SEN9110 Simulation Masterclass Lecture 5. DEVS extensions

Prof.dr.ir Alexander Verbraeck
a.verbraeck@tudelft.nl

Brightspace: SEN9110

Agenda of this lecture

- Questions about paper last class
- Answer to Wymore's model question of lecture 2.2
- DEVS Formalism [lecture 2.2, last topic] (Zeigler)
- Port-based DEVS (Zeigler)
- Hierarchical DEVS (Zeigler)
- Other DEVS extensions (Zeigler, Wainer, e.a.)

Papers to read

- B.P. Zeigler. DEVS Today: Recent Advances in Discrete Event-Based Information Technology. Proceedings of the 11TH IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS'03). IEEE, 2003.
  Retrieved from: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1240652

- B.P. Zeigler. Embedding DEV&DESS in DEVS. DEVS Integrative M&S Symposium (DEVS 2006). Huntsville, Alabama, USA, April 2-6, 2006
  Retrieved from: http://acims.asu.edu/wp-content/uploads/2012/02/Embedding-DEV&DESS-in-DEVS.pdf via official ACIMS web page at https://acims.asu.edu/journal-conference-and-workshop-articles/

DEVS Atomic Model Specification

DEVS = $< X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, ta, \lambda >$

X:	a set of input events
Y:	a set of output events
S:	a set of states
$ta: S \rightarrow R^{+}_{0,\infty}$	time advance function
$\delta_{int}: S \rightarrow S$	internal transition function
$\delta_{ext}: Q \times X^b \rightarrow S$	external transition function
$\delta_{con}: Q \times X^b \rightarrow S$	confluent transition function

where X^b is a set of bags over elements in X

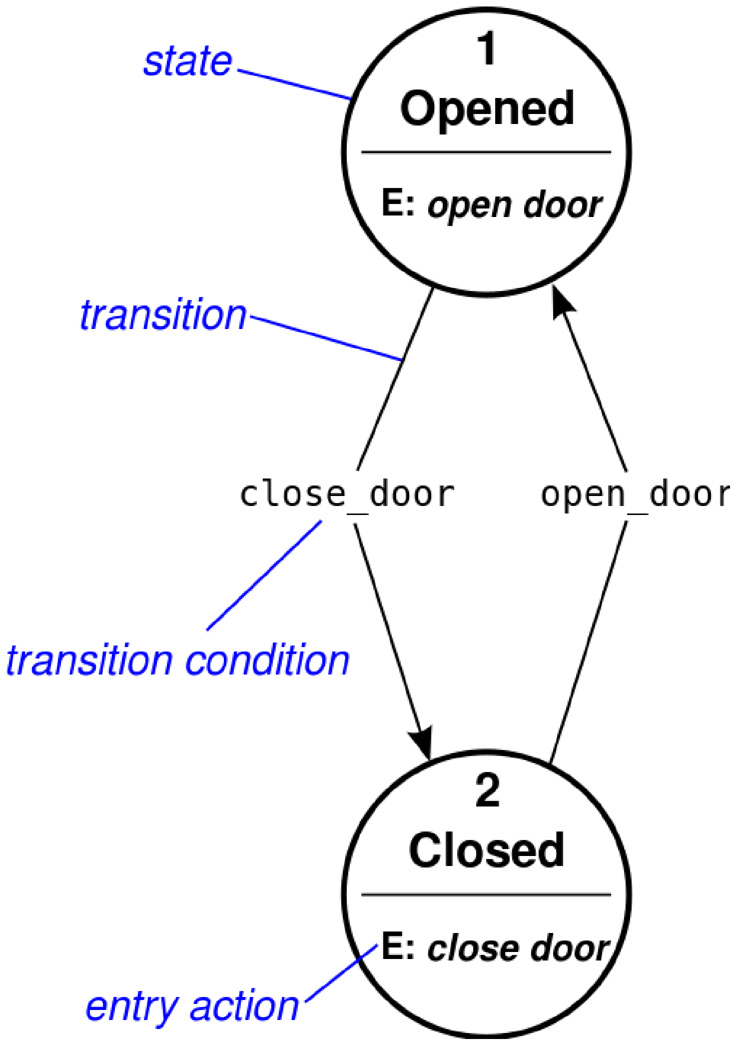
$$Q = \{(s,e) \mid s \in S, 0 \leq e \leq ta(s)\}$$

$\lambda: S \rightarrow Y$	output function

Note: DEVS is an extension of a Mealy machine (1955)

Ref: Prof. B. Zeigler (ACIMS)

