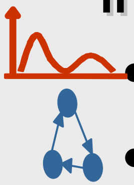
Methodology for Efficient Design of Continuous/Discrete Co-Simulation Tool

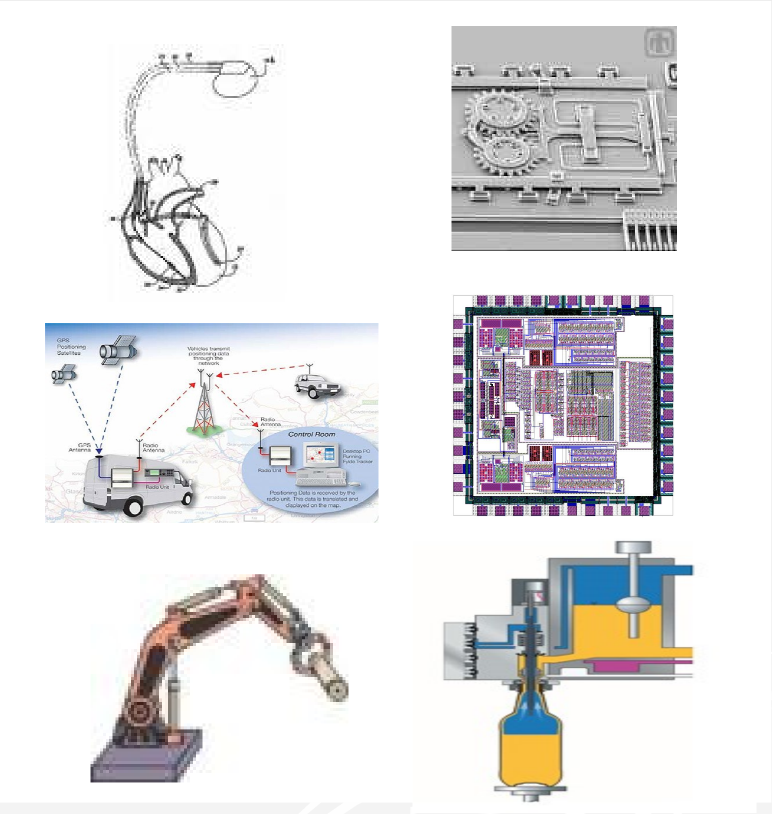
G. Nicolescu, H. Boucheneb,
L. Gheorghe, F. Bouchimma
Ecole Polytechnique de Montréal
Tel : (514) 340 4711 ext 5434
Fax: (514) 340 3240
Email : gabriela.nicolescu@polymtl.ca

ÉCOLE
POLYTECHNIQUE
MONTRÉAL

Continuous/Discrete Systems

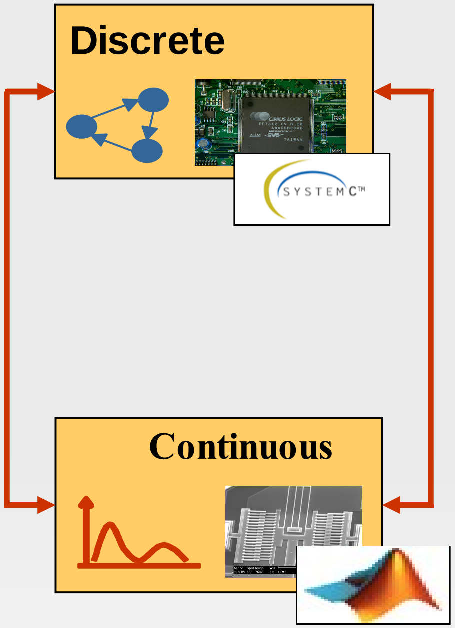


- Two types of components integration
  - Continuous timed
  - Discrete event
- Illustrative systems
  - Electro-mechanical
  - Mixed-signal
  - Radio
  - Hybrid
  - …

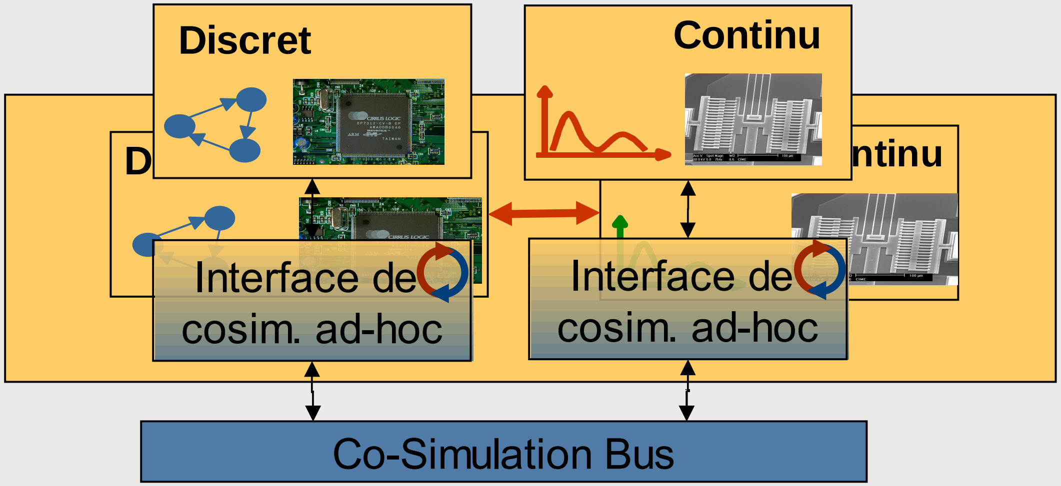


Continuous/Discrete Systems Design

- Collaboration between different teams
- Incremental refinements through different abstraction levels with specific execution models
- Validation requires joint execution of heterogeneous execution models
  - Co-Simulation Technique



Challenges for Continuous/Discrete Co-Simulation



- Defining new tools facilitating cooperation between different teams
  - Exploiting powerful existing tools (Simulink, SystemC, …)
  - Taking into account implementation choices
  - Enabling easy specification, automatic generation for co-simulation interfaces

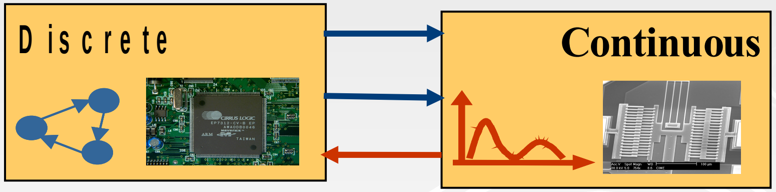
Outline

- Introduction
- Methodology for co-simulation tool design
- Application for CODIS tool design
- Conclusions

