EEE499 – Model-driven Development of Real-Time Systems

UML-RT and Papyrus-RT: Structural Modeling

ROYAL MILITARY COLLEGE OF CANADA
ELECTRICAL & COMPUTER
ENGINEERING

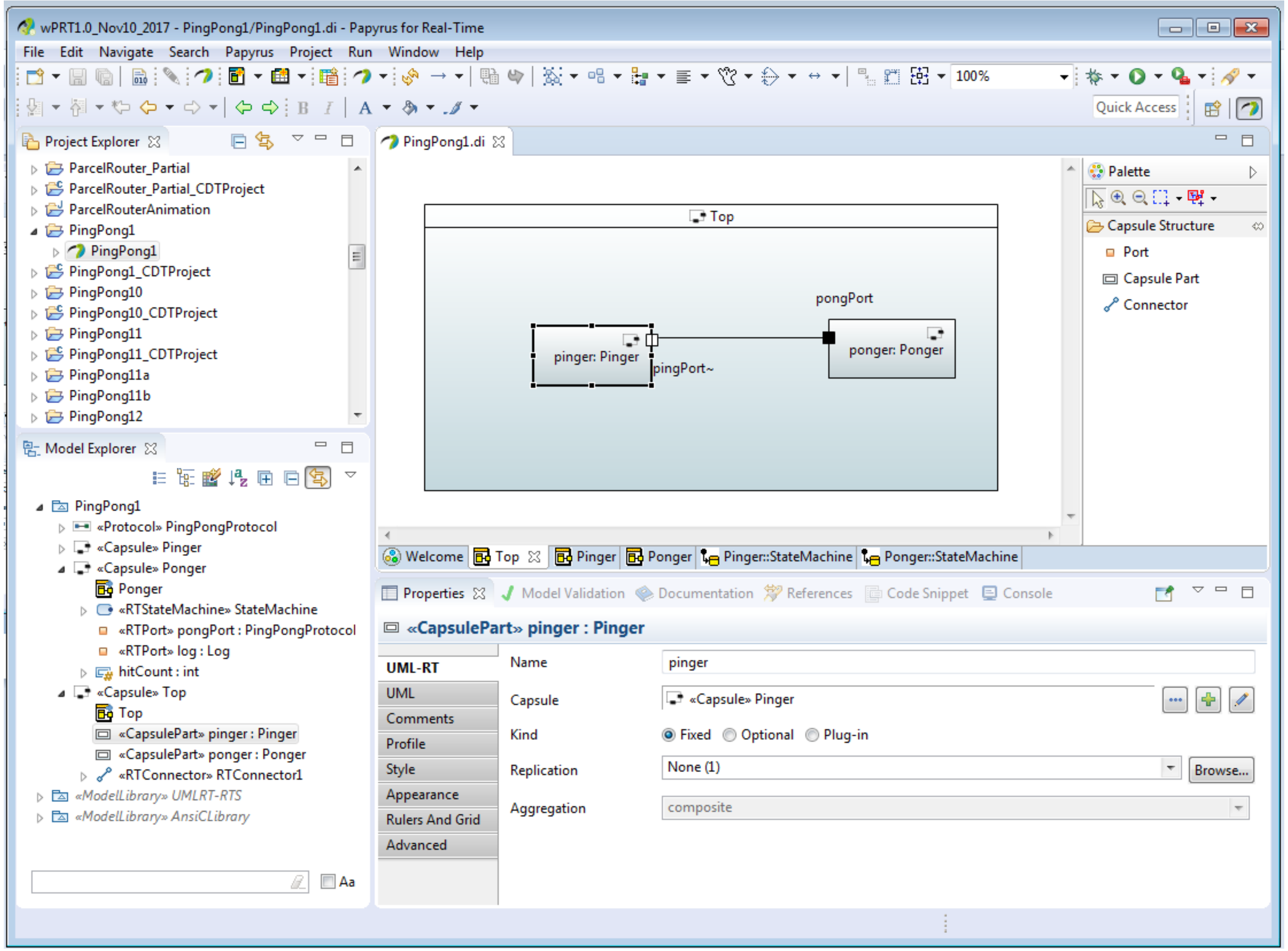
GÉNIE ÉLECTRIQUE
ET GÉNIE INFORMATIQUE
COLLÈGE MILITAIRE ROYAL DU CANADA

RMC CMR

Acknowledgement

The original material for this section was developed by Prof. Juergen Dingel (Queen’s University)

UML-RT and Papyrus-RT: Sneak Peek



«CapsulePart» pinger : Pinger

Name: pinger

Capsule: «Capsule» Pinger

Kind: ● Fixed ○ Optional ○ Plug-in

Replication: None (1)

Aggregation: composite

Modeling Languages

UML

AADL
- Embedded, real-time

UML MARTE
- Embedded, real-time

Modelica
- Physical systems
- Equation-based

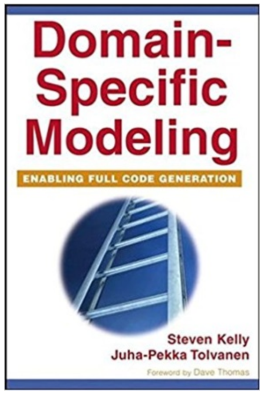
Simulink
- Continuous control, DSP
- time-triggered dataflow

Stateflow
- Reactive systems
- Discrete control
- State-machine-based

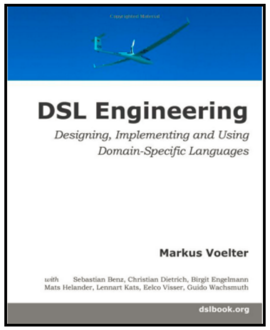
Lustre/SCADE
- Embedded real-time
- Synchronous dataflow