State diagram

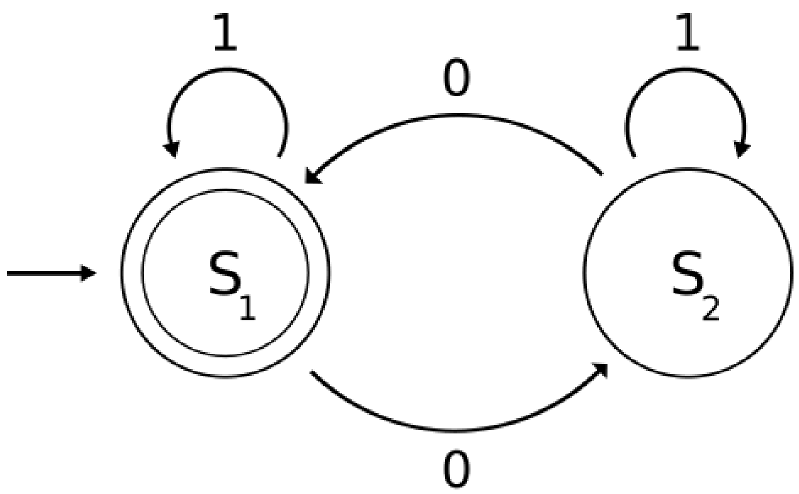


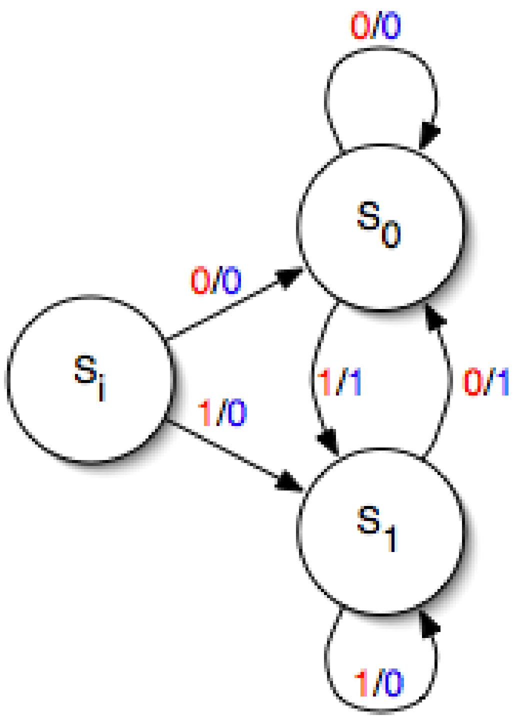
Simple example:

A door that can be opened and closed

Source: Wikipedia - State Diagram

Moore and Mealy Machines





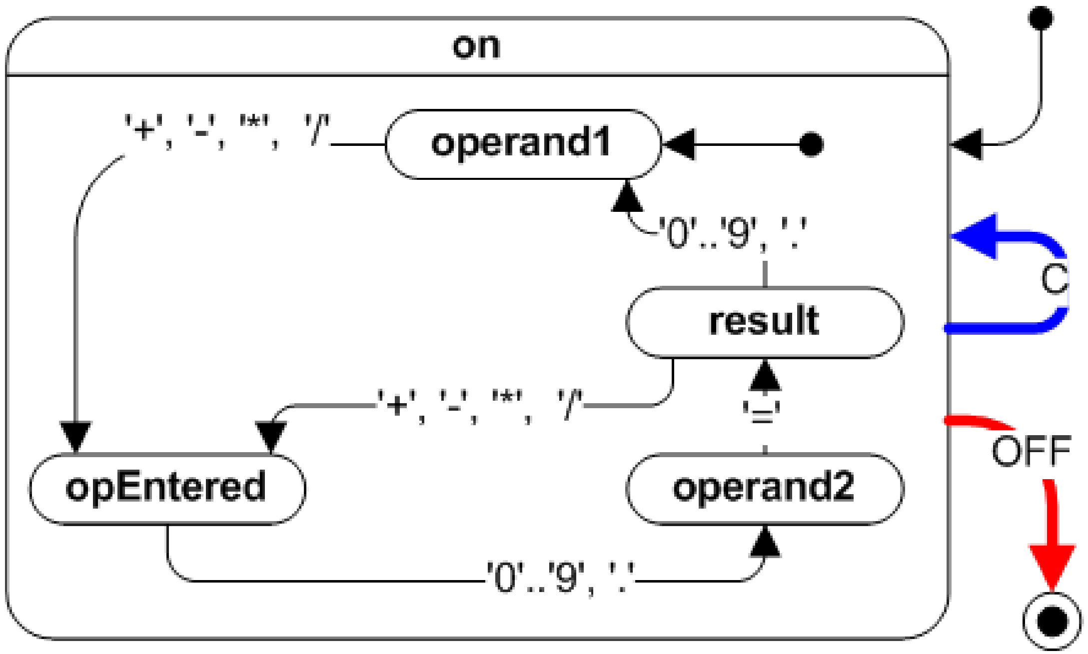
Moore Machine. Edge labeled with i=input. S_1 is the start state (arrow from outside) and the accept state (double circle). The language recognized by *M* is 1*(0 (1*) 0 (1*))*

