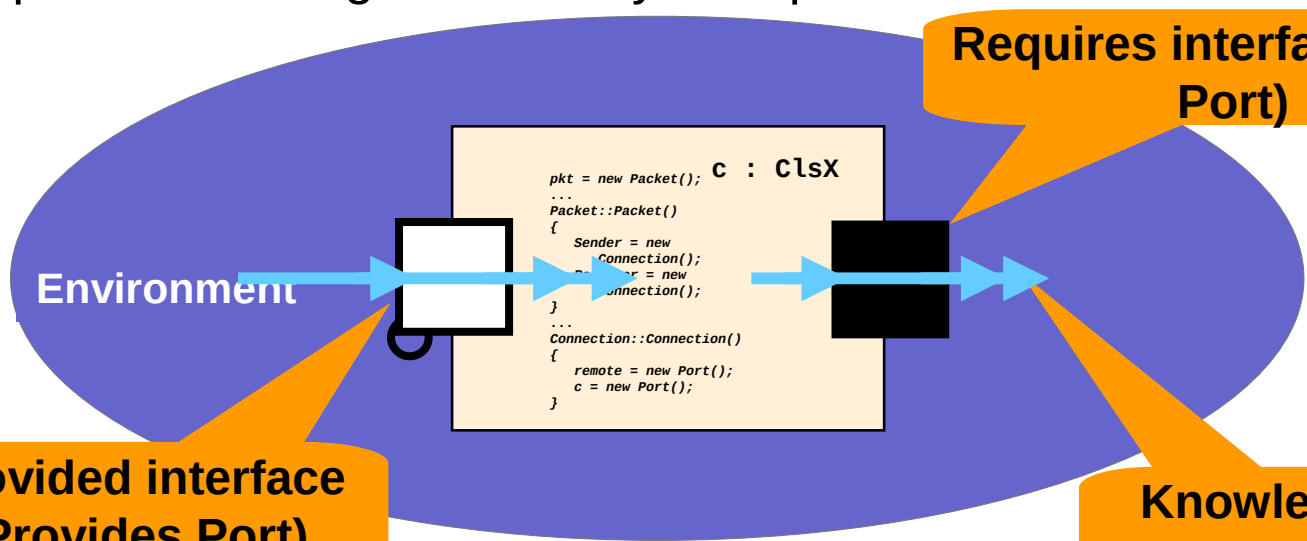
- Roles: the interface contracts a component must fulfill
- Relationships or connections: who speaks to whom
- Reusable, polymorphic concepts



Ports: Two-Way Encapsulation

Note: UML 2.0 ports may be bidirectional and have multiple interfaces. SCA currently supports only single-interface, one-way ports

- Simple interfaces give one-way encapsulation…

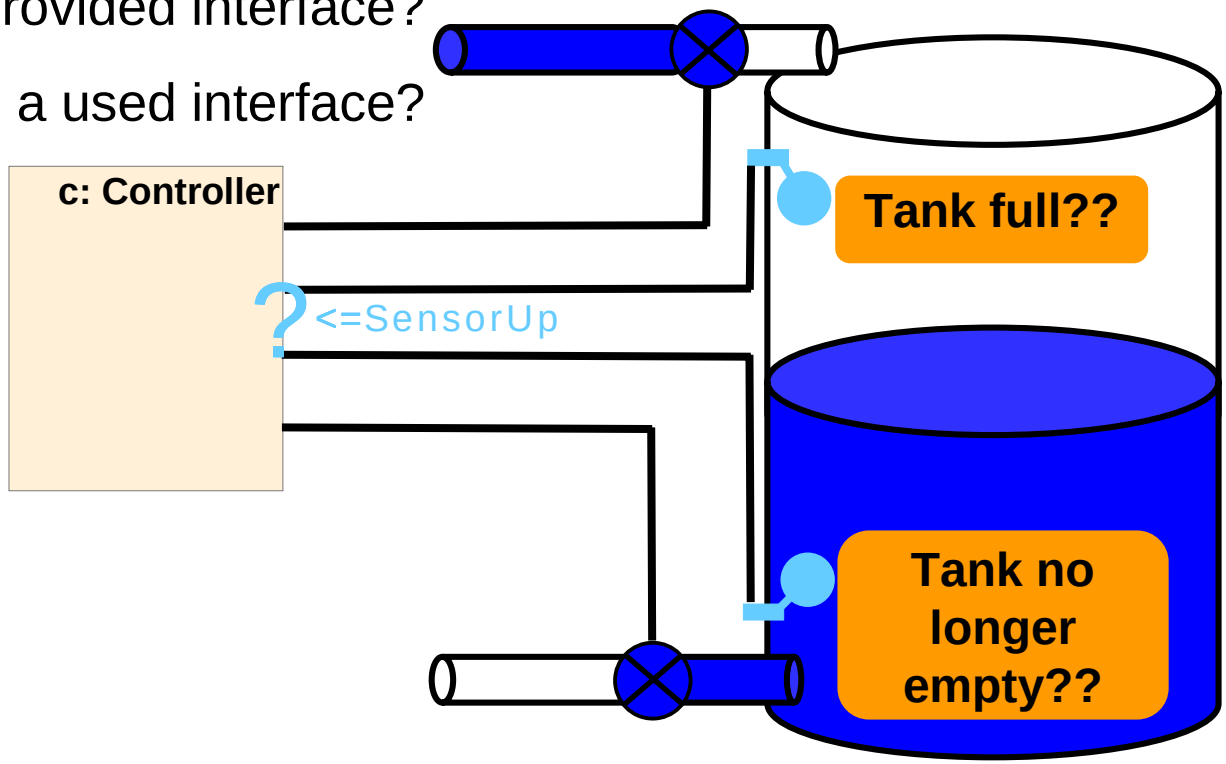


- Ports isolate an object's implementation from its environment and its environment from the object
- Each port has one or more interfaces (Provided and Requires)

