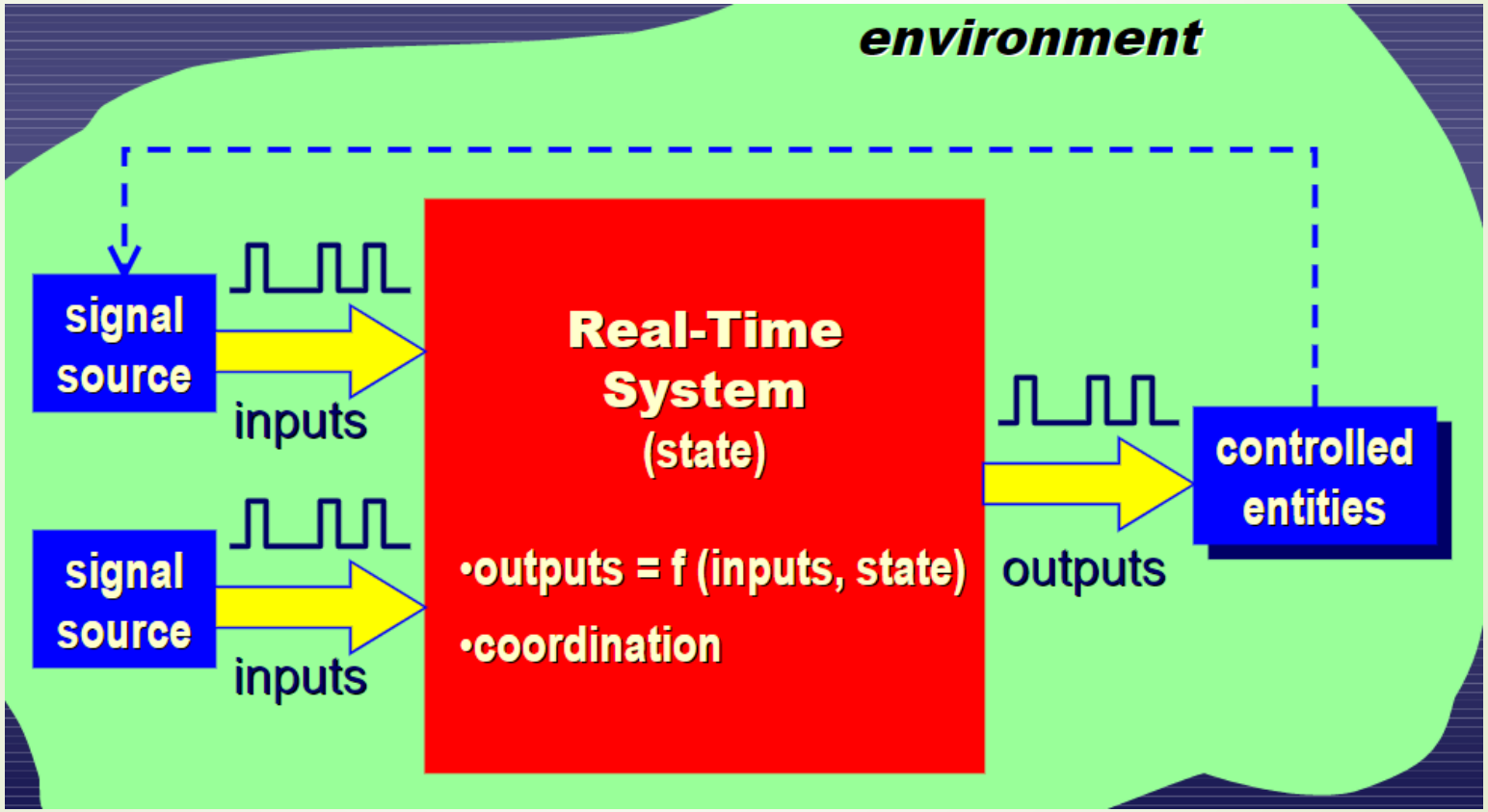
Real-Time System Design in UML

Real Time System ?

- System that maintain an ongoing timely interaction with its environment?