UML-RT
- Embedded, real-time
- State-machine-based

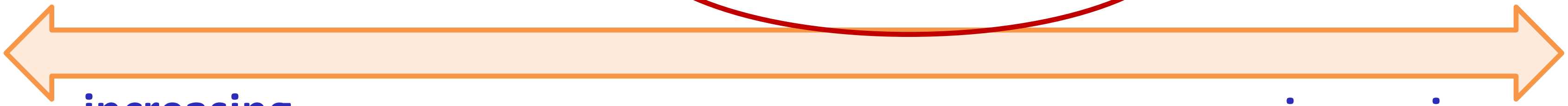
Examples in


Domain-Specific Modeling
ENABLING FULL CODE GENERATION
Steven Kelly
Juha-Pekka Tolvanen
Foreword by Dave Thomas

[Kelly, Tolvanen 2008]


DSL Engineering
Designing, Implementing and Using Domain-Specific Languages
Markus Voelter
with Sebastian Benz, Christian Dietrich, Birgit Engelmann Mats Helander, Lennart Kats, Eelco Visser, Guido Wachsmuth
dslbook.org

[Voelter 2013]

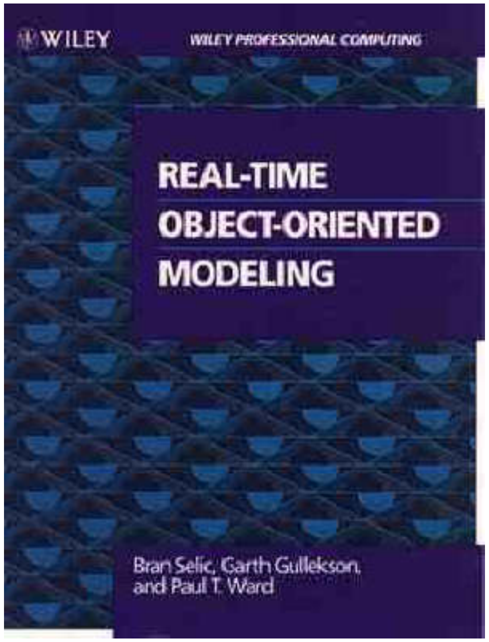


increasing generality

increasing domain-specifity

UML-RT: History

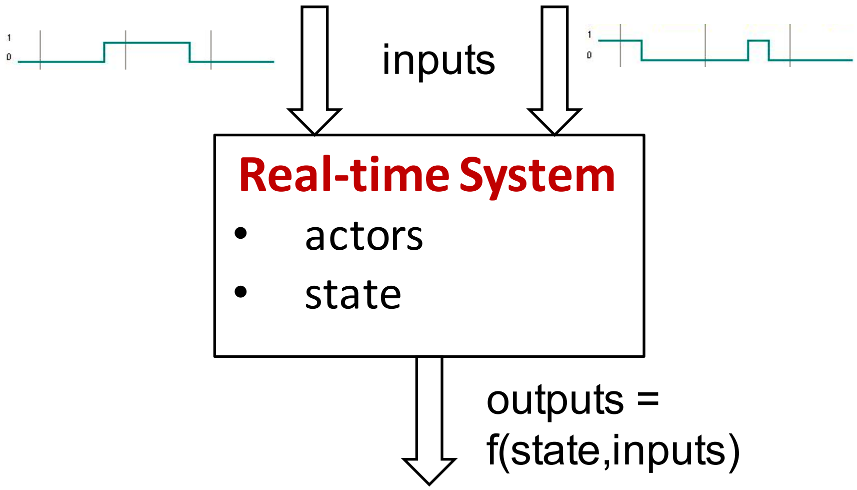
- Real-time OO Modeling (ROOM)
  - ObjecTime, early 1990 ties
- Major influence on UML 2
  - E.g., StructuredClassifier
- “RT subset of UML”
- Tools
  - ObjecTime Developer
  - IBM Rational RoseRT
  - IBM RSA-RTE
  - Eclipse Papyrus-RT


WILEY
WILEY PROFESSIONAL COMPUTING
REAL-TIME
OBJECT-ORIENTED
MODELING
Bran Selic, Garth Gullekson, and Paul T. Ward

[Selic, Gullekson, Ward. *Real-Time Object-Oriented Modellng*. Wiley. 1994]

UML-RT: Characteristics

- Domain-specific
  - Embedded systems with soft real-time constraints
- Graphical, but textual syntax exists
- Small, cohesive set of concepts
- Strong encapsulation
  - Actors (active objects)
  - Explicit interfaces
  - Message-based communication
- Event-driven execution
  - State machines

