Ptera: An Event-Oriented Model of Computation for Heterogeneous Systems

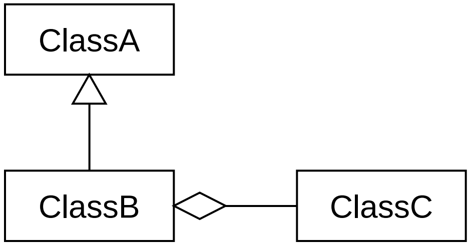
Thomas Huining Feng
Oracle Corp.

Edward A. Lee
EECS, UC Berkeley

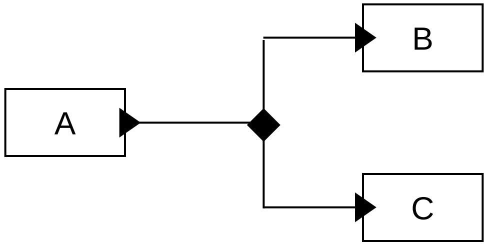
Lee W. Schruben
IEOR, UC Berkeley

EMSOFT 2010, Scottsdale, Arizona, USA *Oct 24-29, 2010*

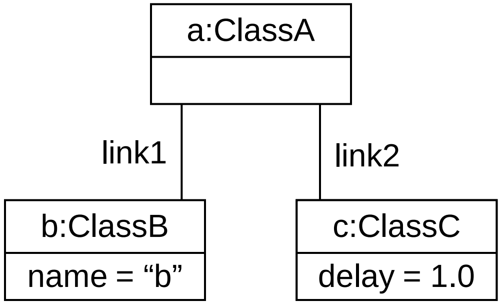
Models of Systems



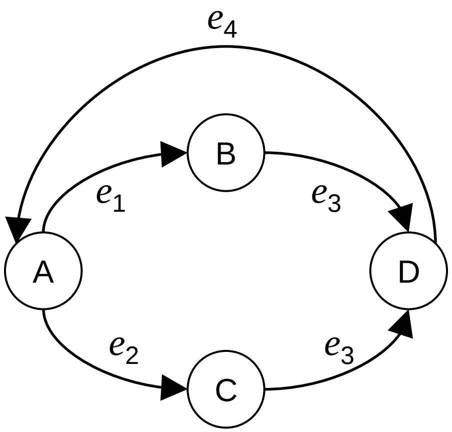
Class Diagram



Actor Diagram

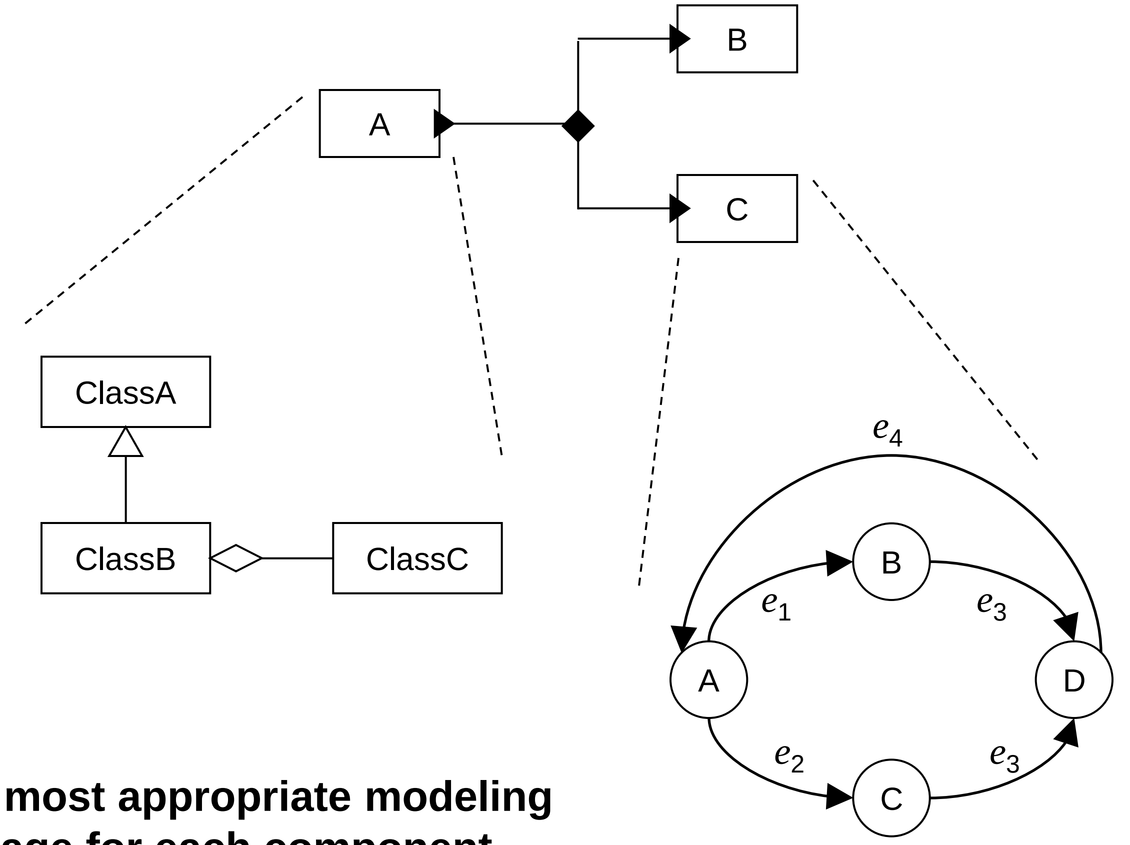


Object Diagram



State Diagram