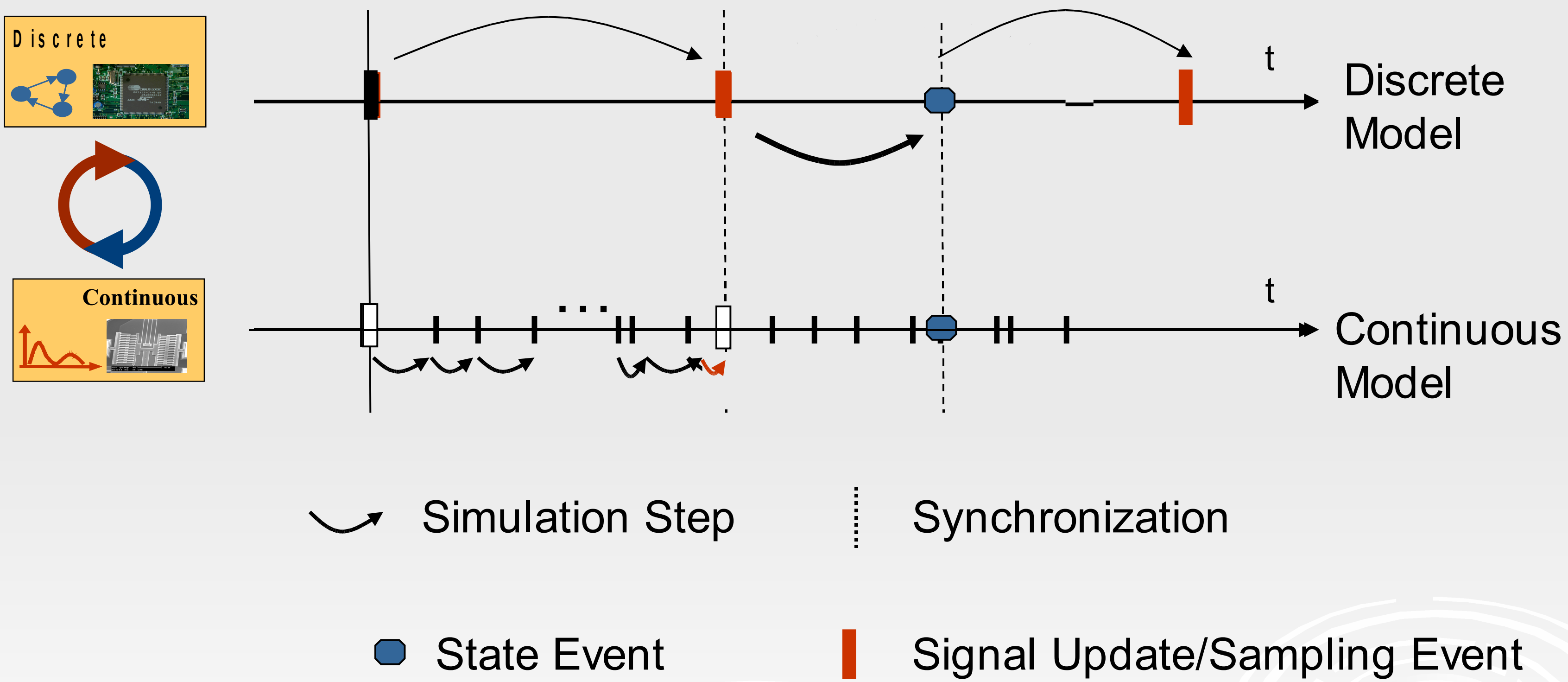
Continuous/Discrete Systems - Heterogeneous Execution Models -

Model \ Concept	Time	Communication means	Processes activation rules
Discrete	It advances discretely	Set of events	Processes are sensitive to events
Continuous	It advances by integration steps (IS)	Piecewise-Continuous signals	Processes are executed at each IS

- Events exchanged between the two models the discrete models
  - Sampling events
  - Update signal events
  - State events



Continuous/Discrete Systems
- Heterogeneous Execution Models -



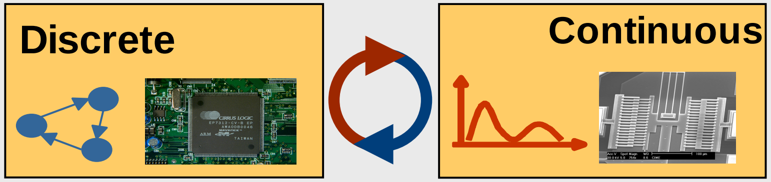
Methodology for Co-Simulation Tool Design

Generic Stage

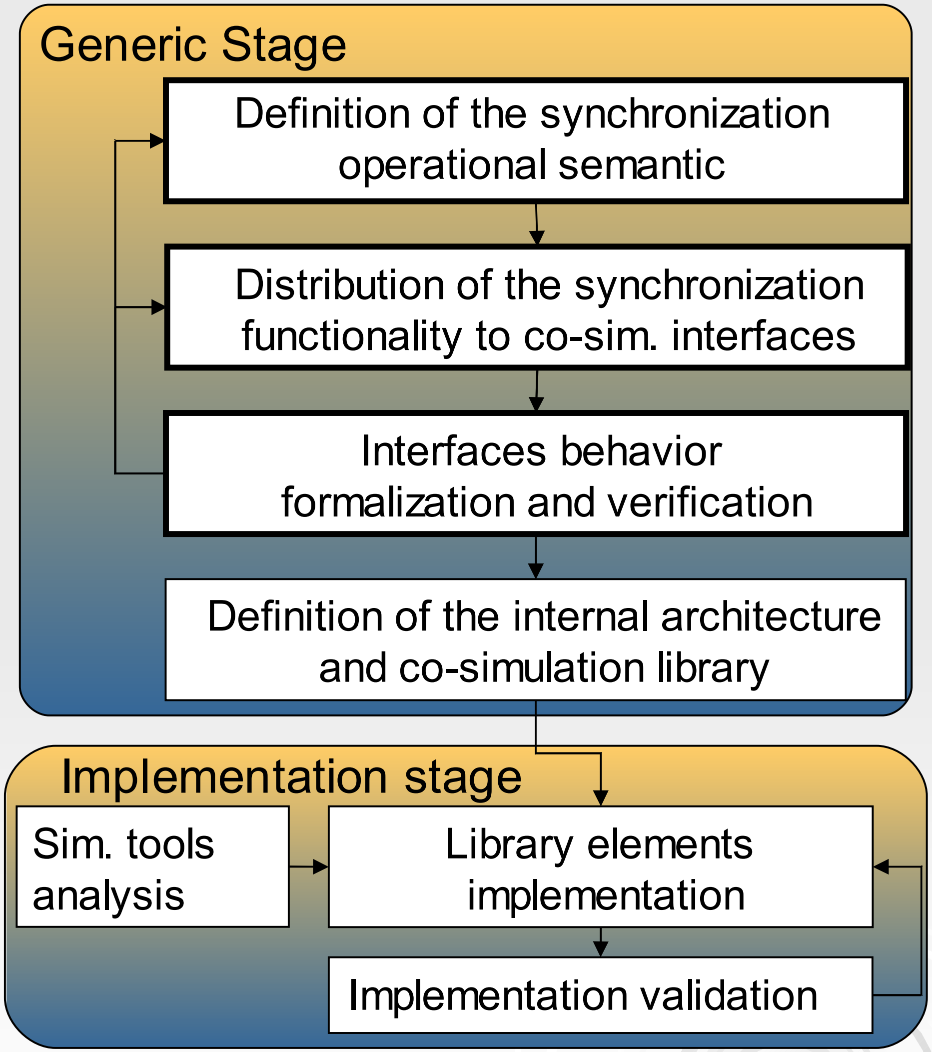
- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