Ports: Directed Interfaces

With simple interfaces,

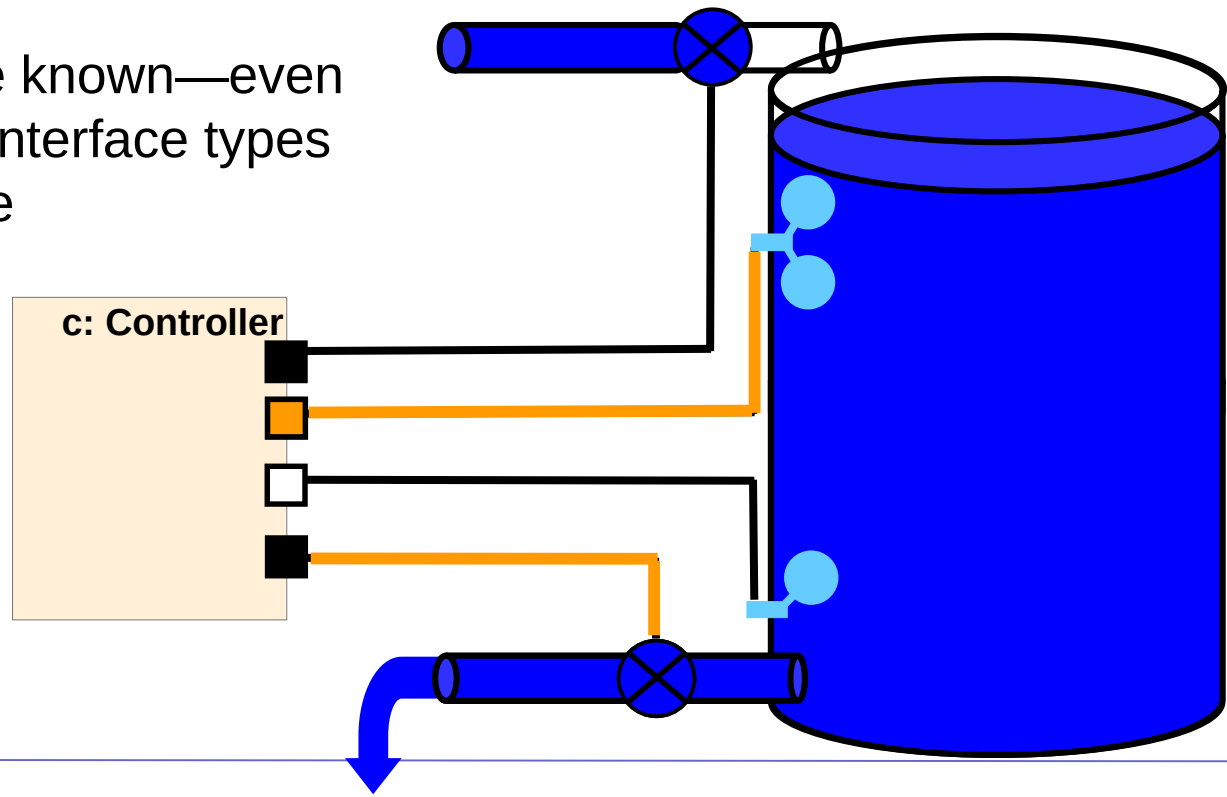
- What used a provided interface?
- What provided a used interface?



Ports: Directed Interfaces

With ports,

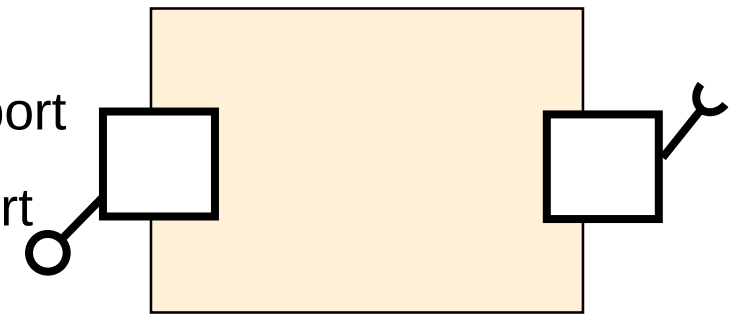
- User role known—even when Provided Interface types are the same
- Provider role known—even when Uses Interface types are the same



Conjugation: Port Direction and Notation

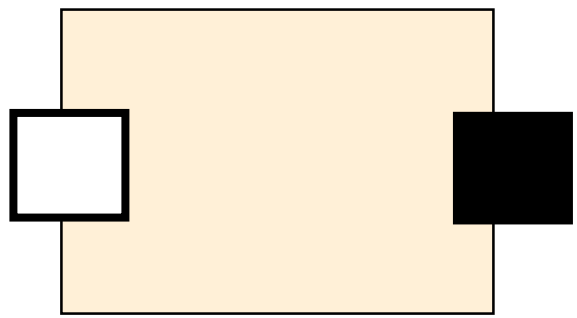
UML 2.0 allows optional port notation:

- “Lollipop” for Provides interface on port
- “Radar” for Requires interface on port