Hierarchical Heterogeneous Modeling



Use the most appropriate modeling language for each component.

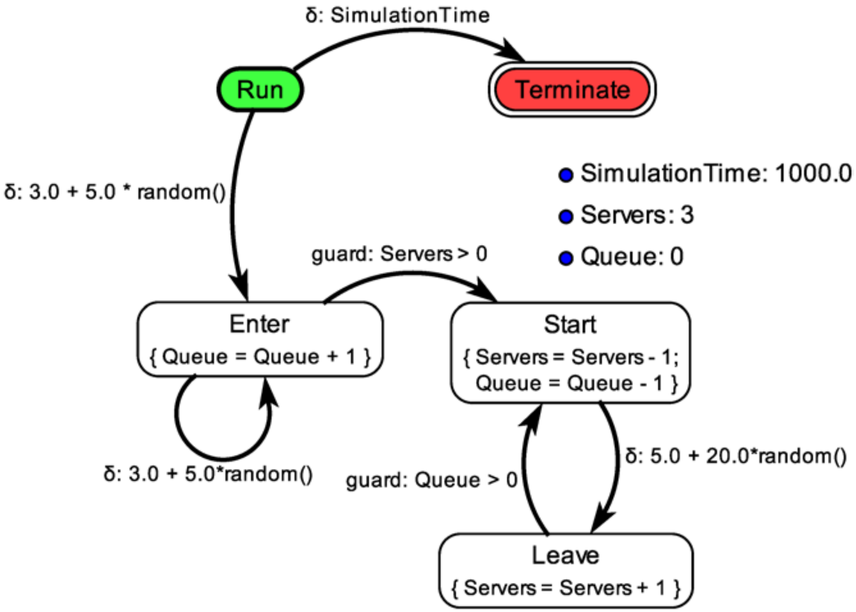
Compose those components to form more complex systems.

The Event-Oriented View

- Ptera (Ptolemy Event-Relationship Actor) Based on event graphs [Schruben 1983]
- Visual representation
  - Nodes are events
  - Edges are scheduling relations

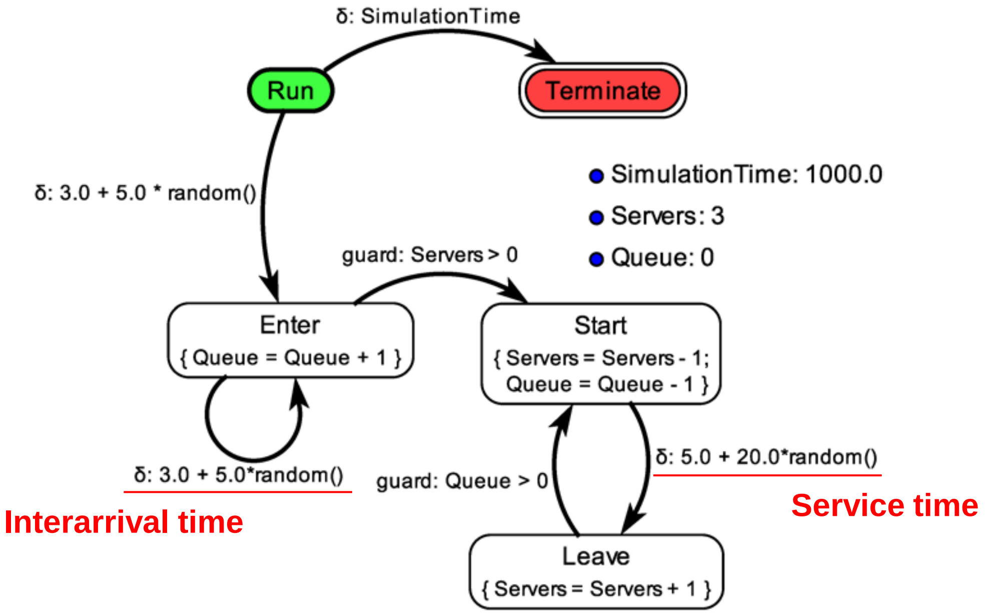
- Compare
  - State diagram
  - UML activity diagram
  - Business process modeling

