EEE499 – Model-driven Development of Real-Time Systems

UML-RT and Papyrus-RT:
Basic Behavioural Modeling

ROYAL MILITARY COLLEGE OF CANADA
ELECTRICAL & COMPUTER ENGINEERING

GÉNIE ÉLECTRIQUE ET GÉNIE INFORMATIQUE
COLLÈGE MILITAIRE ROYAL DU CANADA

Acknowledgement

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

State Machines

States

- Capture relevant aspects of history of object
- Determine how object can respond to incoming messages
- May have invariants associated with them

Pseudo states

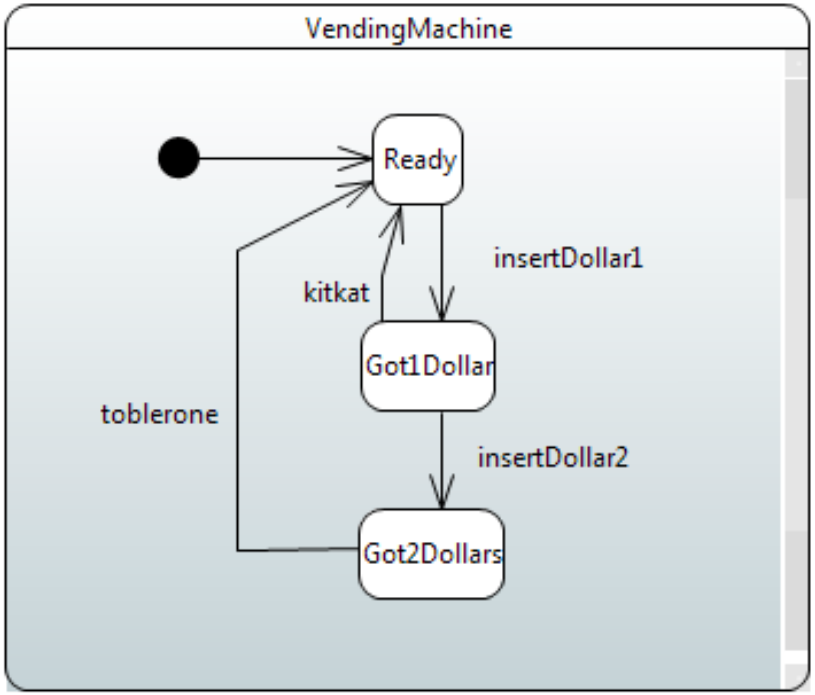
- Don't belong to description of lifetime of object

