Parallel DEVS & DEVSJAVA

Presented by
Ximeng Sun
Mar 16, 2005

References

- Bernard P. Zergler, Herbert Praehofer, and Tag Gon Kim.

  Theory of Modeling and Simulation.

  Academic Press, 2000.
- Bernard P. Zergler, Hessam S. Sarjoughian. *Introduction to DEVS Modeling and Simulation with JAVA.* http://www.acims.arizona.edu/SOFTWARE/software.shtml#DEVSJAVA

Outline

- *Classic DEVS* quick review
- Why *Parallel DEVS*
- *Parallel DEVS* Formalism
  - Atomic Model
  - Coupled Model
  - Closure under Coupling
- *Parallel DEVS* Simulation Protocol
- DEVSJAVA

- *Classic DEVS* quick review
- Why *Parallel DEVS*
- *Parallel DEVS* Formalism
  - Atomic Model
  - Coupled Model
  - Closure under Coupling
- *Parallel DEVS* Simulation Protocol
- DEVSJAVA

Classic DEVS formalism

$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$

Where

X is the set of **inputs**

S is a set of **states**

Y is the set of **outputs**

$\delta_{int} : S \rightarrow S$ is the internal transition function

$\delta_{ext} : Q \times X \rightarrow S$ is the ***external transition* function**, where

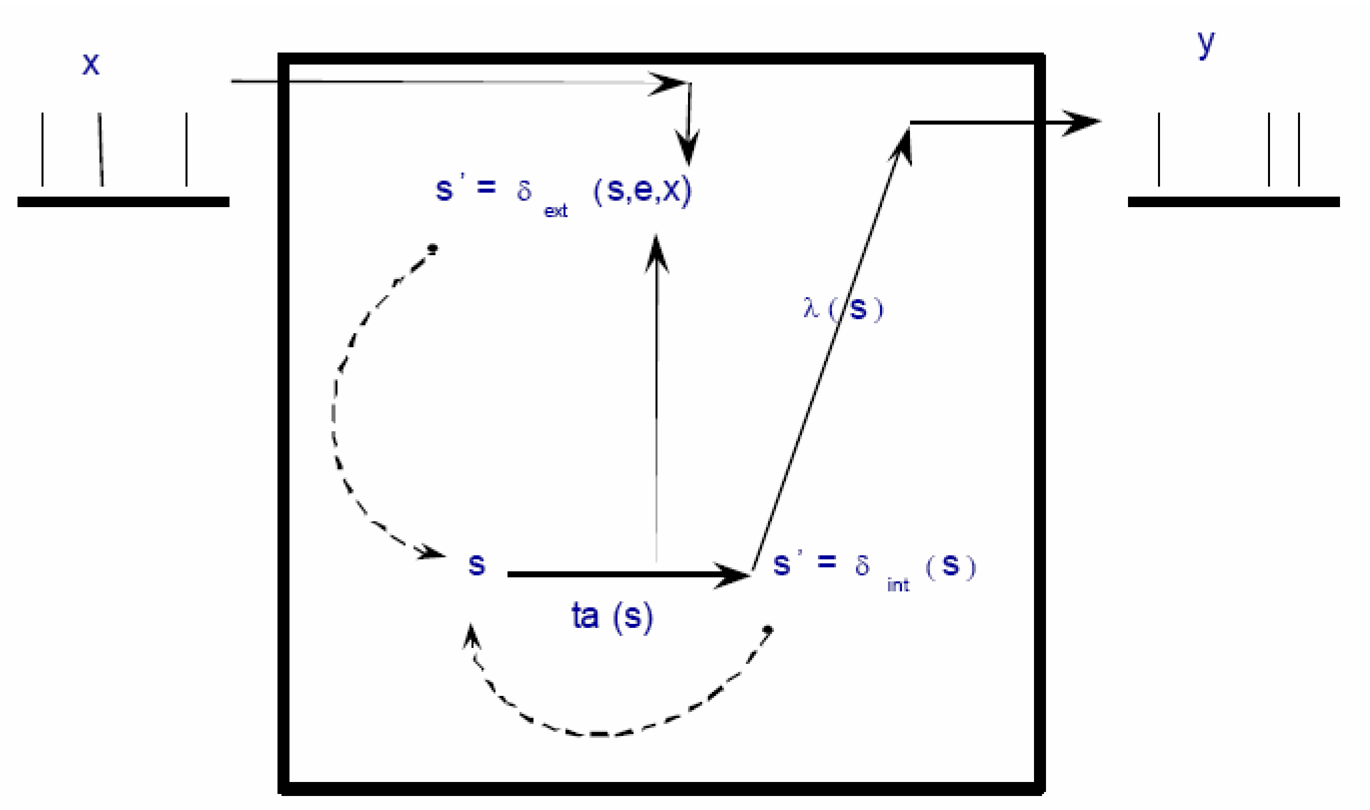
$Q = \{(s,e) \mid s \in S, 0 \leq e \leq ta(s)\}$ is the **total state** set

e is the **time elapsed** since last transition

$\lambda : S \rightarrow Y$ is the **output function**

$ta : S \rightarrow R^{+}_{0,\infty}$ is the ***time advance*** function

DEVS in action



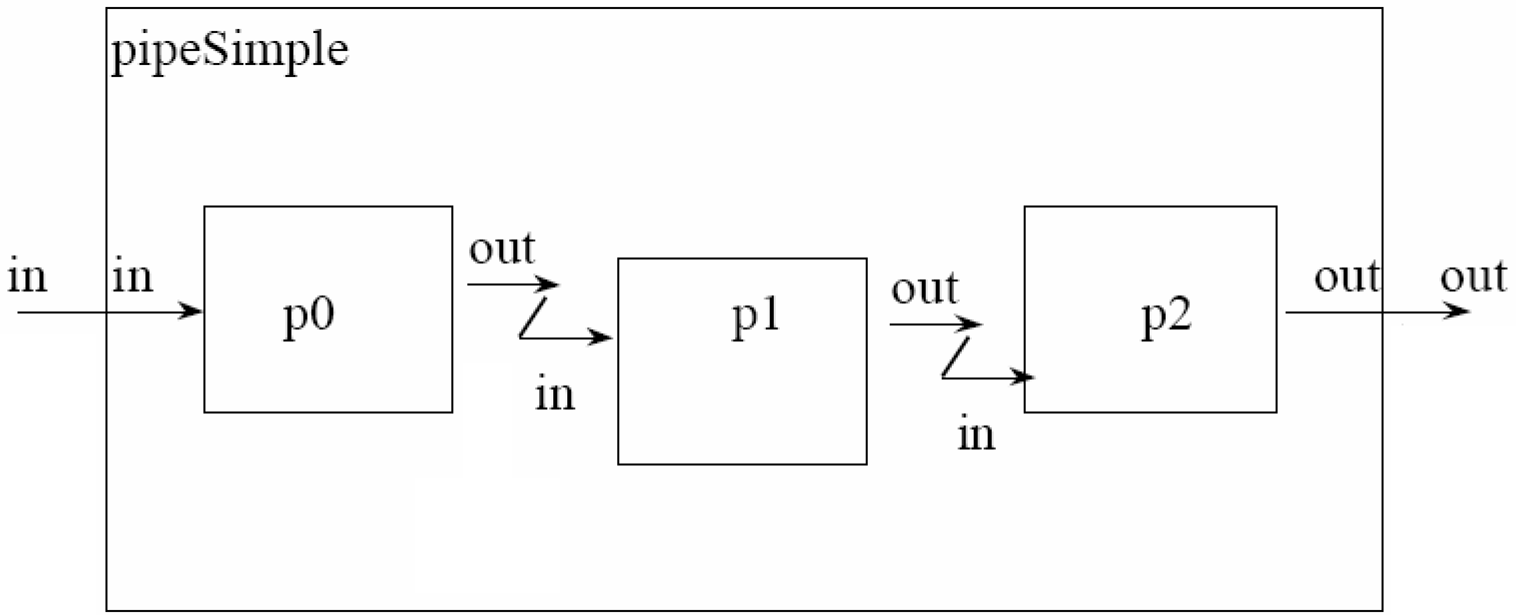
Classic DEVS Coupled Model

$$N = \langle X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC, Select \rangle$$

