freedom TO CREATE

A Preview of UML 2.0

Rational
the software development company

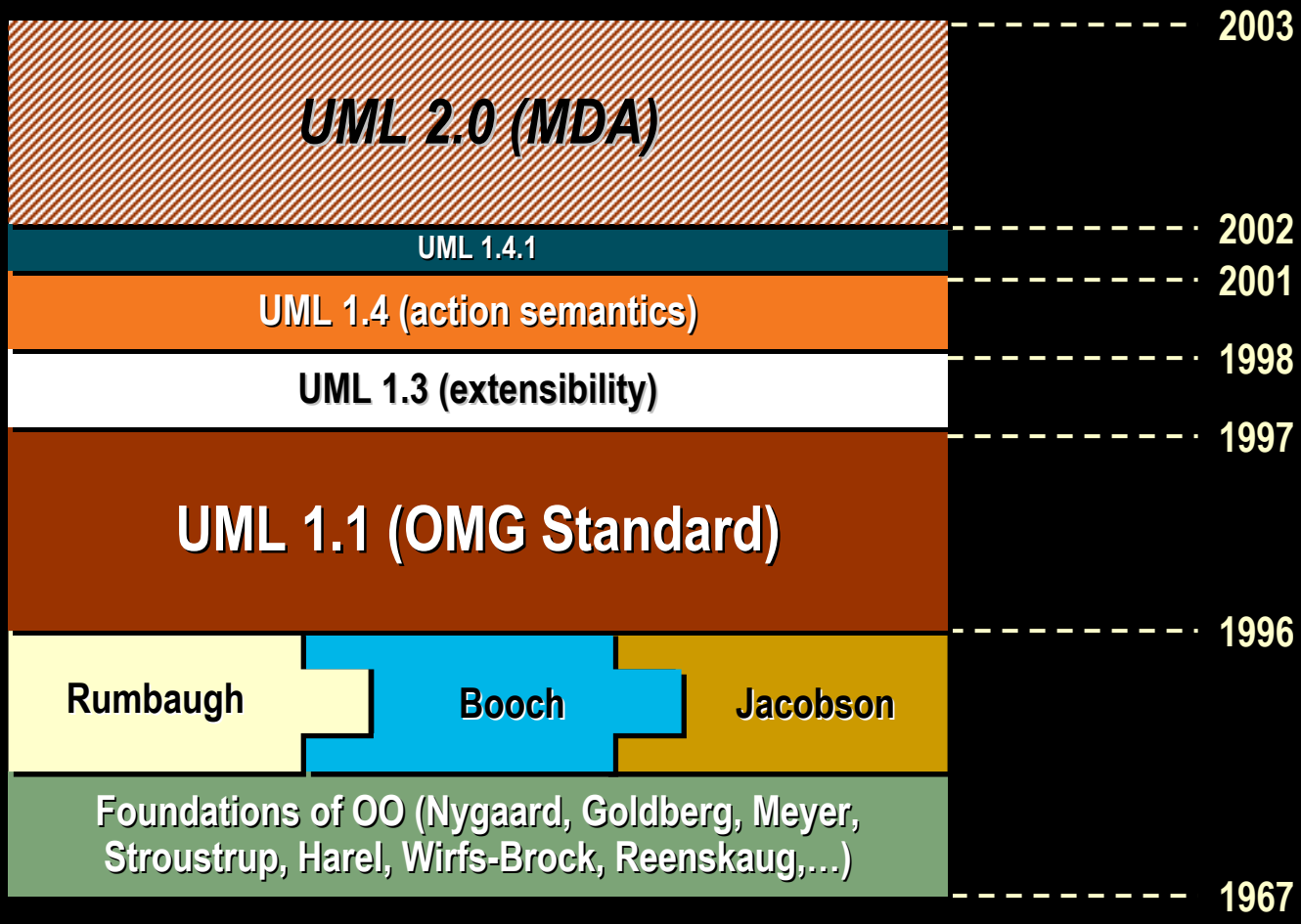
IMPORTANT DISCLAIMER!

The technical material described here is still under development and is subject to modification prior to adoption by the OMG

Overview

- Background
- Requirements for UML 2
- UML 2 Infrastructure Features
- UML 2 Superstructure Features
- Summary

The Evolution of UML



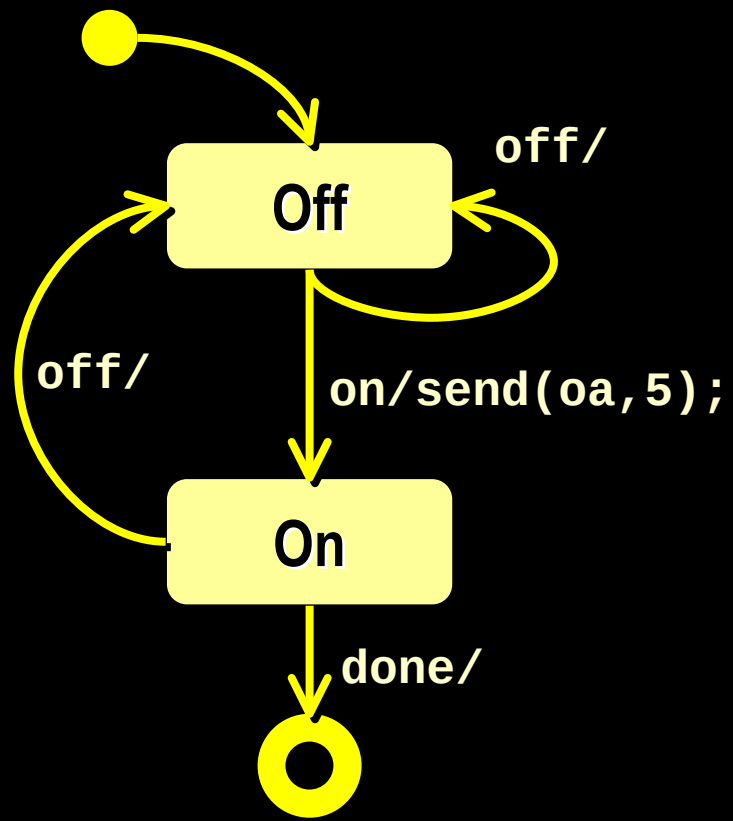
What is UML?

- A language for *modeling* object-oriented software applications
- Why bother?
  - To understand and predict the key characteristics of our design before we go through the expense and effort of building it
  - …and then finding out that it does not work
- Modeling is a key risk mitigation technique shared by all forms of engineering

Models of Software

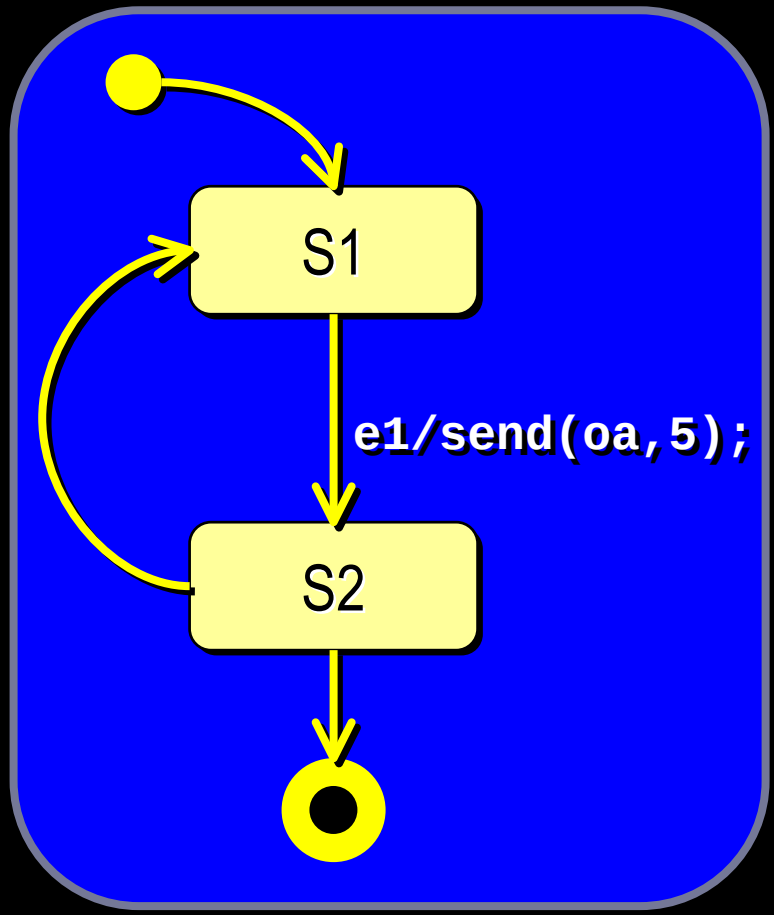
- A description of the software which
  - Abstracts out irrelevant detail
  - Presents the software in problem-domain terms

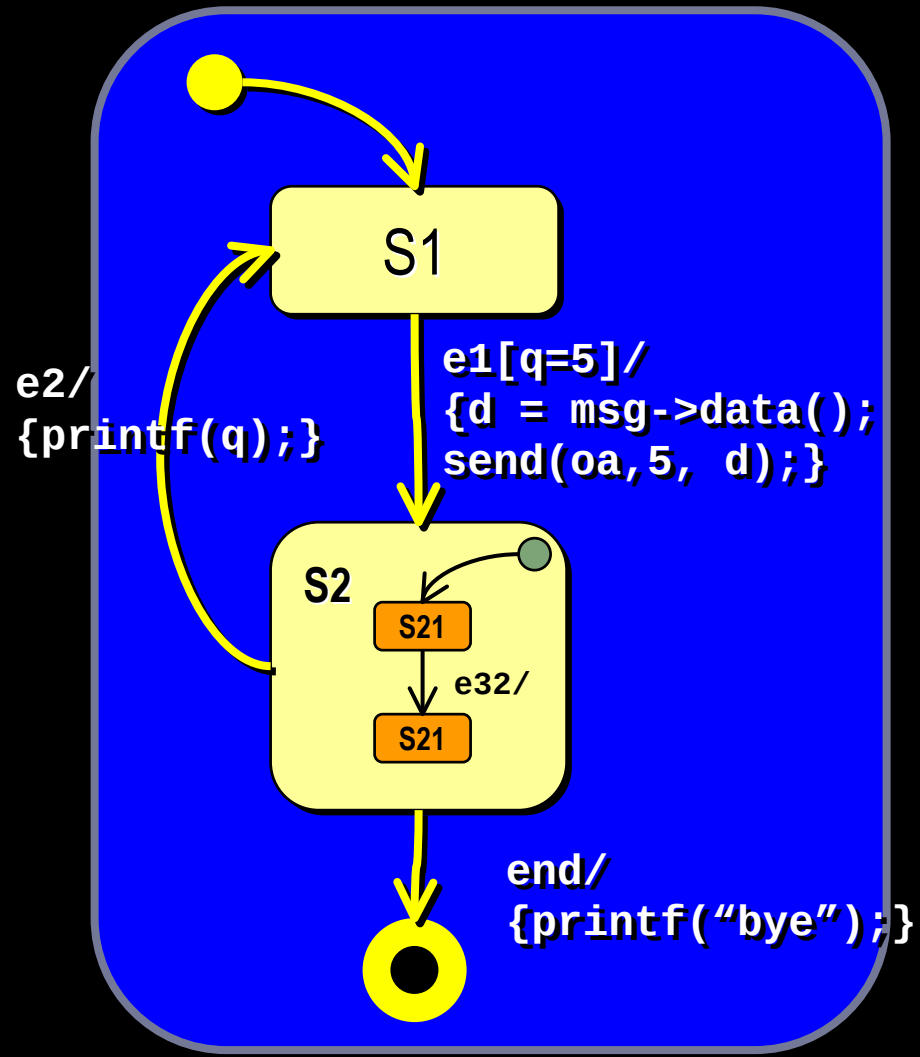
```
case mainState of
   initial: send("I am here");
            end
   Off:     case event of
              on: send(oa,5);
                  next(On);
                  end
              off: next(Off);
                  end

            end
    On:     case event of
              off: next(Off);
                  end
              done: terminate;
                  end
            end
   end
```



Evolving Models

- We can add more detail to make an abstract model more concrete:





The Remarkable Thing About Software

Software has the rare property that allows us to directly evolve models into full-fledged implementations without changing the engineering medium, tools, or methods!