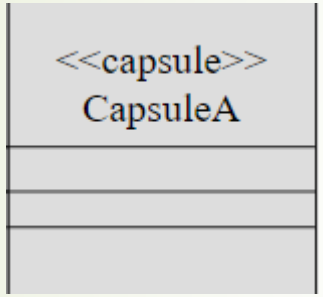
- The **Unified Modeling Language** (UML) is a general-purpose, developmental modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

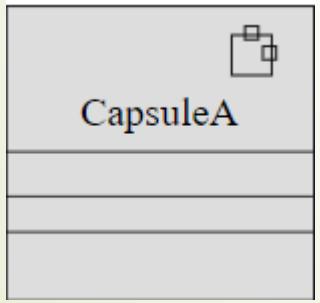
- Logical view diagrams
  - class diagram
  - collaboration deployment
  - sequence charts
  - state diagrams
- Logical view modeling elements
  - classes
  - relationships
  - packages

Modelling RTS in UML

- A system will be modelled as multiple communicating active objects (capsules)
- System behavior will be modeled through the state machines of the capsules
- Capsules can only communicate by sending messages through their ports
- Communication between ports is defined by a protocol
- Messages control(trigger) the transitions in the receiving capsule's state machine

Capsule

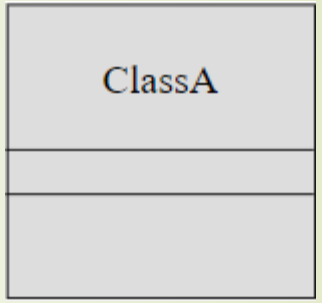




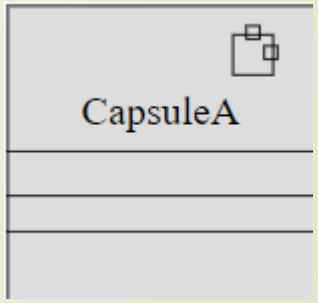
- Based upon a common pattern found in RTS: active object
  - Ultra light weight concurrency
  - Run to completion
  - Executable
- Fundamental modeling element
- A stereotype of a UML class
  - Has attributes
  - Has operations
  - Has a state machine
  - Has ports

Capsule vs class (1)

- Class
  - Contain public, protected, or private attributes and operations

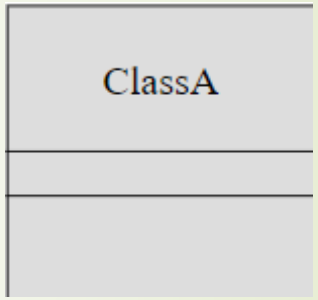


- Capsule
  - Contain only protected or private operations and attributes

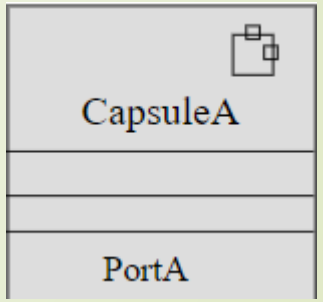


Capsule vs Class (2)

- Class
  - Communicate by calling operations on other classes

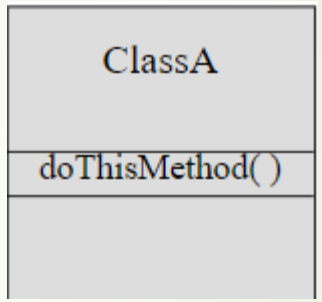


- Capsule
  - Communicate by sending messages (signals) through contained ports

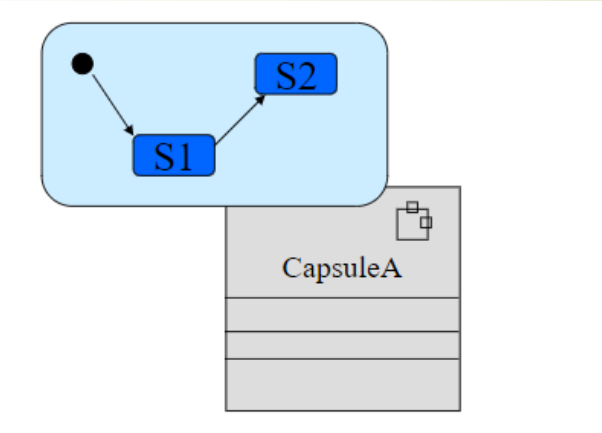


Capsule vs Class (3)

- Class
  - Elemental behavior is specified by operations

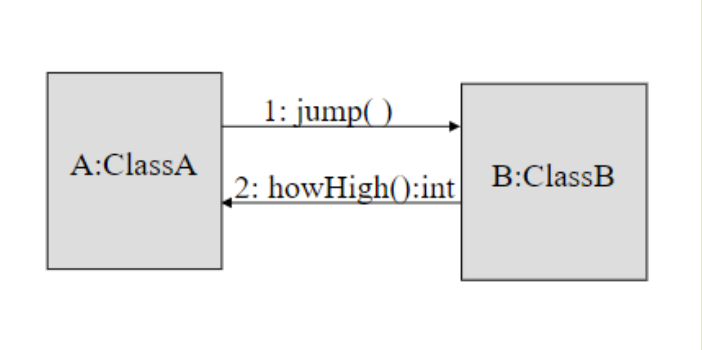


- Capsule
  - Elemental behavior is specified by a capsule's state machine

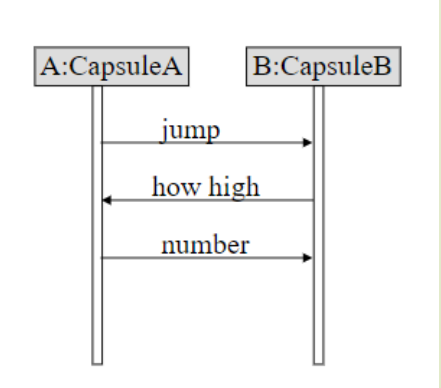


Capsule vs Class (4)