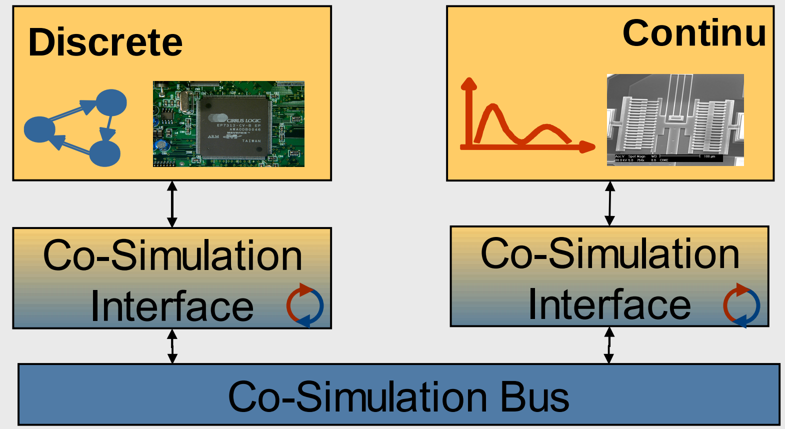
Implementation stage

- Sim. tools analysis
- Library elements implementation
- Implementation validation

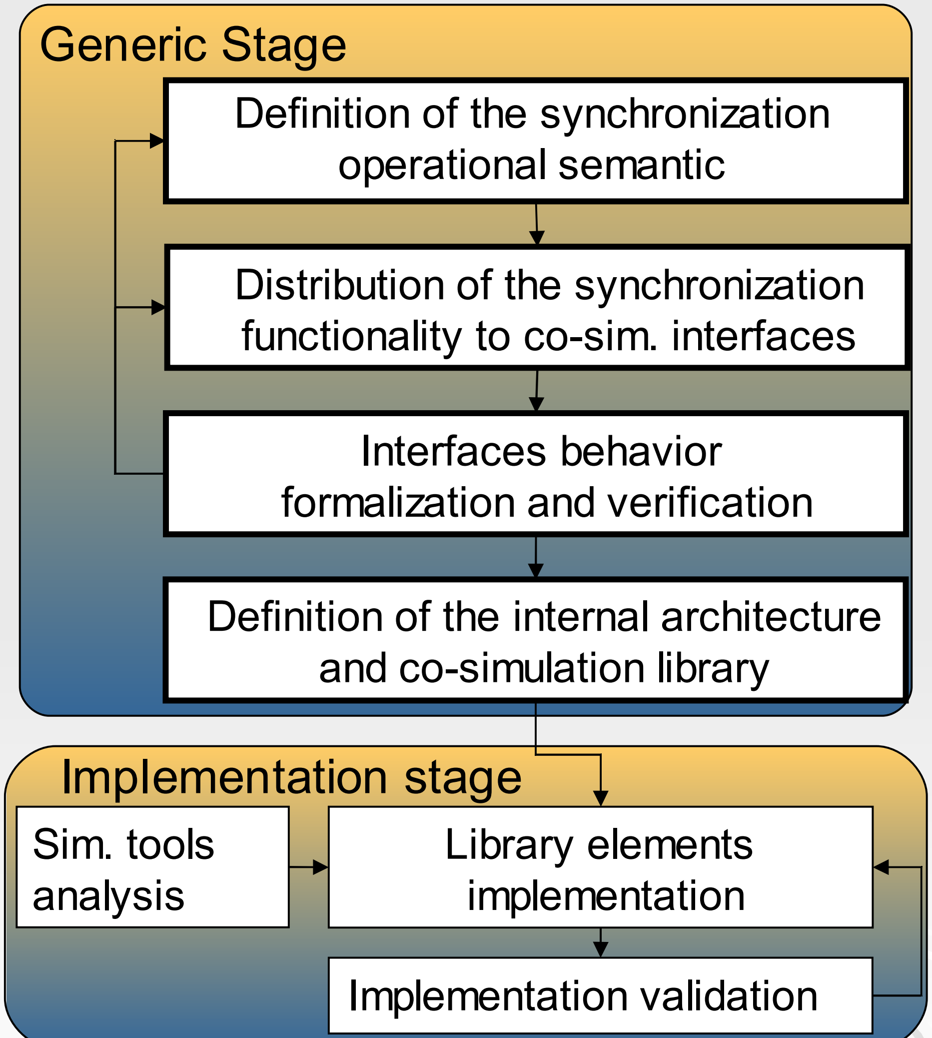


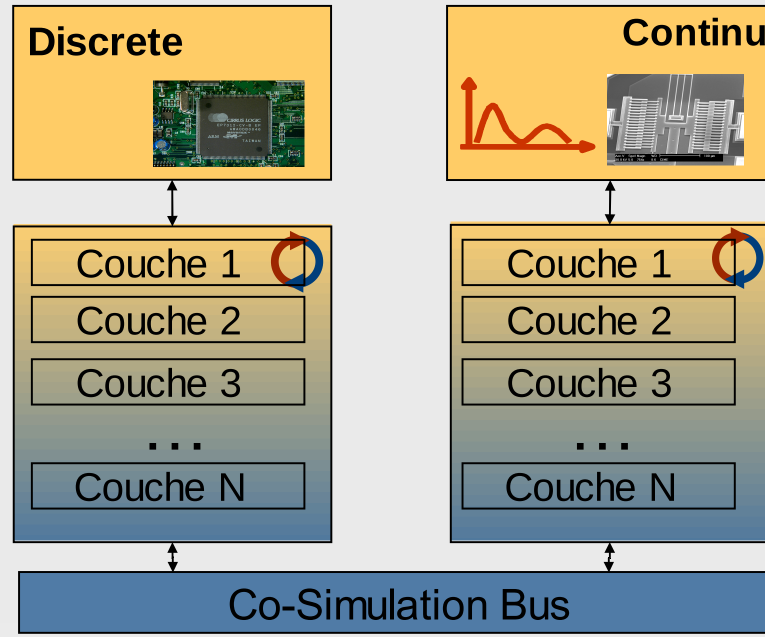
Méthodologie de conception des outils de co-simulation





Méthodologie de conception des outils de co-simulation