CarWash: single queue multiple servers



The CarWash model

Execution with an Event Queue



δ: SimulationTime

Run

Terminate

δ: 3.0 + 5.0 * random()

- SimulationTime: 1000.0
- Servers: 3
- Queue: 0

guard: Servers > 0

Enter
{ Queue = Queue + 1 }

Start
{ Servers = Servers - 1;
Queue = Queue - 1 }

δ: 3.0 + 5.0*random()

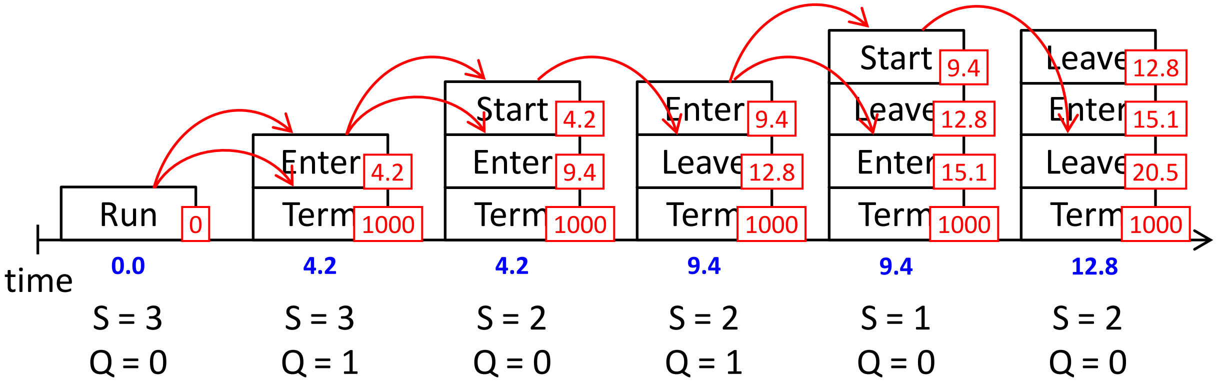
guard: Queue > 0

δ: 5.0 + 20.0*random()

Interarrival time

Service time

Leave
{ Servers = Servers + 1 }



Run 0

Enter 4.2
Term 1000

Start 4.2
Enter 9.4
Term 1000

Enter 9.4
Leave 12.8
Term 1000

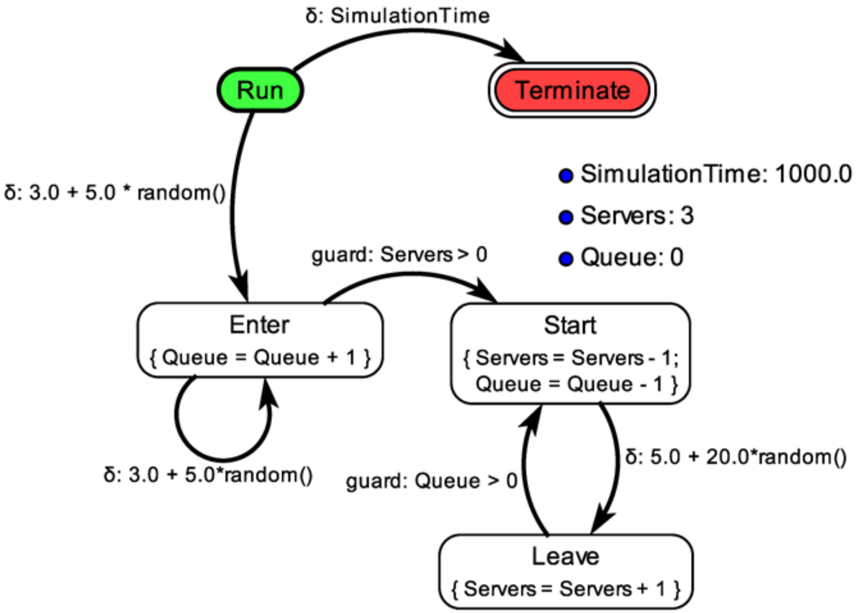
Start 9.4
Leave 12.8
Enter 15.1
Term 1000