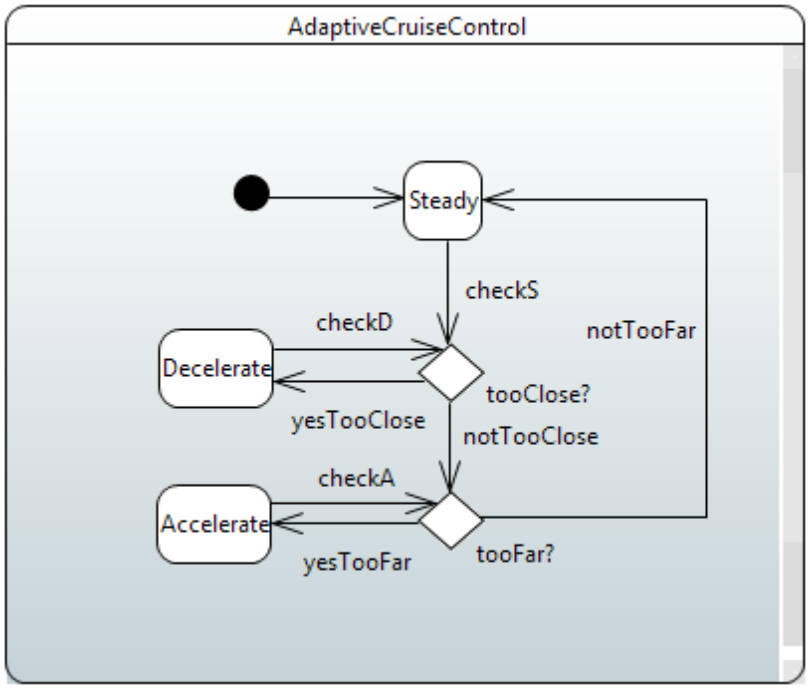
) object cannot be 'in' a pseudo state

- Helper constructs to define complex state changes

Transitions

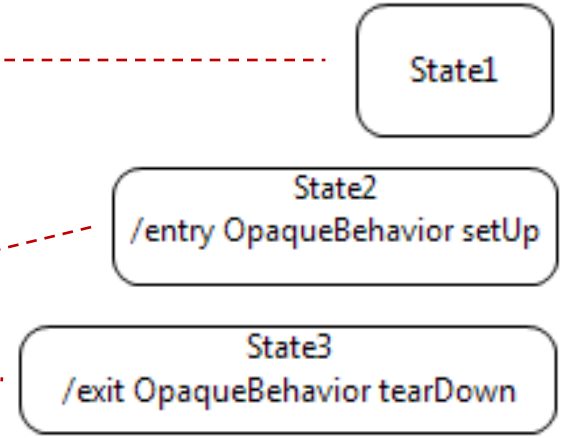
- Describe how object can move from one state to next in response to message input



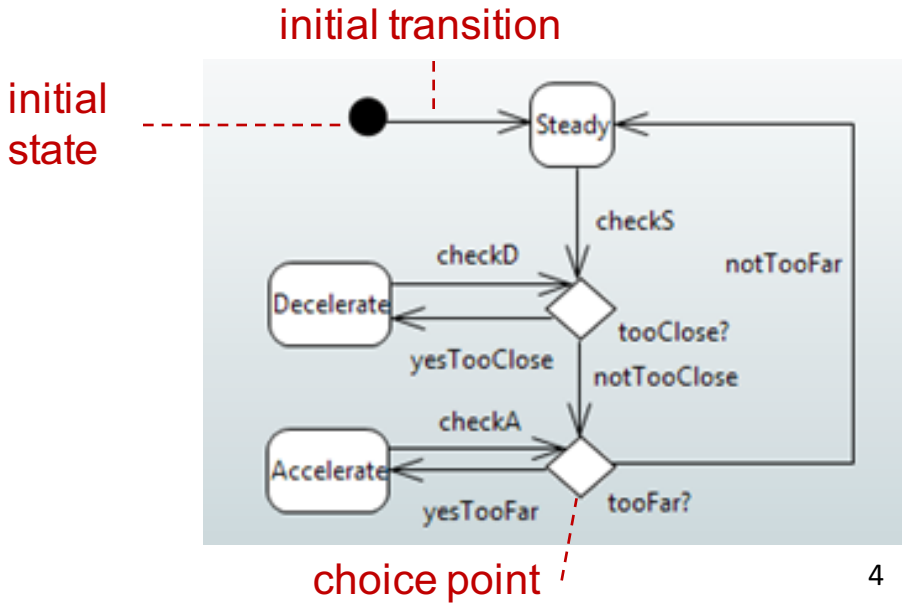


States I: Simple and Pseudo

- States
  - Kinds:
    - Simple
    - Later: composite (in hierarchical state machines)
  - May contain
    - Entry action (written in action language)
    - Exit action (written in action language)

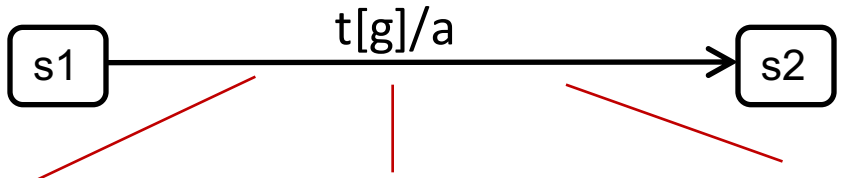


- Pseudo states I
  - initial
  - choice point



Transitions

- Kinds:
  - Basic
  - Later: group (in hierarchical state machines)
- Consists of
  - Triggers
    - Transitions out of pseudo states (initial, choice) don't have triggers
    - Transitions out of non-pseudo state should have at least one trigger
  - Guards (optional, written in action language)
    - Transitions out of initial state should not have guards
  - Effect/Actions (optional, written in action language)