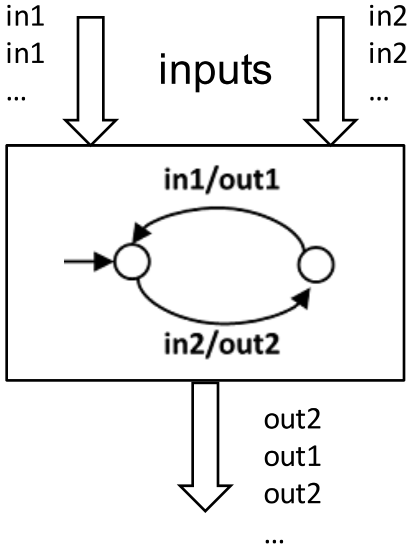


inputs

Real-time System

- actors
- state

outputs = f(state,inputs)



in1
in1
…

inputs

in2
in2
…

in1/out1

in2/out2

out2
out1
out2
…

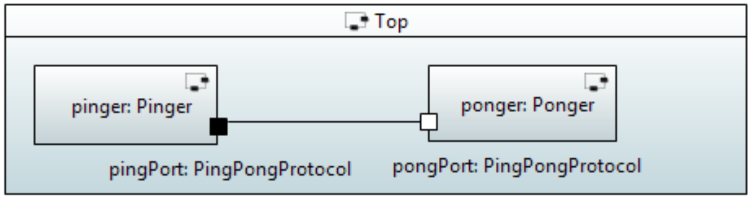
UML-RT: Core Concepts (1)

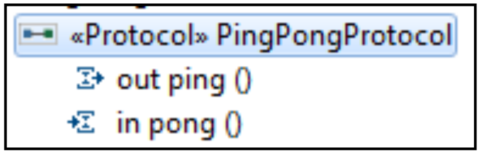
- Types
  - Capsules (active classes)
    - Capsule instances (parts)
  - Passive classes (data classes)
    - Objects
  - Protocols
  - Enumerations
- Structure
  - Attributes
  - Ports
  - Connectors

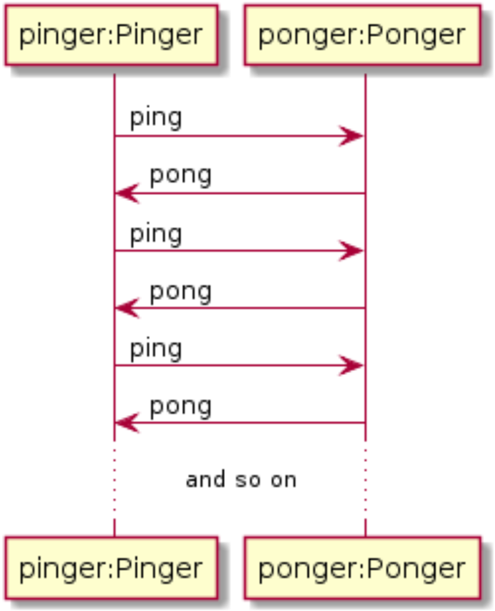
- Behaviour
  - Messages (events)
  - State machines
- Grouping
  - Package
- Relationship
  - Generalization
  - Associations

UML-RT: Core Concepts (2)

- Model
  - Collection of capsule definitions
  - ‘Top’ capsule containing collection of capsule instances (parts)
- Capsules
  - May contain
    - Attributes, ports, or other capsule instances (parts)
  - Behaviour defined by state machine
- Ports
  - Typed over protocol defining input and output messages
- State machine
  - Transition triggered by incoming messages
  - Action code can contain send statements that send messages over certain ports

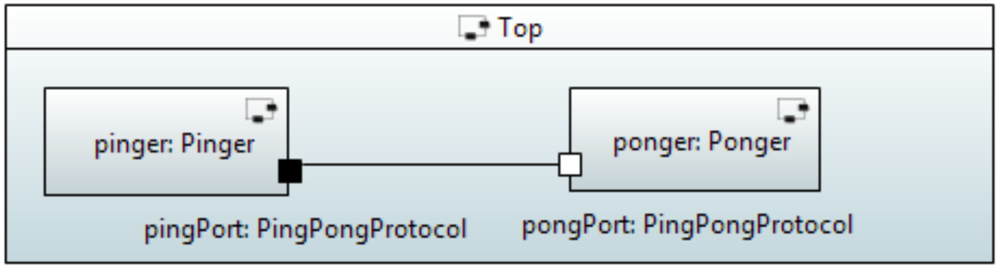



«Protocol» PingPongProtocol
out ping ()
in pong ()

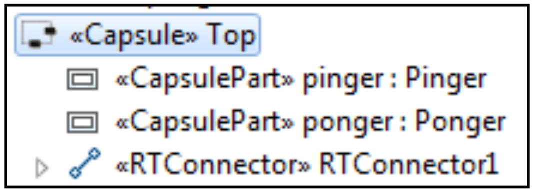


Capsules (1)

- Kind of active class
  - Attributes, operations
  - Own, independent flow of control (logical thread)



- May also contain
  - Ports over which messages can be sent and received
  - Parts (instances of other capsules) and connectors

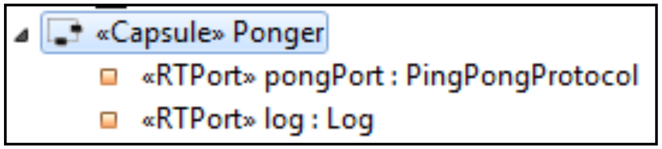


«Capsule» Top
- «CapsulePart» pinger : Pinger
- «CapsulePart» ponger : Ponger
- «RTConnector» RTConnector1

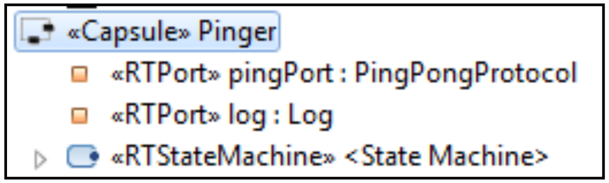
- Creation, use of instances tightly controlled
  - Created by runtime system (RTS)
  - Cannot be passed around
  - Stored in attribute of another capsule (part)
  - Information flow only via messages sent to ports