Leave 12.8
Enter 15.1
Leave 20.5
Term 1000

time

0.0	4.2	4.2	9.4	9.4	12.8
S = 3	S = 3	S = 2	S = 2	S = 1	S = 2
Q = 0	Q = 1	Q = 0	Q = 1	Q = 0	Q = 0

Simulation



δ: SimulationTime

Run

Terminate

δ: 3.0 + 5.0 * random()

- SimulationTime: 1000.0
- Servers: 3
- Queue: 0

guard: Servers > 0

Enter
{ Queue = Queue + 1 }

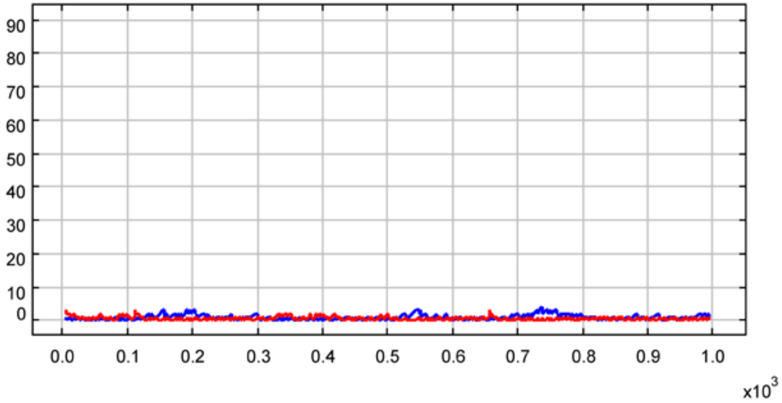
Start
{ Servers = Servers - 1;
Queue = Queue - 1 }

δ: 3.0 + 5.0*random()

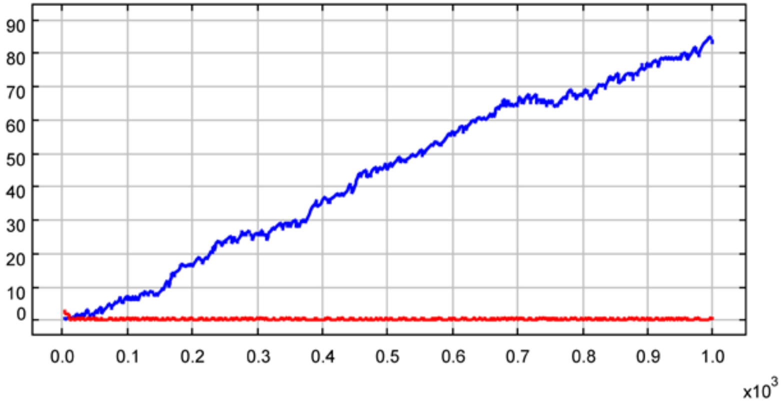
guard: Queue > 0

δ: 5.0 + 20.0*random()

Leave
{ Servers = Servers + 1 }



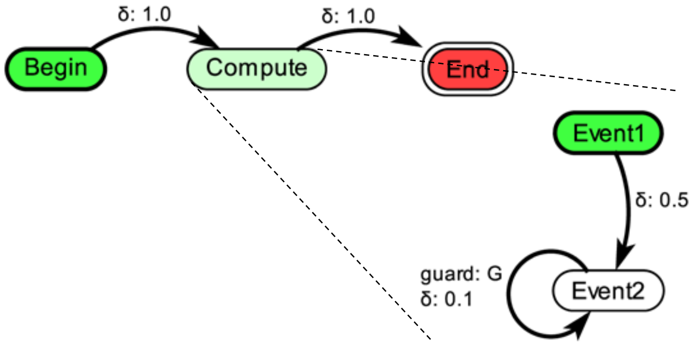
IAT = 3.0 + 5.0 * random()