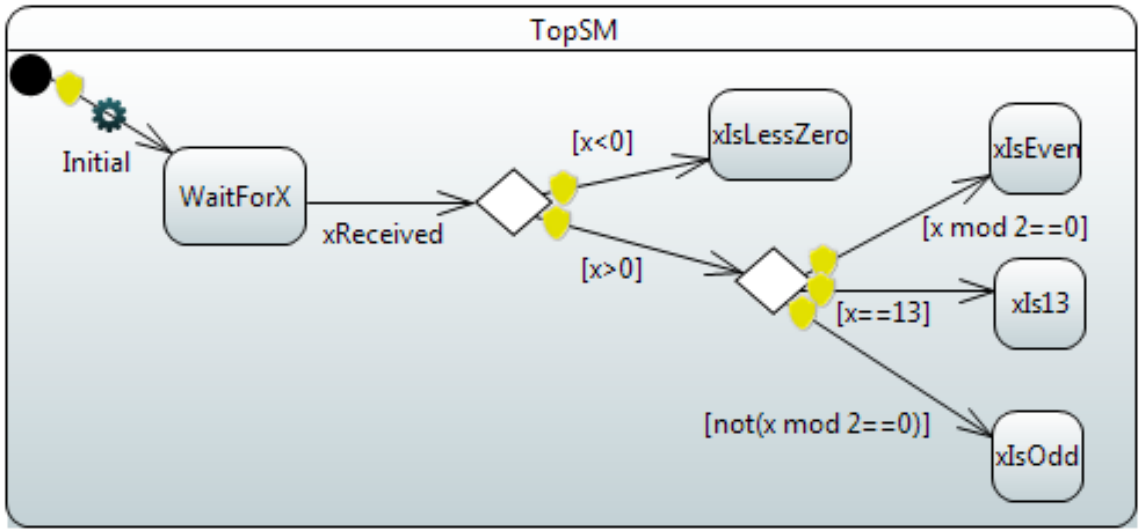
Trigger
Specifies port and message

Guard
Boolean condition that must hold

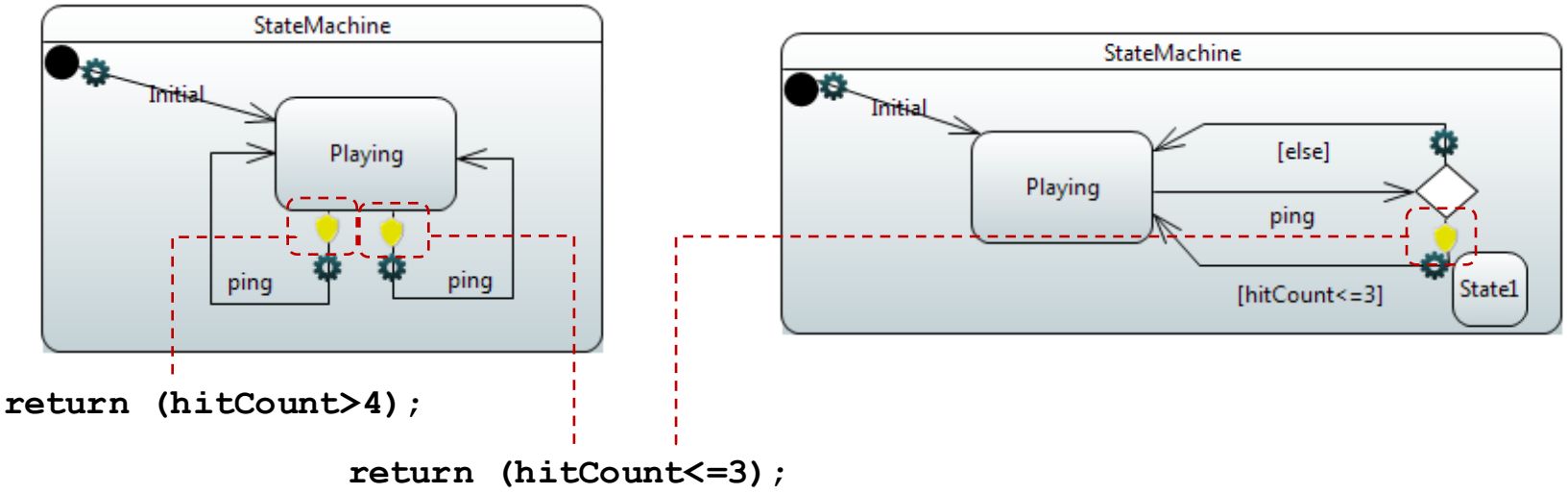
Effect/Actions
Code that is executed when transition is taken