) better concurrency control and encapsulation

- Behaviour defined by state machine



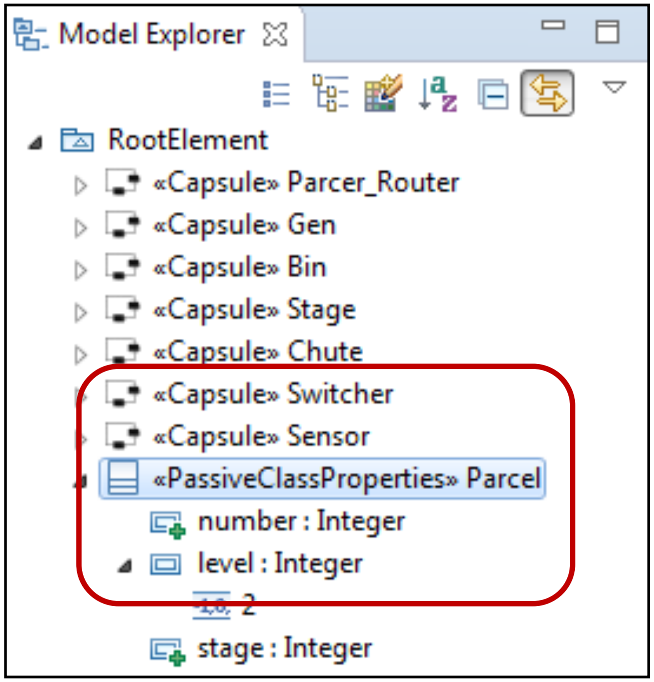
«Capsule» Ponger
- «RTPort» pongPort : PingPongProtocol
- «RTPort» log : Log



«Capsule» Pinger
- «RTPort» pingPort : PingPongProtocol
- «RTPort» log : Log
- «RTStateMachine» <State Machine>

Passive Classes/Data Classes

- Similar to regular classes
- Do not have independent flow of control
- Behaviour defined through operations
- Used to define data structures and operations on them

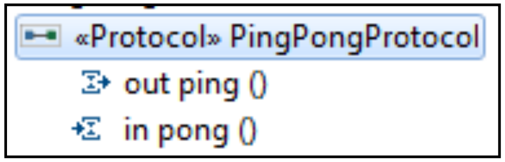


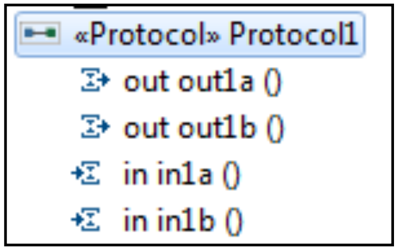
Model Explorer

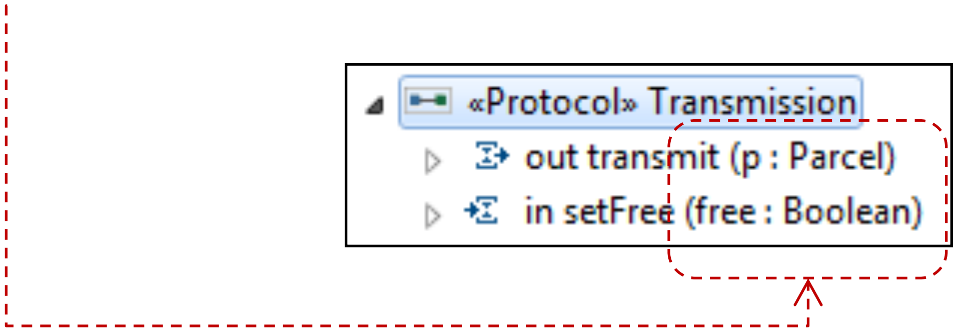
RootElement
«Capsule» Parcer_Router
«Capsule» Gen
«Capsule» Bin
«Capsule» Stage
«Capsule» Chute
«Capsule» Switcher
«Capsule» Sensor
«PassiveClassProperties» Parcel
number : Integer
level : Integer
2
stage : Integer

Protocols

- Provide types for ports
- Define
  - – Input messages
    - Services provided by capsule owning port
  - – Output messages
    - Services required by capsule owning port
  - – Input/output messages
- Messages can carry data


«Protocol» PingPongProtocol
out ping ()
in pong ()


«Protocol» Protocol1
out out1a ()
out out1b ()
in in1a ()
in in1b ()

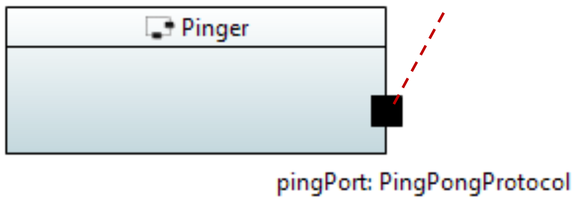

«Protocol» Transmission
out transmit (p : Parcel)
in setFree (free : Boolean)

Ports

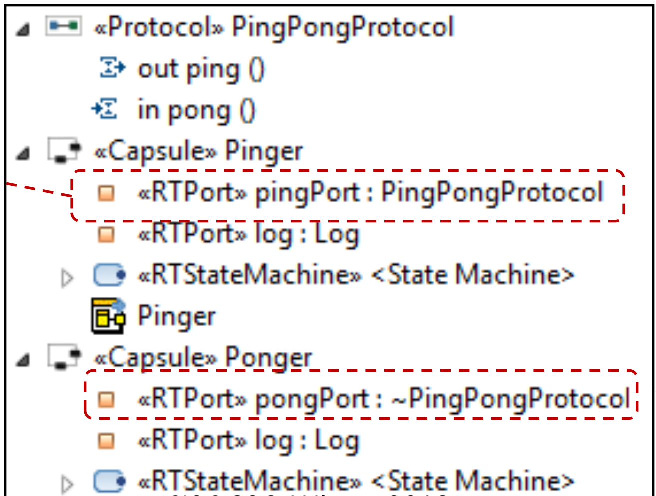
- “Boundary objects” owned by capsule
- Typed over a protocol P
- Have ‘**send**’ operation

```
portName.msg(arg1,...,argn).send()
```