IAT = 1.0 + 5.0 * random()

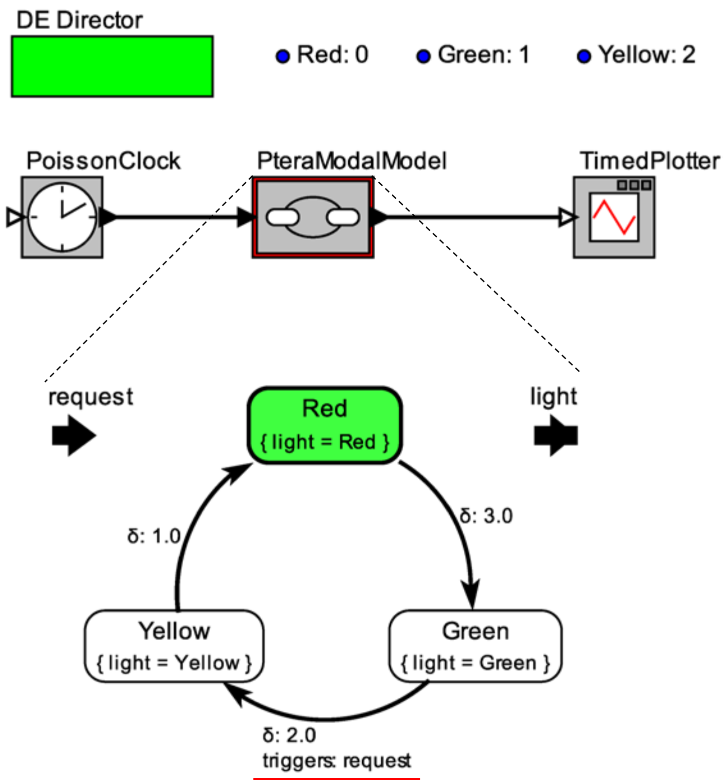
Model Hierarchy: The Ptera Approach

- A submodel is itself a model
  - No difference in syntax
  - Conceptually equipped with an isolated event queue
  - A global notion of model time



- Implication: events (or tasks) are no longer instantaneous
  - Start of an event causes start of its submodel
  - End of the submodel causes end of the event