Mealy Machine. Edge labeled with i/o, i=input, o=output. Suppose S_i is the start. ***What is the input/output relation?***

Source: Wikipedia - State Diagram

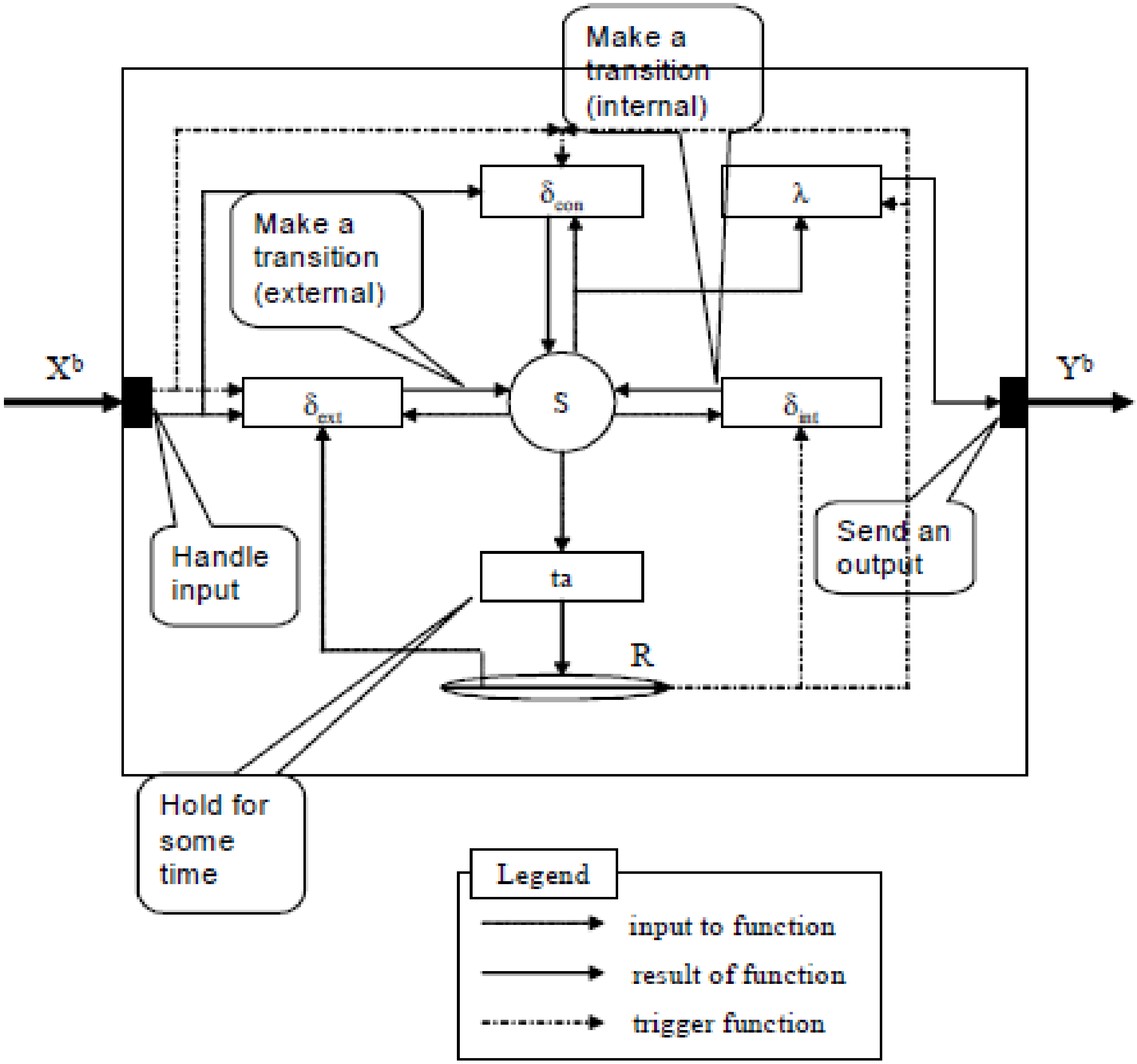
Finite State machine

Is this diagram correct?
(and if nesting is possible?)