The OMG's Model Driven Architecture

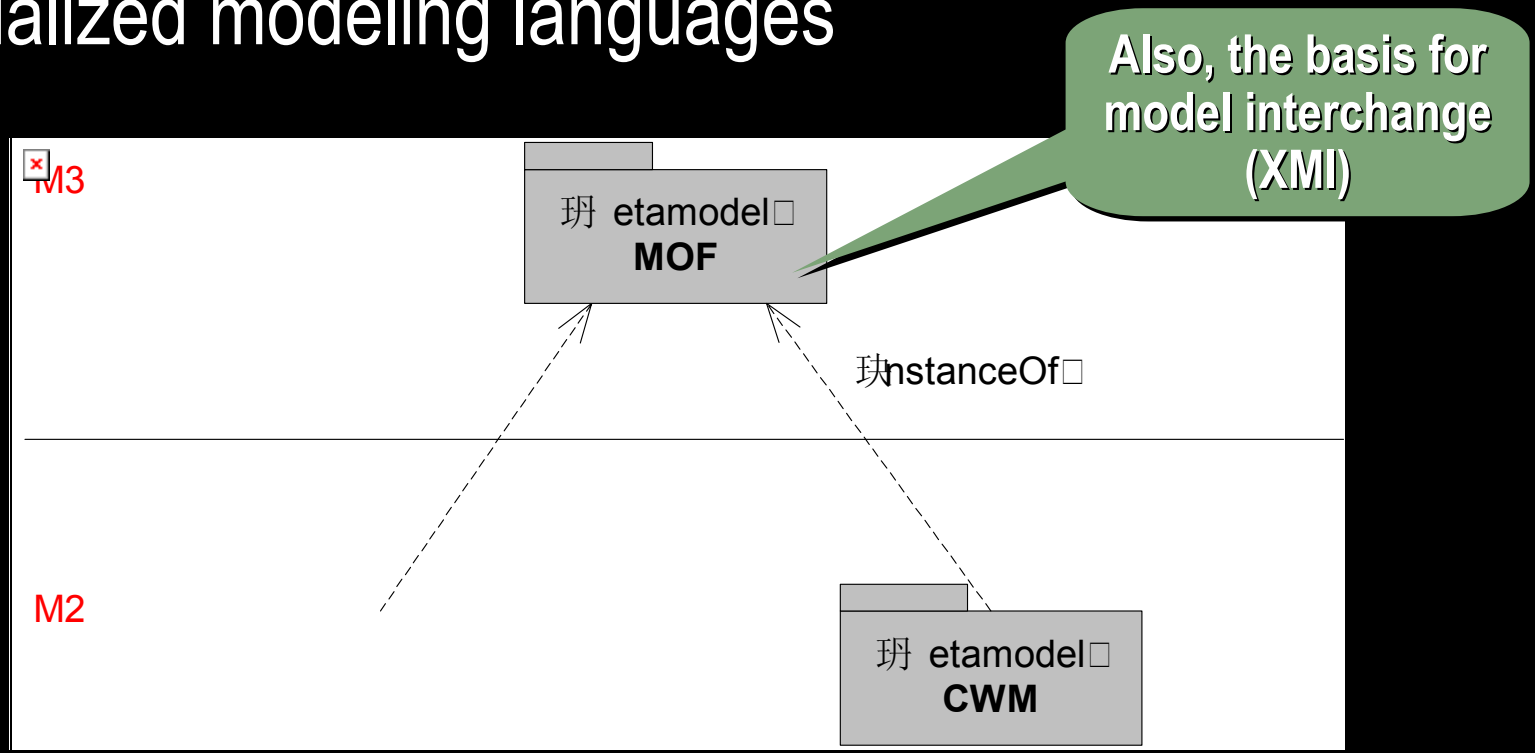
- The OMG has formulated an initiative called "Model-Driven Architecture" (MDA)
  - A framework for a set of standards in support of a *model-centered* style of development
  - Inspired by the widespread public acceptance of UML
- Key characteristic of MDA:
  - *The focus and principal products of software development are models* (instead of programs)
  - Models all the way – the design *is* the implementation
- Rational is a pioneer of model-driven development and is one of the principal drivers of MDA

MDA Implications

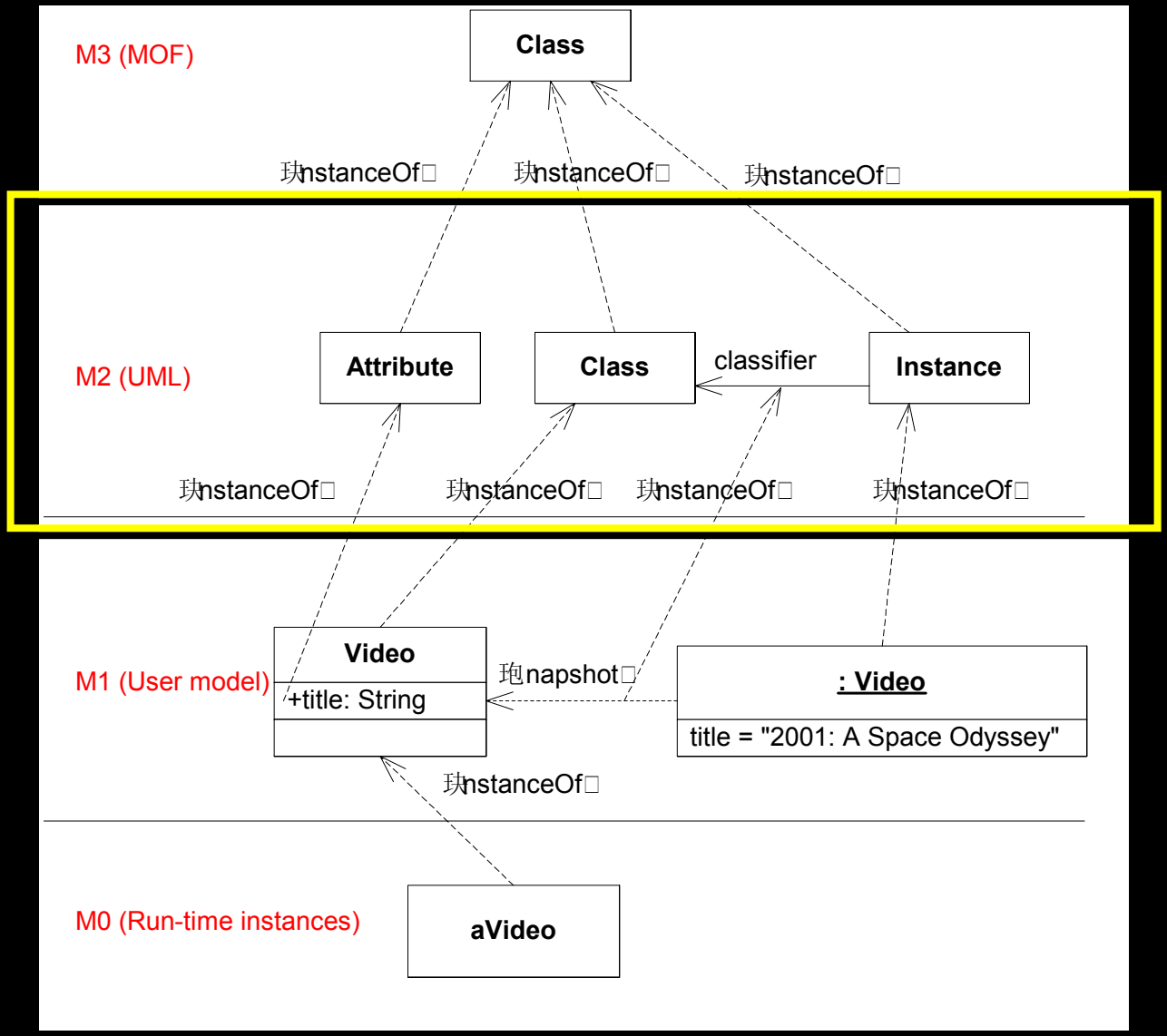
- Ultimately, it should be possible to:
  - Execute UML models
  - Translate them automatically into implementations
  - …possibly for different implementation platforms
  - ⇨Platform independent models (PIMs)
- Modeling language requirements
  - The semantic underpinnings of modeling languages must be precise and unambiguous
  - It should be possible to easily specialize a modeling language for a particular domain
  - It should be possible to easily define new specialized languages

The Meta-Object Facility (MOF)

- A small subset of UML is used to define UML itself
  - Basic concepts: Class, Association, Generalization, Package…
- This subset is also useful for defining other more specialized modeling languages



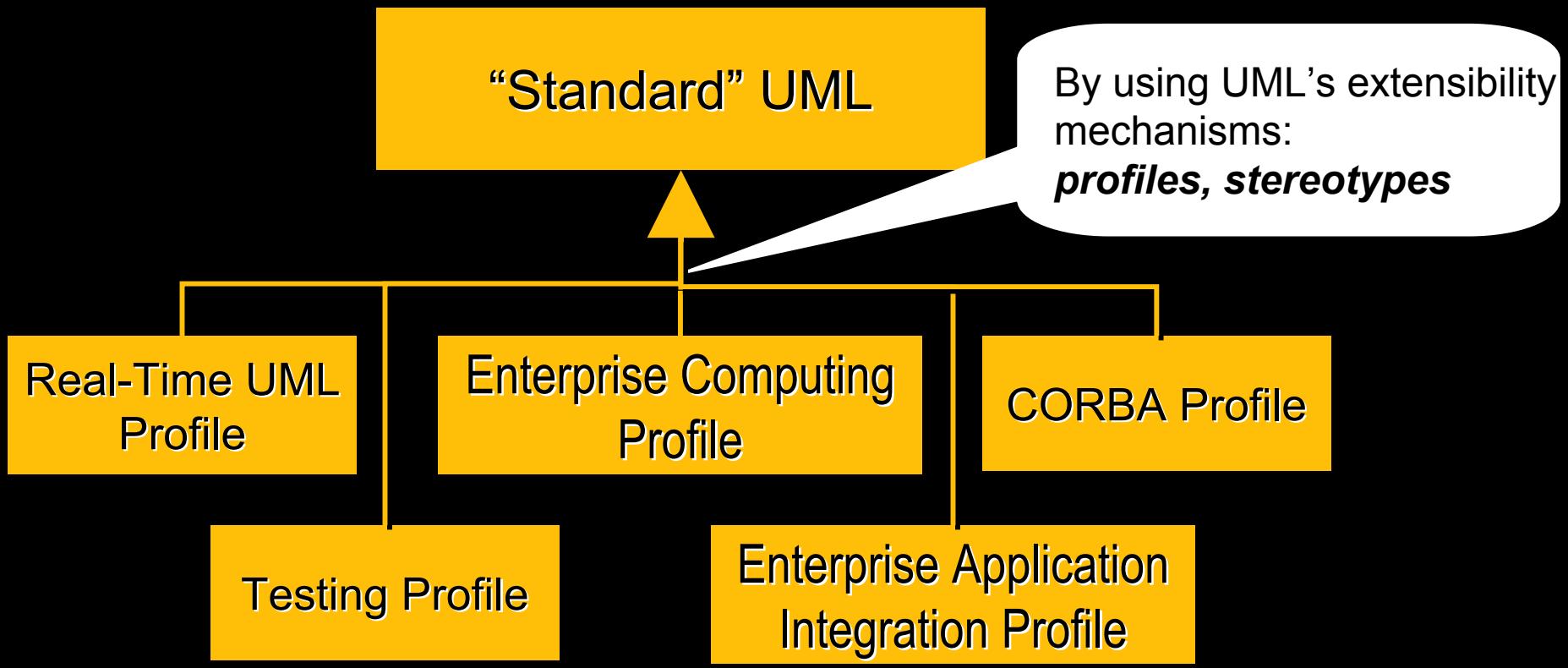
The 4-Layer Modeling Language Architecture



The UML Metamodel: defines semantics and syntax

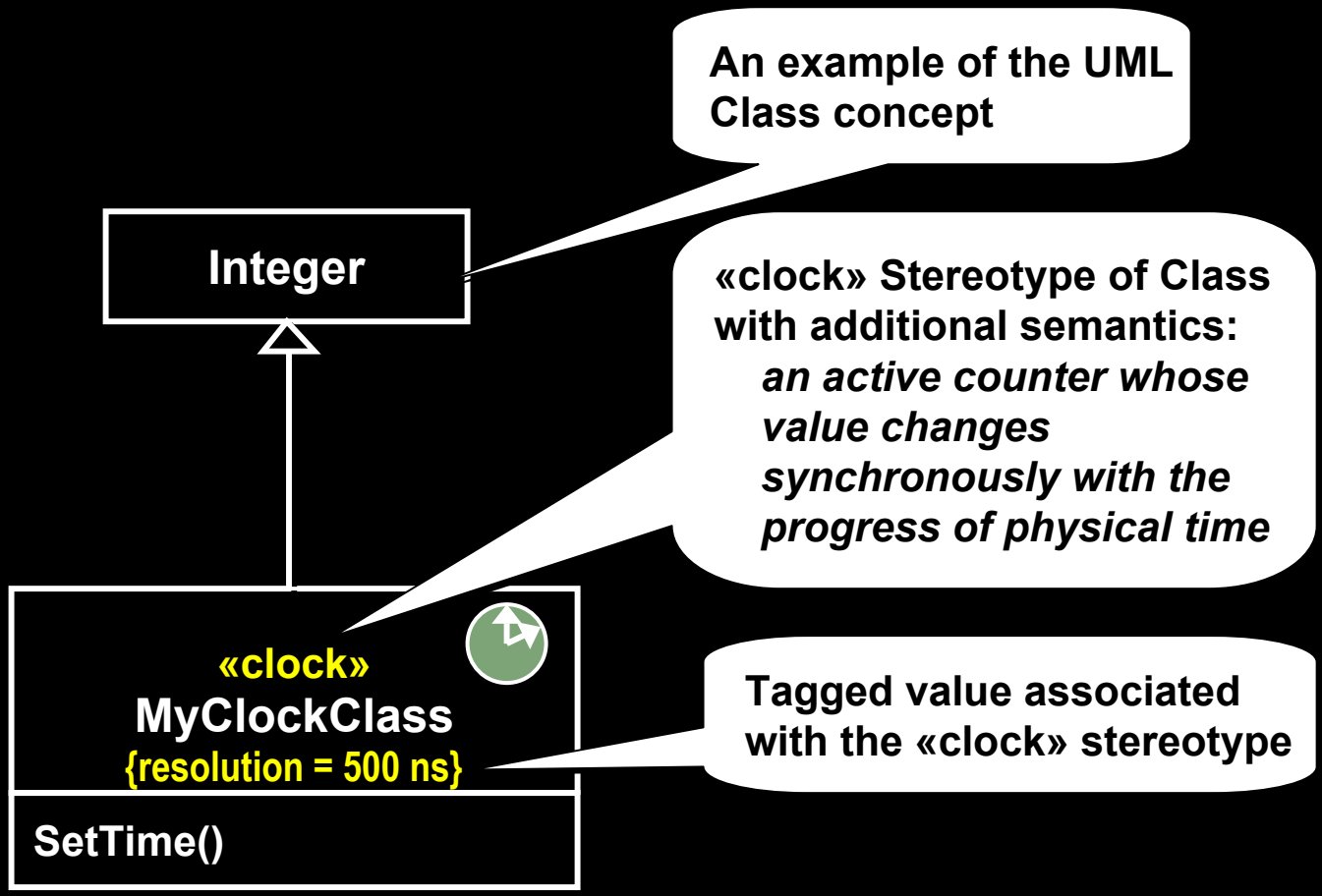
Specializing UML: The Family of Languages