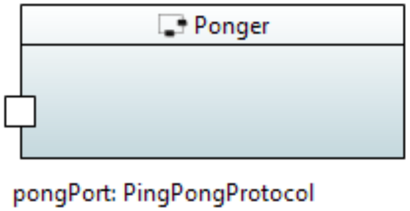
- Can be
  - base (not conjugated)
    - Direction of messages is declared in protocol
    - Notation:
      - textual: P
      - graphical: ¥
  - conjugated
    - Direction of messages declared in protocol is reversed
    - Notation
      - textual: ~P
      - graphical: ¤



base



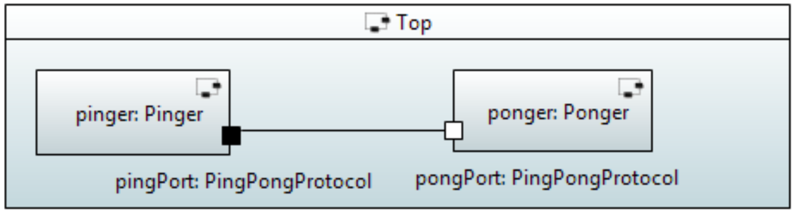
- «Protocol» PingPongProtocol
  - out ping ()
  - in pong ()
- «Capsule» Pinger
  - «RTPort» pingPort : PingPongProtocol
  - «RTPort» log : Log
  - «RTStateMachine» <State Machine>
  - Pinger
- «Capsule» Ponger
  - «RTPort» pongPort : ~PingPongProtocol
  - «RTPort» log : Log
  - «RTStateMachine» <State Machine>
  - Ponger

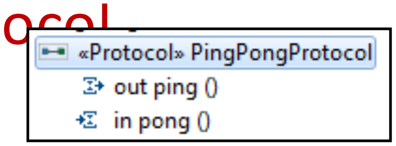


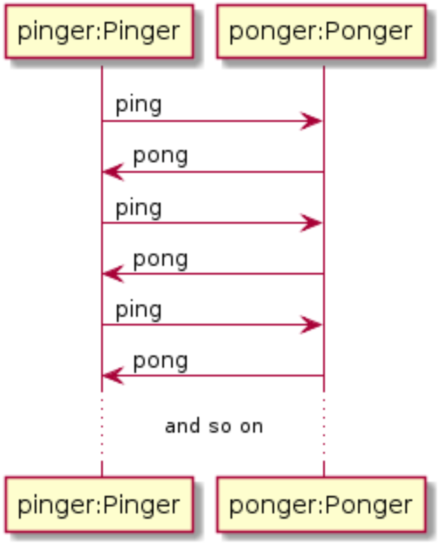
conjugated

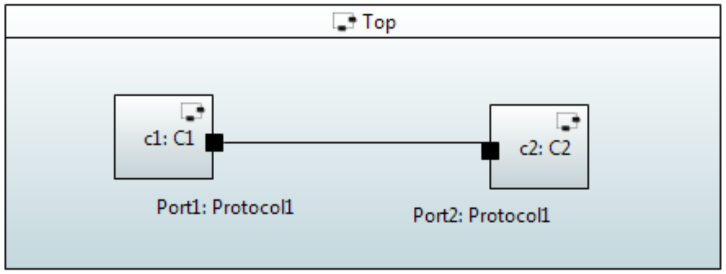
Connectors

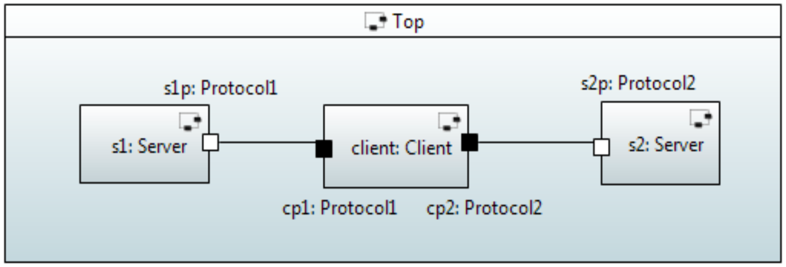
- Connect two ports
- Ports must be compatible
  - Both are instances of same protocol
  - Either (asymmetric)
    - one is 'base' (i.e., not 'conjugated')
      - typically owned by 'client'
    - and the other is 'conjugated'
      - typically owned by 'server'
  - Or (symmetric)
    - only InOut messages




«Protocol» PingPongProtocol
out ping ()
in pong ()