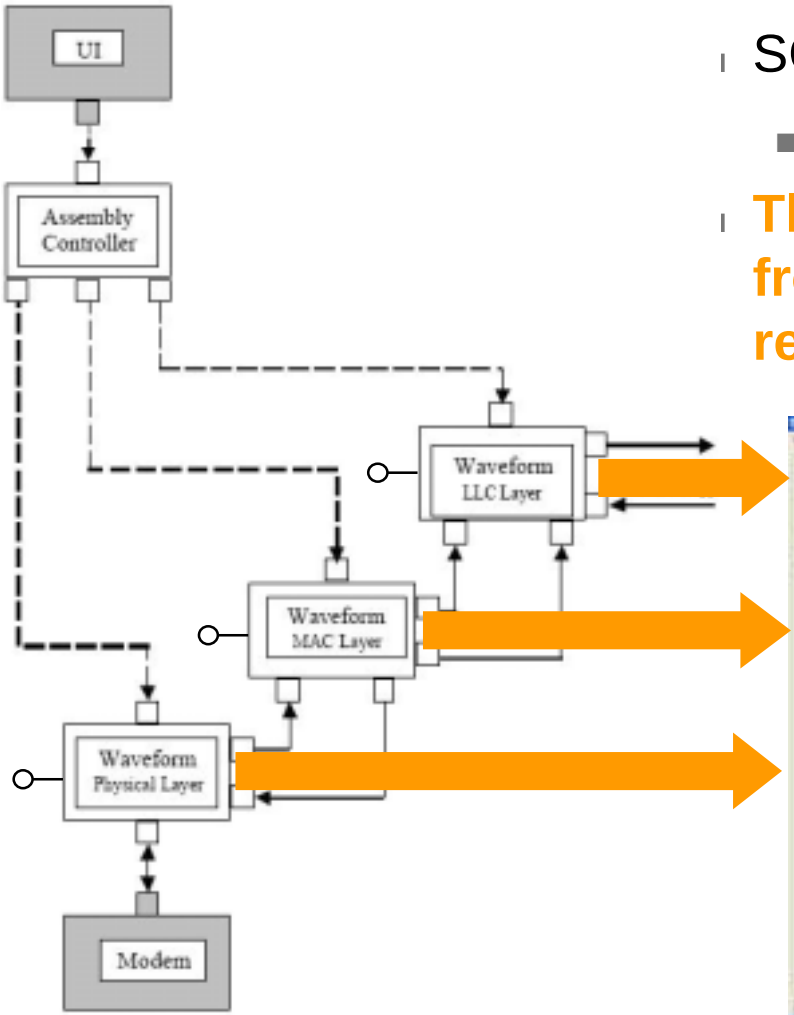
SCA uses color to indicate port interface direction

- White for Provides
- Black for Requires

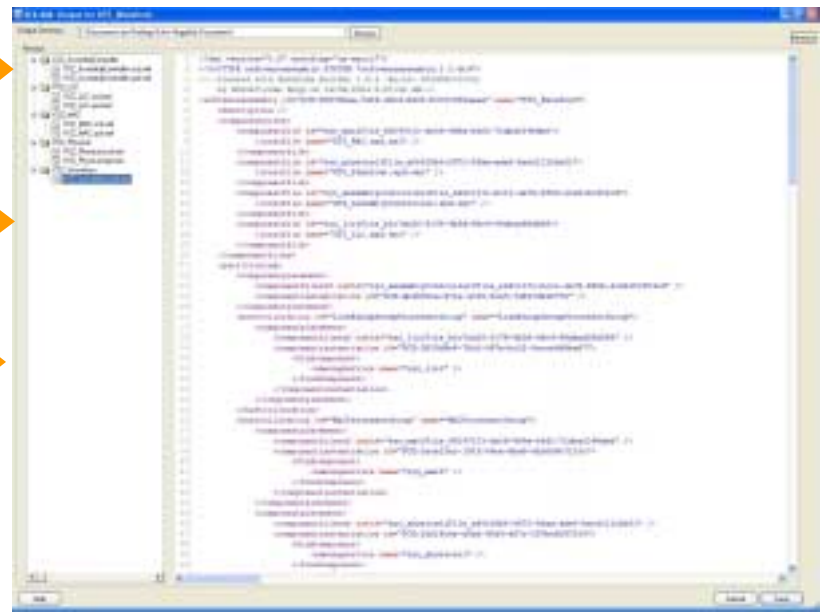


The interface’s direction is called its “conjugation”