- Multiple languages with a common semantic base
  - By narrowing or removing semantic variation points



Specializing UML: Stereotypes

- We can add semantics to any standard UML concept
  - Must not violate standard UML semantics



- Background
- Requirements for UML 2
- UML 2 Infrastructure Features
- UML 2 Superstructure Features
- Summary

Sources of Requirements

- MDA
  - Semantic precision
  - Consolidation of concepts
  - Full MOF-UML alignment
- Practitioners
  - Conceptual clarification
  - New features, new features, new features…
- Language theoreticians
  - My new features, my new features, my new features…
  - Why not replace it with my modeling language instead?
- Dilemma: how to avoid the insidious “language bloat” syndrome

Approach: Slow but Steady

- Evolution rather than revolution
  - Little or no impact on current user base
- Consolidation, improved precision, and a small number of carefully chosen new features
- Feature selection criteria
  - Required for supporting large industrial-scale applications
  - Non-intrusive on UML 1.x users (and tool builders)

Formal RFP Requirements

Four separate but related sets of requirements

1) Infrastructure – UML internals
   - Make the conceptual foundations of UML more precise for better MDA support
2) Superstructure – User-level features
   - New features
   - Consolidation of existing features
3) OCL – Constraint language
4) Diagram interchange standard

Infrastructure Requirements

- Precise MOF alignment
  - Fully shared “common core” metamodel
- Refine the semantic foundations of UML (the UML metamodel)
  - Improve precision
  - Harmonize conceptual foundations and eliminate semantic overlaps
  - Provide clearer and more complete definition of instance semantics (static and dynamic)
- Improve extension mechanisms
  - Profiles, stereotypes
  - Support “family of languages” concept

OCL Requirements

- Define an OCL metamodel and align it (formally) with the UML metamodel
- Add new modeling features available to general UML users
  - E.g., ability to express business rules using OCL

Diagram Interchange Requirements

- Ability to exchange graphical information between tools
  - Currently only non-graphical information is preserved during model interchange
  - Diagrams and contents (size, position, etc.)

Superstructure Requirements (1 of 2)

- More direct support for architectural modeling
  - Based on existing architectural description languages (UML-RT, ACME, etc.)
  - Reusable interaction specifications (UML-RT protocols)
- Behavior harmonization
  - Generalized notion of behavior and causality
  - Support choice of formalisms for specifying behavior
- Hierarchical interactions modeling
- Better support for component-based development
- More sophisticated activity graph modeling
  - To better support business process modeling

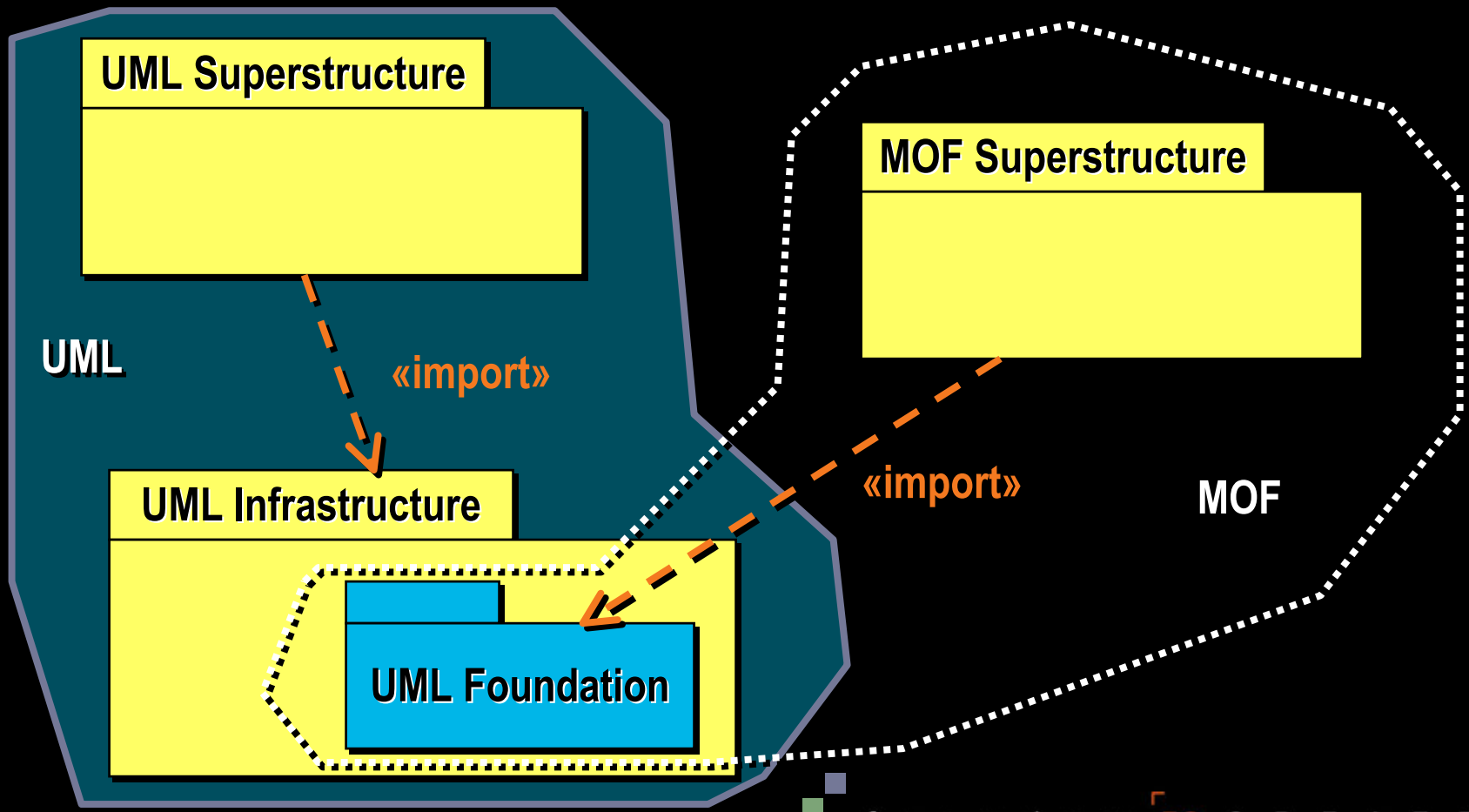
Superstructure Requirements (2 of 2)

- New statechart capabilities
  - Modular states
- Clarification of semantics for key relationship types
  - generalization, refinement, realization, deployment
- Remove unused and ill-defined modeling concepts
- Precise mapping of notation to metamodel
- Backward compatibility
  - Support 1.x style of usage
  - New features only if required

- Background
- Requirements for UML 2
- UML 2 Infrastructure Features
- UML 2 Superstructure Features
- Summary

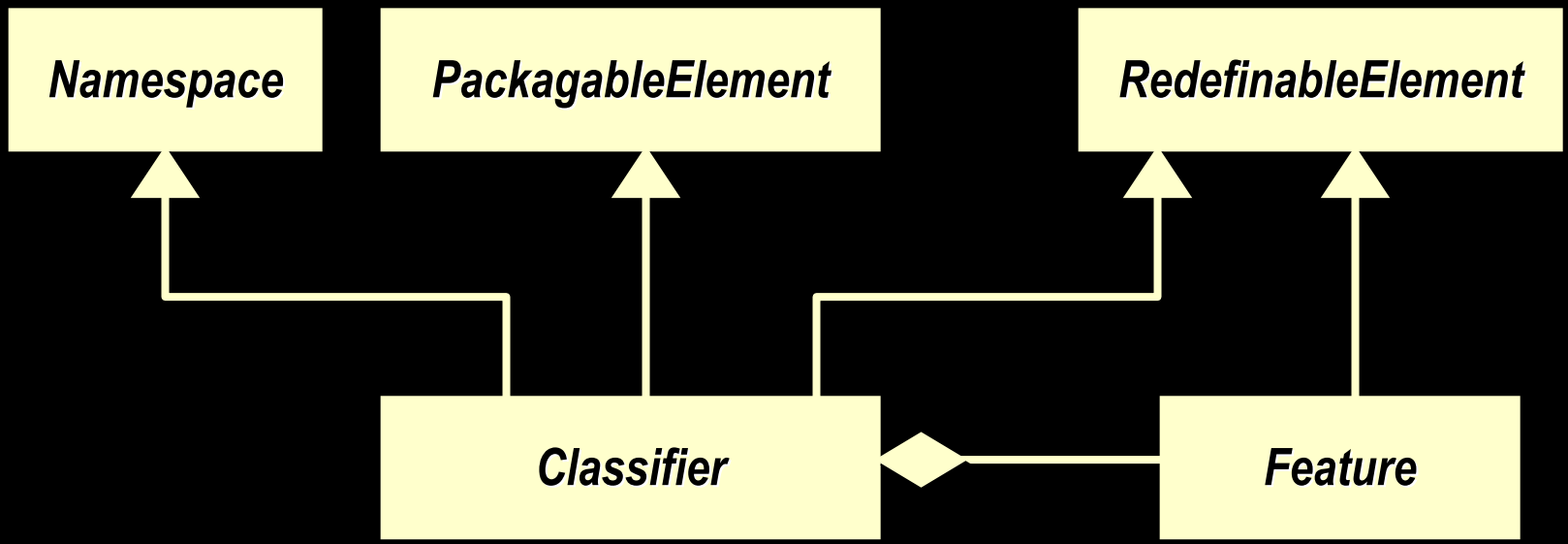
UML-MOF Alignment

- Shared conceptual base
  - MOF: language for defining modeling languages
  - UML: general purpose modeling language



Infrastructure: Consolidation of Concepts

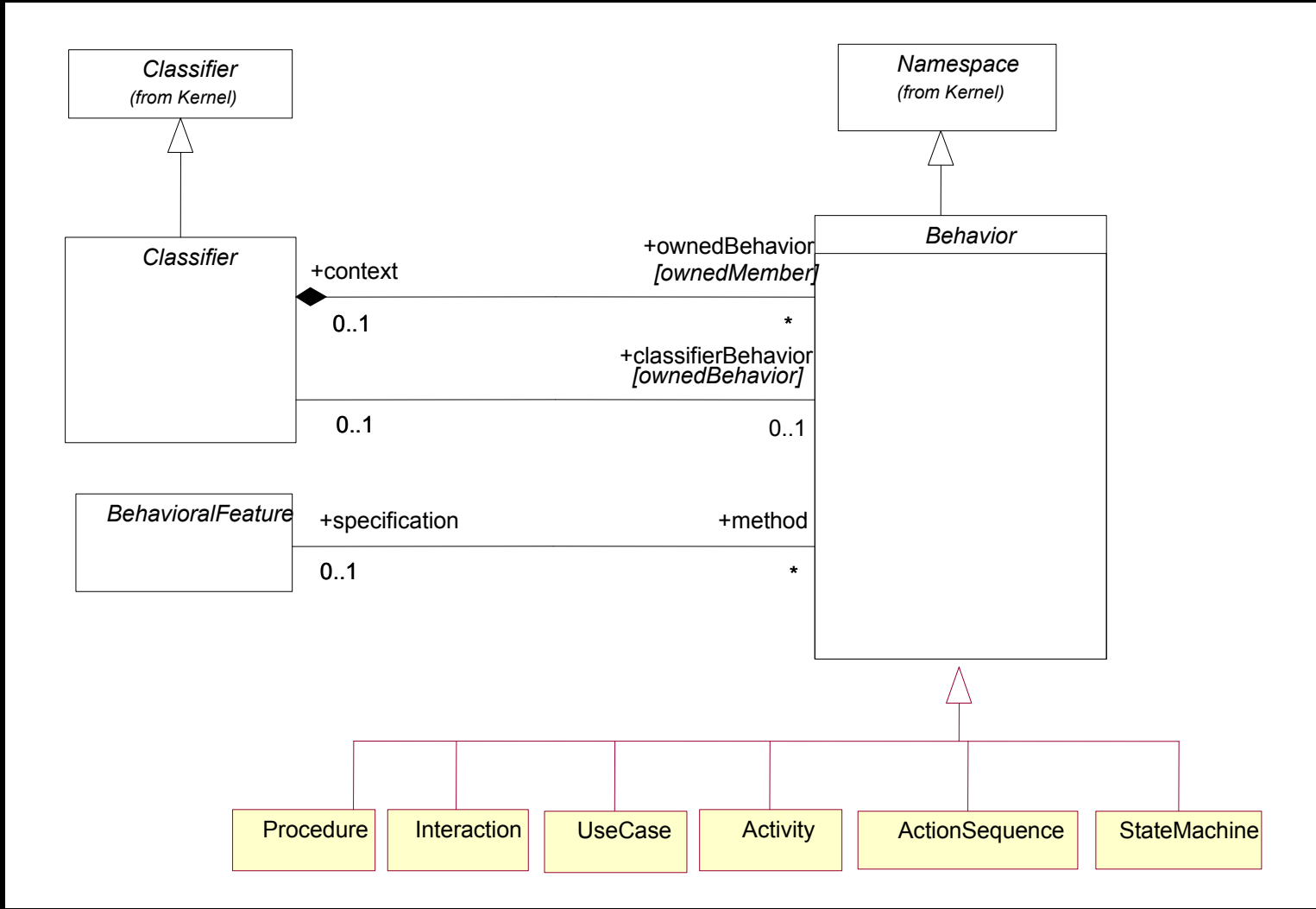
- Breakdown into fundamental conceptual primitives