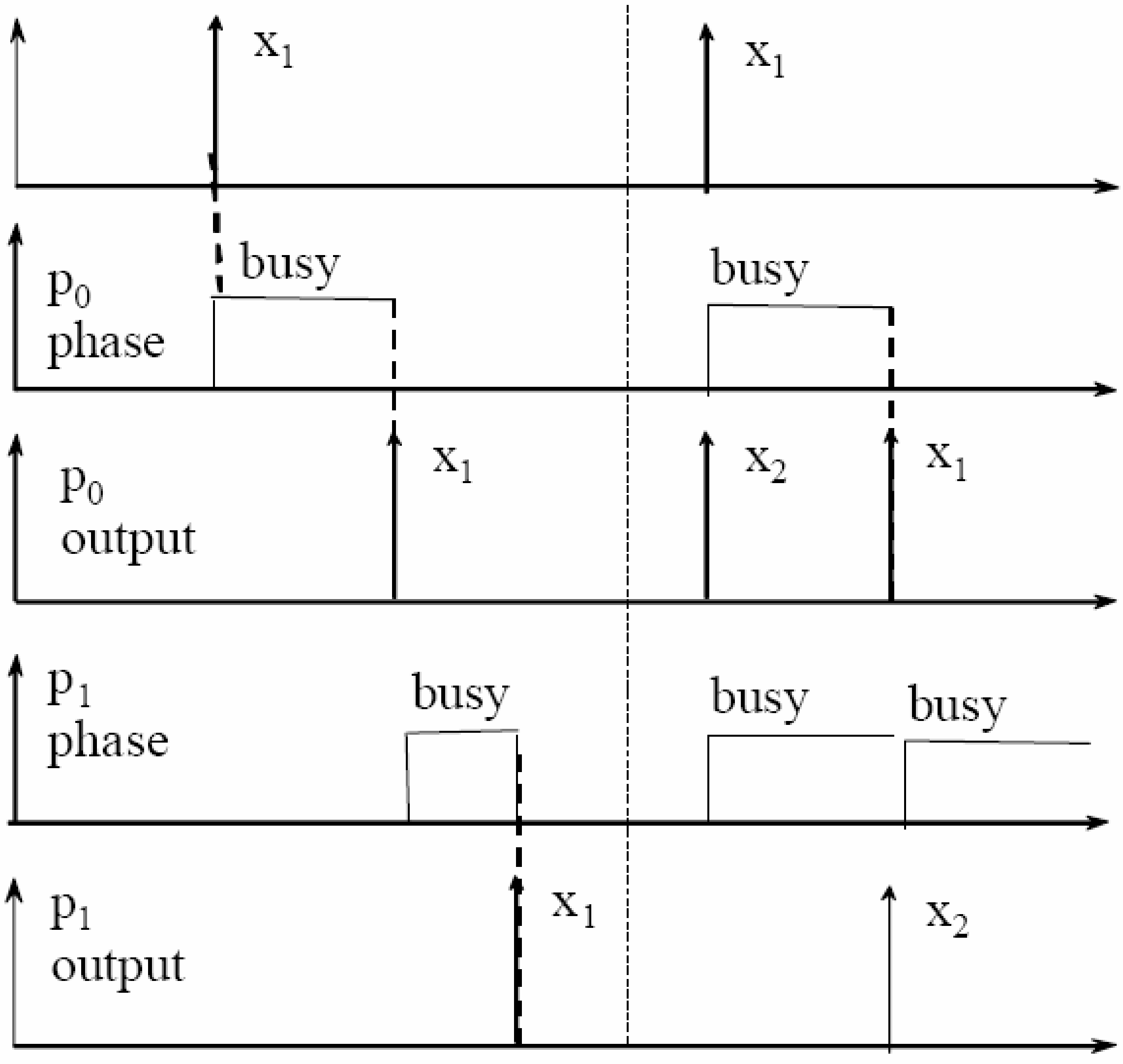
- $X = \{(p, v) | p \in IPorts, v \in X_p\}$ is the set of input ports and values
- $Y = \{(p, v) | p \in OPorts, v \in Y_p\}$ is the set of output ports and values
- D : the set of the components names.
- M_d : component DEVS models
- EIC : external input coupling connects external inputs to component inputs
- EOC : external output coupling connects component outputs to external outputs
- IC : internal coupling connects component outputs to component inputs
- $Select : 2^D - \{\} \rightarrow D,$ the tie-breaking function for imminent components

- *Classic DEVS* quick review
- Why *Parallel DEVS*
- *Parallel DEVS* Formalism
  - Atomic Model
  - Coupled Model
  - Closure under Coupling
- *Parallel DEVS* Simulation Protocol
- DEVSJAVA

Simple Pipeline model

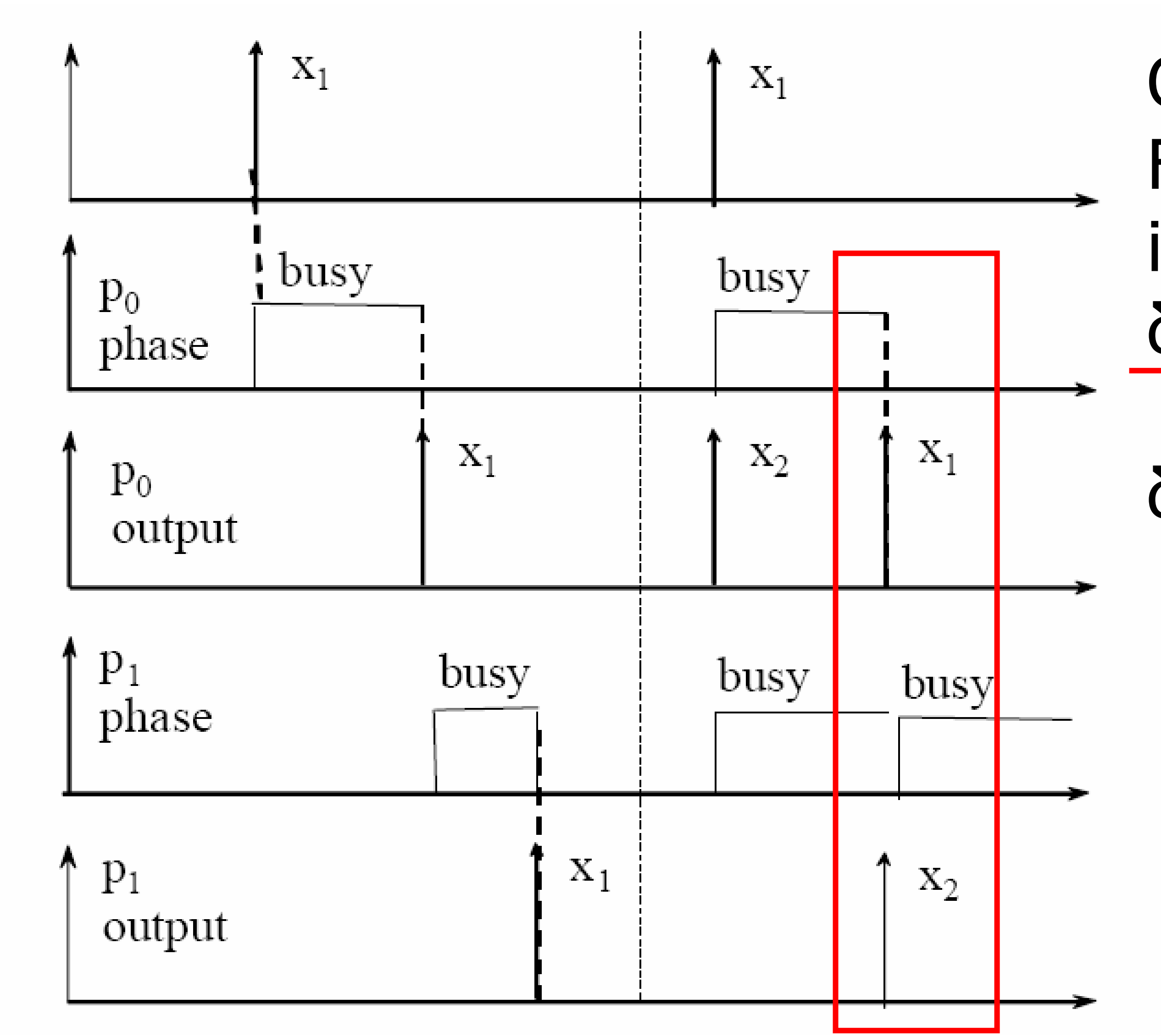


In Action



x_1
x_1
p_0 phase
busy
busy
p_0 output
x_1
x_2
x_1
p_1 phase
busy
busy
busy
p_1 output
x_1
x_2

Simultaneous events



Q?

For p1 which one is correct:

δ int –> δ ext

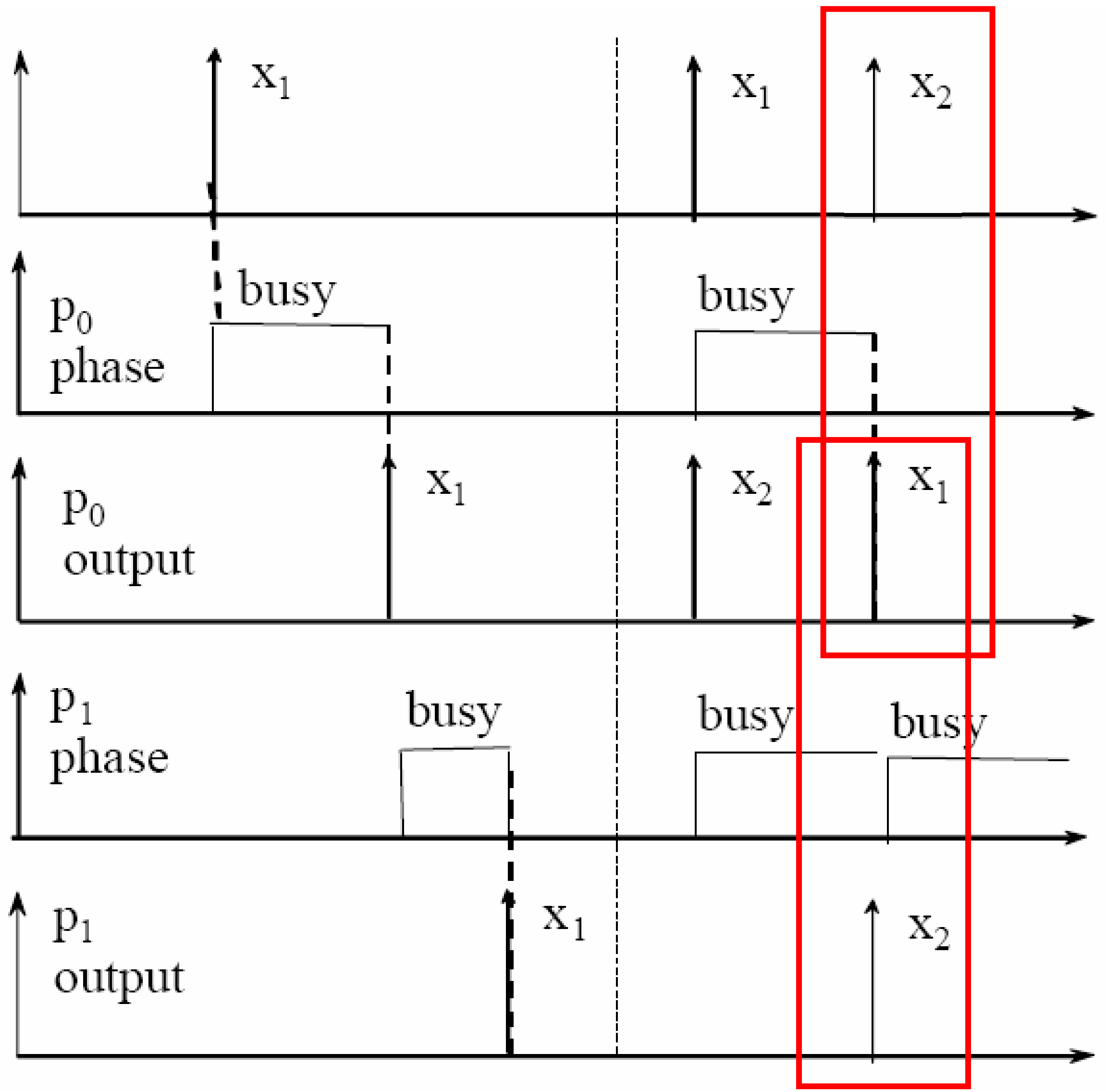
or

δ ext –> δ int

Indirect control

- In Classic DEVS, only one would be chosen to execute by Select function.
  - Select: $s \rightarrow p1 \equiv$ internal-transition-first
  - Select: $s \rightarrow p0 \equiv$ external-transition-first

If there's a feedback…



Lose input anyway

- In Classic DEVS, always make the same choice among imminent components.
  - Select: s –> p0|p1 ≡ p0|p1 loses input

- *Classic DEVS* quick review
- Why *Parallel DEVS*
- *Parallel DEVS* Formalism
  - Atomic Model
  - Coupled Model
  - Closure under Coupling
- *Parallel DEVS* Simulation Protocol
- DEVSJAVA

Parallel DEVS Atomic Model

$$DEVS = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta)$$

where

$X_M = \{(p,v) \mid p \in IPorts, v \in Xp\}$	is the set of *input ports and values*;
$Y_M = \{(p,v) \mid p \in OPorts, v \in Yp\}$	is the set of *output ports and values;*
S	is the set of sequential states;
$\delta_{ext} : Q \times \boxed{X_M^b} \rightarrow S$	is the *external state transition function*;
$\delta_{int} : S \rightarrow S$	is the internal state transition function;
$\delta_{con} : S \times X_M^b \rightarrow S$	is the *confluent transition function*;
$\lambda : S \rightarrow \boxed{Y^b}$	is the output function;
$ta : S \rightarrow R_0^+ \cup \infty$	is the *time advance function*;

With $Q := \{(s,e) \mid s \in S, 0 \le e \le ta(s)\}$ the set of *total states*.

***Xb* is a set of bags over elements in *X*.**

Extensions of Classic DEVS

- Allowing bags of inputs to the external function
  - Inputs may arrive in any order
  - Inputs with the same identity may arrive from one or more sources
- Introducing confluent transition function
  - Localize collision tie-breaking control

Confluent Transition Function

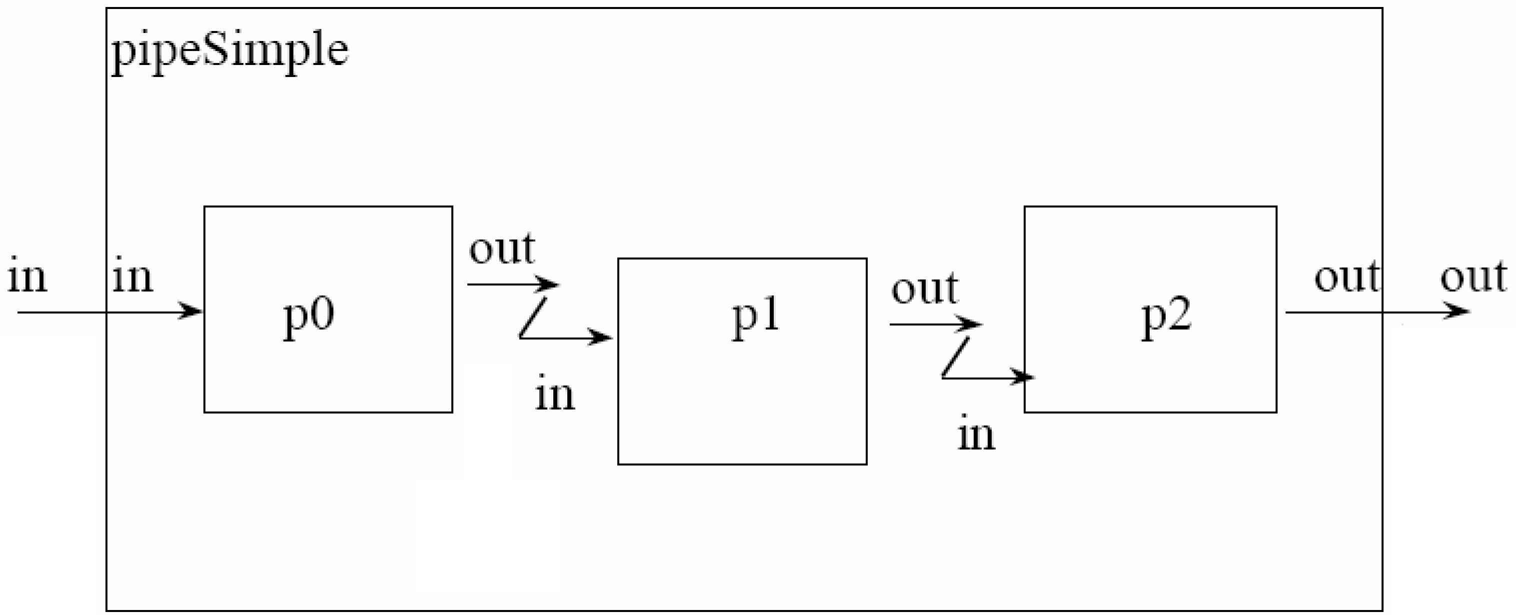
- Collision: $e = ta(s)$
- Classic DEVS: by Select function, at coupled model level – Global decision
- Parallel DEVS: by δ_{con}, to each individual component – Local decision
  - Default: $\delta_{con}(s,x) = \delta_{ext}(\delta_{int}(s),0,x)$
  - Or: $\delta_{con}(s,x) = \delta_{int}(\delta_{ext}(s,ta(s),x))$

Parallel DEVS Coupled Model

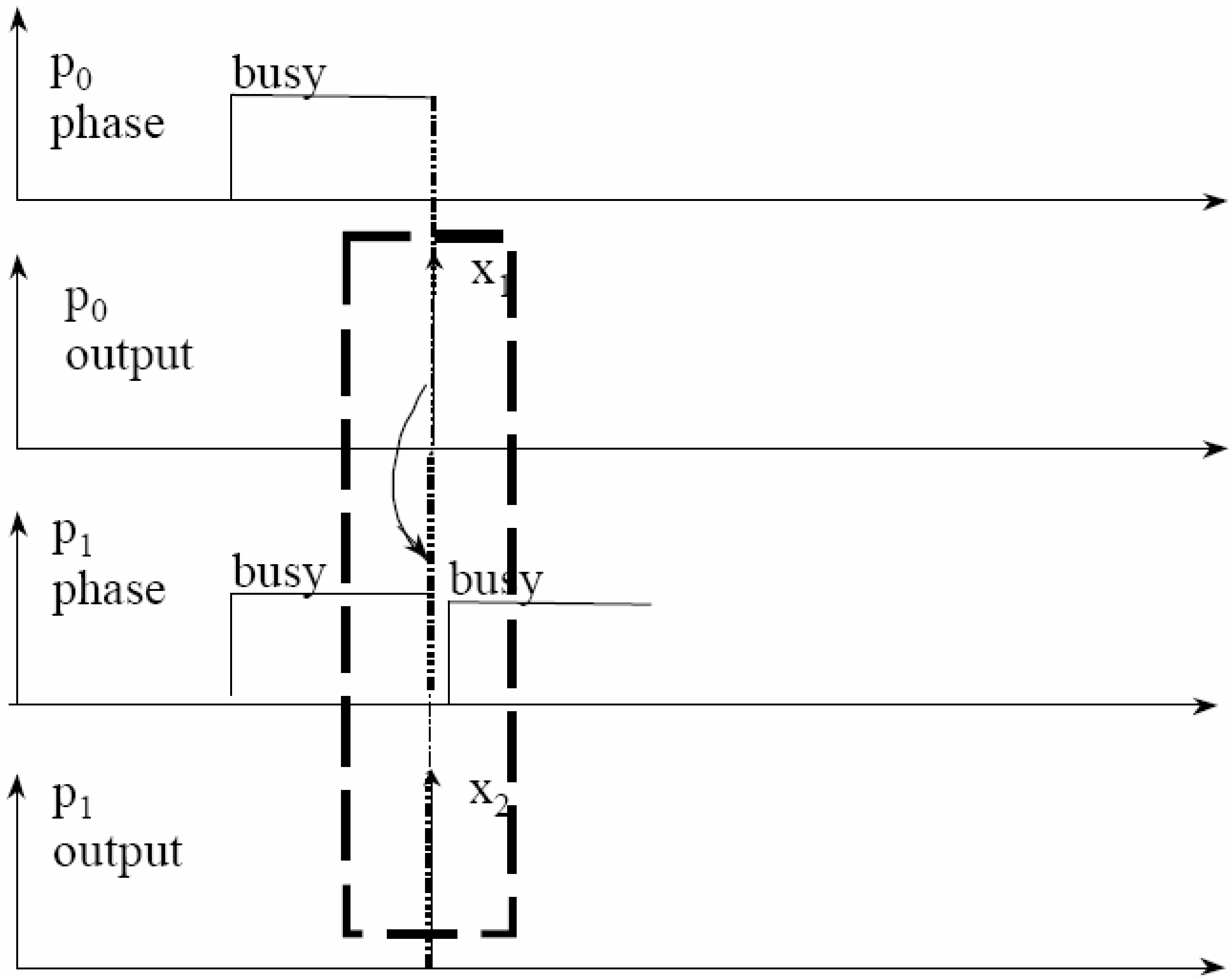
$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} \rangle$$