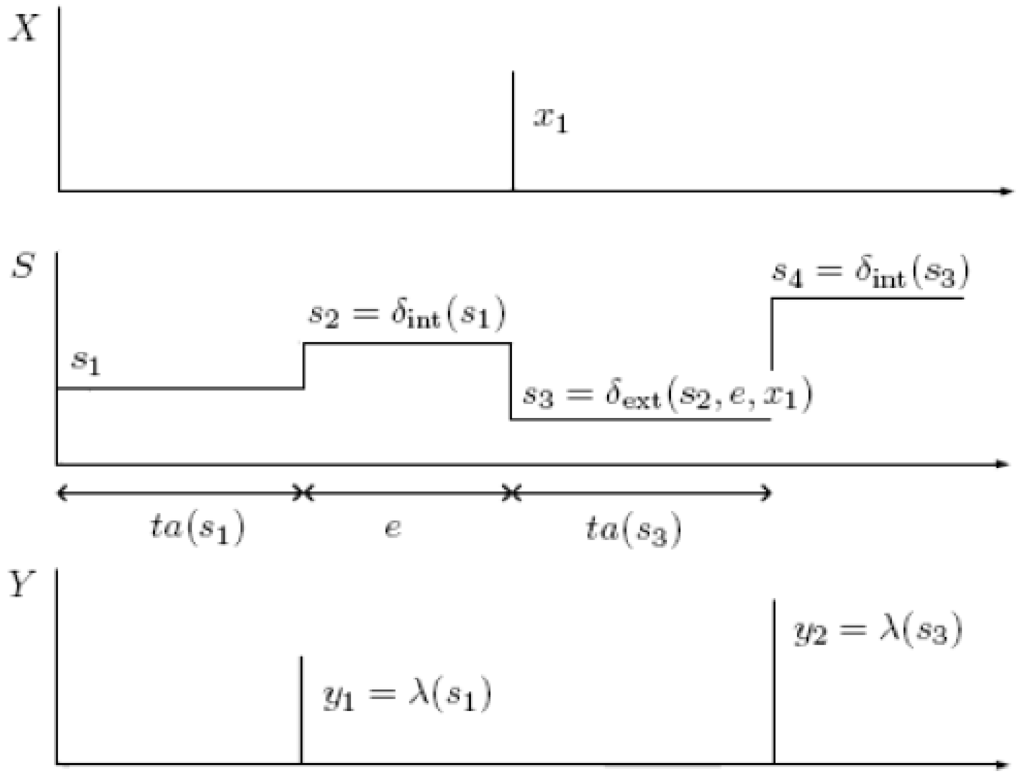


Source: Wikipedia - UML State Machine

DEVS operation: Moore plus time



DEVS model trajectories



$$M = \langle X, Y, S, \delta_{int}(s), \delta_{ext}(s, e, x), \lambda(s), ta(s) \rangle$$

T. Beltrame, F.E. Celllier. Quantised State System Simulation in Dymola / Modelica using the DEVS Formalism. Modelica 2006 Workshop, Modelica Association, 2006

Port-based DEVS

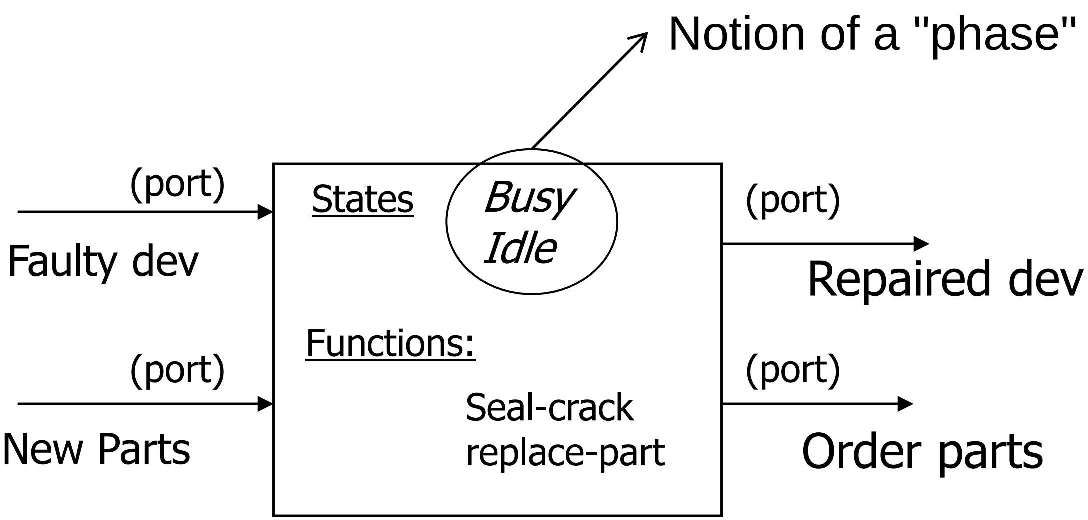
DEVS = $< X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, ta, \lambda >$

$X = \{(p, v) \mid p \in IPorts, v \in X_p\}$
IPorts is the set of input ports
X_p is the set of values for the input ports