Communication via Ports



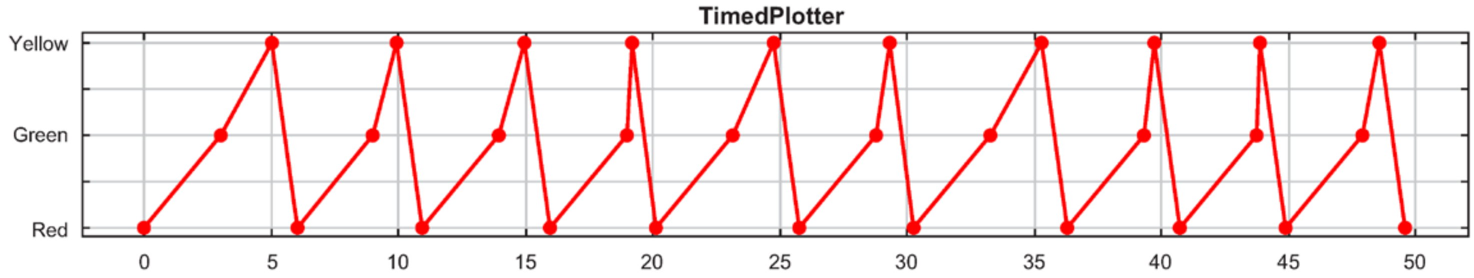
Triggering port

Event processing conditions

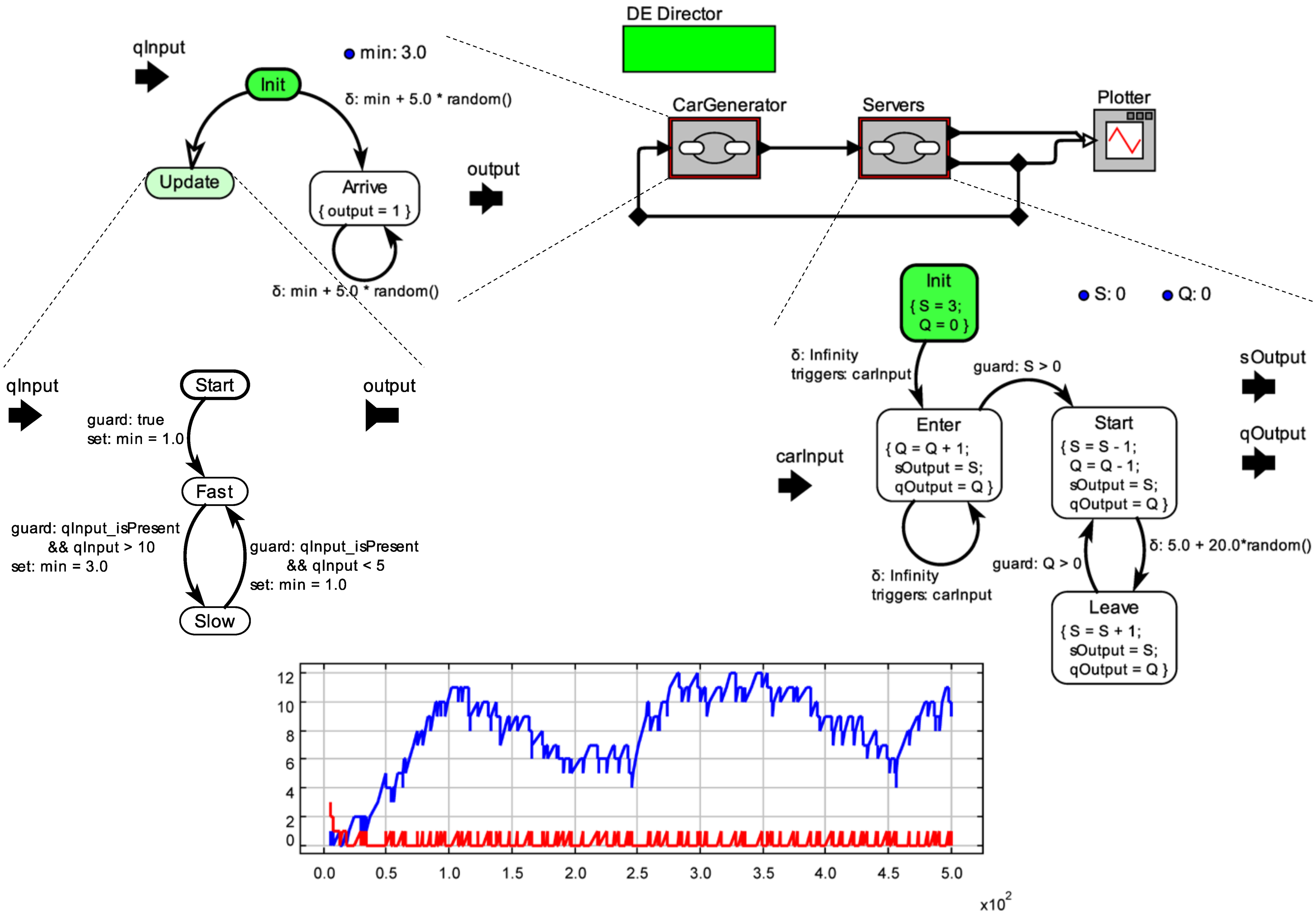
1. Scheduled time is reached, or
2. Tokens received at one or more triggering ports

Inputs not triggering any event are ignored.

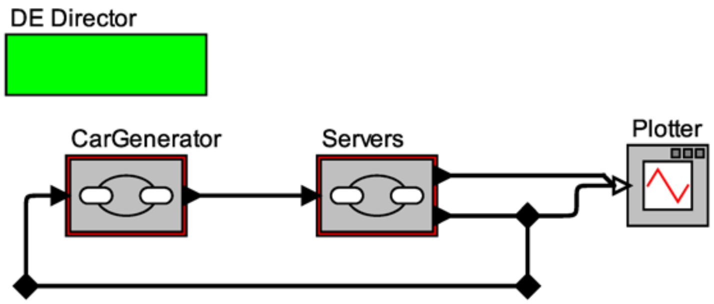
Outputs can be sent in actions.