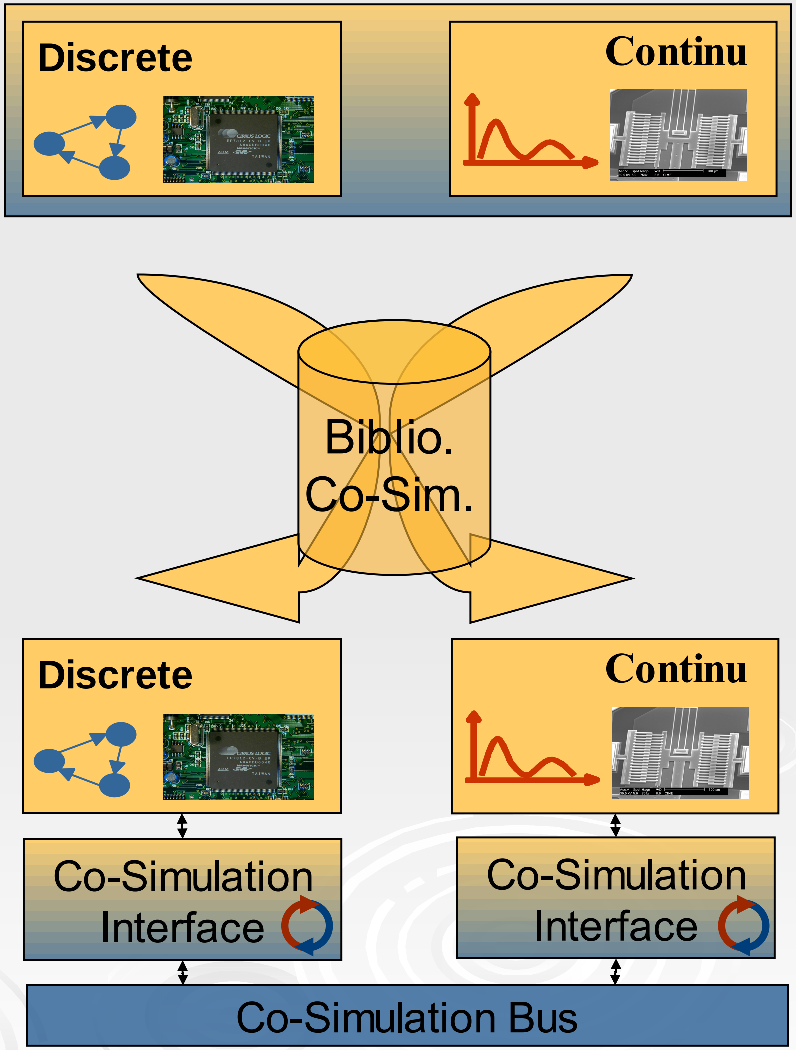
Méthodologie de conception des outils de co-simulation

Generic Stage

- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

Implementation stage

- Sim. tools analysis
- Library elements implementation
- Implementation validation



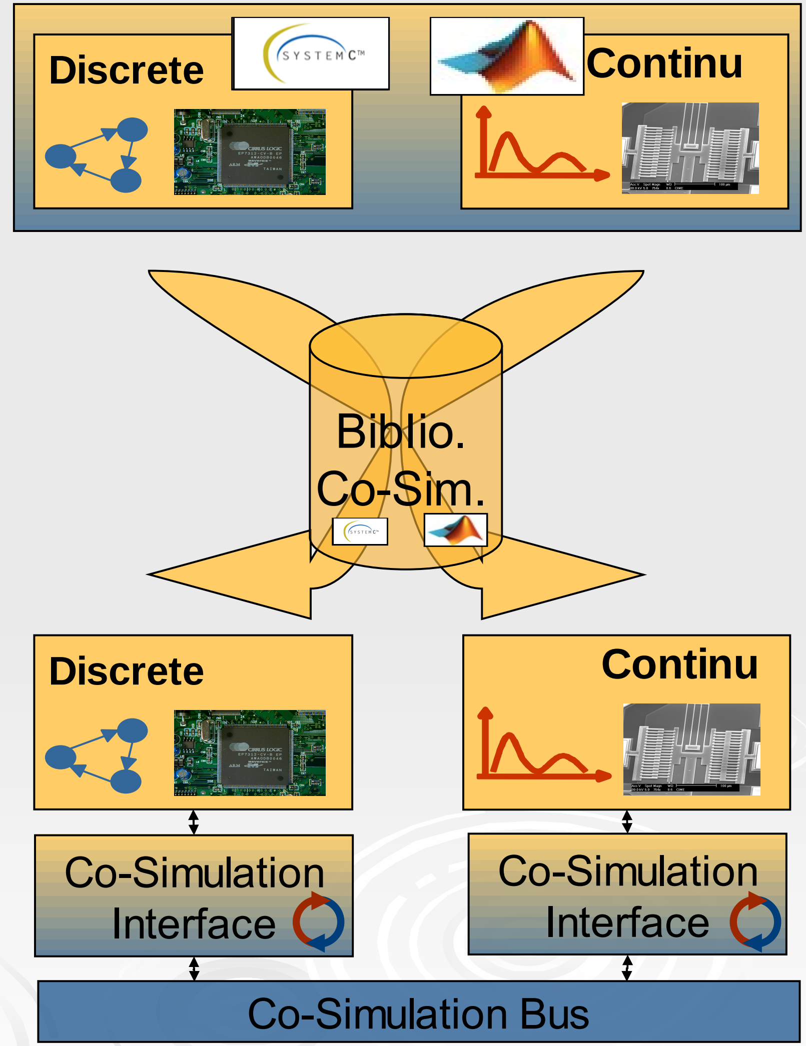
Méthodologie de conception des outils de co-simulation