Back at the SCA Structures…

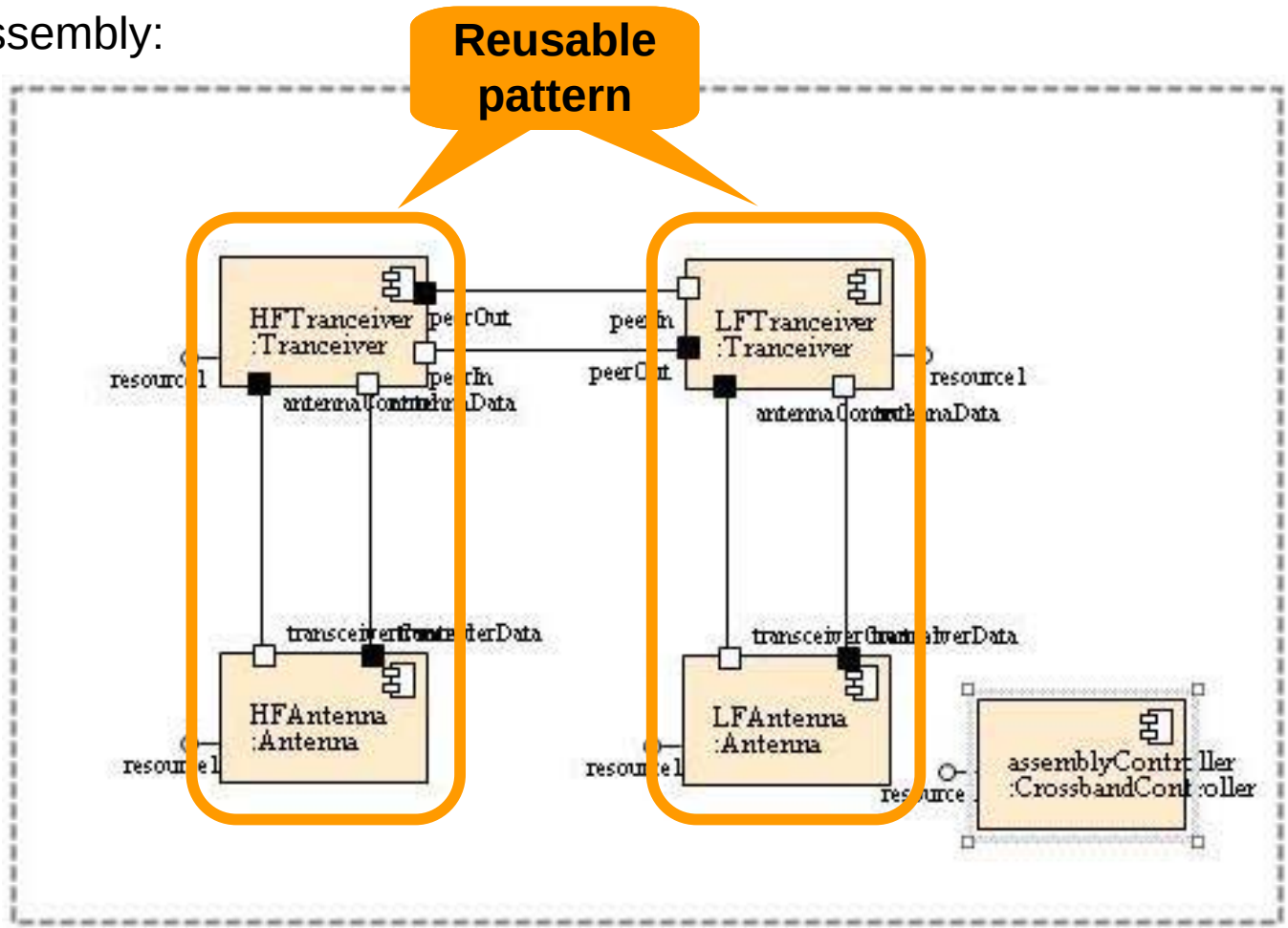


- SCA SAD files are flat
  - Therefore, not reusable
- **This does not prevent structures from being composed of reusable elements**



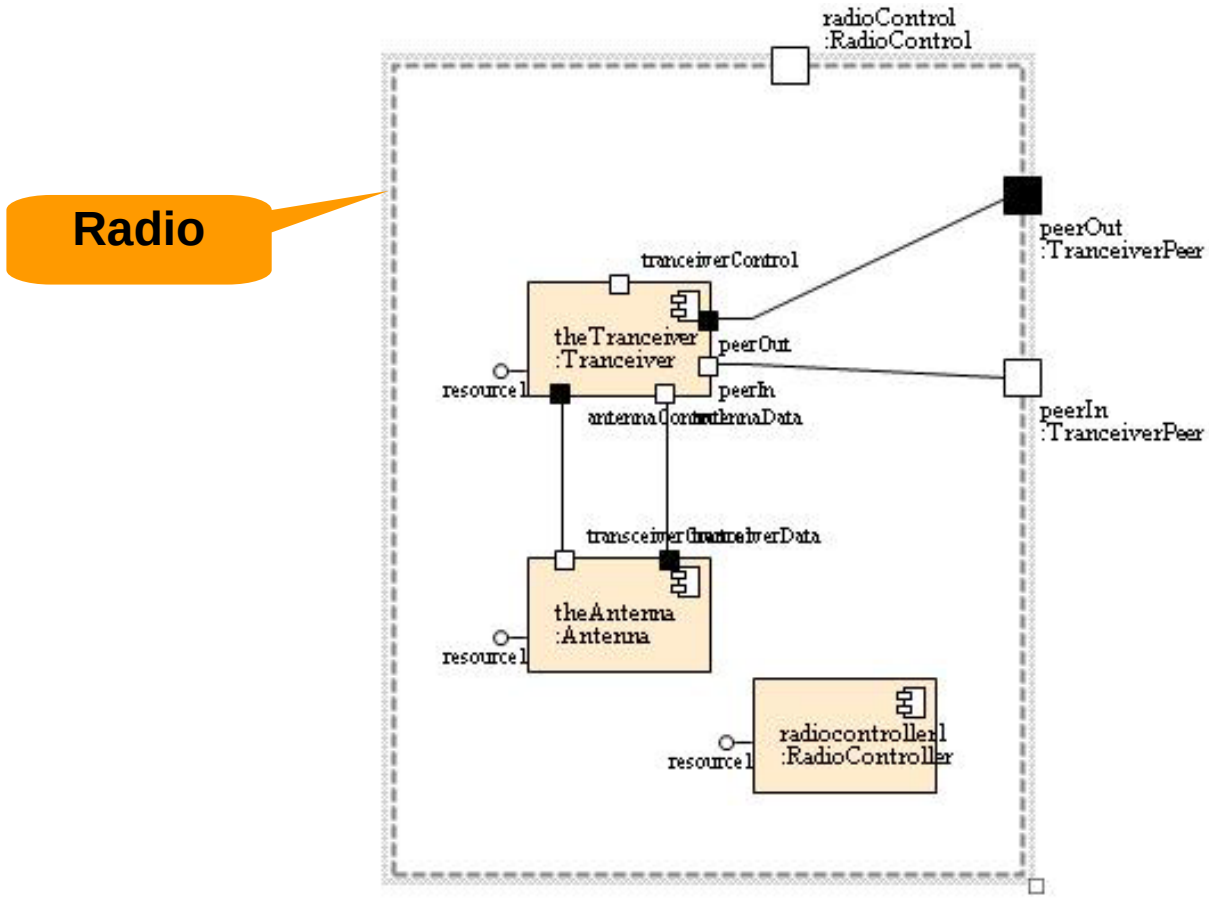
Example: Cross-Banding Radio

- A flat assembly:



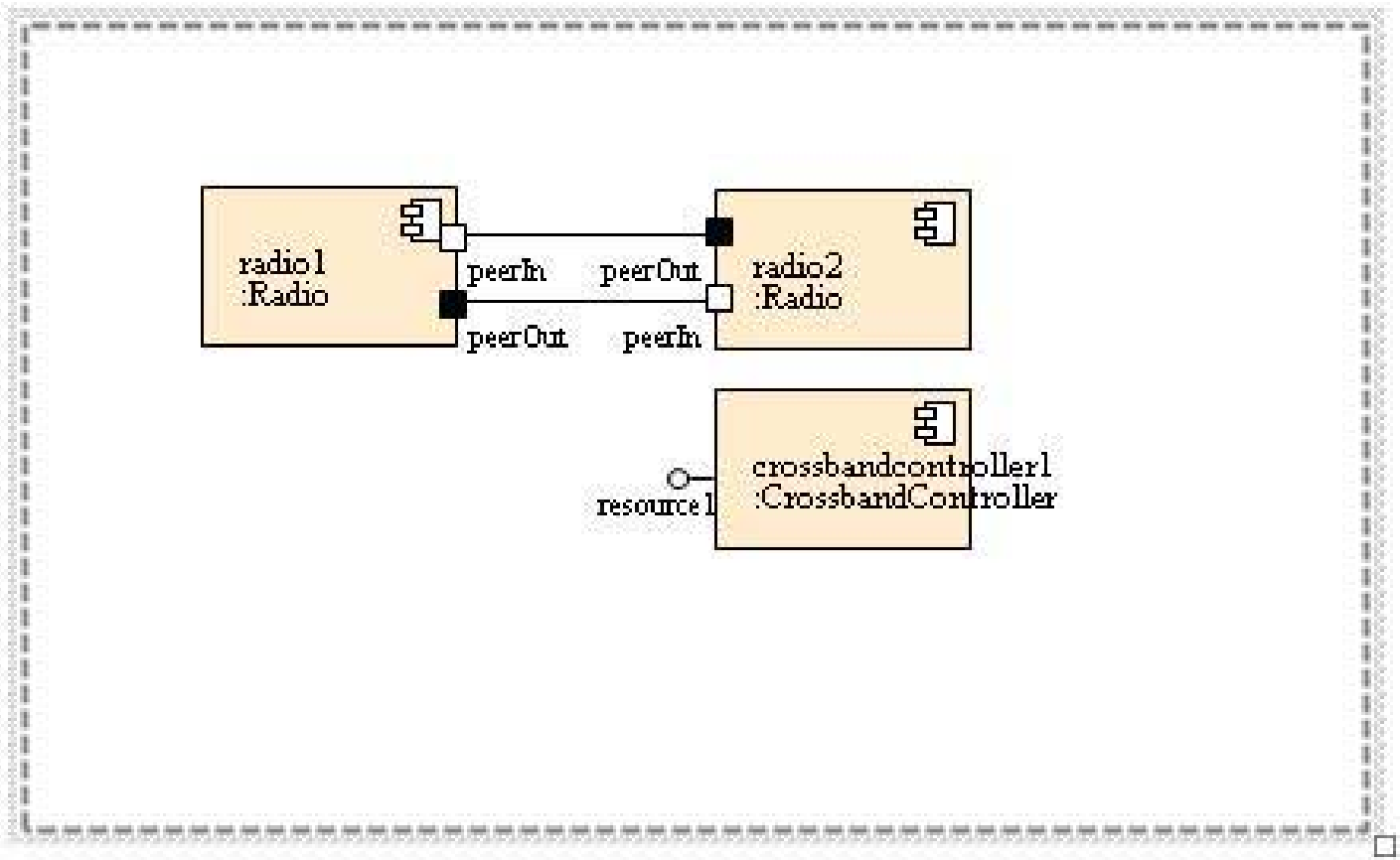
The Reusable Structure

- Abstracting the pattern:



Applying the Structure

- Two connected Radio structures = one cross-banding assembly

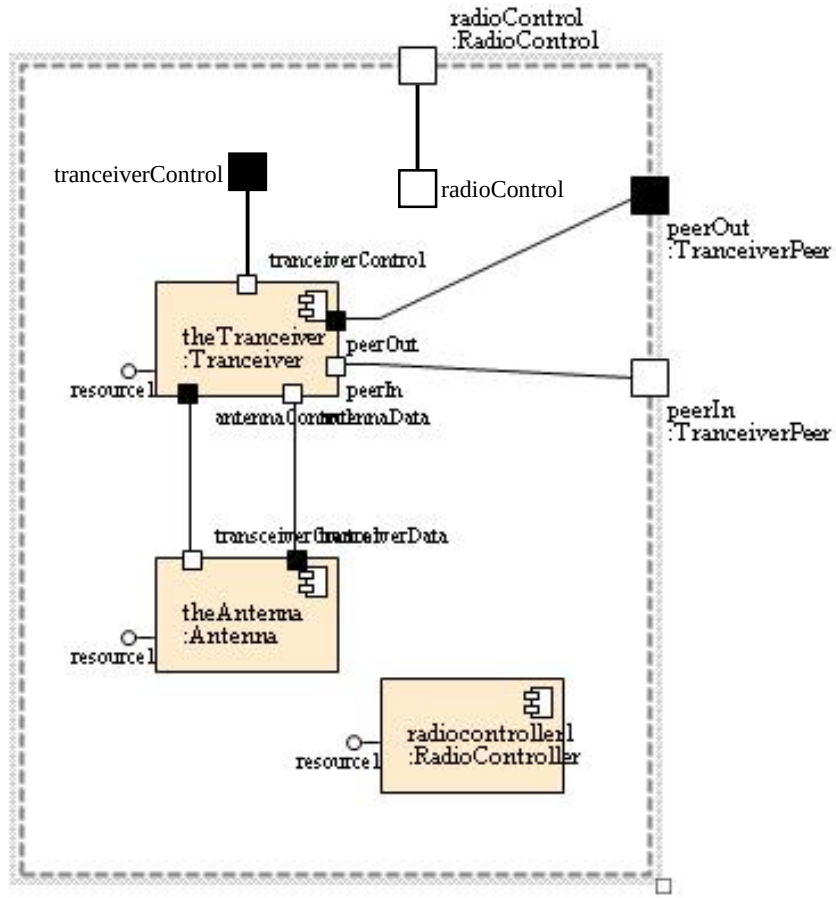


The Key Questions

- Where should the structured component itself go?
  - Where does it appear in diagrams?
  - How are its communication channels shown?
- What should the structured component do?