- Identical to Classic DEVS, except for the absence of the *Select* function
  - X : a set of input events
  - Y : a set of output events
  - D : a set of component references
  - M_d : a *Parallel DEVS* model, for each $d \in D$
  - I_d : a set of influencers of d , $I_d \subseteq D \cup \{N\}, d \notin I_d$ for each $d \in D \cup \{N\}$
  - $Z_{i,d}$: a set of output-to-input translation functions, for each $i \in I_d$

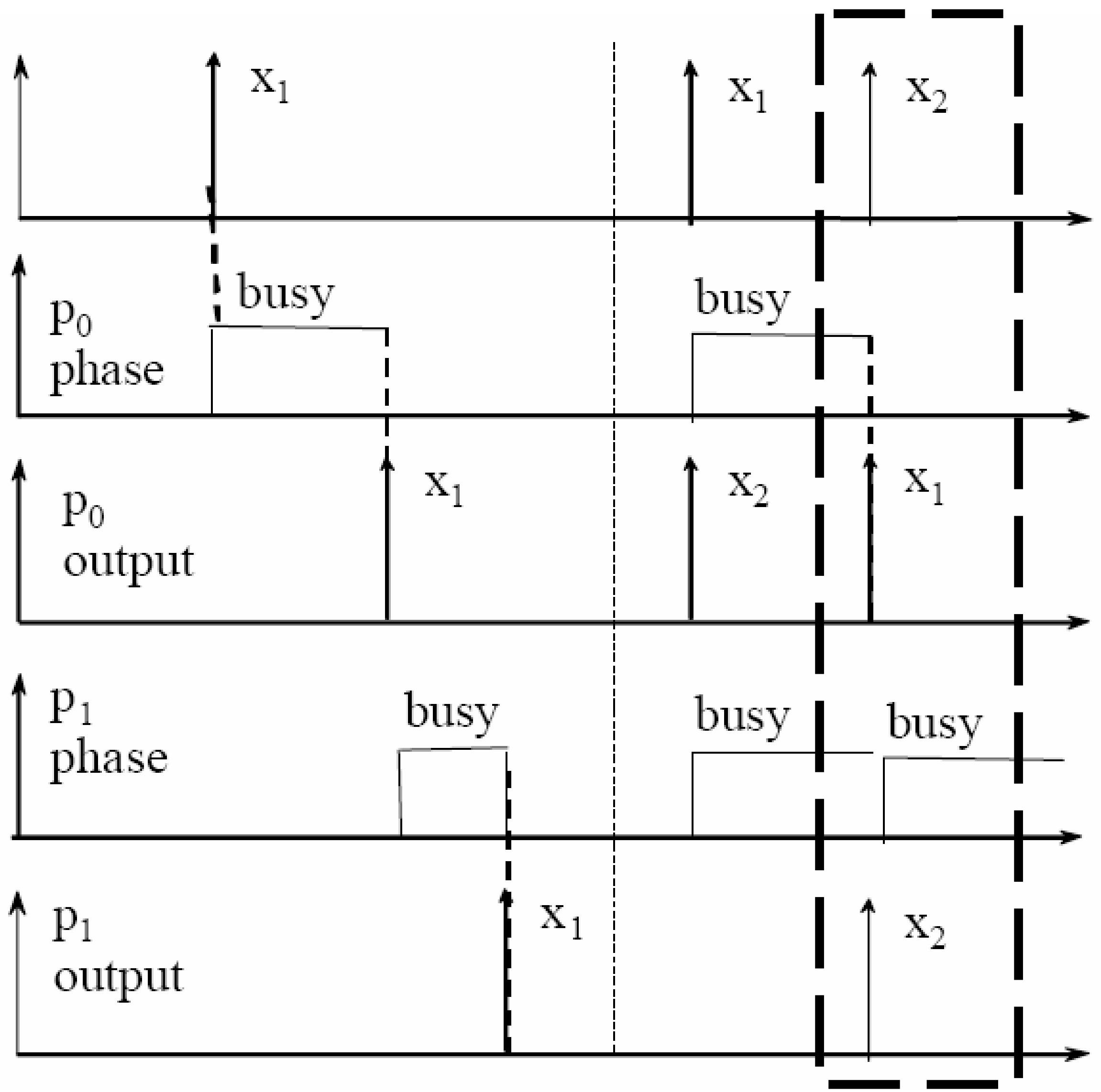
Previous example



Handling of imminent components in Parallel DEVS



Problem in Classic DEVS solved



Closure under Coupling of Parallel DEVS

- Resultant of the coupled model:

$$DEVS_N = \langle X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta \rangle$$

- Partition components into 4 sets:

$$ta(s) = minimum\{\sigma_d | d \in D\},$$
$$\text{where } s \in S \text{ and } \sigma_d = ta(s_d) - e_d$$

$$IMM(s) = \{d | \sigma_d = ta(s)\}$$ imminent components

$$INF(s) = \{d | i \in I_d, i \in IMM(s) \wedge x_d^b \neq \Phi\},$$
$$\text{where } x_b^d = \{Z_{i,d(\lambda_i(s_i))|i \in IMM(s) \cap I_d}\}$$ components about to receive inputs

- $CON(s) = IMM(s) \cap INF(s)$ (confluent components)
- $INT(s) = IMM(s) - INF(s)$ (imminent components receiving no input)
- $EXT(s) = INF(s) - IMM(s)$ (components receiving input but not imminent)
- $UN(s) = D - IMM(s) - INF(s)$ (remaining components)

Closure under Coupling of Parallel DEVS

- Functions of the Resultant:
  - Output Function: $\lambda(s) = \{Z_{d,N}(\lambda_d(s_d)) | d \in IMM(s) \wedge d \in I_N\}$
  - Internal Transition Function:

We define

$$\delta_{int}(s) = (..., (s'_d, e'_d), ...),$$

where

$$
\begin{aligned}
(s'_d, e'_d) &= (\delta_{int,d}(s_d), 0) && \text{for } d \in INT(s), \\
(s'_d, e'_d) &= (\delta_{ext,d}(s_d, e_d + ta(s), x^b_d), 0) && \text{for } d \in EXT(s), \\
(s'_d, e'_d) &= (\delta_{con,d}(s_d, x^b_d), 0) && \text{for } d \in CONF(s), \\
(s'_d, e'_d) &= (s_d, e_d + ta(s)) && \text{otherwise}
\end{aligned}
$$

Closure under Coupling of Parallel DEVS

- External Transition Function:

We define

$$\delta_{ext}(s, e, x^b) = (\ldots, (s'_d, e'_d), \ldots),$$

where

$$(s'_d, e'_d) = (\delta_{ext,d}(s_d, e_d + e, x^b_d), 0) \qquad \text{for } N \in I_d \wedge x^b_d \neq \Phi,$$

$$(s'_d, e'_d) = (s_d, e_d + e) \qquad \text{otherwise}$$

- Confluent Transition Function:

$$INF'(s) = \{d|(i \in I_d, i \in IMM(s) \vee N \in I_d) \wedge x_d^b \neq \Phi\},$$

$$\text{where } x_b^d = \boxed{\{Z_{i,d}(\lambda_i(s_i))|i \in IMM(s) \wedge i \in I_d\} \cup \\ \{Z_{N,d}(x)|x \in x^b \wedge N \in I_d\}}$$