- Eliminates semantic overlap
- Better foundation for a precise definition of concepts and semantics

Infrastructure: Behavior Harmonization

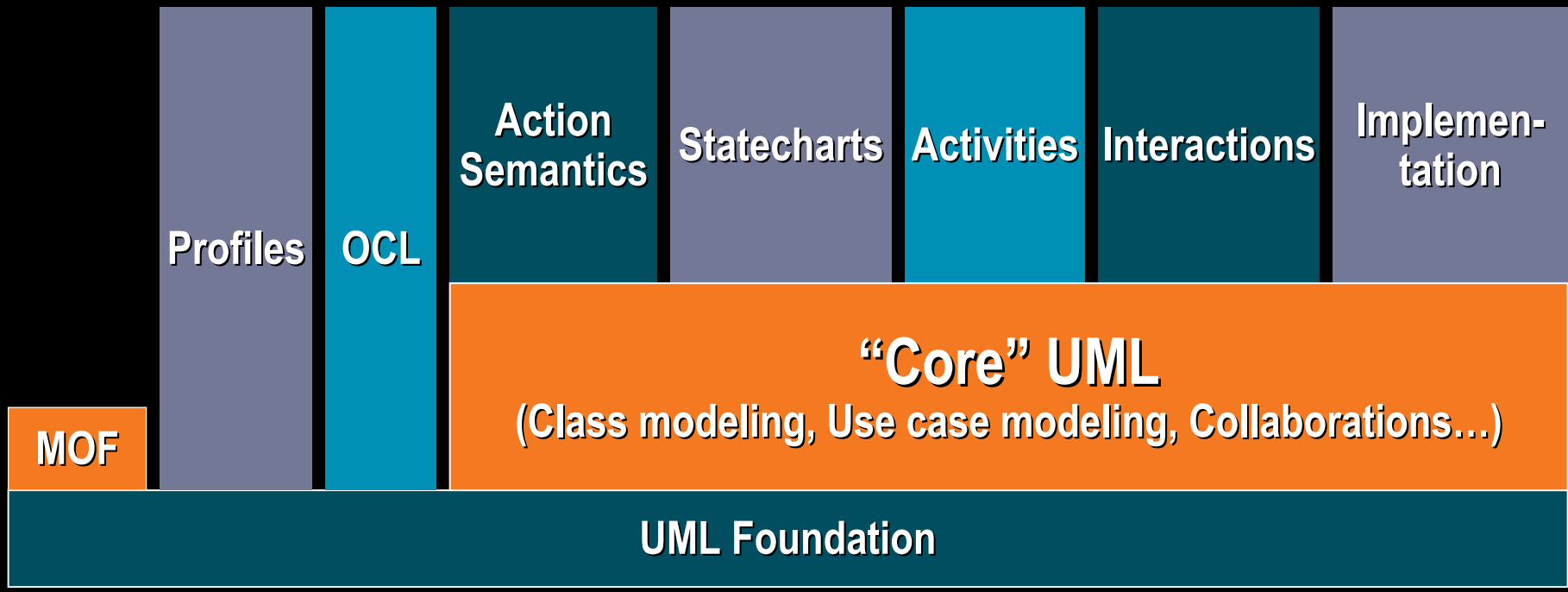
- Consolidation of different behavioral formalisms



- Background
- Requirements for UML 2
- UML 2 Infrastructure Features
- UML 2 Superstructure Features
- Summary

Components of UML 2.0

- A core language + a set of optional specialized "sub-languages"

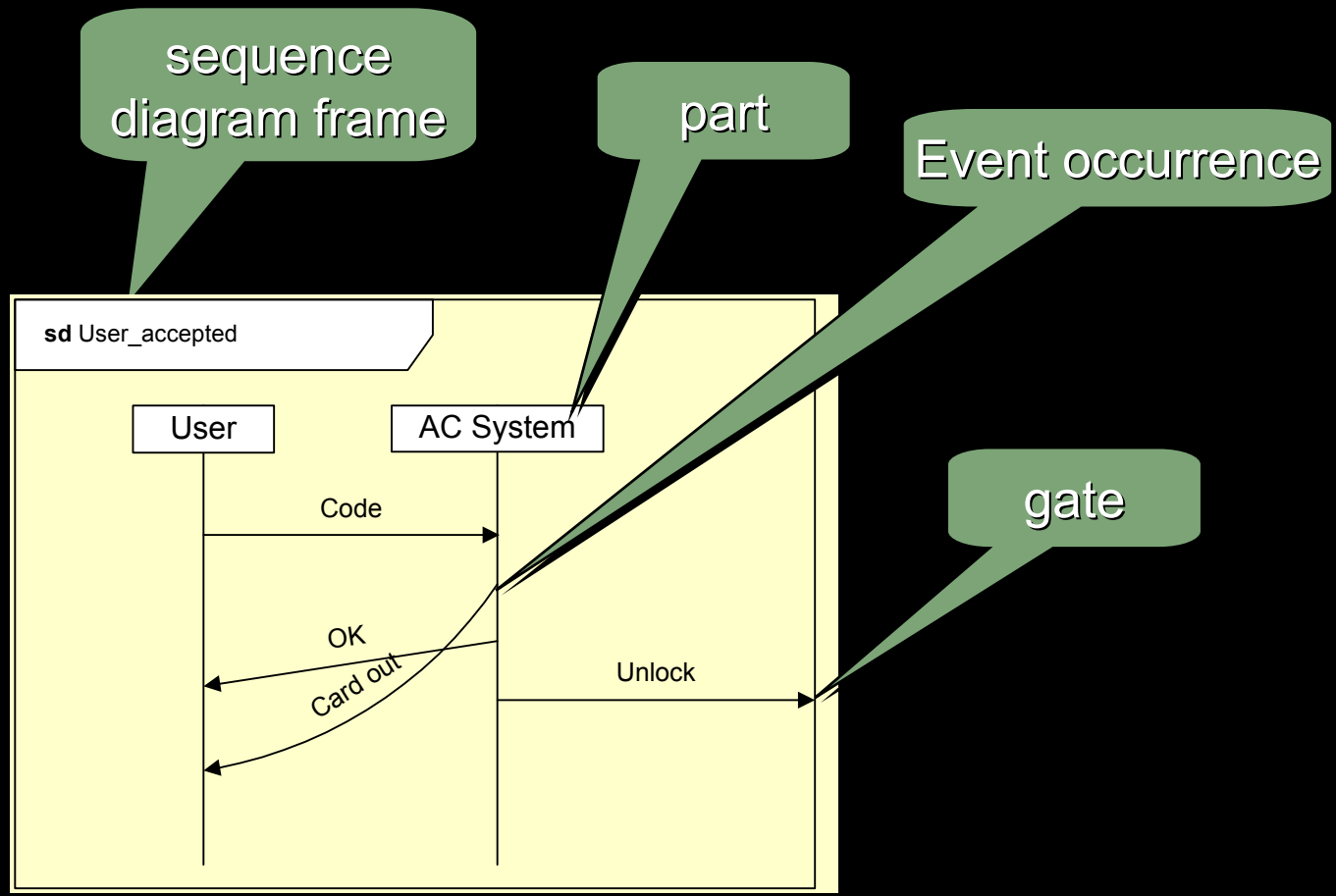


Important New Features

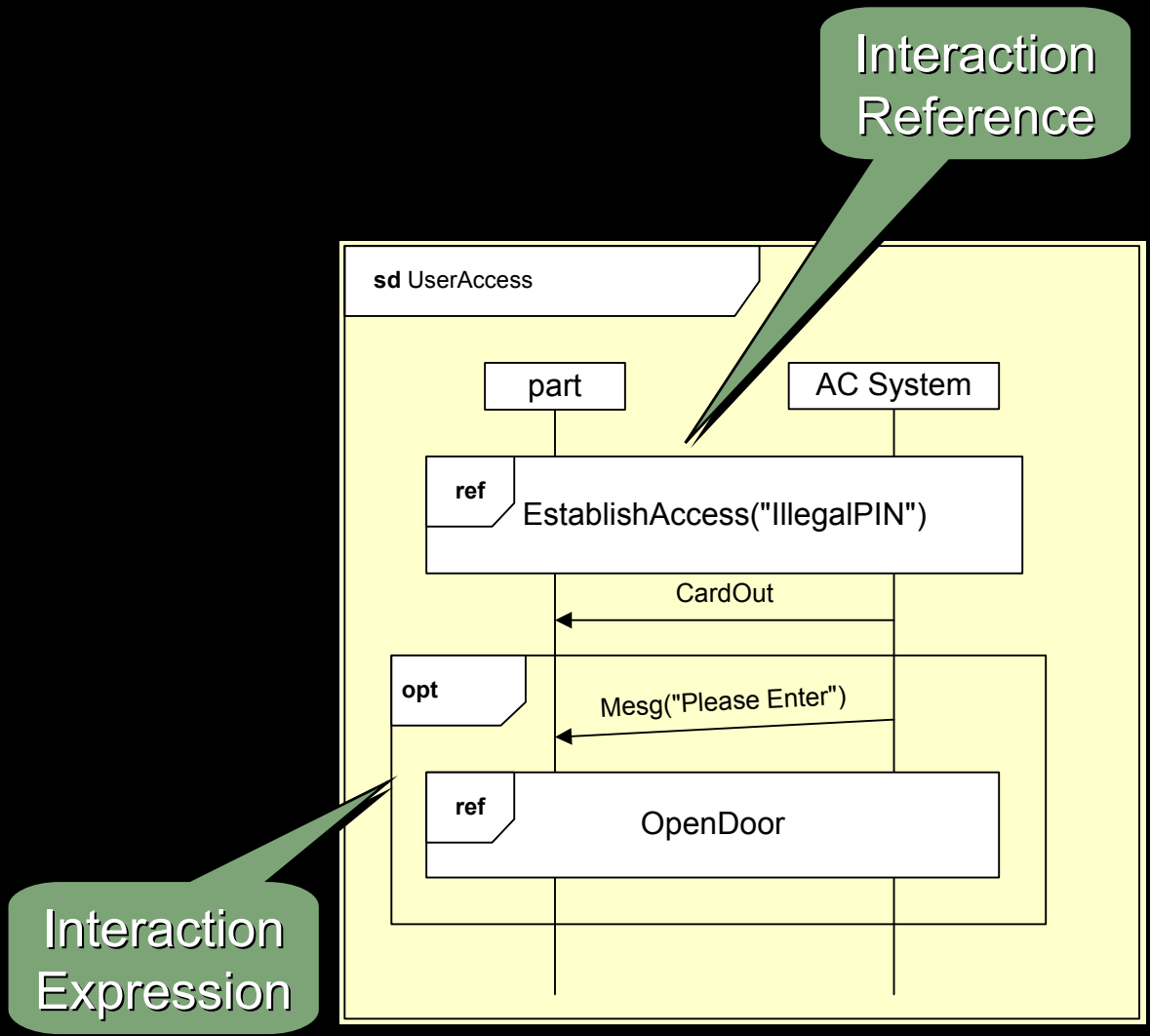
- Interactions
  - Overlays on collaboration structures
  - Need to support complex interactions (conditional sequences, loops, etc.)
  - Hierarchical composition
- Activities
  - New conceptual foundation for greater flexibility
  - Improved support for business process modeling
- Instance-oriented structure modeling
  - Structures of collaborating parts (roles)
  - Classes with internal structure