Transitions Into and Out of Pseudo States

- Initial
  - Incoming transition: impossible
  - Outgoing transition: no guard, no trigger, but can have action code
- Choice point
  - Incoming transitions: can have guard, triggers, action code
  - Outgoing transitions:
    - No trigger, but should have guard
    - Guards should be pairwise disjoint (i.e., non-overlapping)
    - Collection of guards should be exhaustive



Guards on Transitions out of Basic States



- Dangerous: Easy to make mistakes
  - Hard to put trigger and guard info in name of transition
  - Forget that guards are there, what exactly they are
  - Have non-exhaustive or overlapping guards
- Better to use choice points

Action Language

- Language used in
  - guards to express Boolean expressions
  - entry action, exit action, transition effects to read and update attribute values, send messages
- Typically: C/C++, Java

State machines are a hybrid notation combining

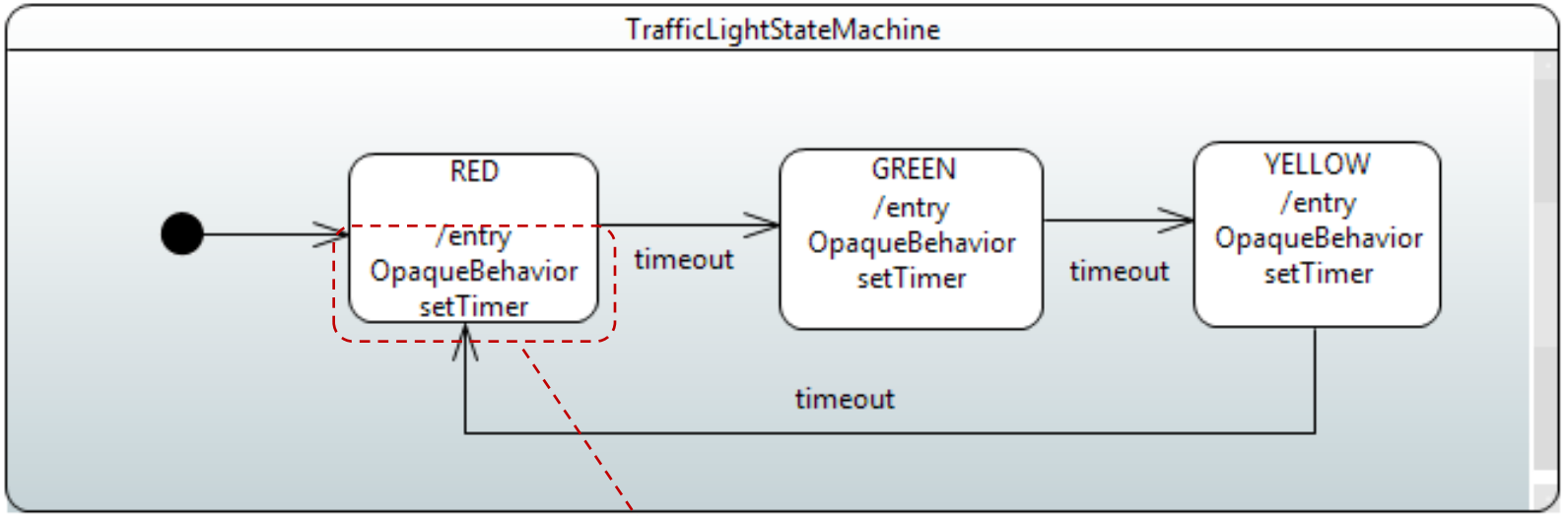
- graphical notation for state machines and
- textual notation for source code in actions

UML and UML-RT State Machines

- different from, e.g., Finite Automata
- closer to 'extended hierarchical communicating state machines' [6]

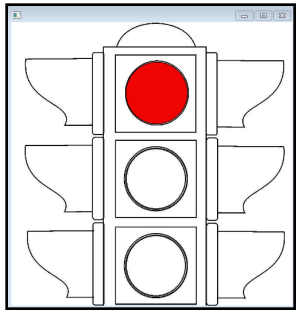
[6] R. Alur. Formal Analysis of Hierarchical State Machines. Verification: Theory and Practice. 2003.