Then we have

$$CON'(s) = IMM(s) \cap INF'(s)$$

$$INT'(s) = IMM(s) - INF'(s)$$

$$EXT'(s) = INF'(s) - IMM(s)$$

We define

$$\delta_{con}(s, x^b) = (..., (s'_d, e'_d), ...),$$

where

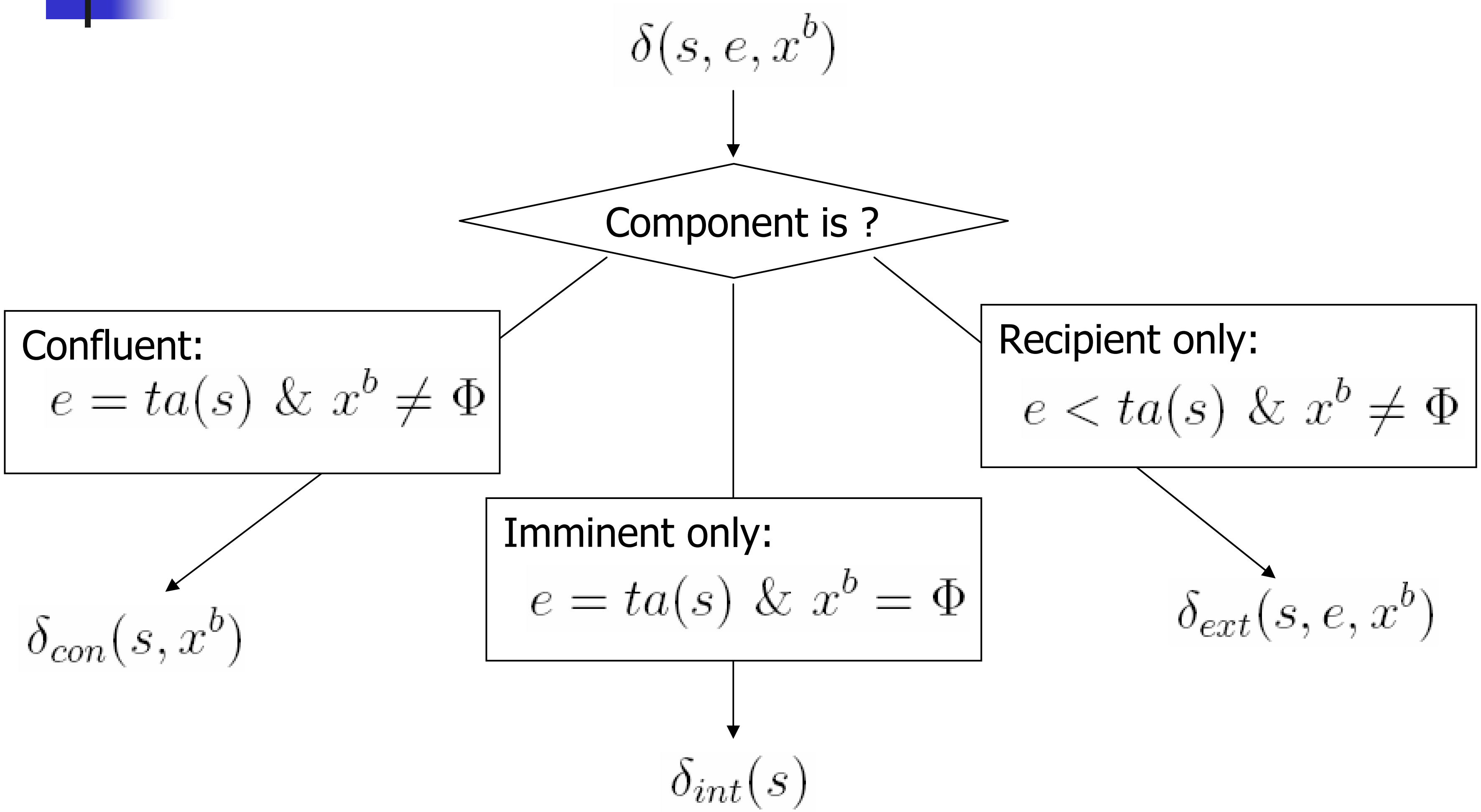
$$(s'_d, e'_d) = (\delta_{int,d}(s_d), 0) \qquad \text{for } d \in INT'(s),$$

$$(s'_d, e'_d) = (\delta_{ext,d}(s_d, e_d + ta(s), x_d^b), 0) \qquad \text{for } d \in EXT'(s),$$

$$(s'_d, e'_d) = (\delta_{con,d}(s_d, x_d^b), 0) \qquad \text{for } d \in CONF'(s),$$

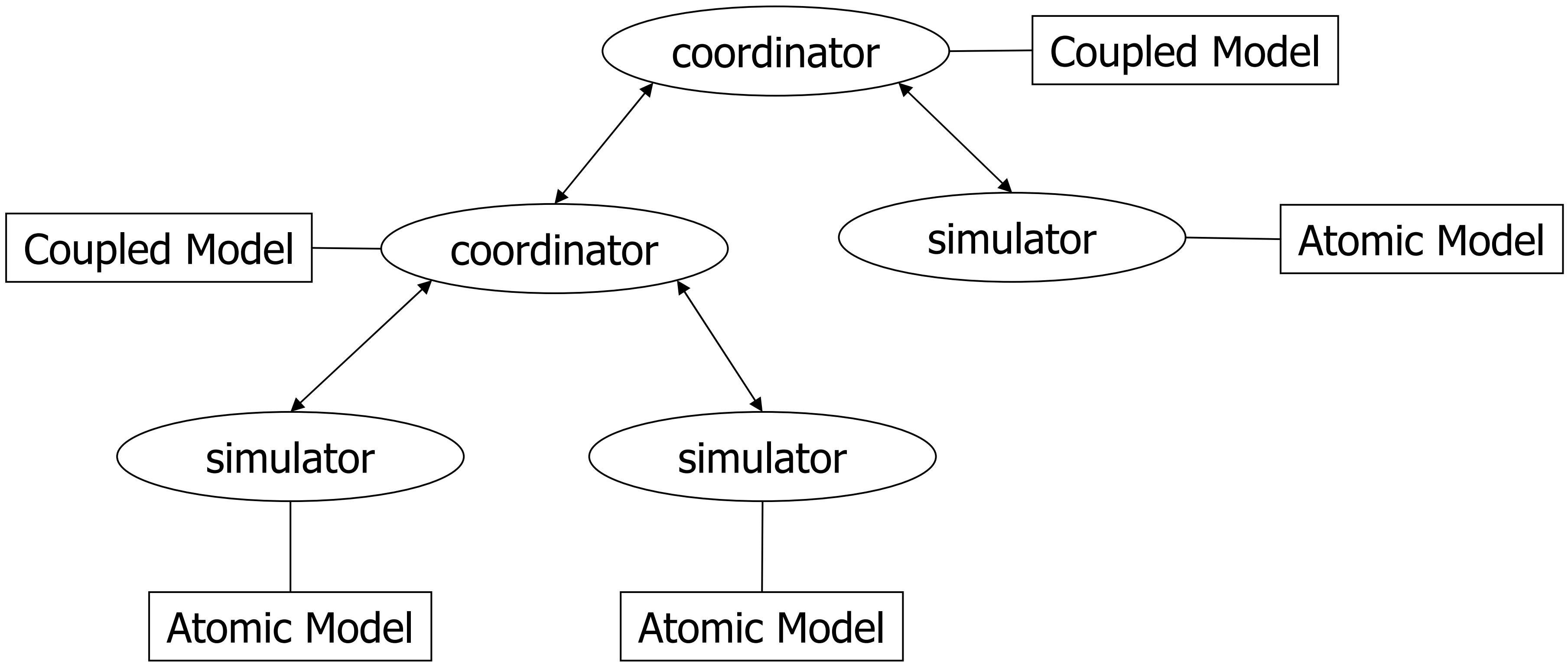
$$(s'_d, e'_d) = (s_d, e_d + ta(s)) \qquad \text{otherwise}$$

Generic Transition Function

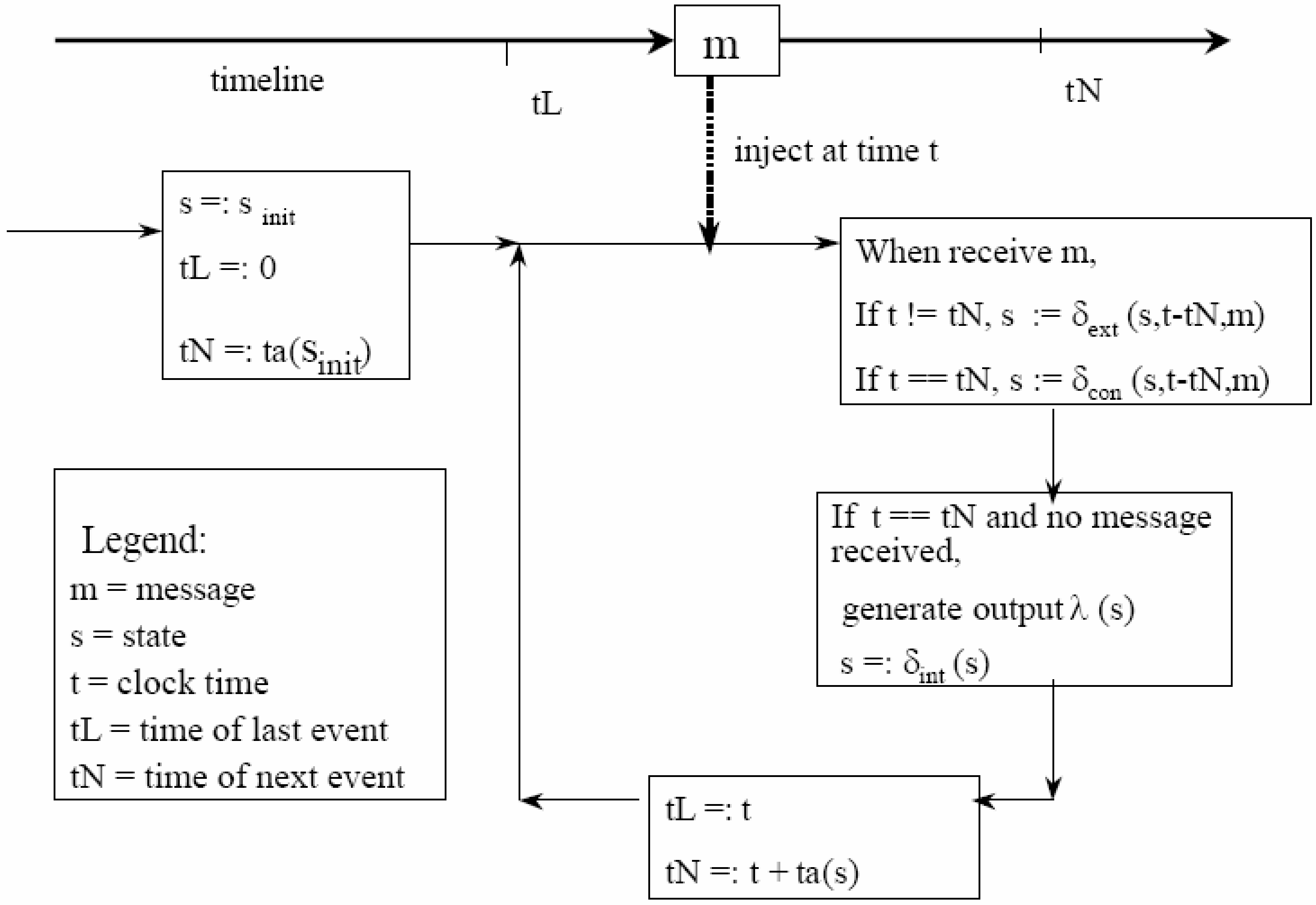


- *Classic DEVS* quick review
- Why *Parallel DEVS*
- *Parallel DEVS* Formalism
  - Atomic Model
  - Coupled Model
  - Closure under Coupling
- *Parallel DEVS* Simulation Protocol
- DEVSJAVA