Interactions: Frames

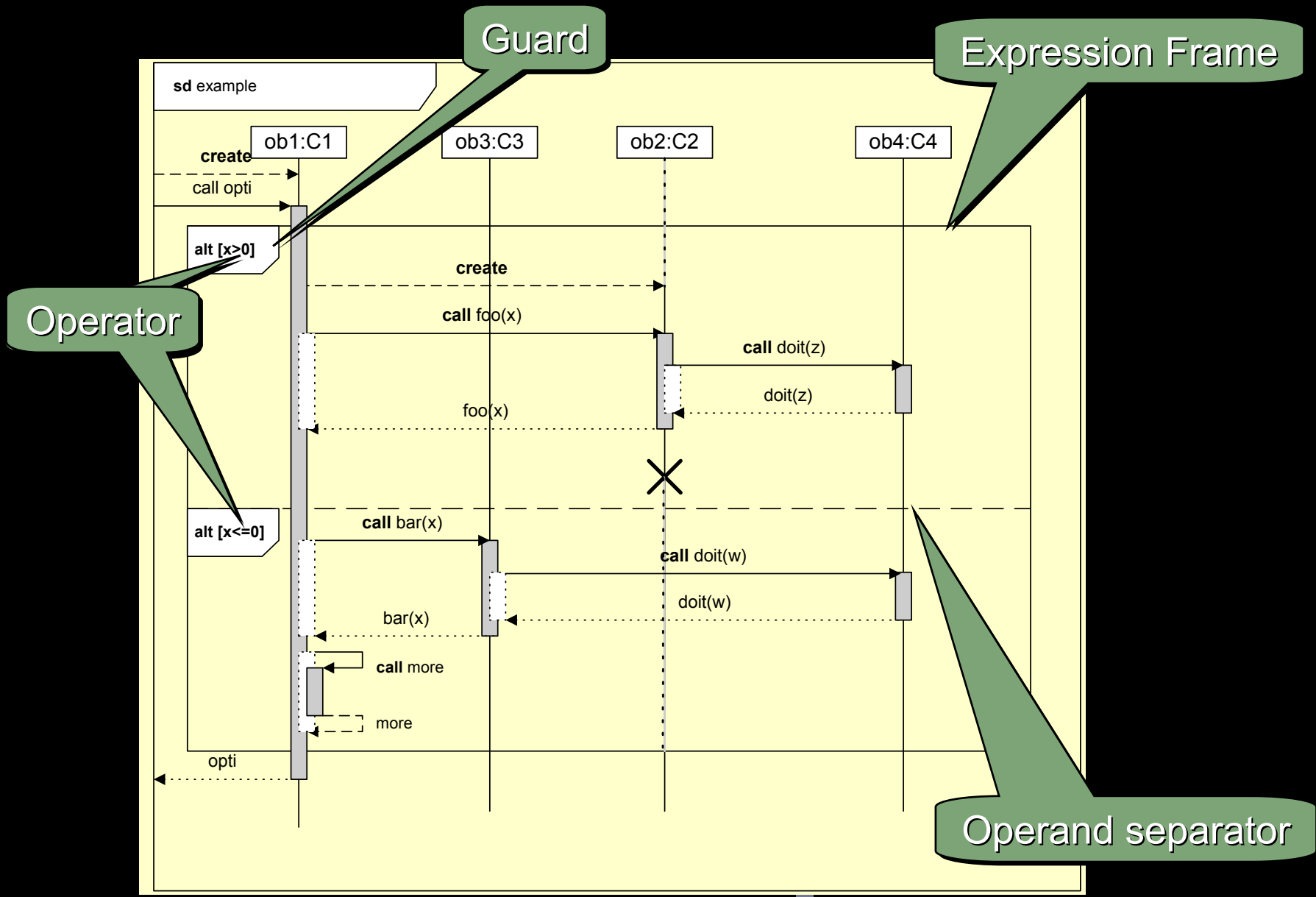
- Reusable interaction diagram fragments
  - Based on proven ITU-T standard Z.120



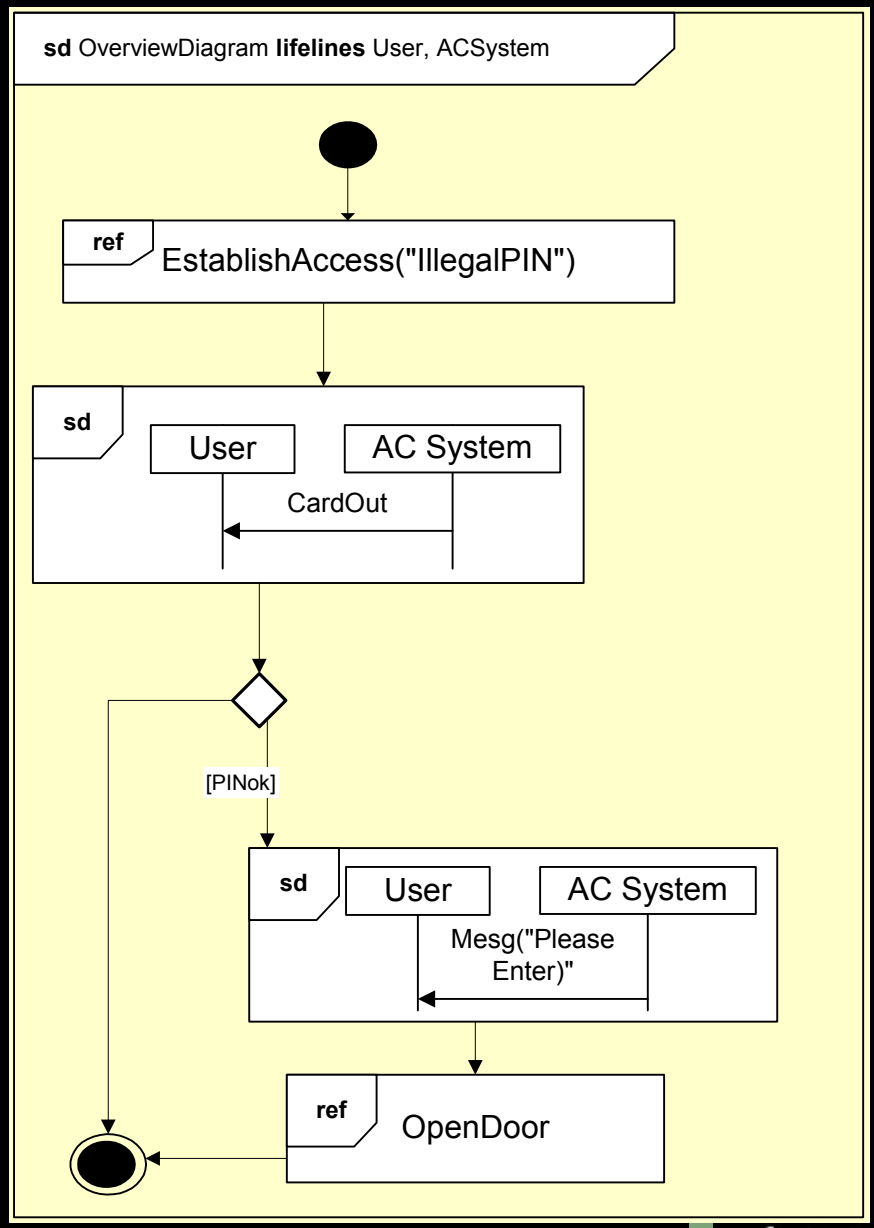
Interactions: Frame Composition



Interactions: Complex Expressions

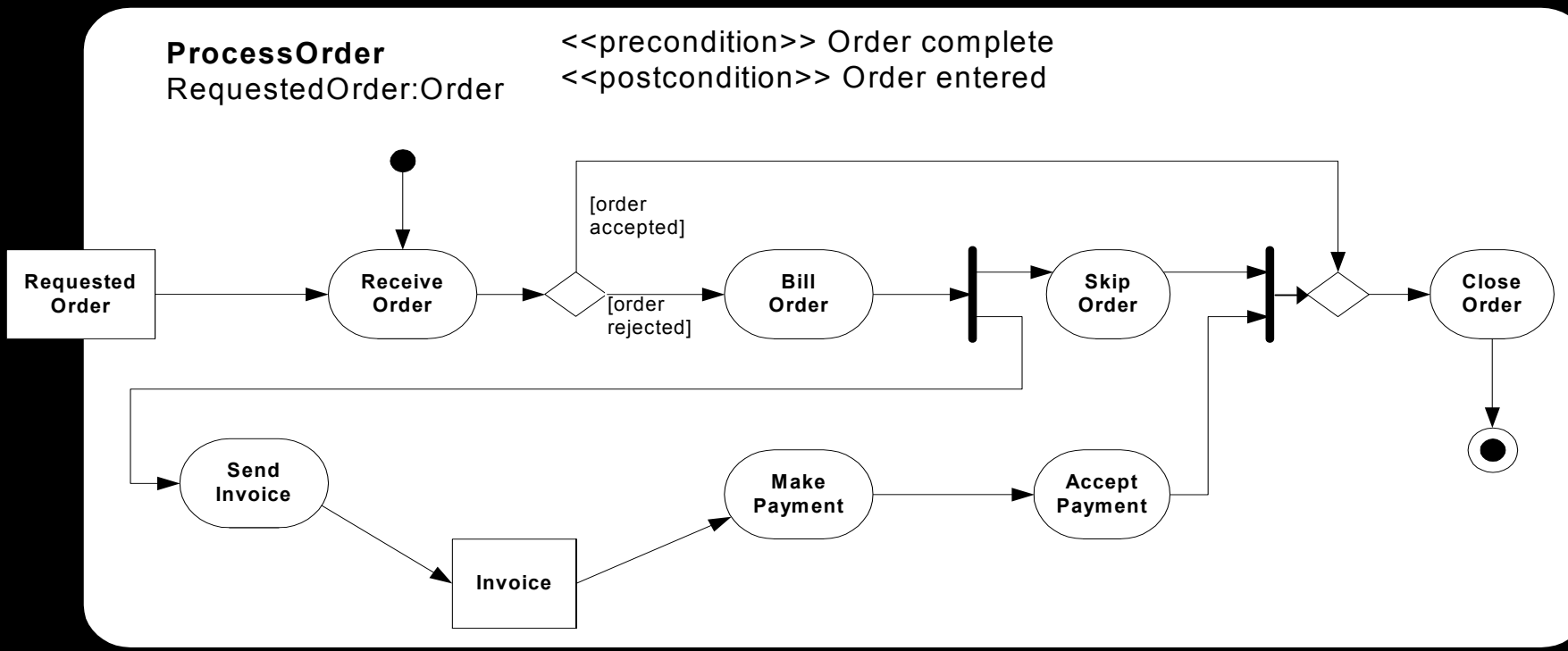


Interaction Overview Diagram

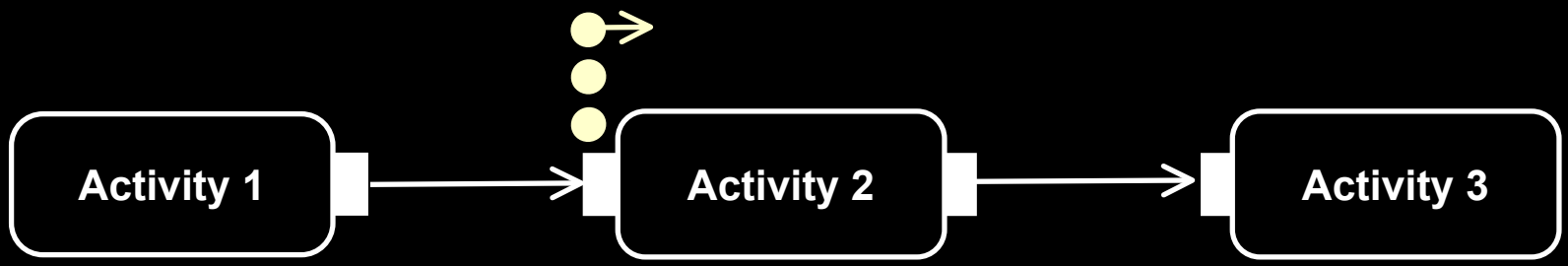


Activities: New Semantic Foundation

- Petri Net foundation (vs. statecharts) enables
  - Un-structured graphs (graphs with “go-to’s”)
  - True concurrency

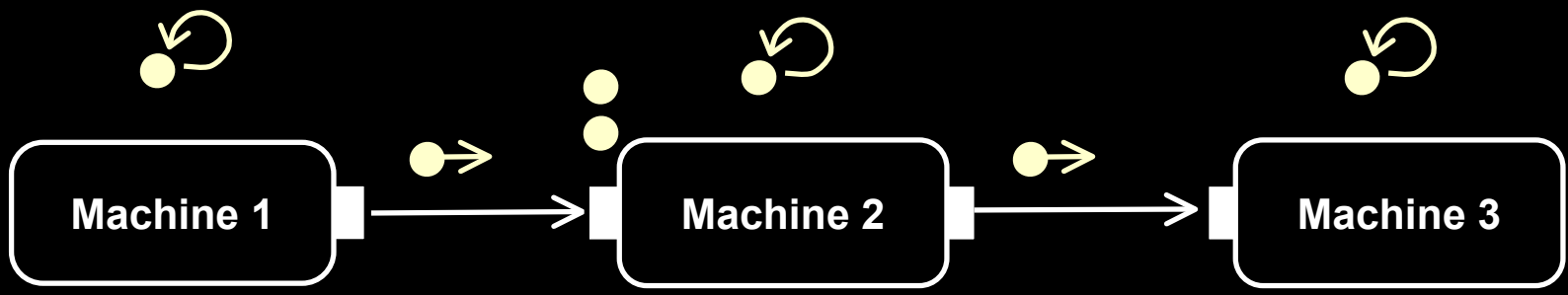


Activities: Queuing Capabilities



- Tokens can
  - stack up in “in/out” boxes.
  - backup in network.
  - prevent upstream behaviors from taking new inputs.
- For modeling systems with significant resource constraints, such as physical systems.