$Y = \{(p, v) \mid p \in OPorts, v \in Y_p\}$
OPorts is the set of output ports
Y_p is the set of values for the output ports

Example (Atomic Model)

Repair Shop: A repair shop receives devices from a customer. Each device requires a certain length of time for it to be repaired. Upon the repair of the device, it is sent to customer



Ref: Prof. B. Zeigler (ACIMS)

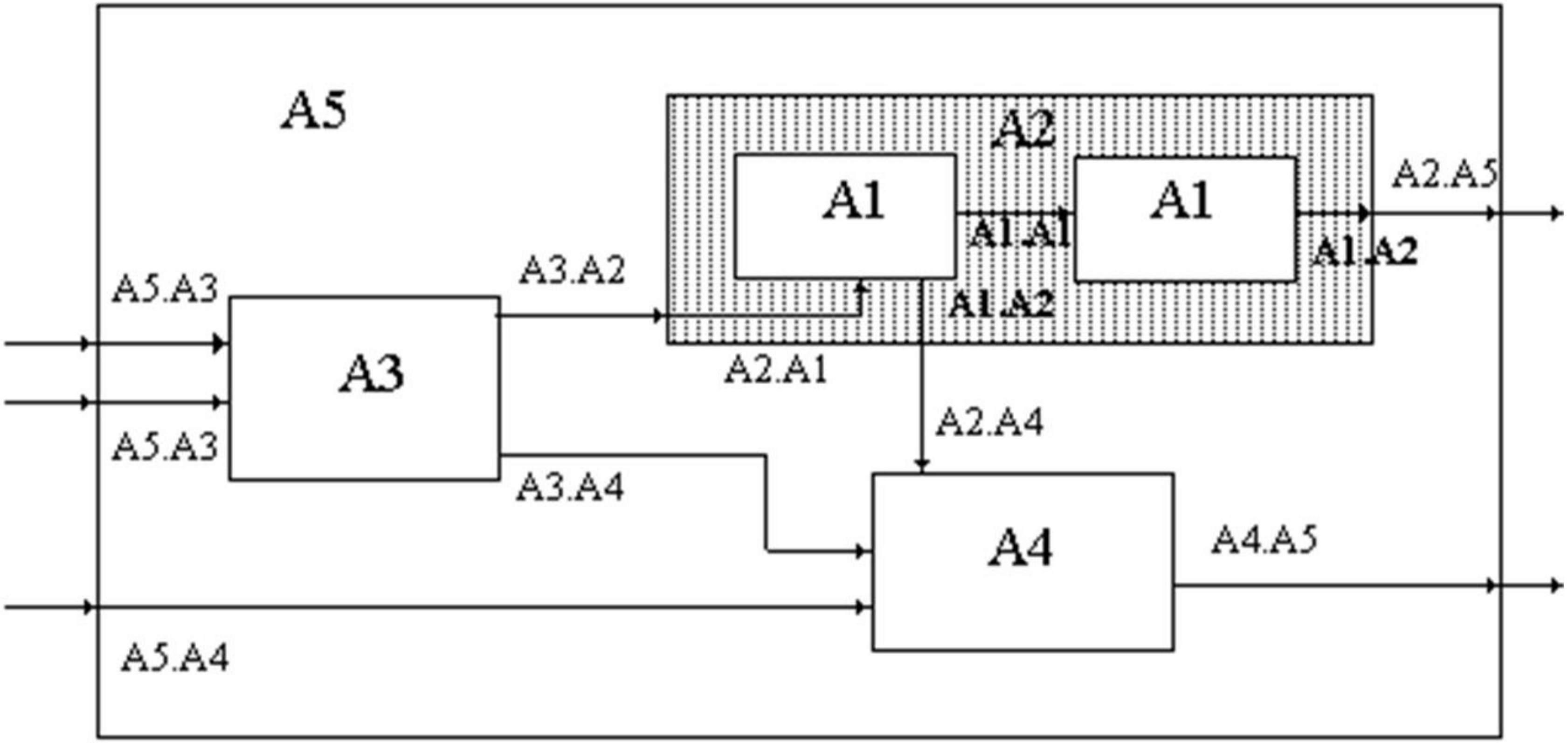
DEVS models

Two Types:

- **Atomic Model**
  Basic models (atomic) from which larger ones (coupled) are built

- **Coupled Model**
  How these models are connected together in hierarchical fashion using port-based DEVS