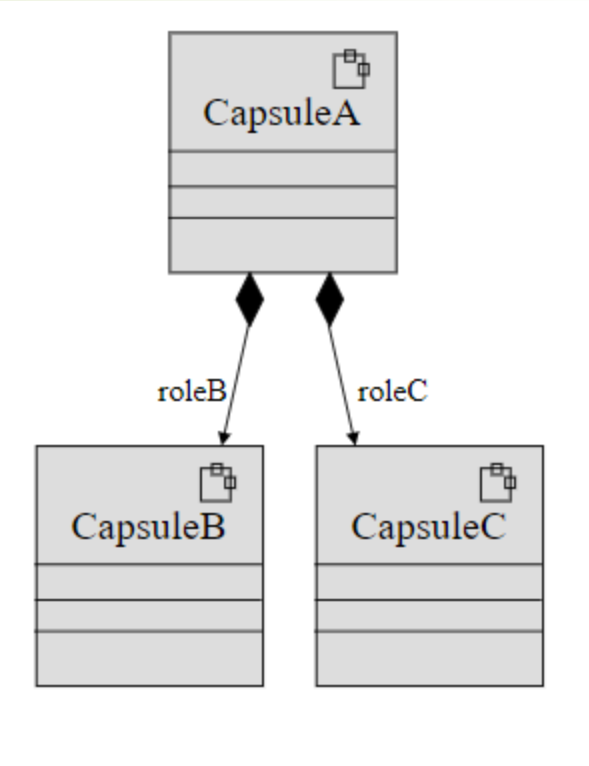
- Class
  - system behavior is expressed as a group of collaborating objects



- Capsule
  - system behavior is expressed as a sequence of inter-capsule messages



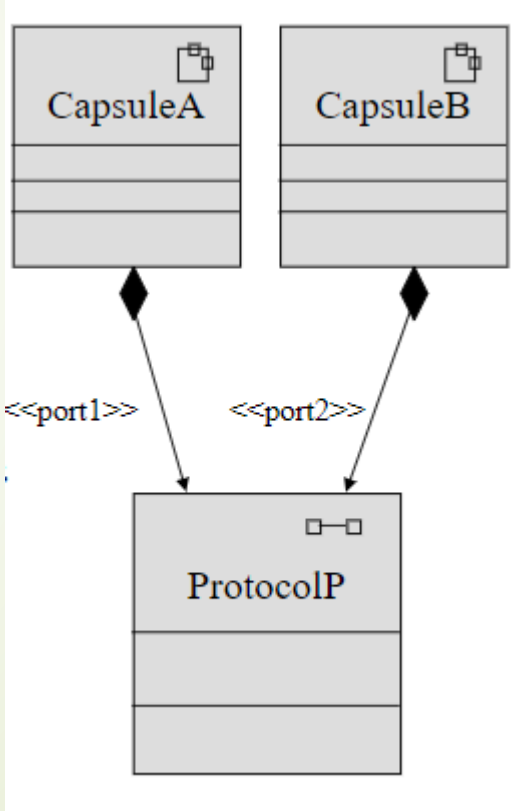
Capsule Role



- An instance of a capsule class
  - Changes to the role only affect the role not the class
  - Has cardinality
- Strongly owned by the containing capsule
  - Composition – fixed role
  - Aggregation – optional or plug-in

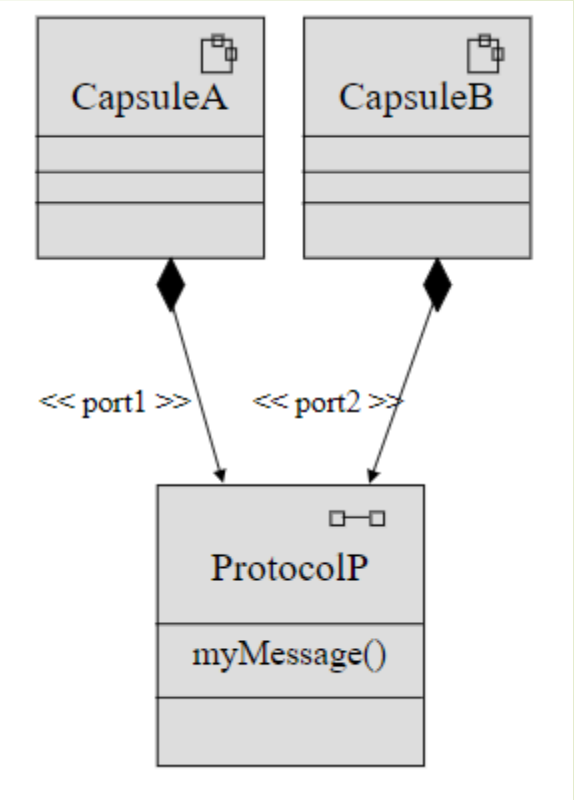
PORT (1)