Example: Action Code, Timers, Logging




```
$ ./TopMain.exe
Controller "DefaultController" running.
address: localhost, port: 8080
Switched to red
Switched to green
Switched to yellow
Switched to red
Switched to green
Switched to yellow
Switched to red
```

```
timer.informIn(UMLRTTimespec(5,0));
log.log("Switched to red");
```



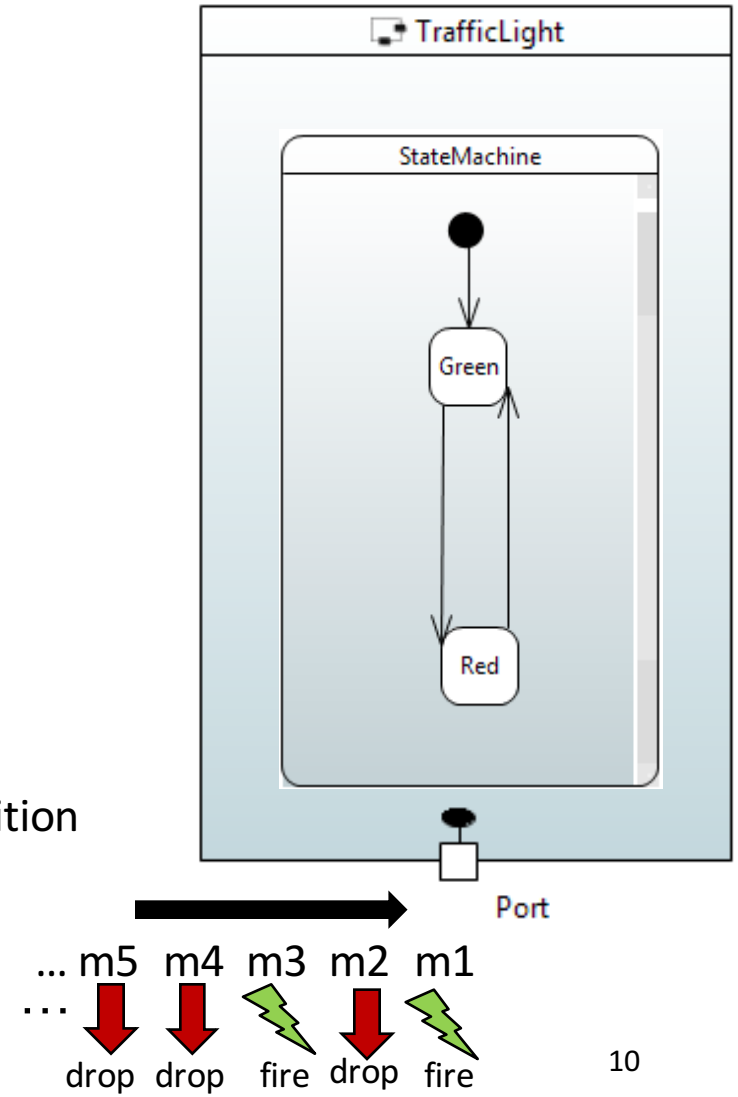
Execution Semantics I

// machine is in stable state configuration

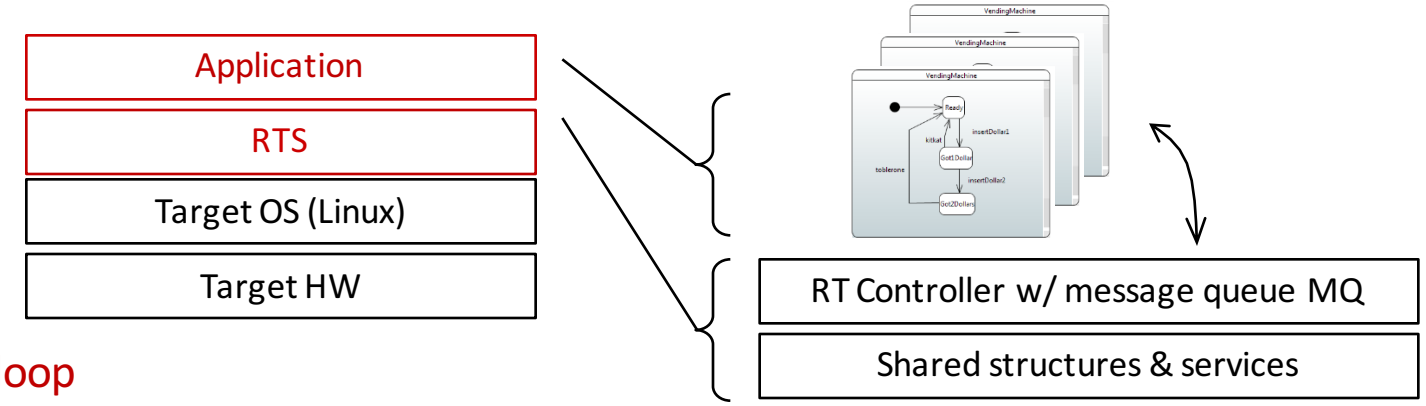
1. Message m1 has arrived and is dispatched
2. If dispatching enables no transition, m1 is 'dropped'
3. If dispatching enables transition t,
   - Source state of t active,
   - message matches trigger of t, and
   - guard evaluates to 'true'
4. then transition t executed
   a. Execute exit action of source state of t (if any)
   b. Execute action code of t (if any)
   c. Execute entry code of target state of t (if any)
5. If target of t is pseudo state,

   continue by choosing and executing outgoing transition (i.e., goto 5.)