Coupled Model Specification

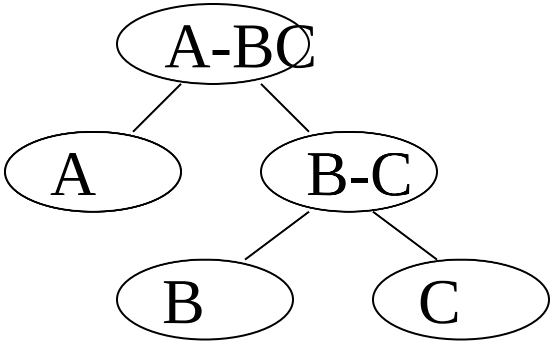


Atomic Models:
M = <X, S, Y, δ_{int}, δ_{ext}, λ, ta>
Coupled models:
CM = <X, Y, D, {M_i}, EIC, EOC, IC, select>

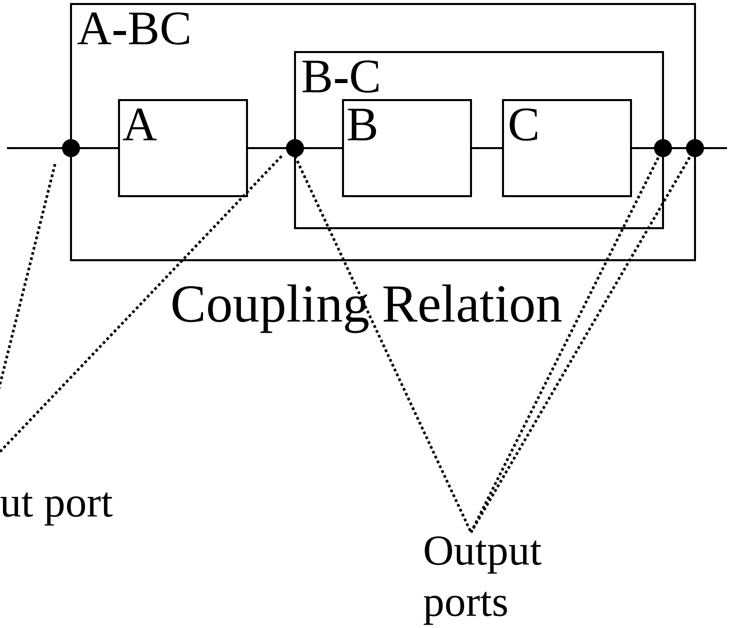
Source: Gabriel Wainer, Dagstuhl workshop 04041

Coupling Relation

Coupling:



Hierarchal tree



Ref: Prof. B. Zeigler (ACIMS)

Closure Under Coupling

HD: CM = <**X**, **Y**, **D**, {$\mathbf{M}_i$}, **EIC**, **EOC**, **IC**, select>

→

DEVS
$< \mathbf{X}, \mathbf{S}, \mathbf{Y}, \delta_{int}, \delta_{ext}, \delta_{con}, ta, \lambda >$

DEVS
$< \mathbf{X}, \mathbf{S}, \mathbf{Y}, \delta_{int}, \delta_{ext}, \delta_{con}, ta, \lambda >$

Every DEVS coupled model has a DEVS Basic equivalent

Ref: Prof. B. Zeigler (ACIMS)

Coupled Model

- Atomic Models coupled together. Coupled Model tells how to connect several atomic models (or lower level coupled models) together

- It contains:
  - Set of components(lower level models)
  - Set of it's own input and output ports
  - The coupling specifications between models and to/from the outside

Ref: Prof. B. Zeigler (ACIMS)

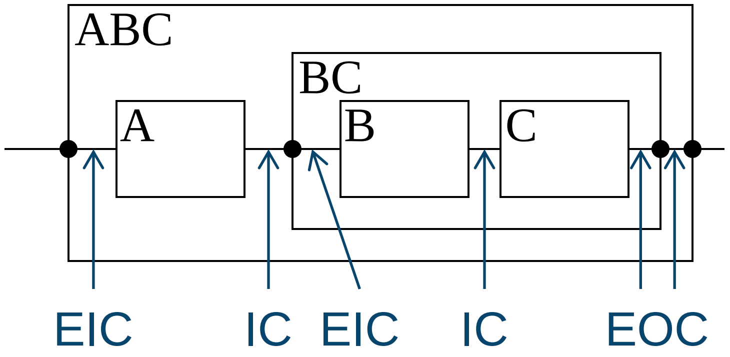
Coupled Model

Elements of a coupled model:

- Components $<M_i>$ where $\boldsymbol{i} \in \boldsymbol{D}$. D is a set of model names or identifiers