Generic Stage

- Definition of the synchronization operational semantic
- Distribution of the synchronization functionality to co-sim. interfaces
- Interfaces behavior formalization and verification
- Definition of the internal architecture and co-simulation library

Implementation stage

- Sim. tools analysis
- Library elements implementation
- Implementation validation



Outline

- Introduction
- Methodology for co-simulation tool design
- Application for CODIS tool design
- Conclusions

Synchronization – Operational Semantic –

- **Based on DEVS**

$$DEV = \langle X, S, Y, \delta int, \delta ext, \lambda, ta \rangle$$

X = { (pd, vd) | pd∈InPorts, vd∈Xpd } - set of input ports and their values in the discrete domain

Y = {(pd, vd) | pd∈OutPorts, vd∈Ypd } - set of output ports and their values in the discrete domain

S = is the set of sequential state (locations)

e = the elapsed time from the last transition

Q = {(s,e)|s ∈S, 0 ≤e≤ ta(s)} is the total state set

δint : S→ S - internal transition function

δext : Q x X→S - external transition function where:

λ : S→Y - output function

ta : time advance function – real and positive, can be 0 and ∞.

Synchronization – Operational Semantic –

- **DESS = ‹X, Y, Q, f, λ›**

X = $\{(p_c, v_c) | p_c \in \text{InPorts}, v_c \in \Re\}$ set of input ports and their values in the continuous domain

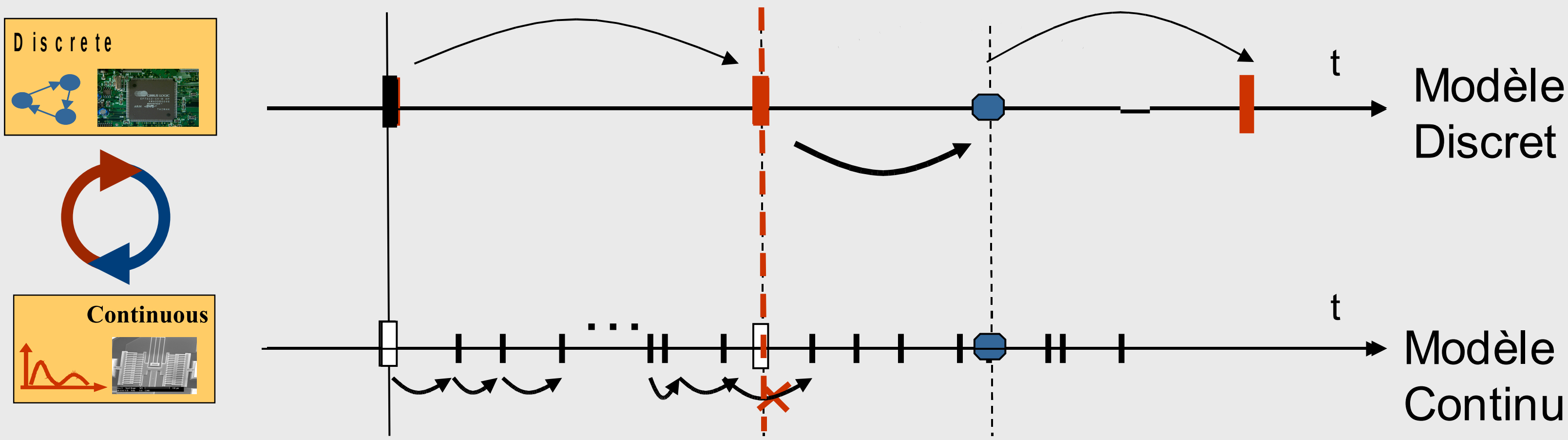
Y = $\{(p_c, v_c) | p_c \in \text{OutPorts}, v_c \in \Re\}$ set of output ports and their values in the continuous domain

Q = set of states

f: $Q \times X \rightarrow Q$ is the rate of change function

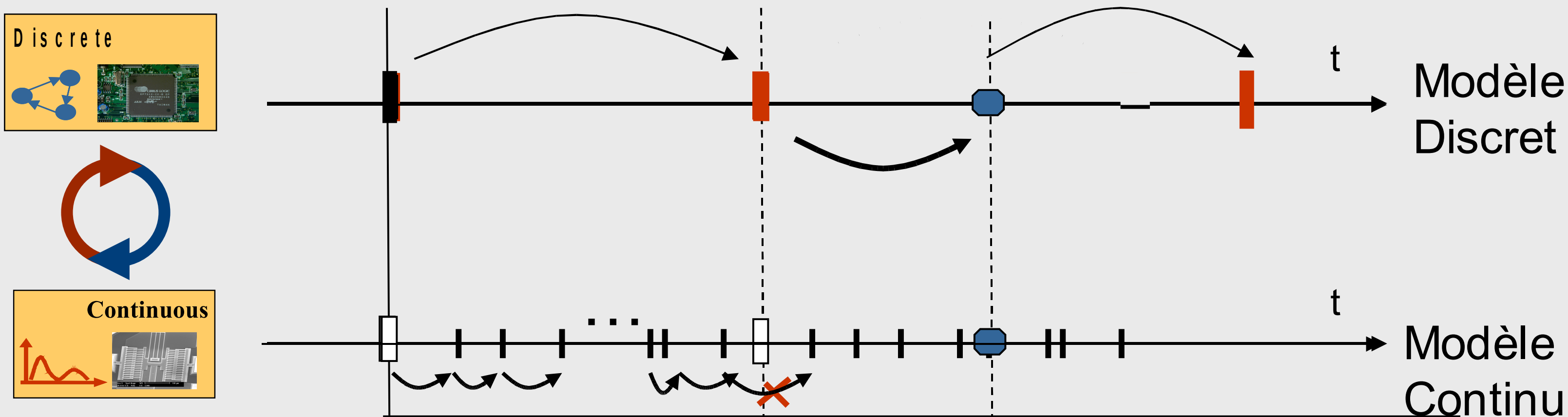
λ: $Q(+X) \rightarrow Y$ output function