Streaming Parameters



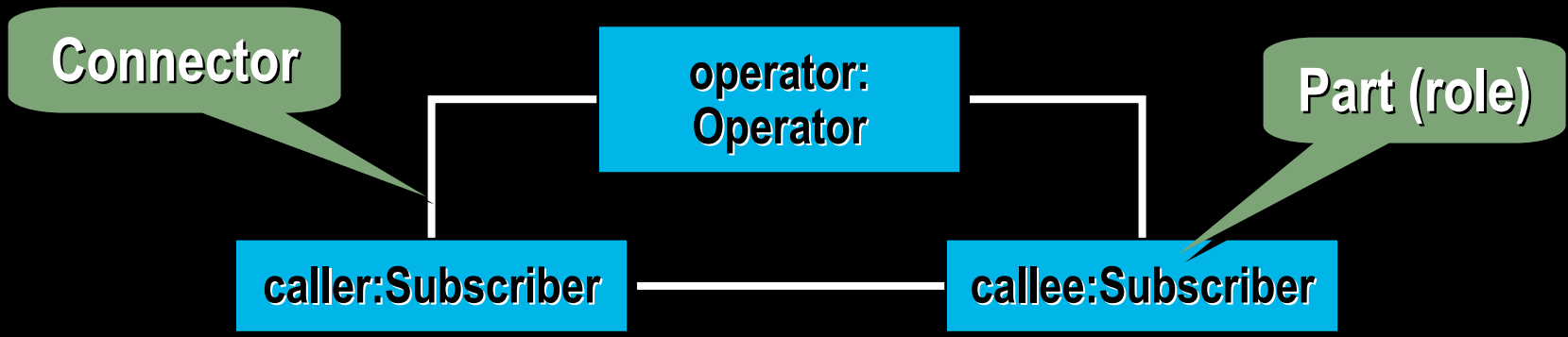
- Tokens can be
  - taken as input while behavior is executing.
  - given as output while behavior is executing.
- For systems of independent, interacting agents.

Structured Classes in UML 2.0

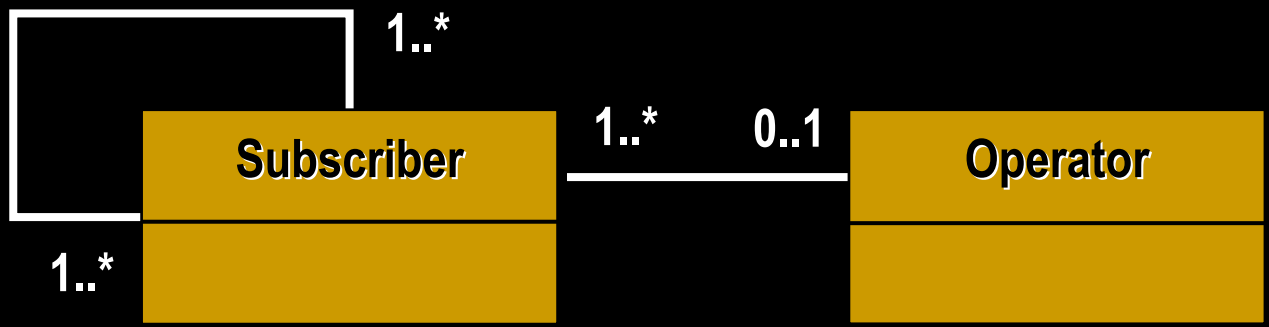
- *Structure*:
  a specification of a set of parts and the communication, composition, and layering relationships between them
- Structured classes:
  - Classes that contain a collaboration structure of interconnected parts and ports (specialized “interface” parts)
  - Parts can be instances of structured or unstructured classes
  - Based on modeling concepts found in popular architectural description languages (UML-RT, Acme…)

Modeling Collaboration Structures

- Based on the Collaboration concept
  - A structure of interconnected parts playing specialized roles

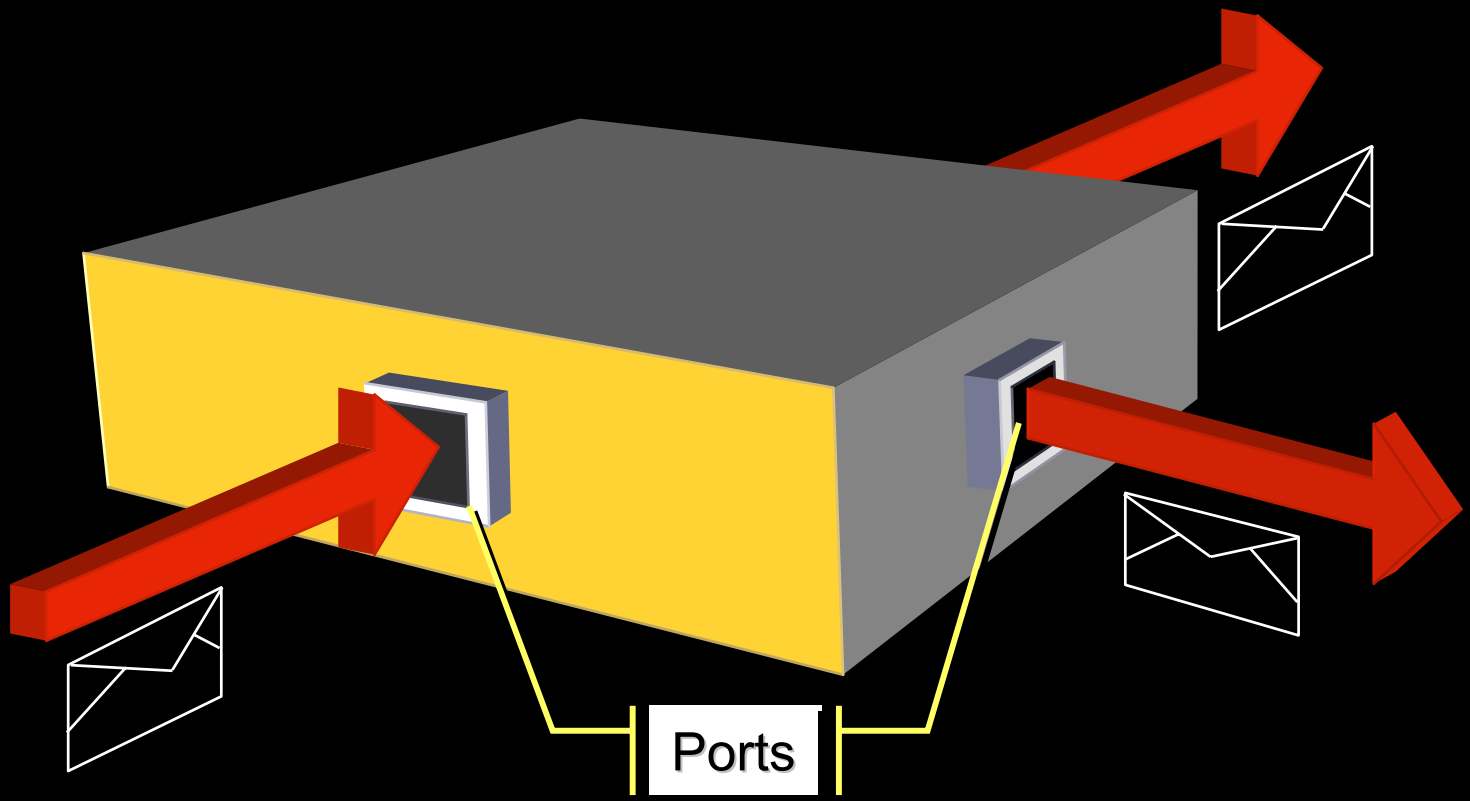


- Corresponding class diagram



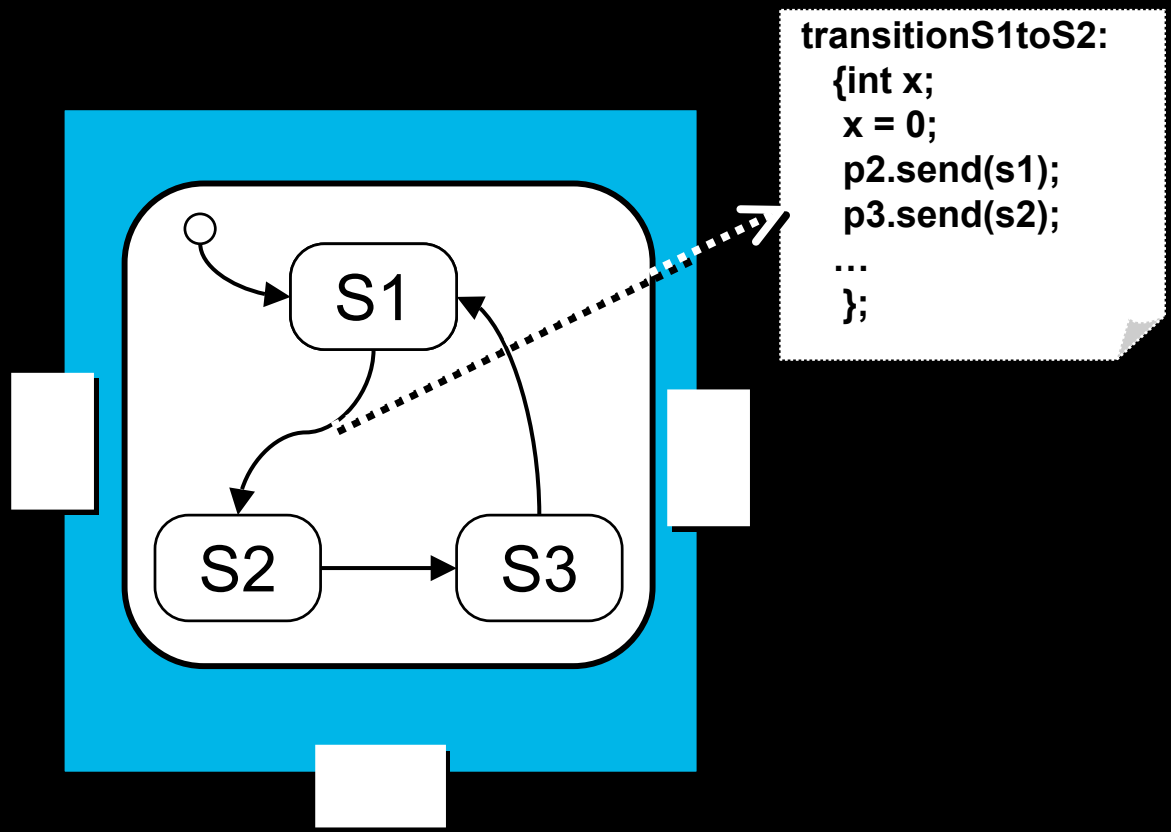
Structured Classes: External Structure

- Objects that may have multiple interaction points: *ports*
  - For accessing the functional capabilities of the object
  - Different ports may offer different capabilities



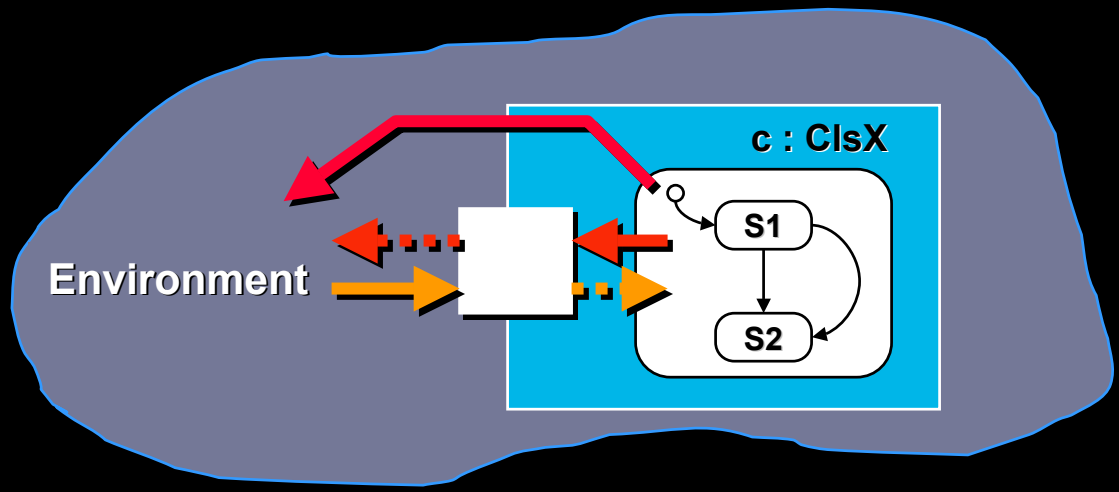
Structured Classes: Internal Behavior

- Events may occur on any one of the ports
  - Events are handled by the implementation (e.g., state machine)