Hierarchical Multimodeling



Hierarchical Multimodeling

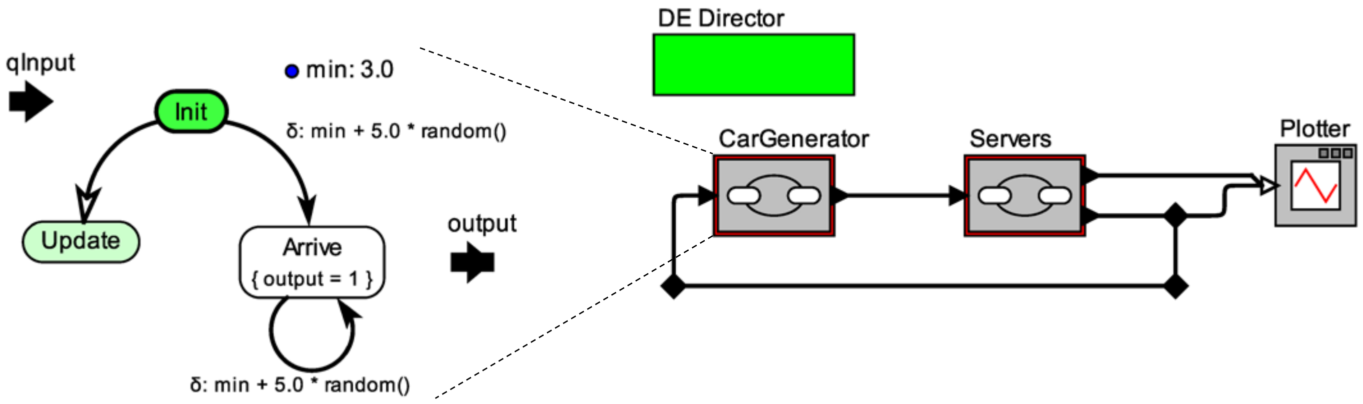


Goal: reusable, robust and flexible design

Choose DE at top level for

- Concurrency
- Concern separation
- Encapsulation
- Fixpoint semantics
- Out-of-order execution
- Distributed execution

Hierarchical Multimodeling



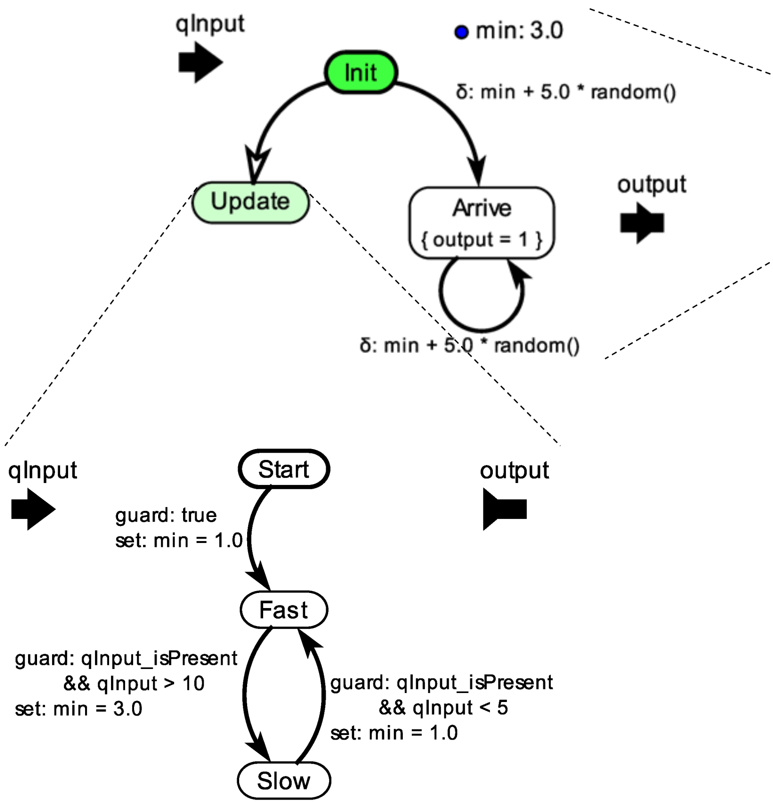
Choose Ptera to model a random process

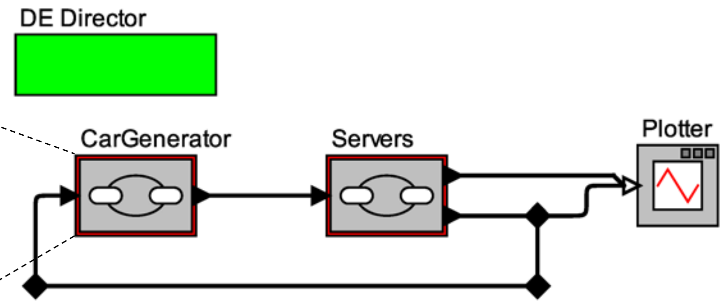
- No need to depend on predefined actors
- Easy to control the exact behavior
- Totally sequential (but concurrency may be possible)