// machine in stable state configuration



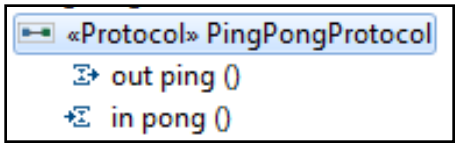
Execution Semantics I (Cont'd)

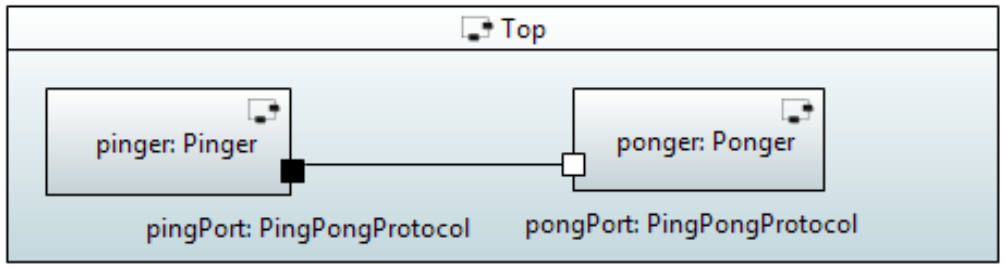


Controller main loop

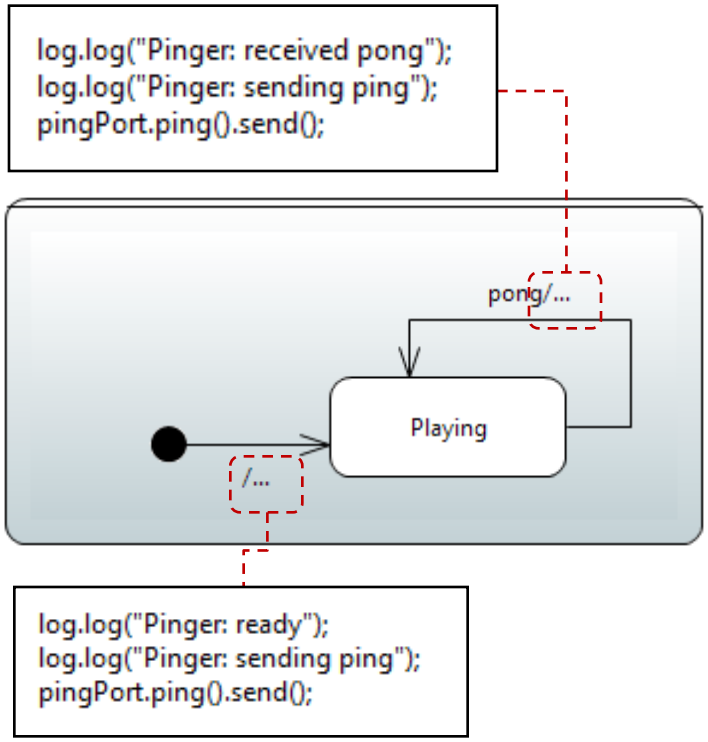
```
WHILE (1) {
  m = dequeue(MQ);
  IF can find transition t such that enabled(m,t) THEN
     targetState = execChain(t);
     mark targetState as active;
  ELSE
     report 'Unexpected message m';
}
WHERE
enabled(m,t) = source(t) is active, trigger(t) matches m, and eval(guard(t))='true'
execChain(t) = execute exit of source(t), if any;
               execute effect of t, if any;
               execute entry of target(t), if any;
          WHILE target(t) is choice point {
             find t' such that source(t')=target(t) and eval(guard(t'))='true';
                  execute effect of t', if any;
             execute entry of target(t'), if any;
                  t = t';
               }
               RETURN target(t);
```