Ports

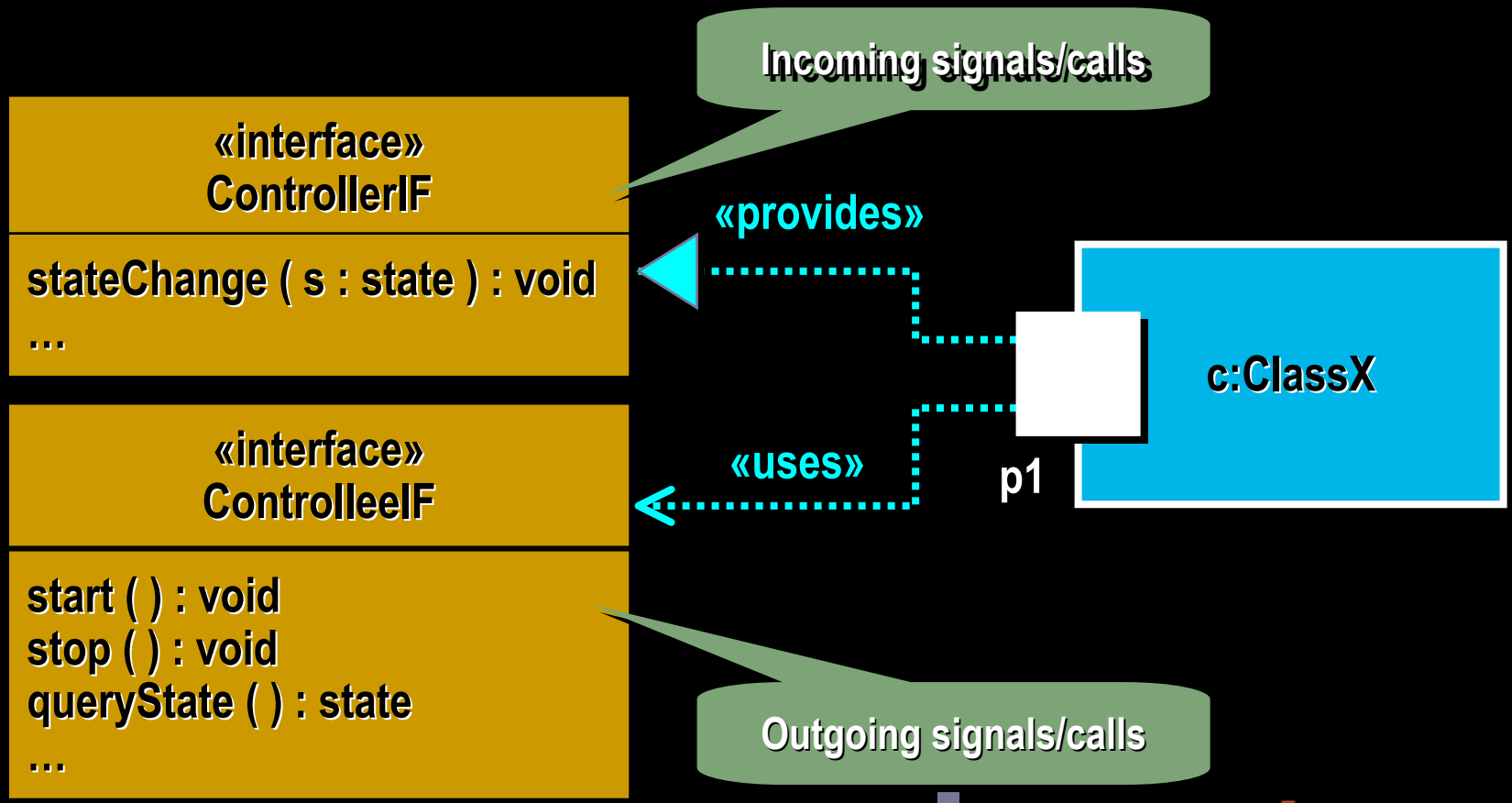
- Serve to fully isolate a structured object's implementation from its environment *(in both directions!)*



"There are very few problems in computer science that cannot be solved by adding an extra level of indirection"

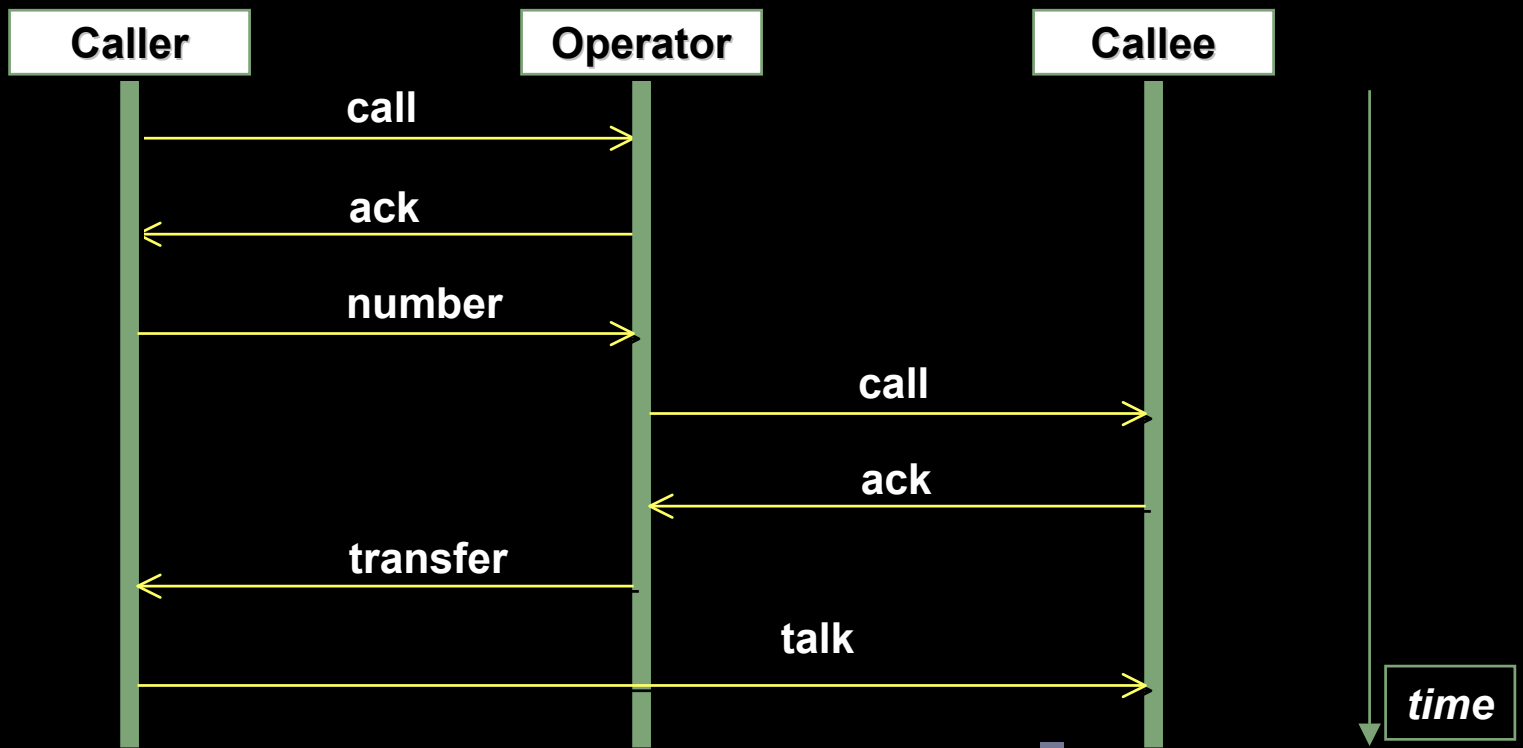
Port Semantics

- A port can support multiple interface specifications
  - Provided interfaces (what the object can do)
  - Required interfaces (what the object needs to do its job)



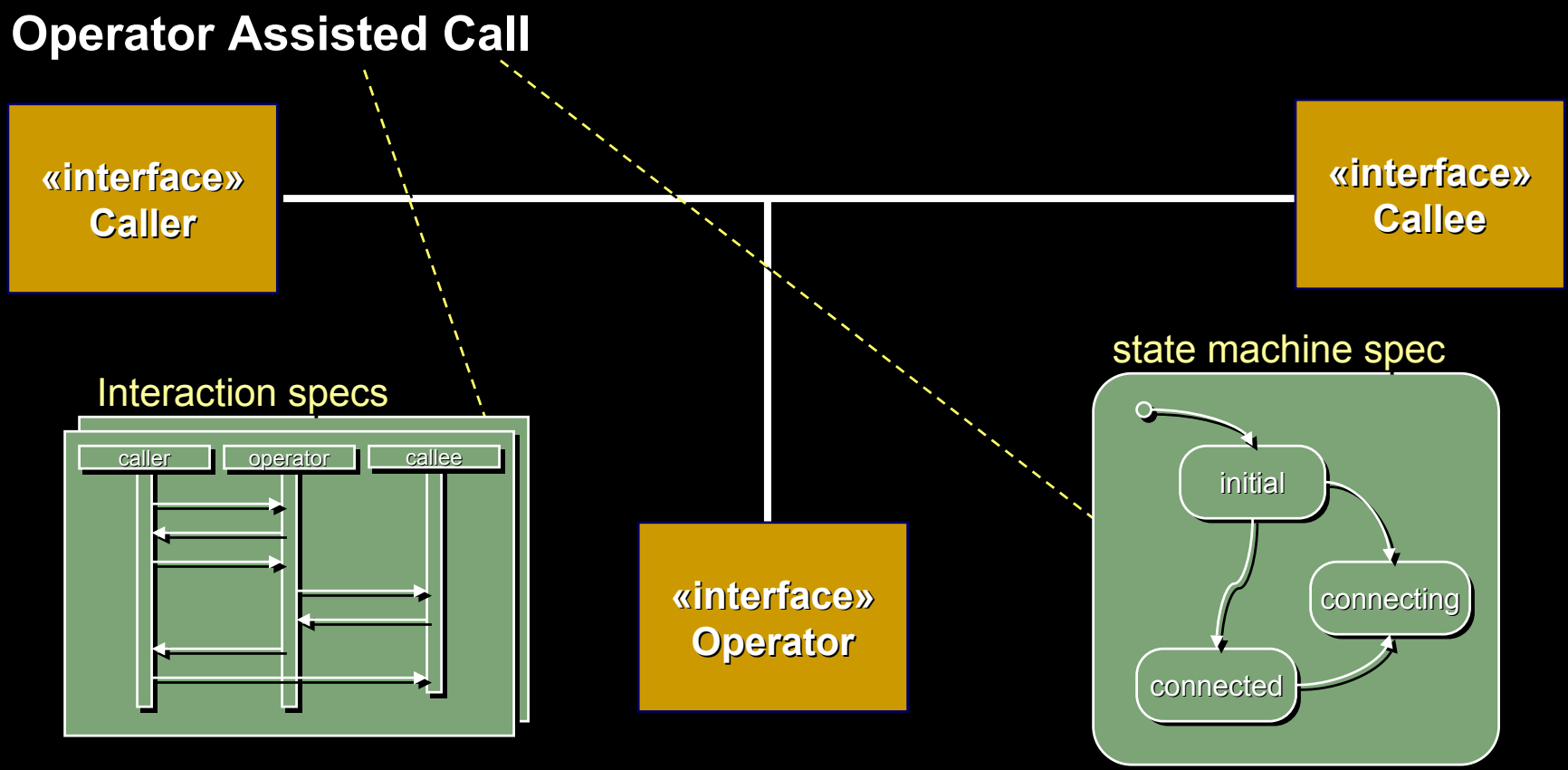
Dynamics of Interface Usage (Protocols)

- Interface specifications define what objects can do
  - For greater architectural control, it is also necessary to define (constrain) the order in which things are done
  - e.g., operator-assisted call



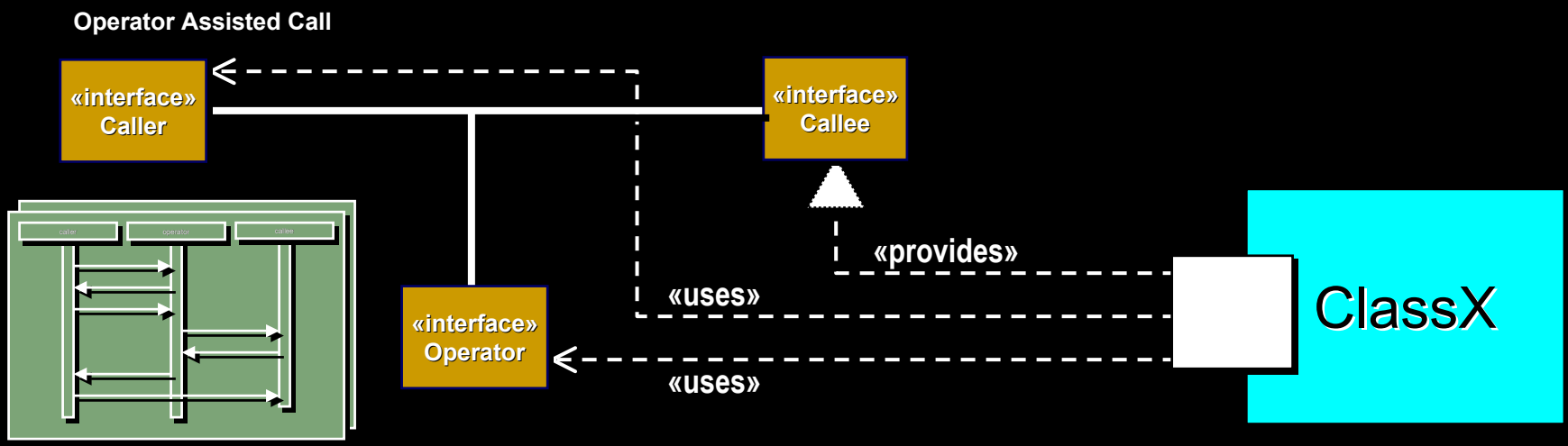
Specifying Protocols with UML 2.0

- A collaboration involving a set of interfaces and set of related behavior specifications (e.g., interactions)