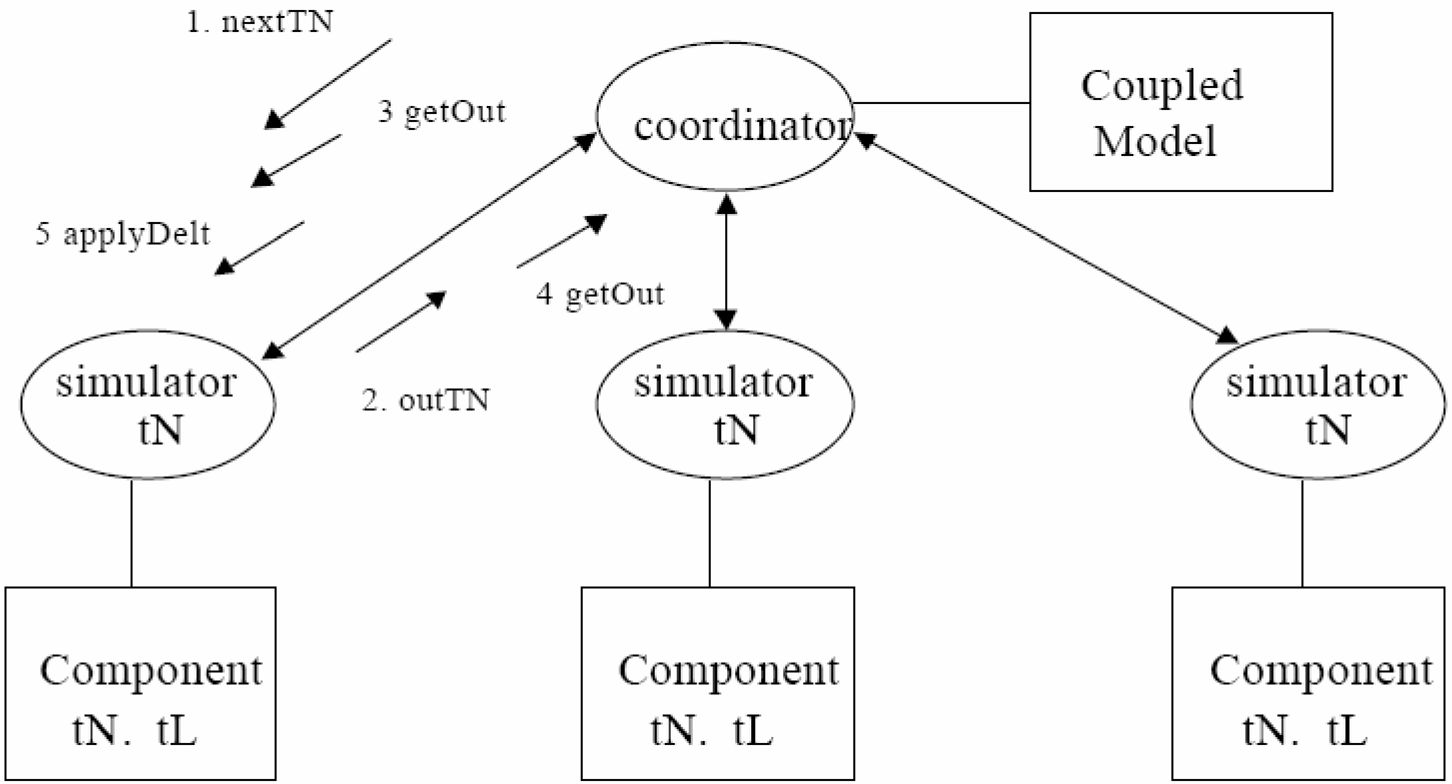
Hierarchical Model



Atomic Model Simulator



Coupled Model Simulator



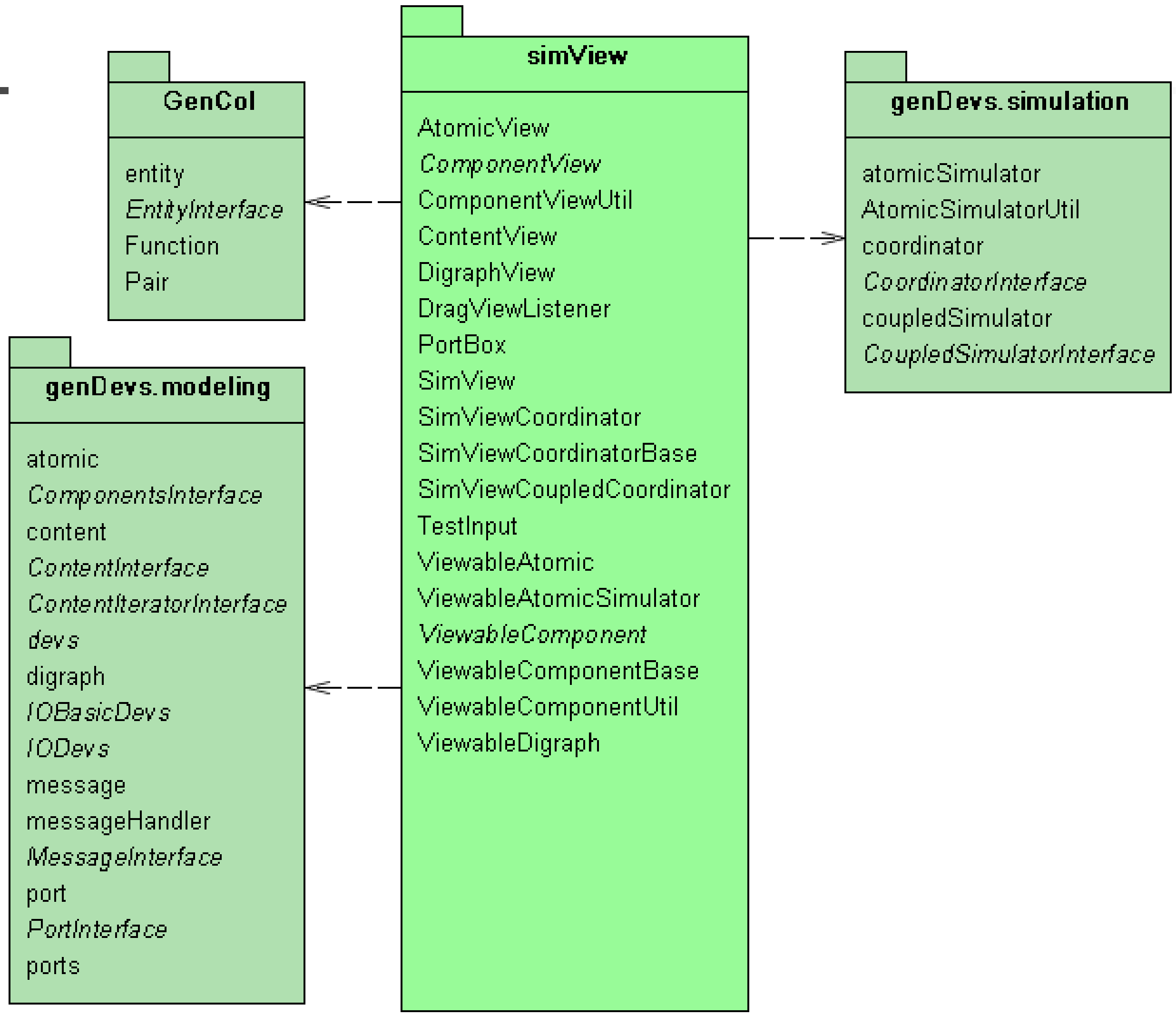
After each transition
tN = t + ta(), tL = t

- *Classic DEVS* quick review
- Why *Parallel DEVS*
- *Parallel DEVS* Formalism
  - Atomic Model
  - Coupled Model
  - Closure under Coupling
- *Parallel DEVS* Simulation Protocol
- DEVSJAVA