Some predefined actors can be designed in this way

- Source actors
- Math actors
- Time delay actors
- Flow control actors

Hierarchical Multimodeling





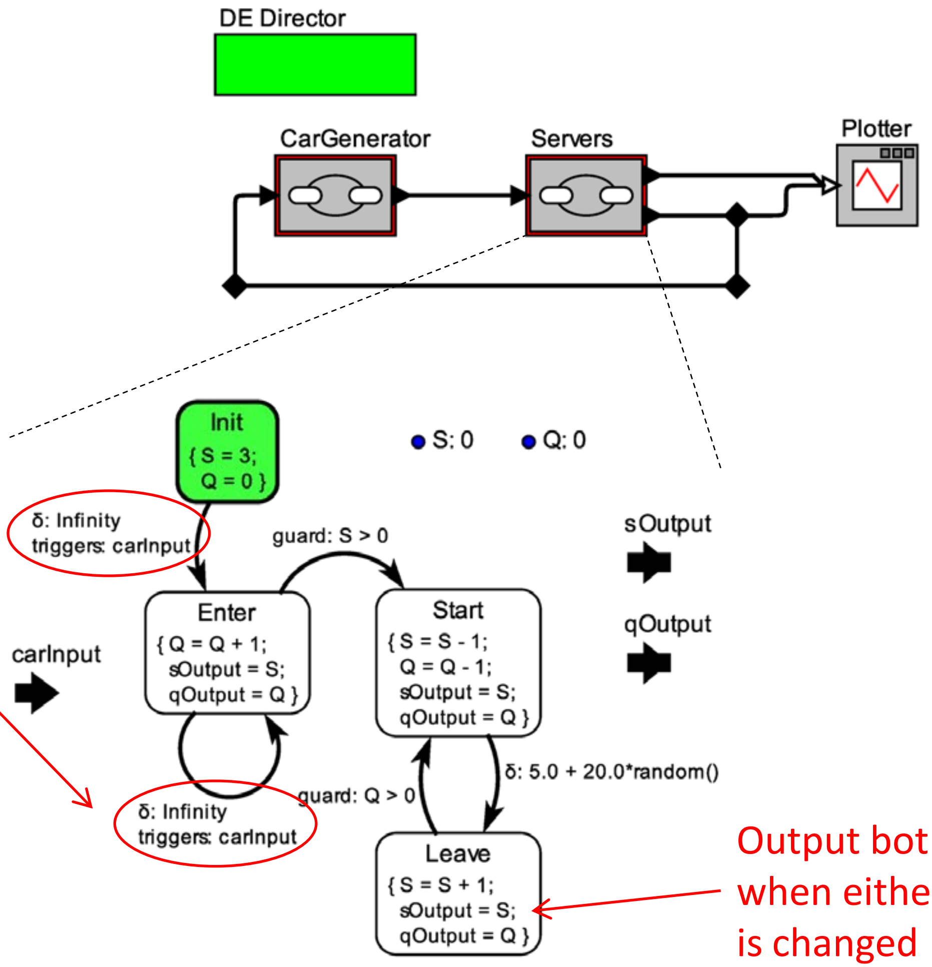
Use FSM to capture two modes

- Easy to understand
- Easy to model check, debug, convert into other languages, ...

Submodel firing conditions

- The submodel itself requests (not in this case), or
- Input is received at a port, or
- The event containing the submodel is processed

Hierarchical Multimodeling



Indefinitely wait for change of car number.

Output both S and Q when either of them is changed

A Ptera model similar to the standalone version.

Listen to the input port for car arrivals.

Opportunities

- Composition with other MoCs
  (Especially, Ptides and continuous time)
- Formal analysis
  (Bound of event queue, simultaneous events, termination condition, model categorization, …)
- Behavior-preserving concurrent and distributed execution
- Other application domains
  (Currently studied: statistical analysis, model transformation)
- Tool support
  (Debugging and testing, code generation)
- Design patterns
  (Currently studied: Input, Output, LoopForCount, ParallelTasks, SingleQueueMultipleServers)