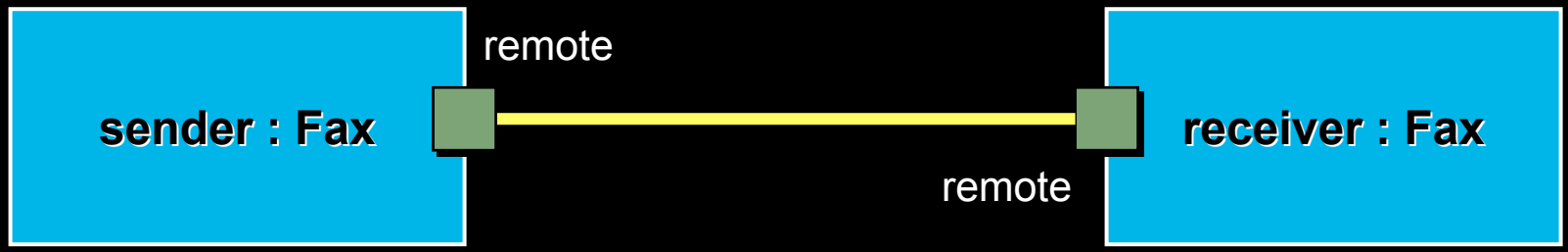
Ports and Protocols

- Ports can be assigned specific roles in protocols
  - By supporting interfaces involved in protocol specifications
  - The type of a port is determined by the corresponding protocol collaboration



Connecting Ports

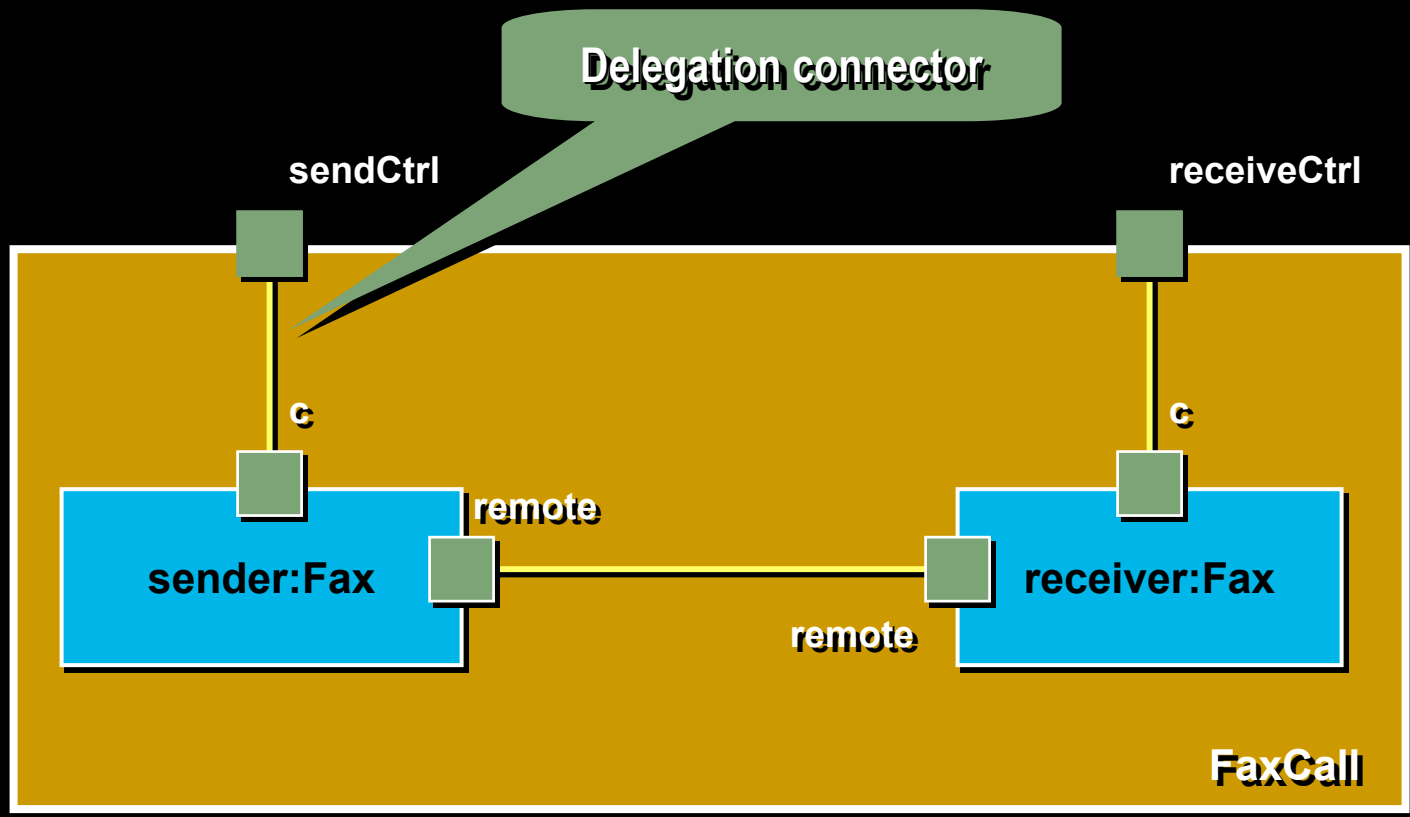
- Ports can be joined by connectors to create peer collaborations composed of structured classes



Connectors model communication channels
A connector is constrained by a protocol
Static typing rules apply (compatible protocols)

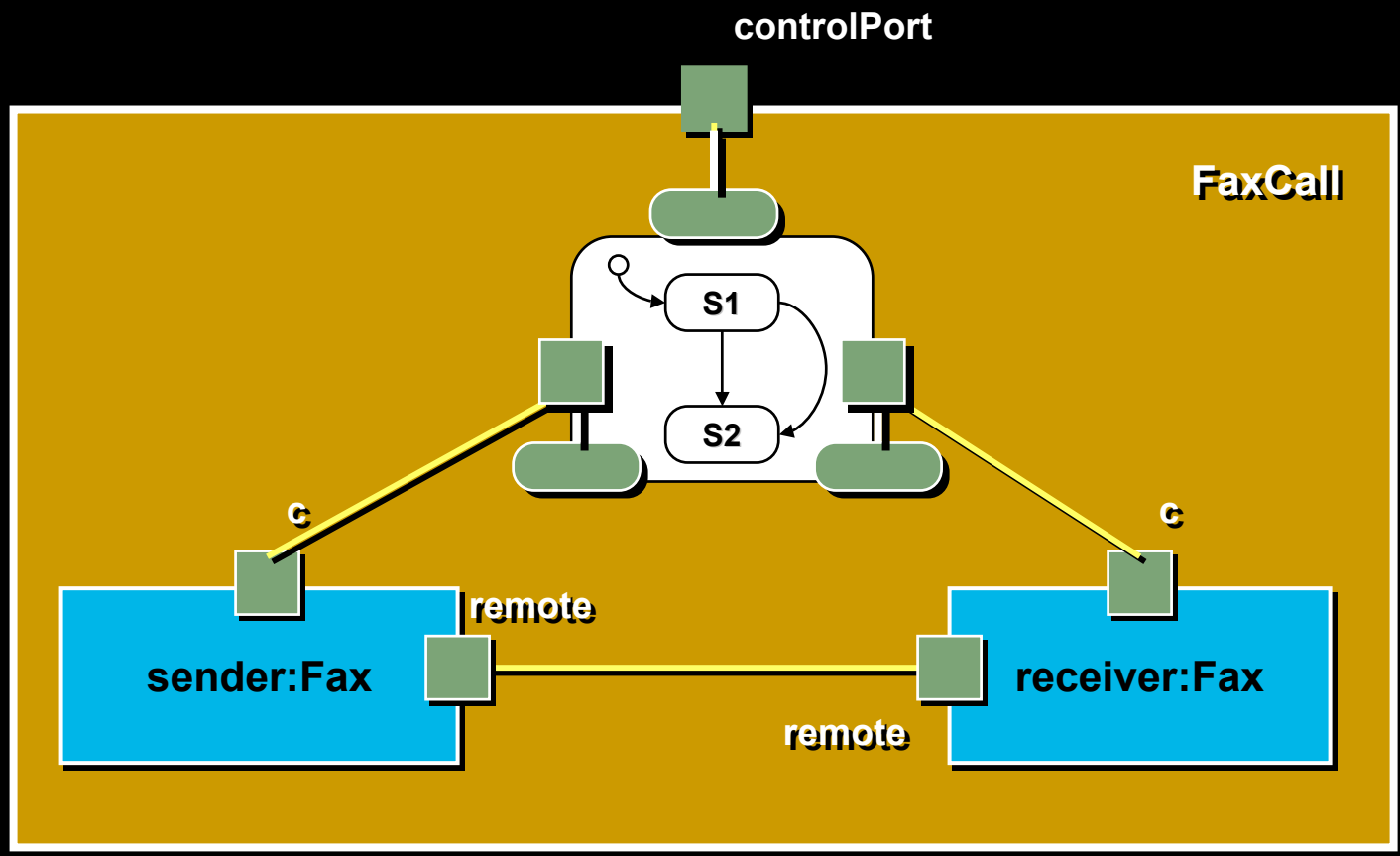
Structured Classes: Internal Structure

- Structured classes may have an internal structure of (structured class) parts and connectors



Behavior Ports

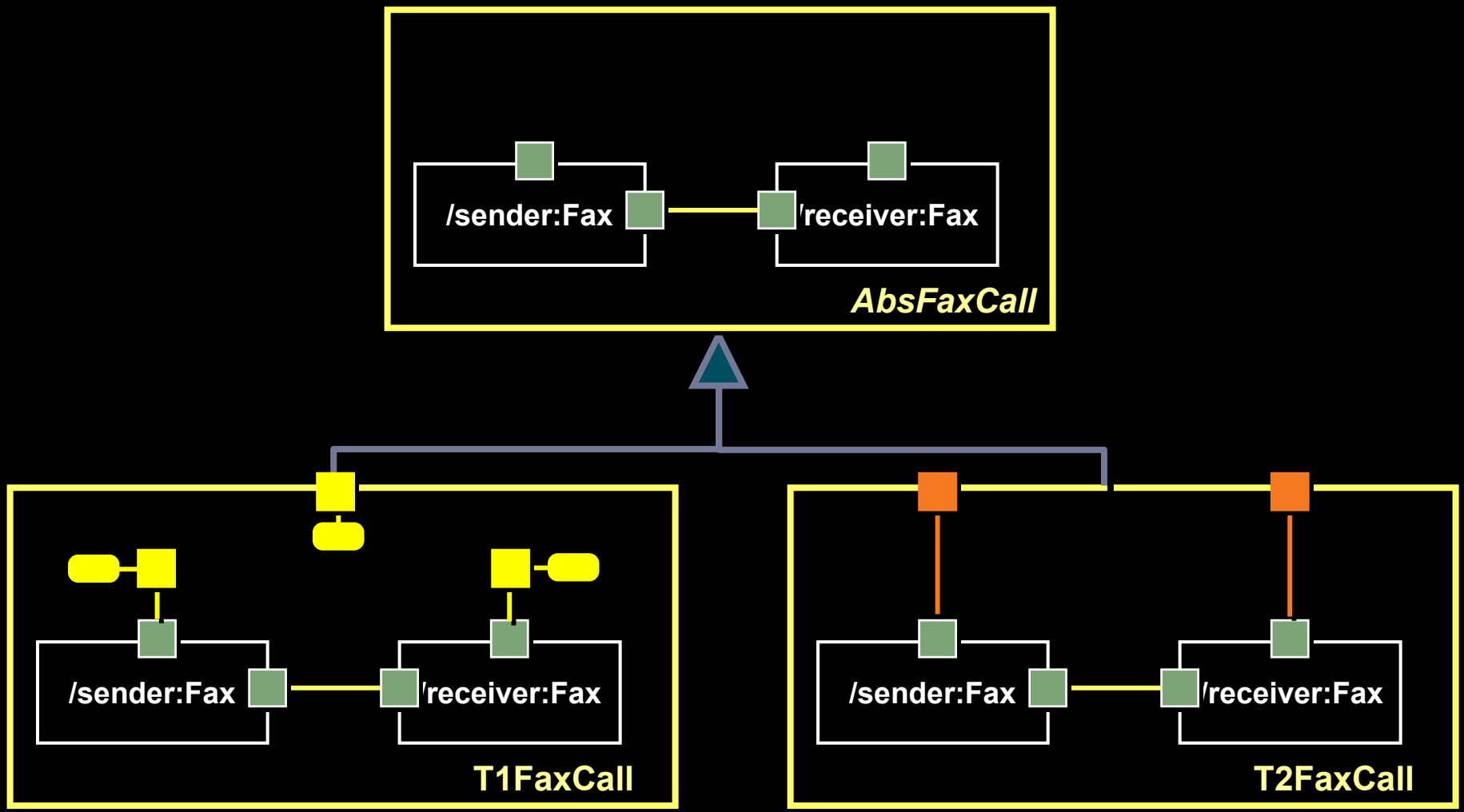
- Ports that are connected directly to a behavior
  - Require a special notation



- Internal behavior ports can be used to model layering

Inheritance of Structure

- Structured classes allow the inheritance of complex structures



Summary

- The “next generation” UML represents a significant evolutionary step:
  - Balance of consolidation and feature extensions
  - Modularized (core + optional specialized sub-languages)
  - Increased semantic precision and conceptual clarity
  - Supports full diagram interchange
  - Full alignment with MOF
  - Suitable MDA foundation (executable models, full code generation)
- New modeling features:
  - Large-scale system support (architecture-level structure, complex behavior and interaction modeling)
  - Extended business process modeling
- Expected availability: 2003