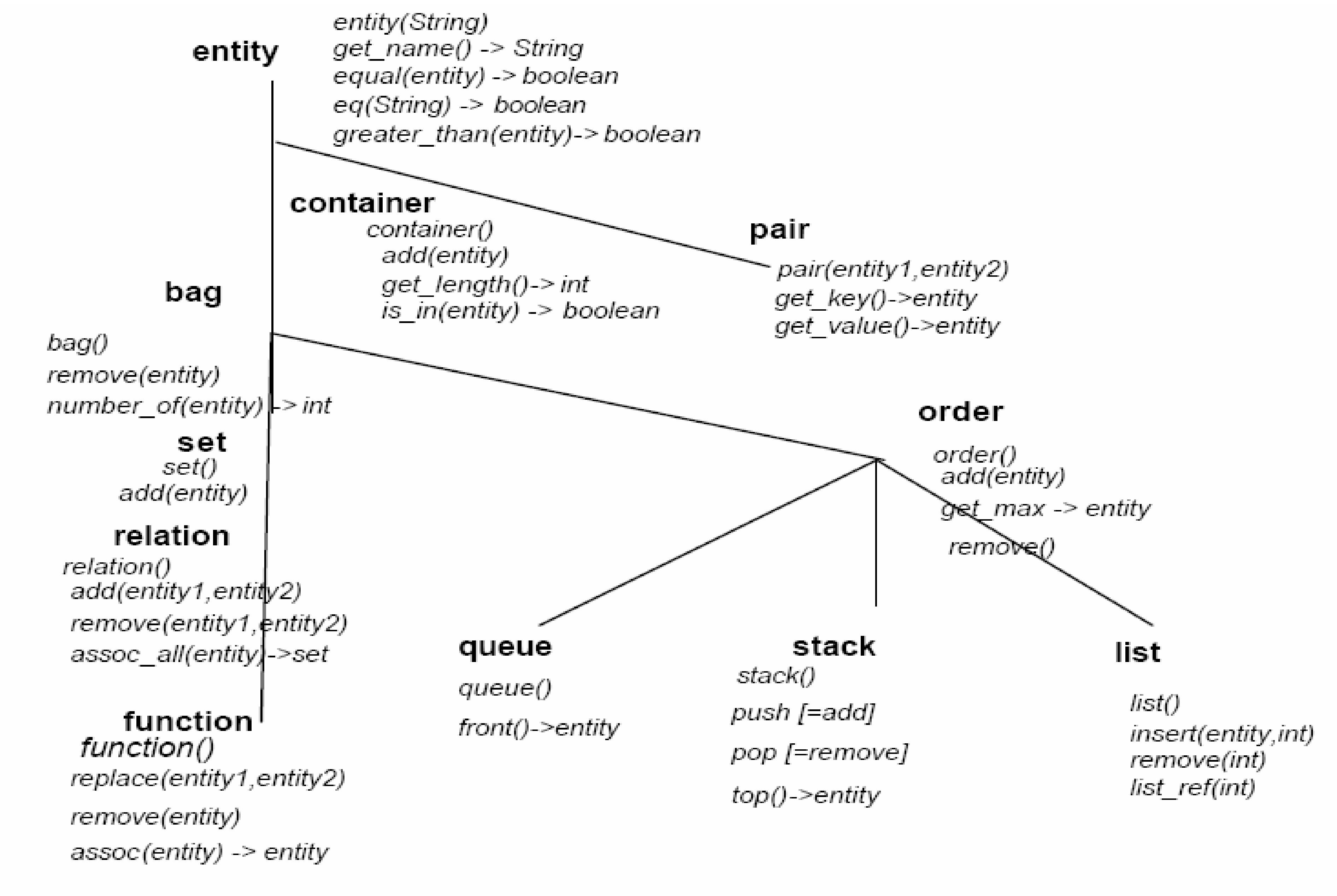
DEVSJAVA

- DEVS-based, Object-Oriented Modeling and Simulation environment.
- Written in Java and supports parallel execution on a uni-processor
- Simulation Viewer for animating simulation in V2.7

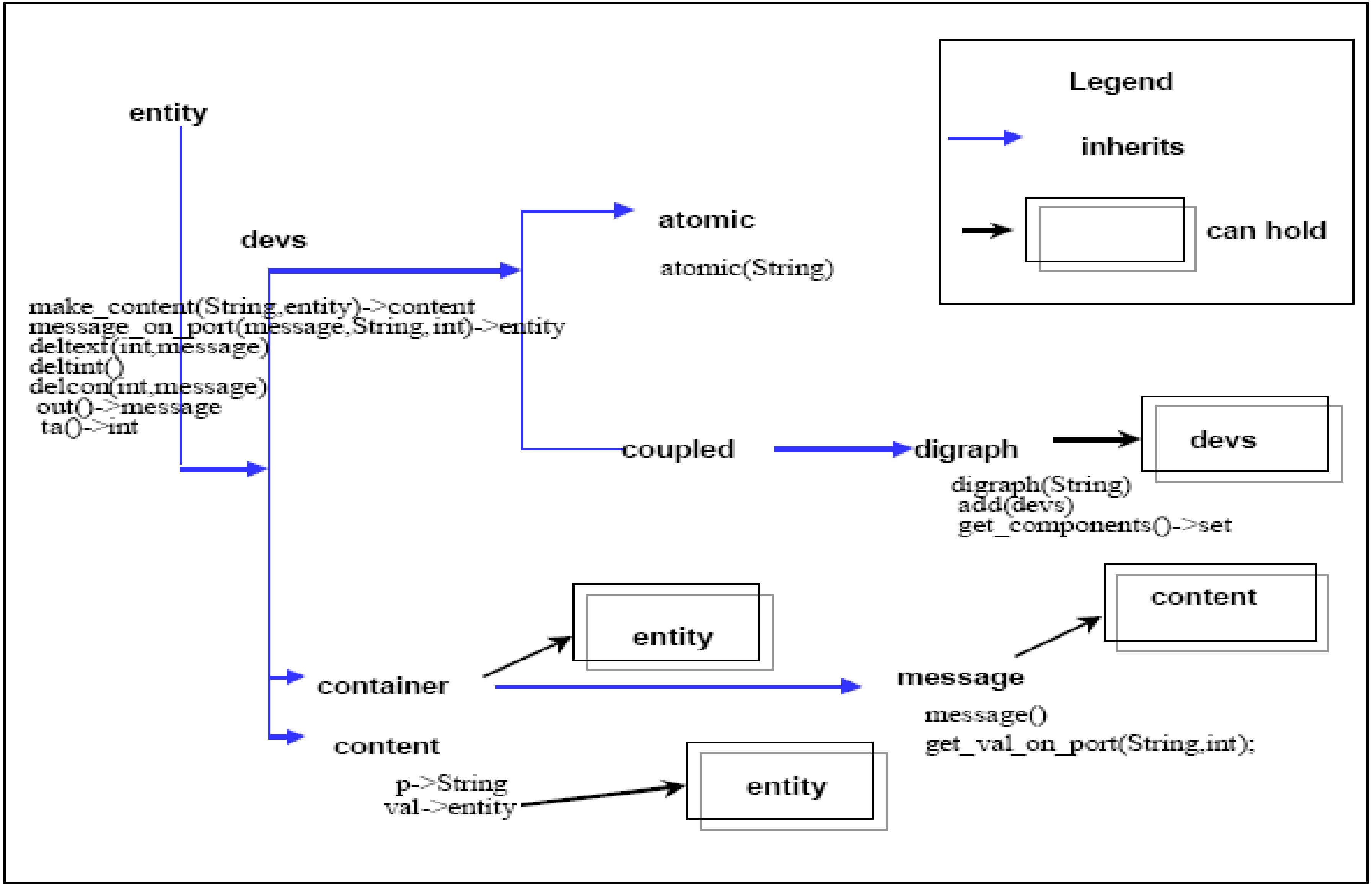
Package Diagram



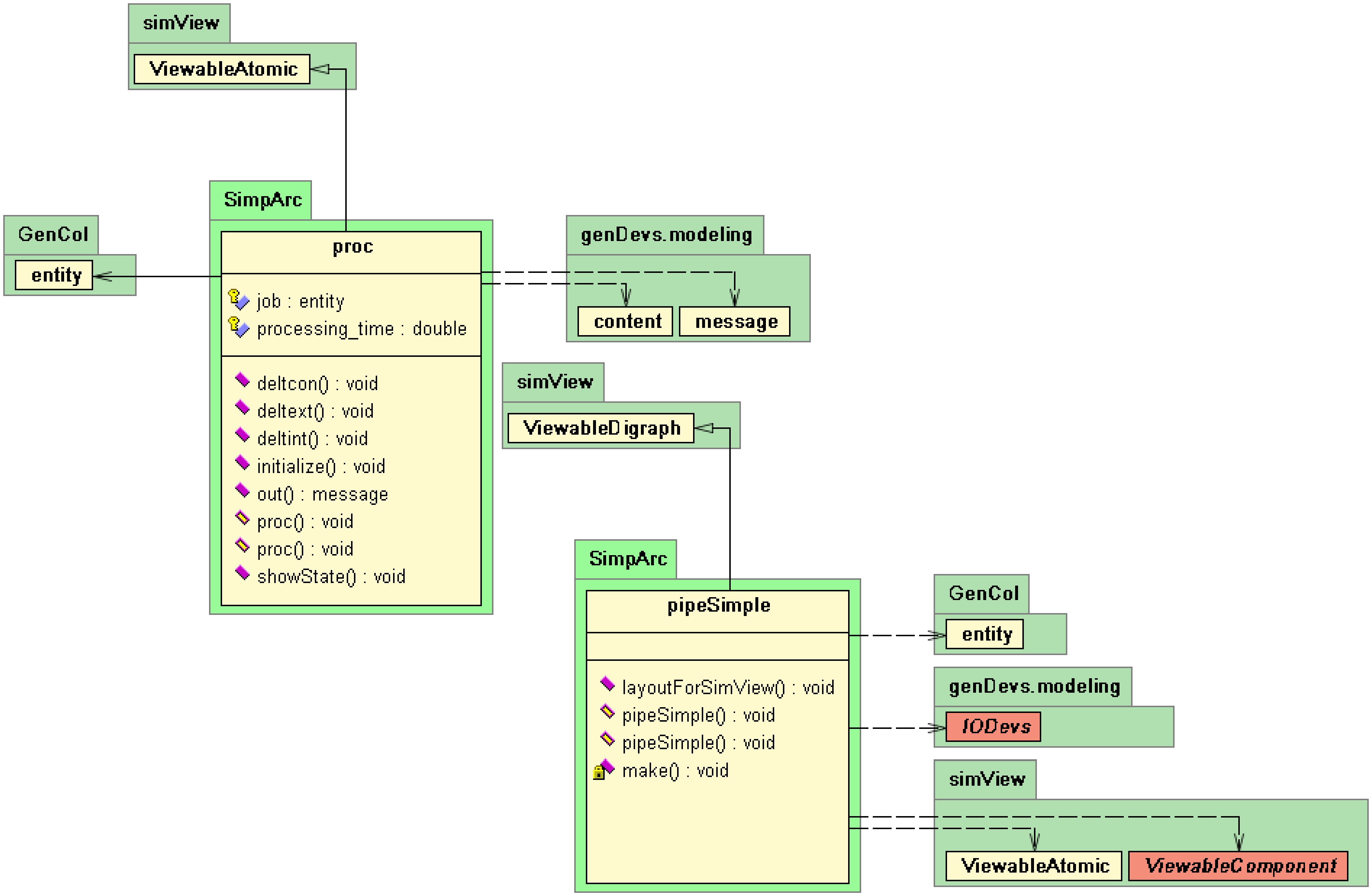
DEVSJAVA Class hierarchy of container classes