- Isolates a capsule's implementation
- The means by which capsules communicate
  - Send and receive messages
- Are owned by the capsule instance
  - Created and receive messages



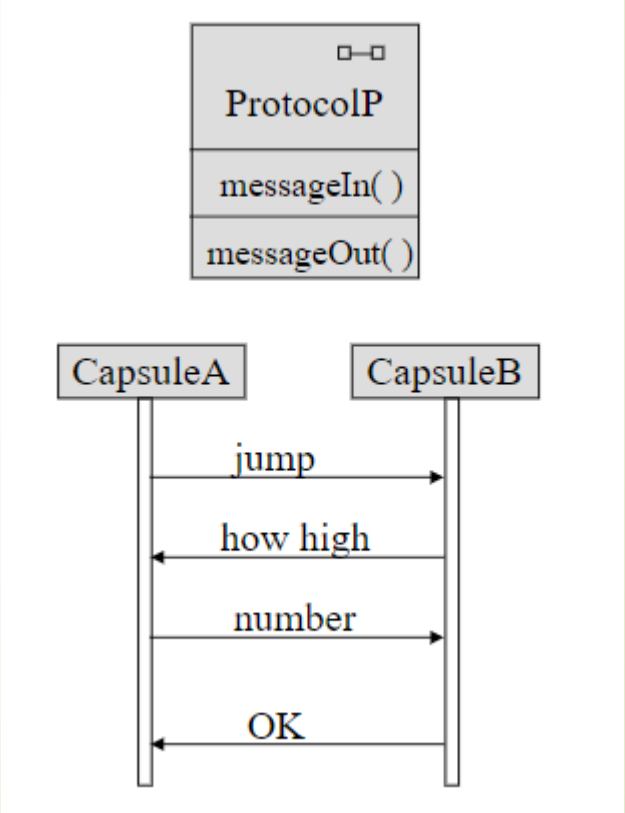
Port 2

- Defines a capsule interface
- Is a protocol role
  - An instance of a protocol
- Only compatible ports may communicate
- All ports have a “send” operation
  - port1.myMessage().send()



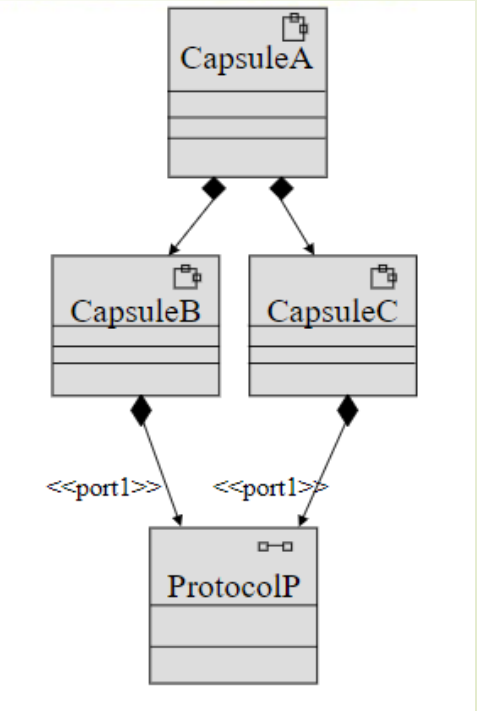
Protocol (1)

- Contract between capsules
  - A specification of a set of messages received (in) and sent (out) from the port
- Defines the port type
  - recall port compatibility
- A stereotype of a UML collaboration



Protocol (2)

- Each capsule role typically has an associated protocol for every other capsule role with which it associates
- Defines the services one capsule role provides another
  - A set of signals (and associated data) required to perform the capsule role's job



Capsule Structure Diagrams

- Visually defines the structure of a capsule
- Stereotype of a UML collaboration diagram
- Protocols not visible (only their instances – ports)
- Provides the support necessary to add “code generation”

Summary
Overview of a real-time UML modelling

- A system will be modelled as multiple communicating capsules
- System behavior will be modeled through the state machines of the capsules
- Capsules can only communicate by sending messages through their ports
- Communication between ports is defined by a protocol
- Message control (trigger) the transitions in receiving capsule's state machine.