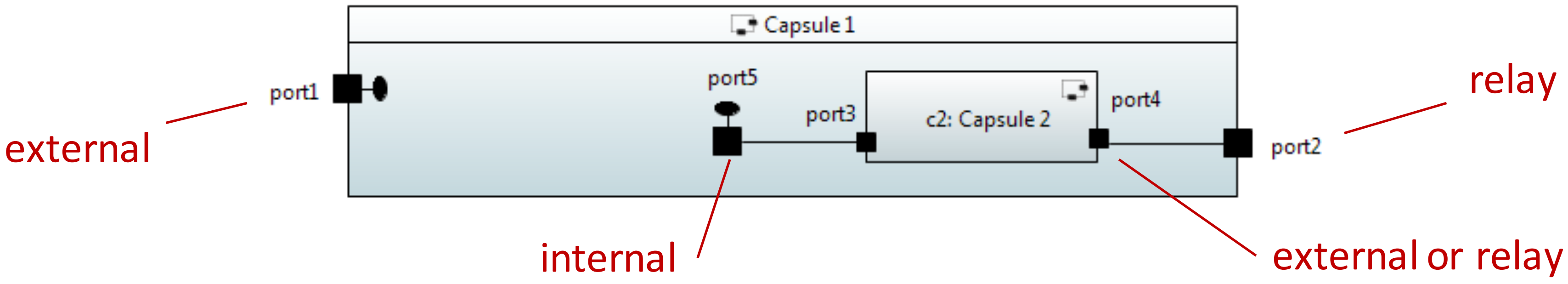




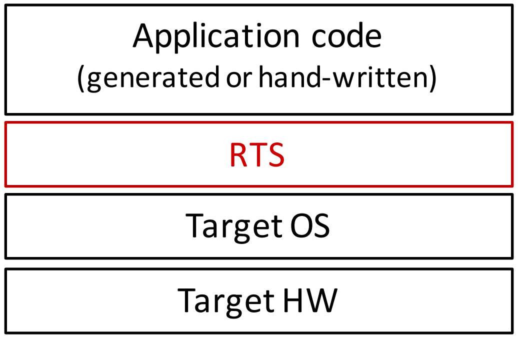


Ports: External, Internal, Relay

- External behaviour
  - Provides (part of) externally visible functionality (isService=true)
  - Incoming messages passed on to state machine (isBehaviour=true)
  - Must be connected (isWired=true)
- Internal behaviour
  - As above, but not externally visible (isService=false)
  - Connect state machine with a capsule part
- Relay
  - Pass external messages to and from capsule parts



Ports: System



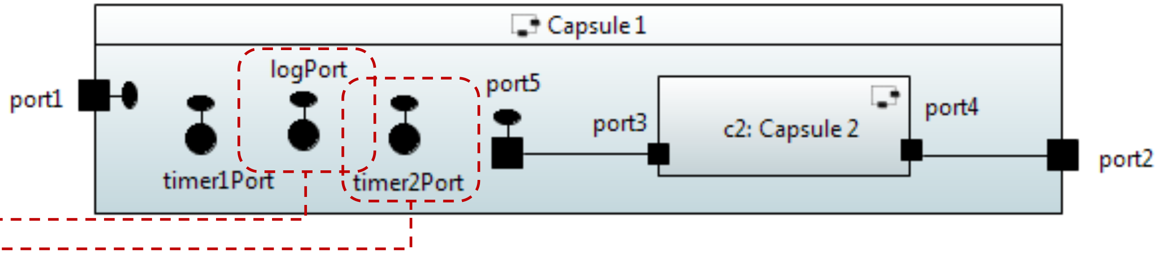
Application code
(generated or hand-written)

RTS

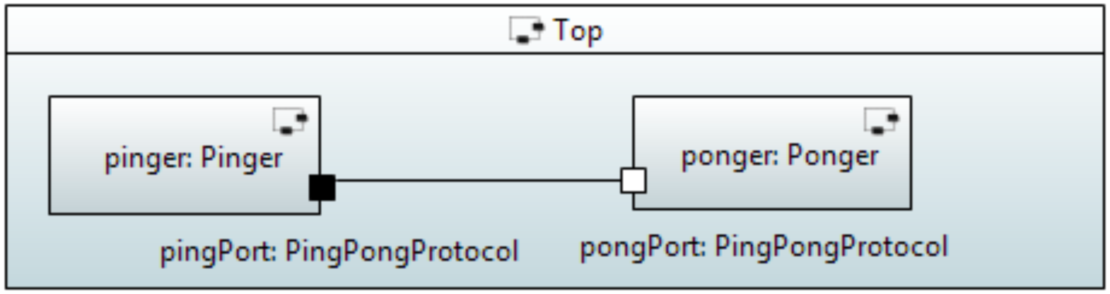
Target OS

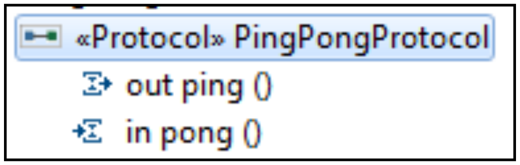
Target HW

- Connects capsule to Runtime System (RTS) library via corresponding system protocol
- Provides access to RTS services such as
  - Timing: setting timers, time out message
    - `timer2Port.informIn(UMLRTTimespec(10, 0));`
      `// set timer that will expire in 10 secs and 0 nanosecs`
    - When timer expires, '`timeout`' message will be sent over `timer2Port`
  - Log: sending text to console
    - `logPort.log("Ready to self-destruct")`
  - Frame: incarnate, destroy capsule instances