Hétérogénéité continu/discret



$$\frac{q = \delta_{ext}(q) \wedge synch = 1 \wedge flag = 1}{(s_d, e_d) \xrightarrow{!(\lambda(s_d), t_a(s_d)); flag := 0} (s_d, e_d); q \xrightarrow{?(\lambda(s_d), t_a(s_d)); synch := 0} q}$$

Hétérogénéité continu/discret



$$\frac{synch = 1 \wedge flag = 1 \wedge q = \delta_{ext}(q)}{(s_d, e_d) \xrightarrow{!(\lambda(s_d), t_a(s_d)); flag:=0} (s_d, e_d); q \xrightarrow{?(\lambda(s_d), t_a(s_d)); synch:=0} q}$$

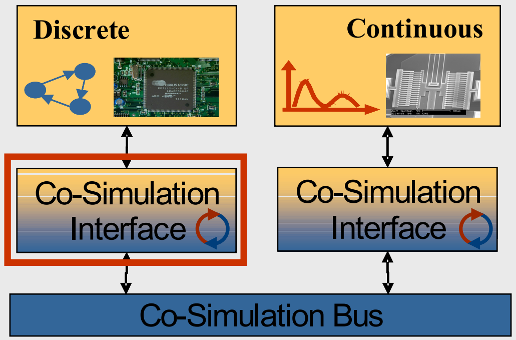
$$\frac{flag = 0 \wedge \neg statevent(t) \wedge q' = \delta_{int}(q)}{q \xrightarrow{!\lambda(q)} q' \xrightarrow{!\lambda(q); flag:=1} q'}$$

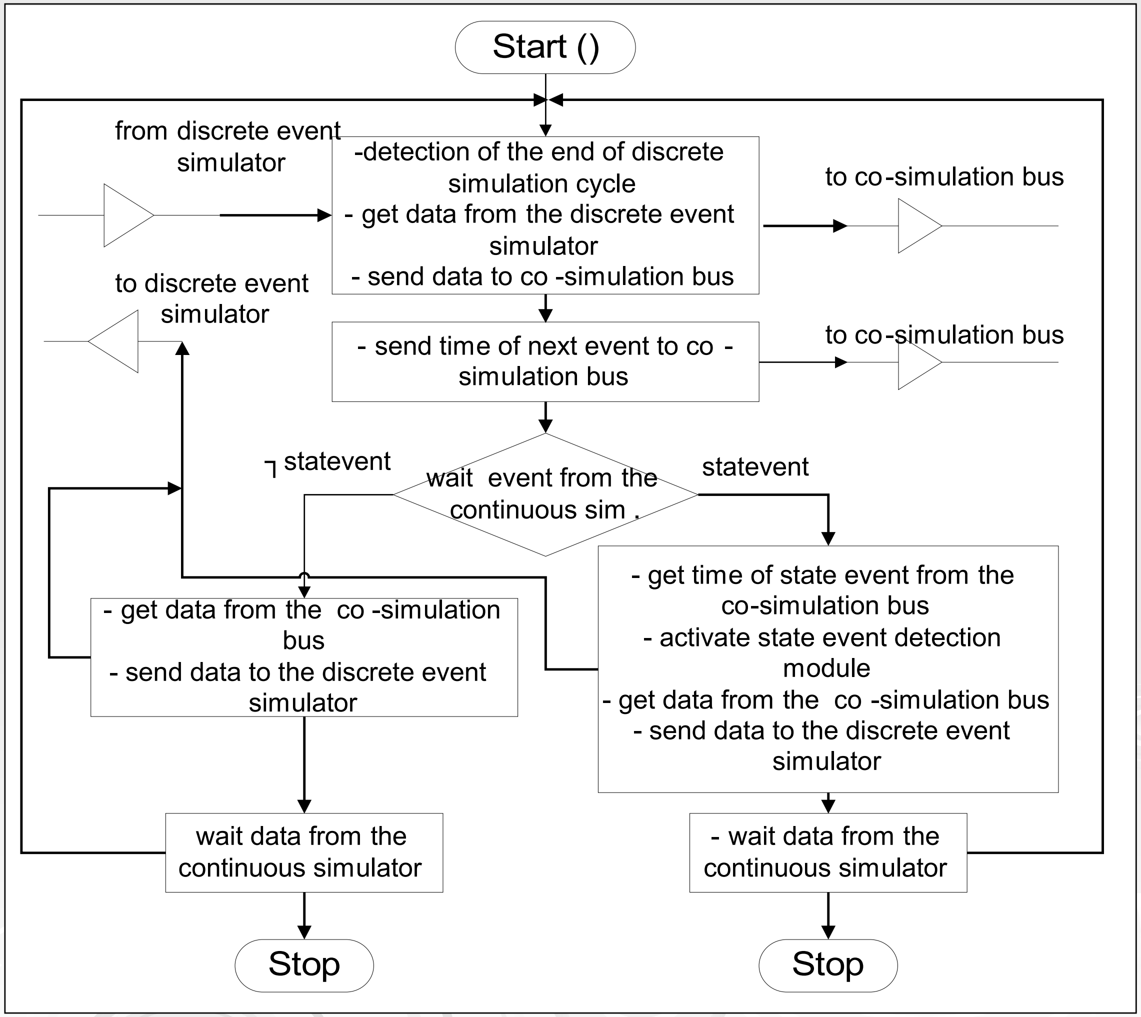
$$\frac{synch = 0 \wedge flag = 1 \neg statevent \wedge s'_d = \delta_{ext}(s_d)}{(s_d, e_d) \xrightarrow{t_a(s_d) - e_d} (s_d, t_a(s_d)) \xrightarrow{?\lambda(q); \delta_{int}(s'_d); \lambda(s'_d); synch:=1} (s'_d, 0)}$$

$$\frac{flag = 0 \wedge statevent \wedge q' = \delta_{int}(q)}{q \xrightarrow{!\lambda(q)} q' \xrightarrow{!\lambda(q); !t_{se}; flag:=1} q'}$$