Connecting Components To Their Structures

- How do components with structure communicate?
- New* graphical concept: internal end port
  - Provides…
  - …or Uses
- Internal ports can connect to internal structure…
- …or to external ports

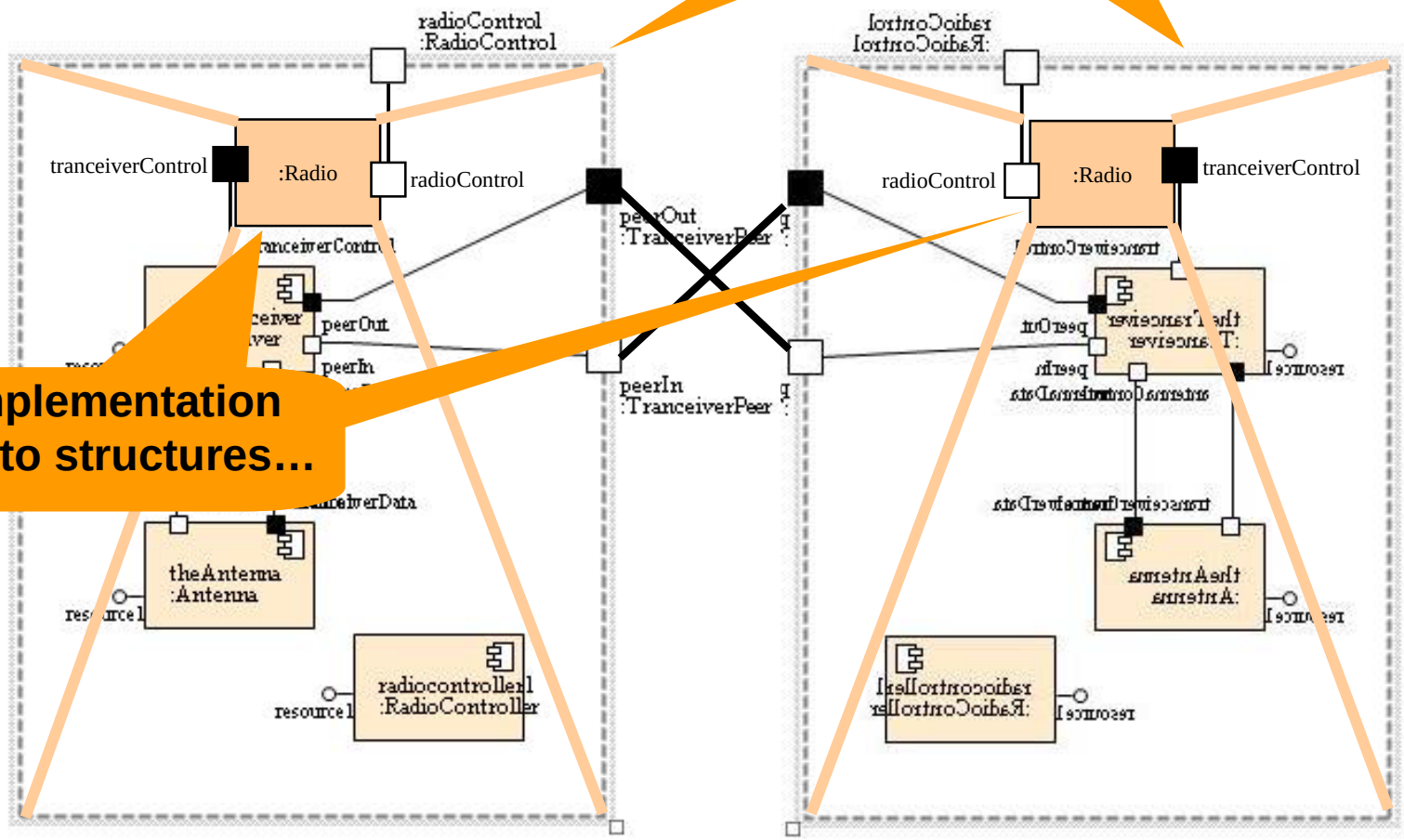


*to SCA; well-established notation

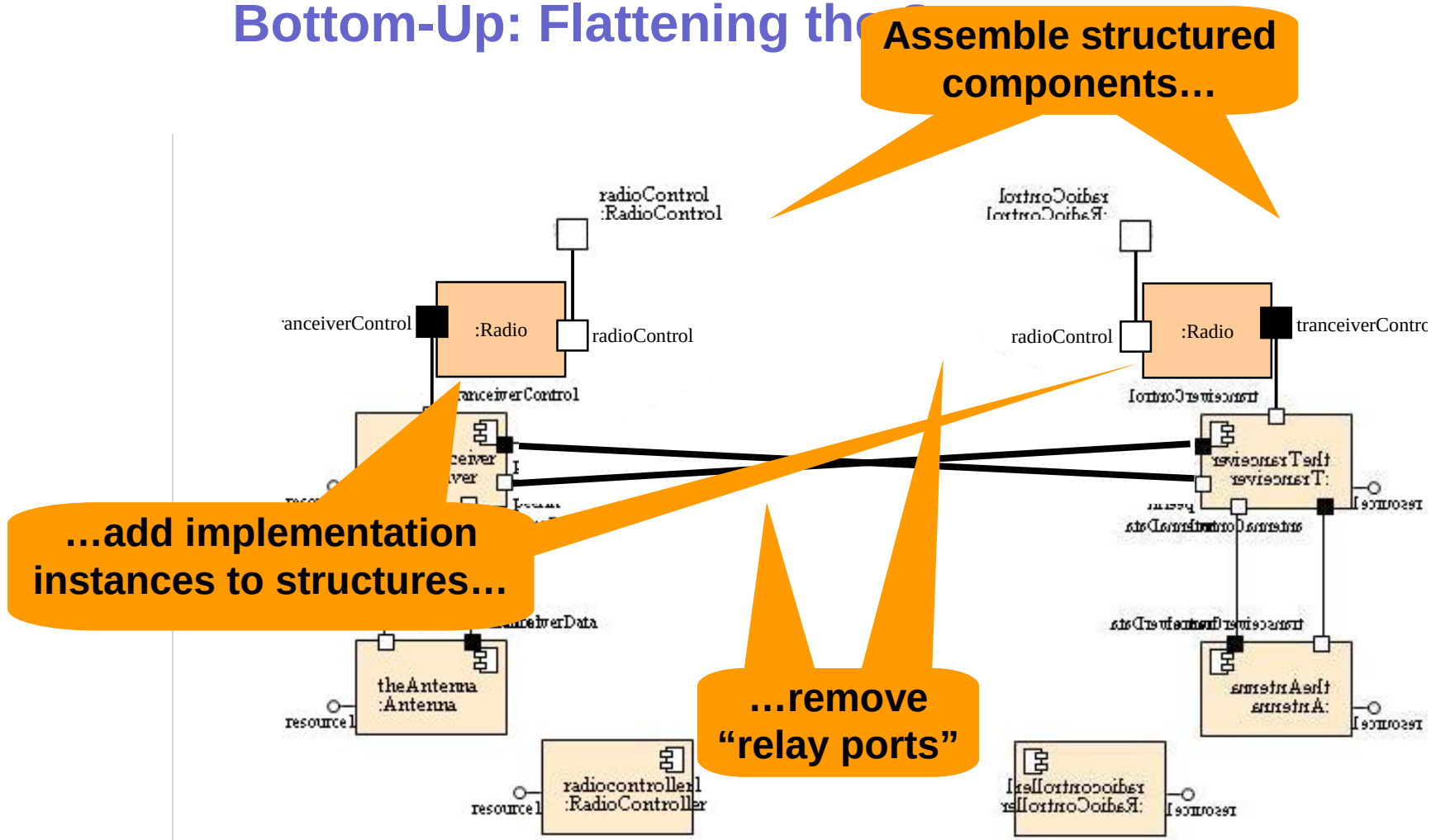
Bottom-Up: Flattening the

Assemble structured components…

…add implementation instances to structures…

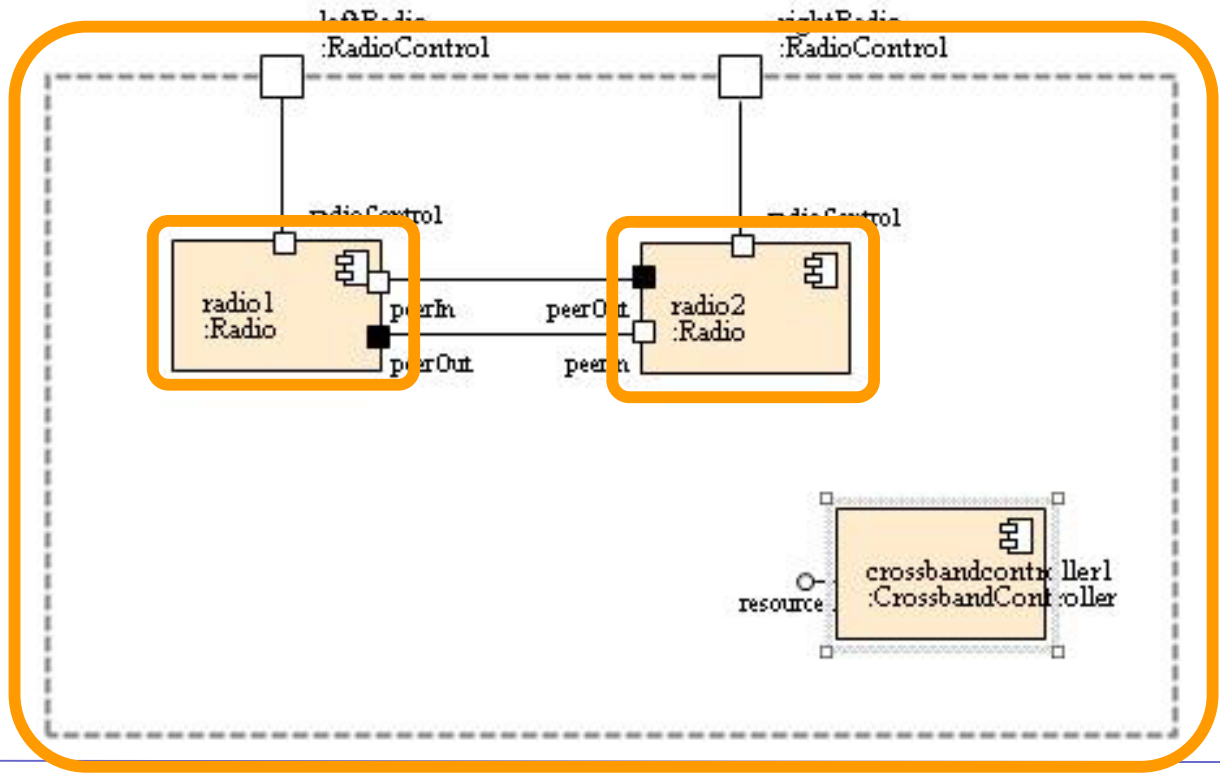


Bottom-Up: Flattening the C



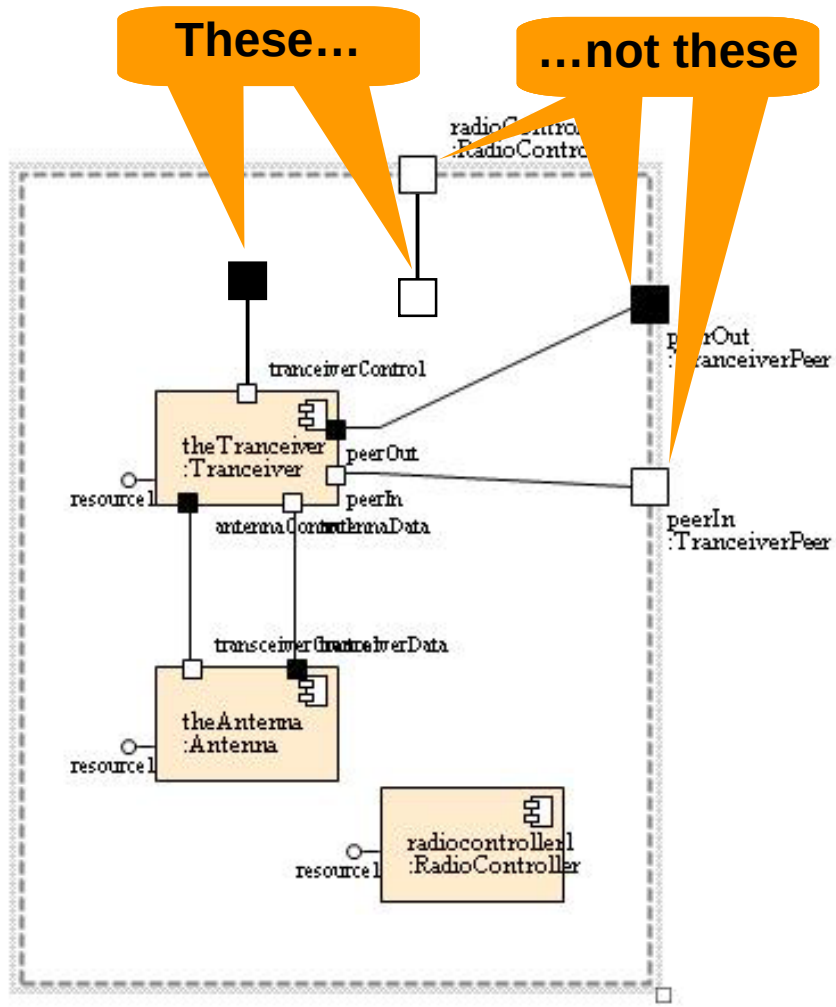
Again, Structures are Fractal

- When an assembly has structured components…
- …maybe it's somebody else's component



Structure and Behaviour

- Structure and implementation behaviours are tightly linked
- Implementation behaviours knows about structure
  - Ports of flattened model, not ports of external structure