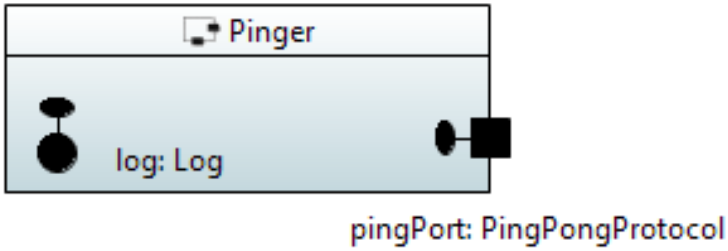
Example: PingPong

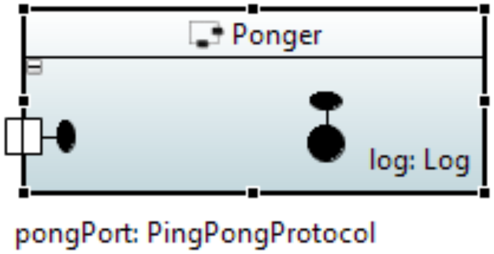



«Protocol» PingPongProtocol

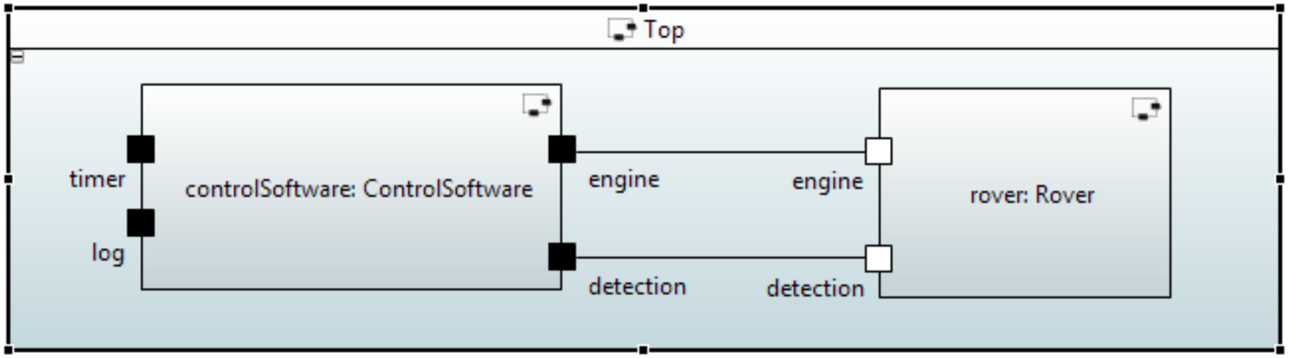
out ping ()

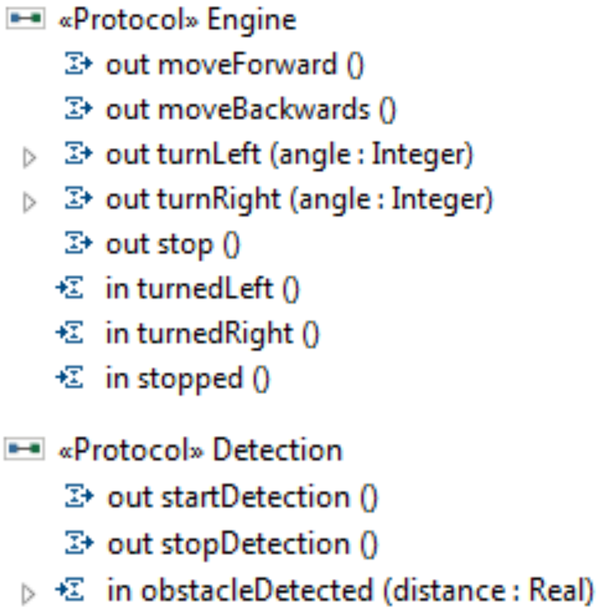
in pong ()