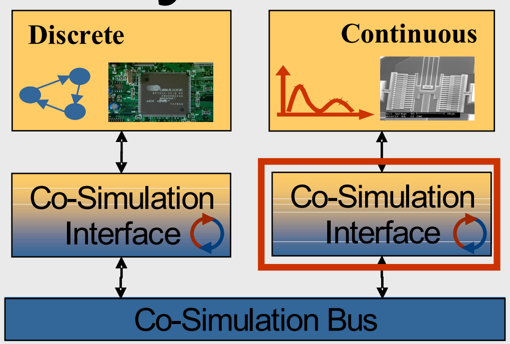
$$\frac{synch = 0 \wedge flag = 1 \wedge statevent \wedge s'_d = \delta_{ext}(s_d, t)}{(s_d, e_d) \xrightarrow{?t_{se}} (s_d, t_{se}) \xrightarrow{?\lambda(q); \delta_{int}(s'_d); \lambda(s'_d); synch:=1} (s'_d, 0)}$$

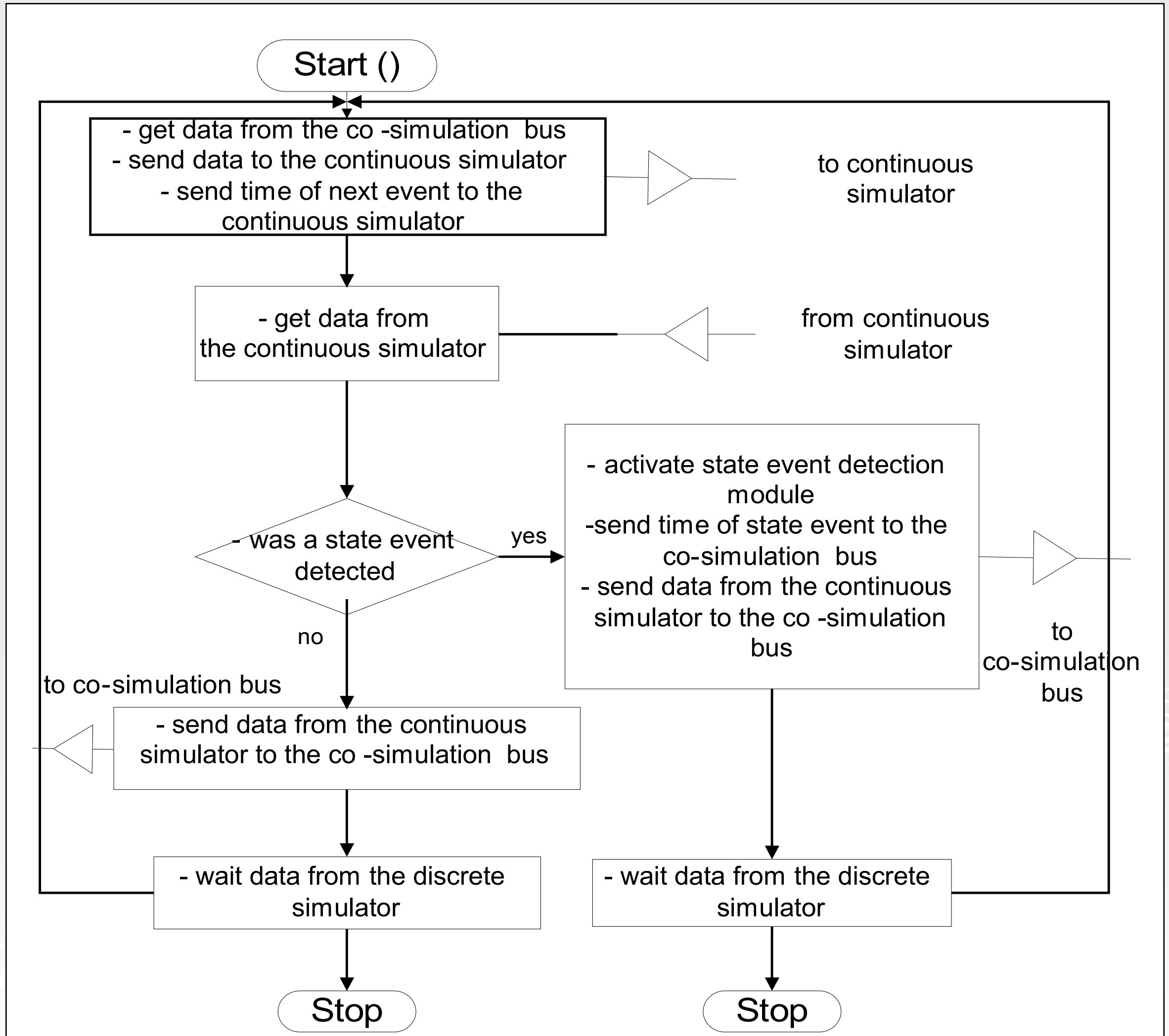
Distribution de la fonctionnalité de synchro aux interfaces de co-sim.