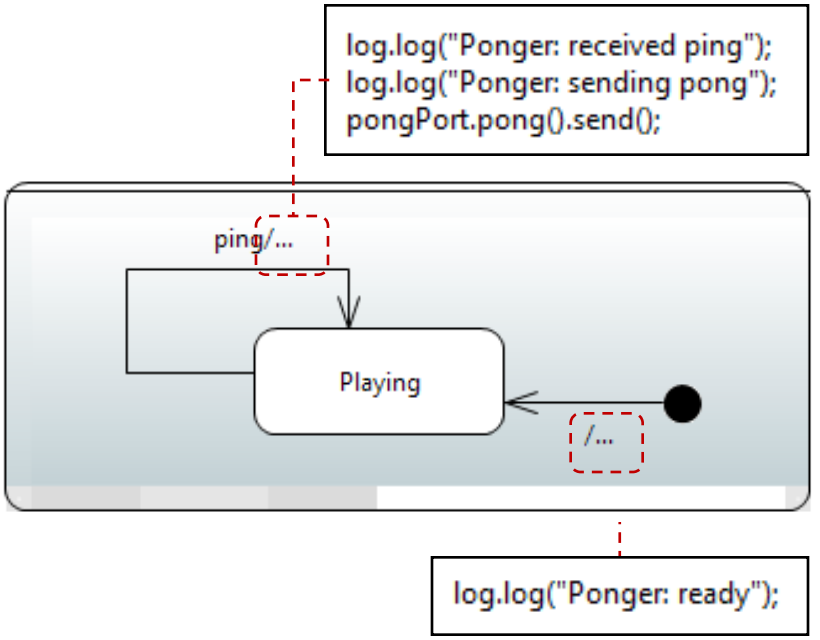
Example: Ping Pong





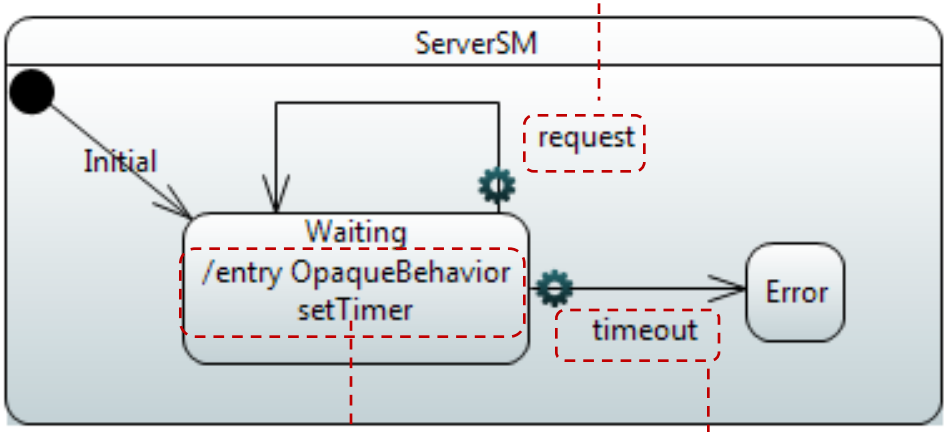

```
$ ./TopMain.exe
Controller "DefaultCon
Pinger: ready
Pinger: sending ping
Ponger: ready
Ponger: received ping
Ponger: sending pong
Pinger: received pong
Pinger: sending ping
Ponger: received ping
Ponger: sending pong
Pinger: received pong
Pinger: sending ping
Ponger: received ping
Ponger: sending pong
Pinger: received pong
Pinger: sending ping
```