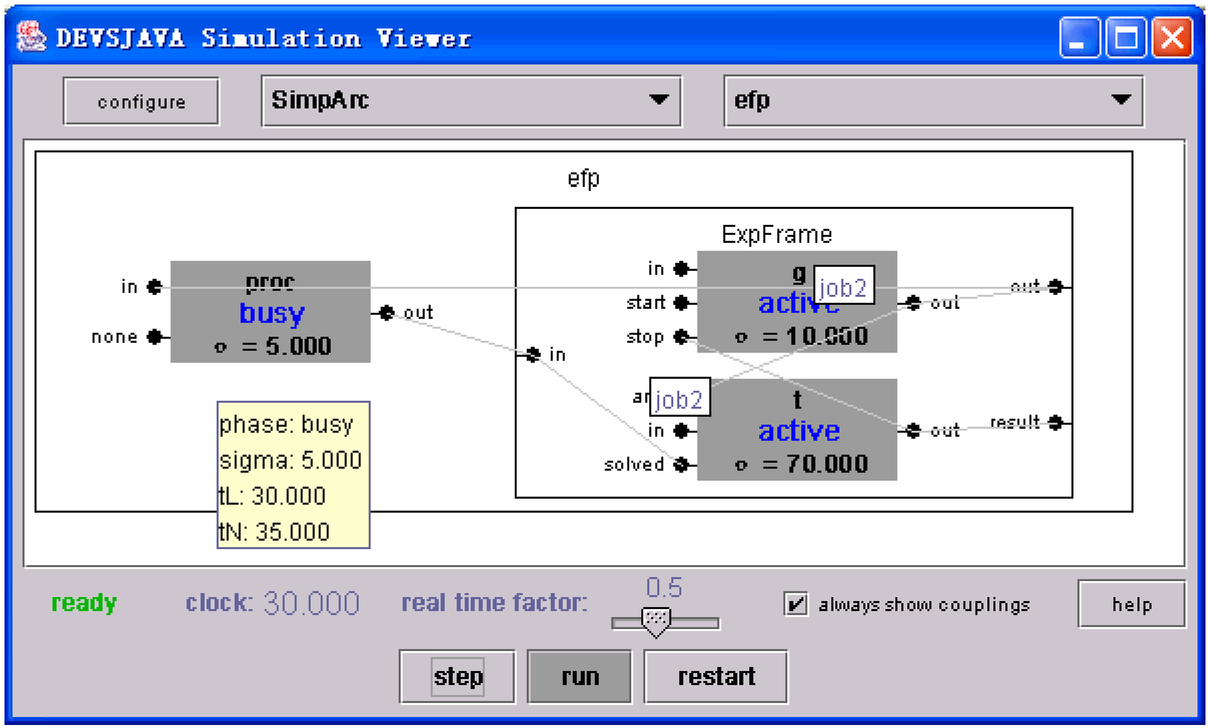
DEVSJAVA Class hierarchy of DEVS classes



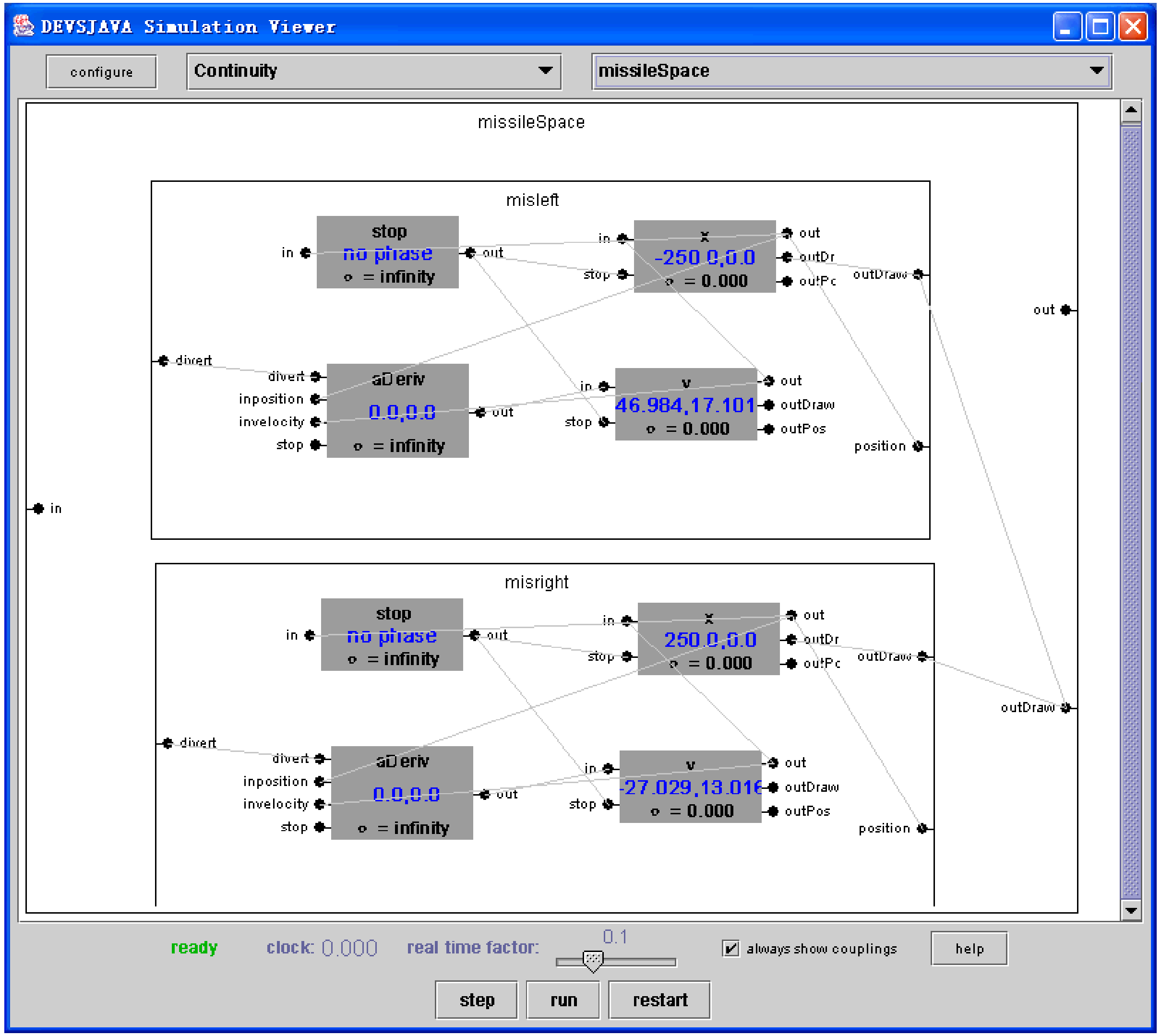
Simple Pipeline in DEVSJAVA

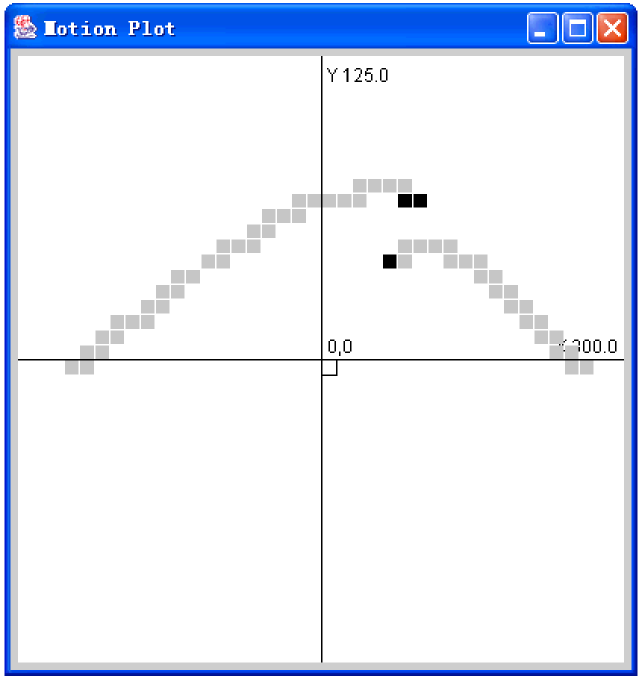


Simulation Viewer



More complicated example





Sources

- DEVSJAVA - Modeling and Simulation environment for developing DEVS-based models by Hessam Sarjoughian, Bernard Zeigler.
  - http://www.acims.arizona.edu/SOFTWARE/software.shtml#DEVSJAVA (need a license)

Question?