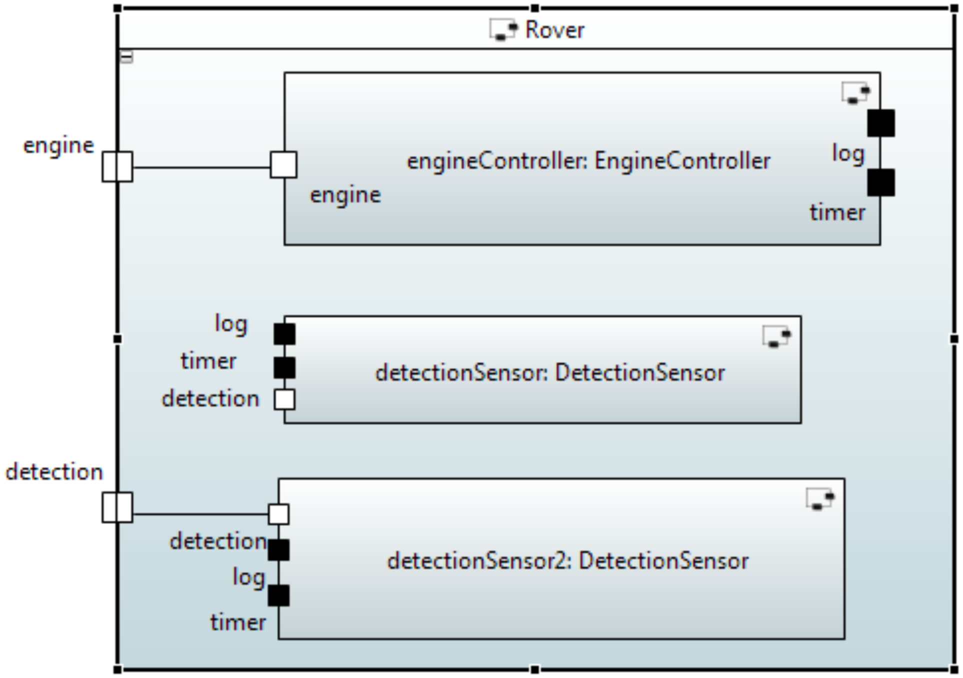


Example: Rover



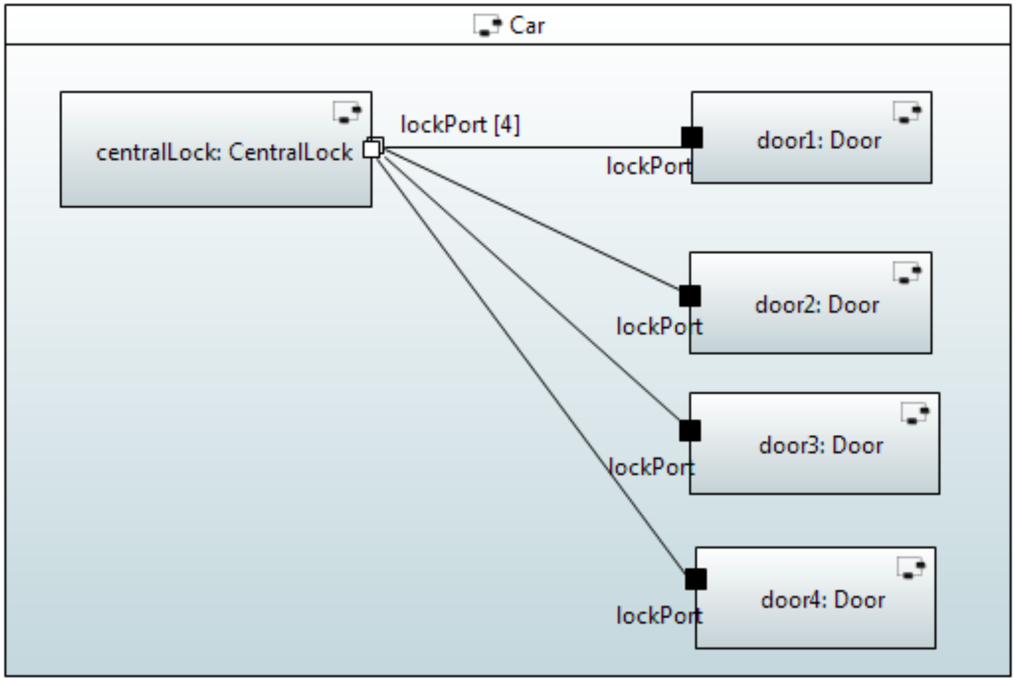


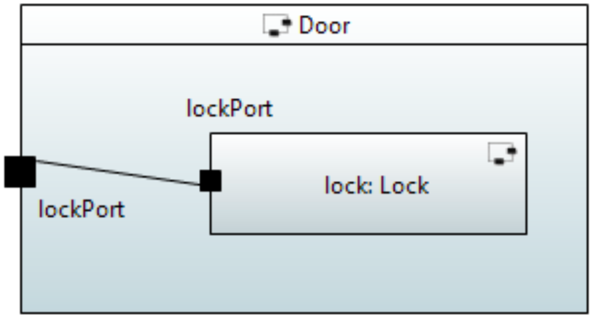
- «Protocol» Engine
  - out moveForward ()
  - out moveBackwards ()
  - ▷ out turnLeft (angle : Integer)
  - ▷ out turnRight (angle : Integer)
  - out stop ()
  - in turnedLeft ()
  - in turnedRight ()
  - in stopped ()
- «Protocol» Detection
  - out startDetection ()
  - out stopDetection ()
  - ▷ in obstacleDetected (distance : Real)

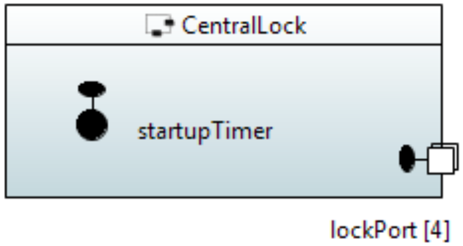

Rover
engine
engine
engineController: EngineController
log
timer
log
timer
detection
detectionSensor: DetectionSensor
detection
detection
log
timer
detectionSensor2: DetectionSensor

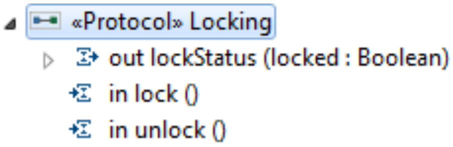
UML-RT

Example: Door Lock System

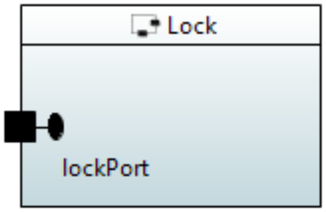


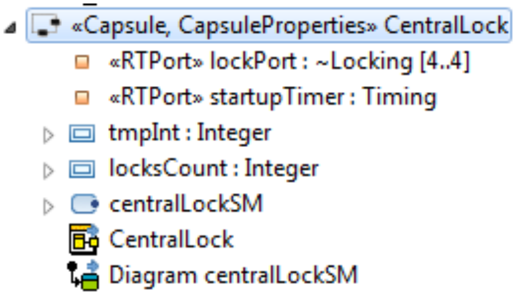






«Protocol» Locking
- out lockStatus (locked : Boolean)
- in lock ()
- in unlock ()





«Capsule, CapsuleProperties» CentralLock
- «RTPort» lockPort : ~Locking [4..4]
- «RTPort» startupTimer : Timing
- tmpInt : Integer
- locksCount : Integer
- centralLockSM
- CentralLock
- Diagram centralLockSM