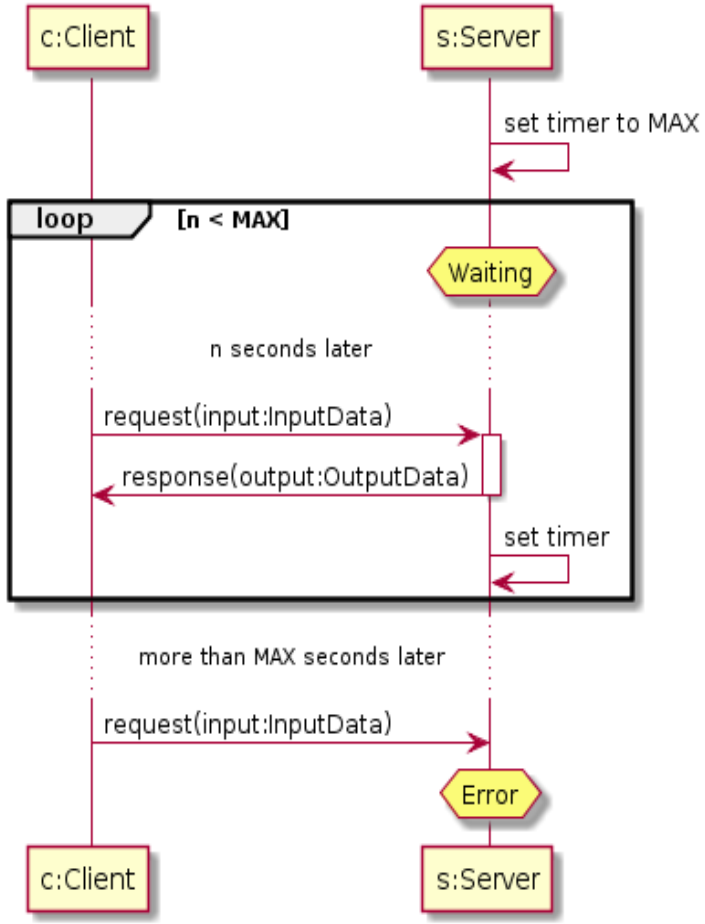
Example: Timers

```
logger.log("Processing request");
// cancel timer
timer.cancelTimer(timerId);
// compute output
p.response(output).send();
```




```
logger.log("Too late!");
```

```
logger.log("setting timer");
timer.informIn(UMLRTTimespec(MAX, 0));
```



Papyrus-RT

- Download
  - https://www.eclipse.org/papyrus-rt/content/download.php
  - Java 8, 64 bits
- Installation, tutorials
  - https://wiki.eclipse.org/Papyrus-RT/User
  - 2 parts:
    a) How to create models, generate code
    b) How to build generated code (easiest under Linux)
- Q&A forums
  - For Papyrus-RT: www.eclipse.org/forums/index.php/f/314/
  - For assignment: CISC 836 pages on http://onq.queensu.ca/

Papyrus-RT (Cont'd)

- Use
  - (model, generate, build, run)^*
- Generated code
  - `<workspace>/<projectName>_CDTProject/src`
- Building generated code
  - Papyrus-RT executable:
    - `C:\Users\Juergen Dingel\Programs\pRT1_Nov10_2017\Papyrus-RT\papyrusrt.exe`
  - UMLRTS_ROOT (under Cygwin)
    - `/cygdrive/c/Users/Juergen Dingel/Programs/pRT1_Nov10_2017/Papyrus-RT/plugins/org.eclipse.papyrusrt.rts_1.0.0.201707181457/umlrts`

Papyrus-RT (Cont'd)

- Tips and tricks
  - Common mistakes
    - Forgot: 'send' statement, trigger
    - When using 'code snippet':
      - don't confuse 'effect' and 'guard' tab
      - ensure changes saved properly
  - Timing quite imprecise when using Cygwin under Windows 7 and Vista
- Examples