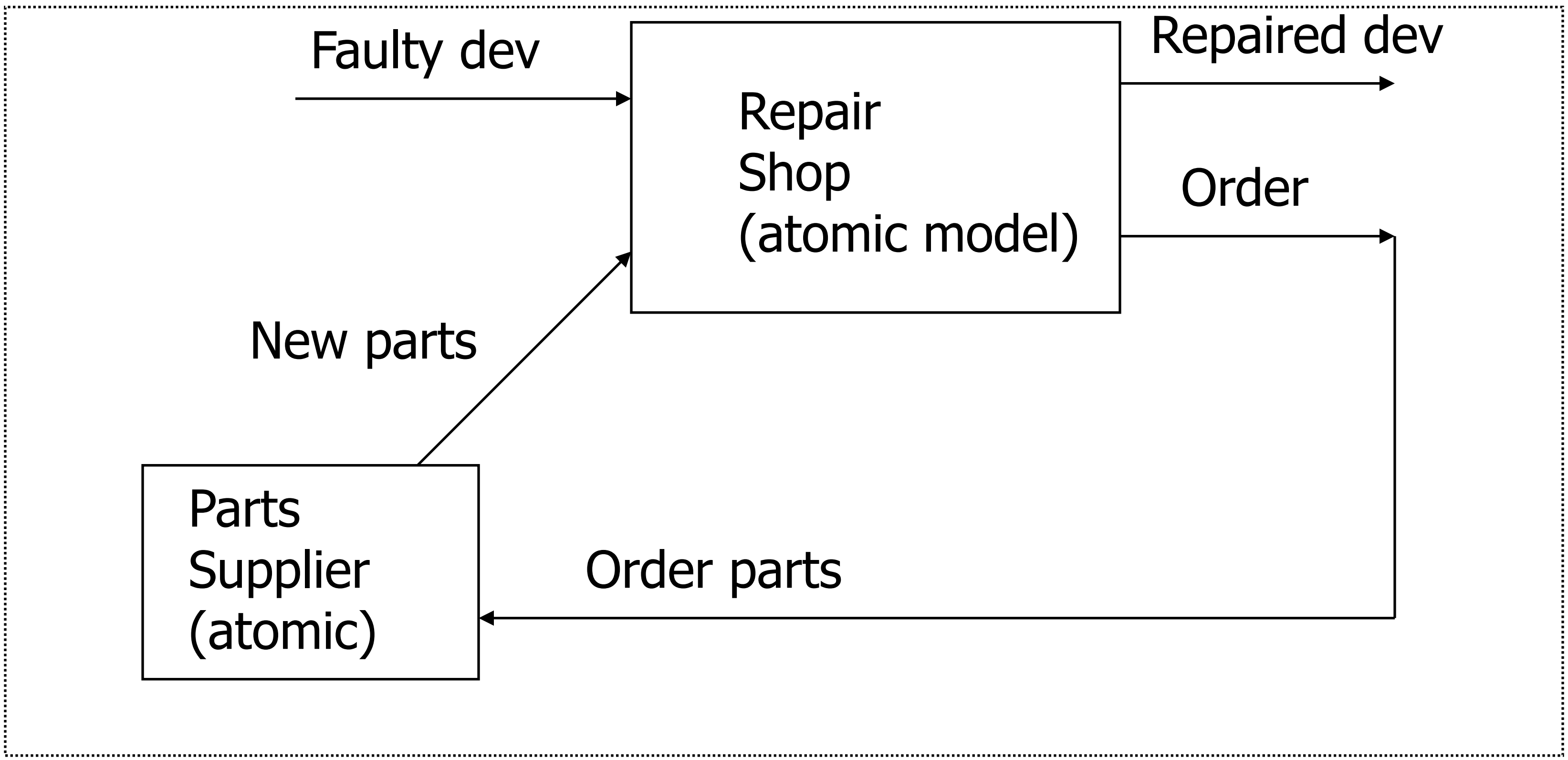
Interconnections

- Internal Couplings $<IC>$
- External Input Couplings $<EIC>$
- External Output Couplings $<EOC>$



Example (Coupled model)