- NB: relay ports don't exist in flattened model
  - They are an **external** contract interface
- The **internal** ports form the implementation contract interface



Structured Component Behaviour Best Practices

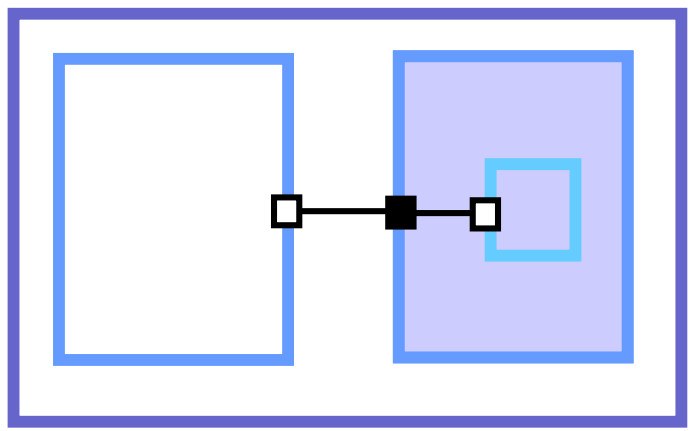
- What should a structured component do?
  - Components should be responsible for creating, controlling and destroying their sub-components
  - I.e., Assembly Controller and Resource Factory functions
- Components with structure should do nothing apart from this structure maintenance
  - If necessary, create a sub-component to perform additional tasks
- A ResourceFactory must tell its components to create or destroy themselves
  - Black-box view of sub-component structures

Reusable Artifacts

- XML descriptor sets are not reusable within structures
  - Reusable at waveform level only
- Structures *are* reusable
  - Generate flattened descriptor set from hierarchy of structured components
- Not surprising:
  - Structures are PIMs
  - Profiles are PSMs
- Not necessary to actually generate flattened structure; generate XML files instead
  - Analogy: compilers generate complete executables from sets of source files

Polymorphism

- Can SCA subcomponents be polymorphic?
  - Can different subcomponent structures be dynamically created and destroyed?
- Yes
  - The solution involves fan-out of connectors crossing relay ports in the generated profile



Summary

- SCA structures are role models—not class models and not object models
- Powerful abstraction and reuse require hierarchical structures
- The SCA can support these today—if the right transformations are applied
- Work at the PIM level, not the PSM level

Thank You.

John Hogg
hogg@zeligsoft.com
www.zeligsoft.com