Repair shop on a higher systemic level



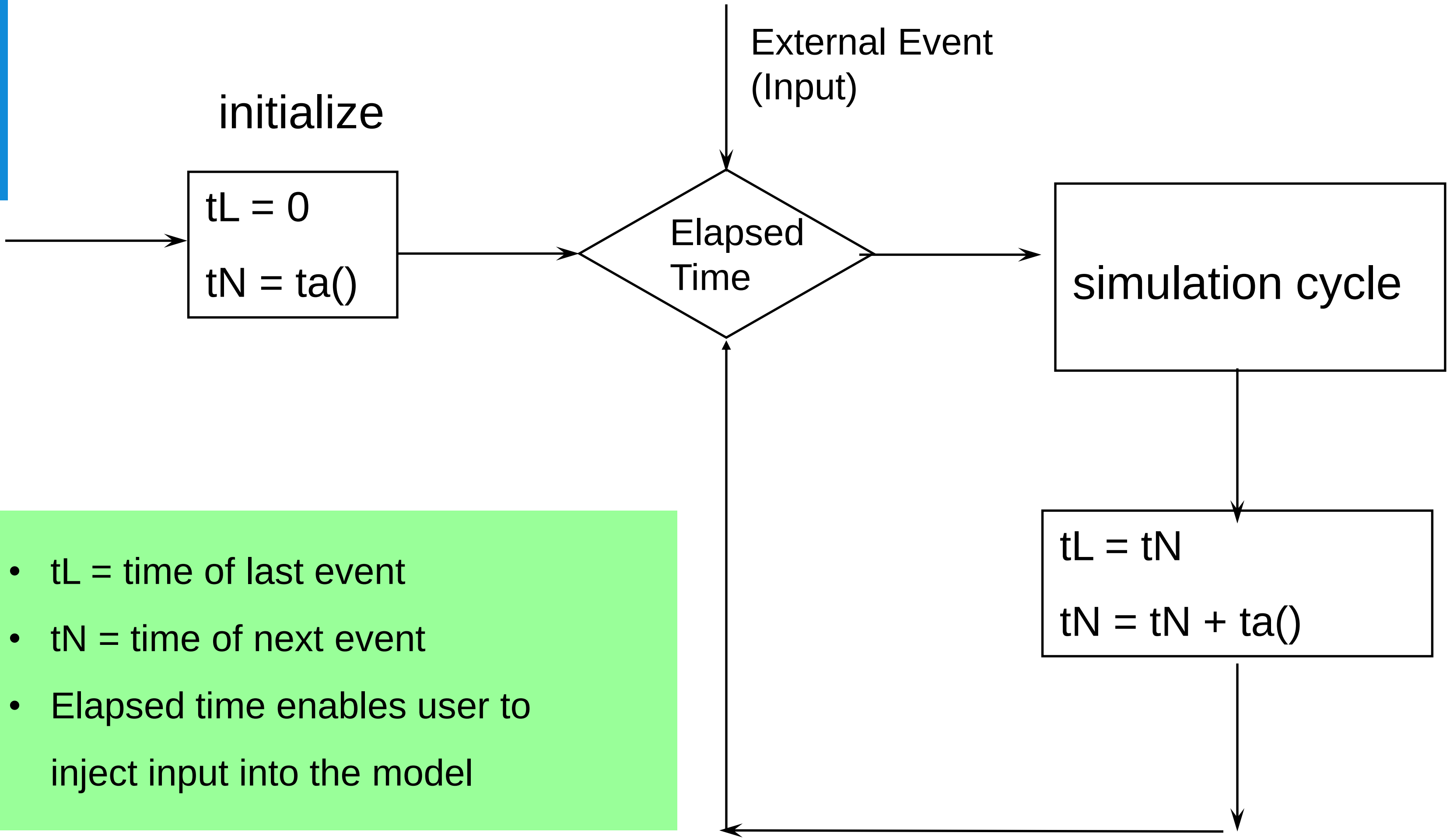
Ref: Prof. B. Zeigler (ACIMS)

Simulator/Simulation Concepts

- **Simulator** is responsible for executing a model's dynamics (represented as instructions) in a given formalism.
- **Abstract simulator** is a characterization of what needs to be done in executing a model's instructions
  - atomic simulator
  - coupled simulator
- **Simulation engines** enforce particular realizations of an abstract simulator
- Simulations can be **executed**:
  - Sequential
  - Parallel
  - Distributed (sequential/parallel)

Ref: Prof. B. Zeigler (ACIMS)

Atomic DEVS Simulator



- tL = time of last event
- tN = time of next event
- Elapsed time enables user to inject input into the model

Ref: Prof. B. Zeigler (ACIMS)

The DEV&DESS Simulator

- How to combine continuous (DESS) components (solved by a numerical integrator)
- and discrete event (DEVS) components (scheduled by a DEVS simulator)
- into one coupled model and preserve behavioral aspects of the components?

Many other extensions

- Parallel DEVS (PDEVS)
  (handling simultaneous events)
- Quantized DEVS (QDEVS)
  (discretizaton of continuous models based on discretized **state**)
- Generalized DEVS (GDEVS)
  (replace continuous behavior by n-th order polynomial segments)
- Dynamic Structure DEVS (DSDEVS)
  (input/output port mapping changes in simulation)
- Cell-DEVS
  (cellular automaton - example of the forest)

Homework

- Read paper about DEVS extensions
- Prepare questions if unclear

- Work on term papers
- Work on simulation package

Sources:

- Gabriel Wainer, Dagstuhl Workshop 04041 contribution (see www.dagstuhl.de)
- Presentations from Zeigler and Sarjoughian (www.acims.arizona.edu) about DEVS
- B.P. Zeigler, H. Praehofer, T.G. Kim, Theory of Modeling and Simulation. 2nd Edition. Academic Press, 2000
- D. Hild, Discrete Event System Specification (DEVS) Distributed Object Computing (DOC) Modeling and Simulation. Ph.D. thesis, University of Arizona, 2000