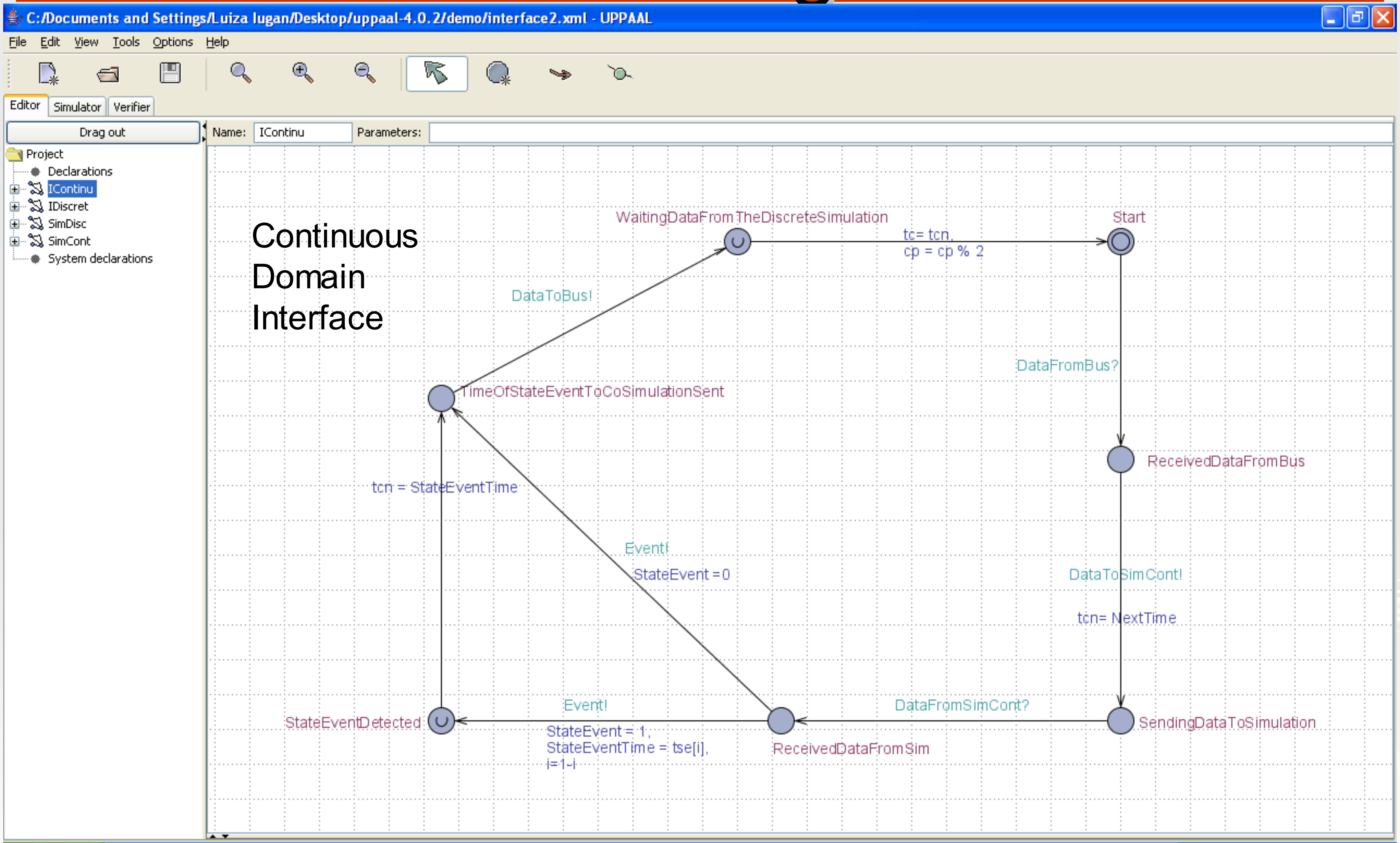


Distribution de la fonctionnalité de synchro aux interfaces de co-sim.

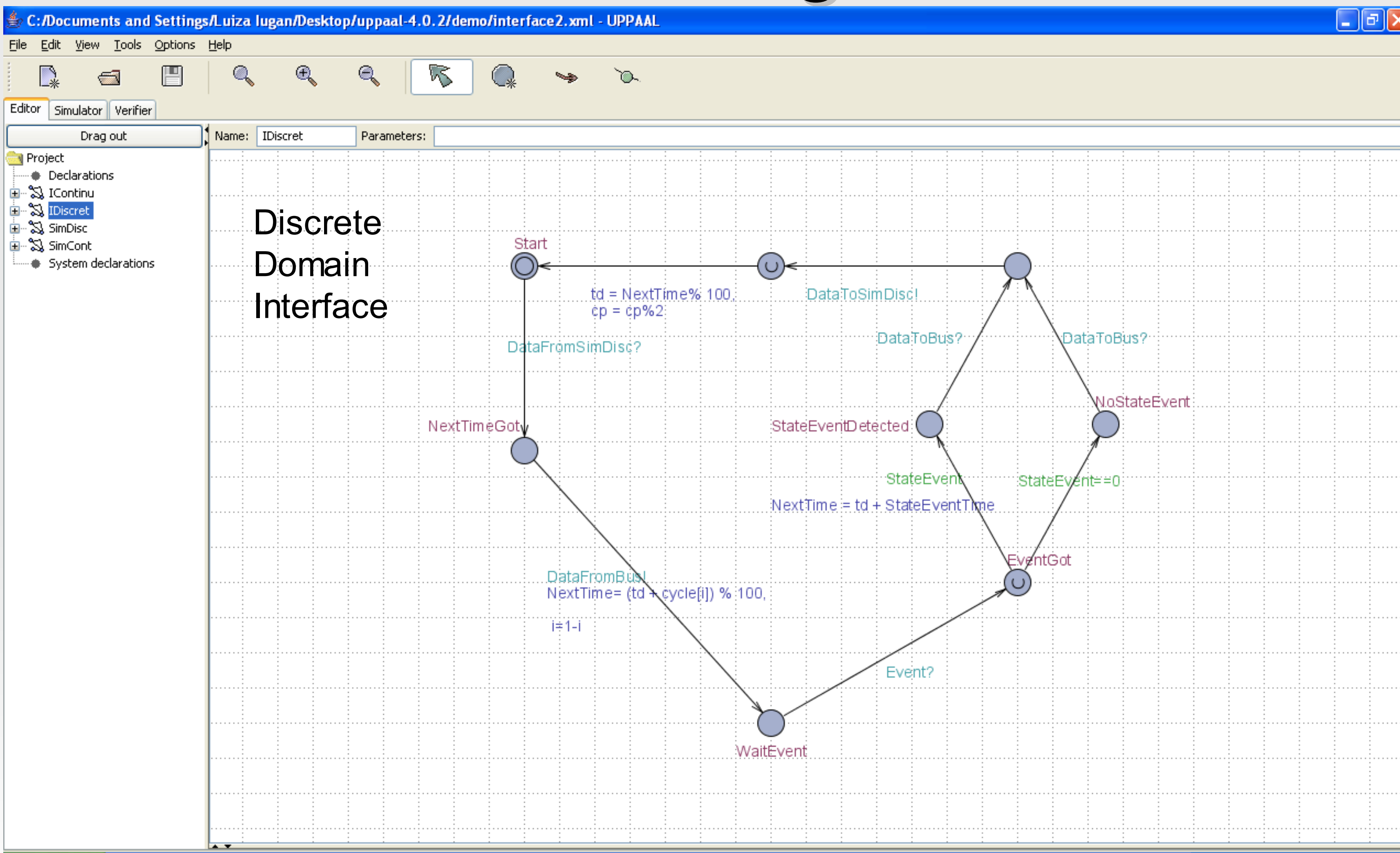




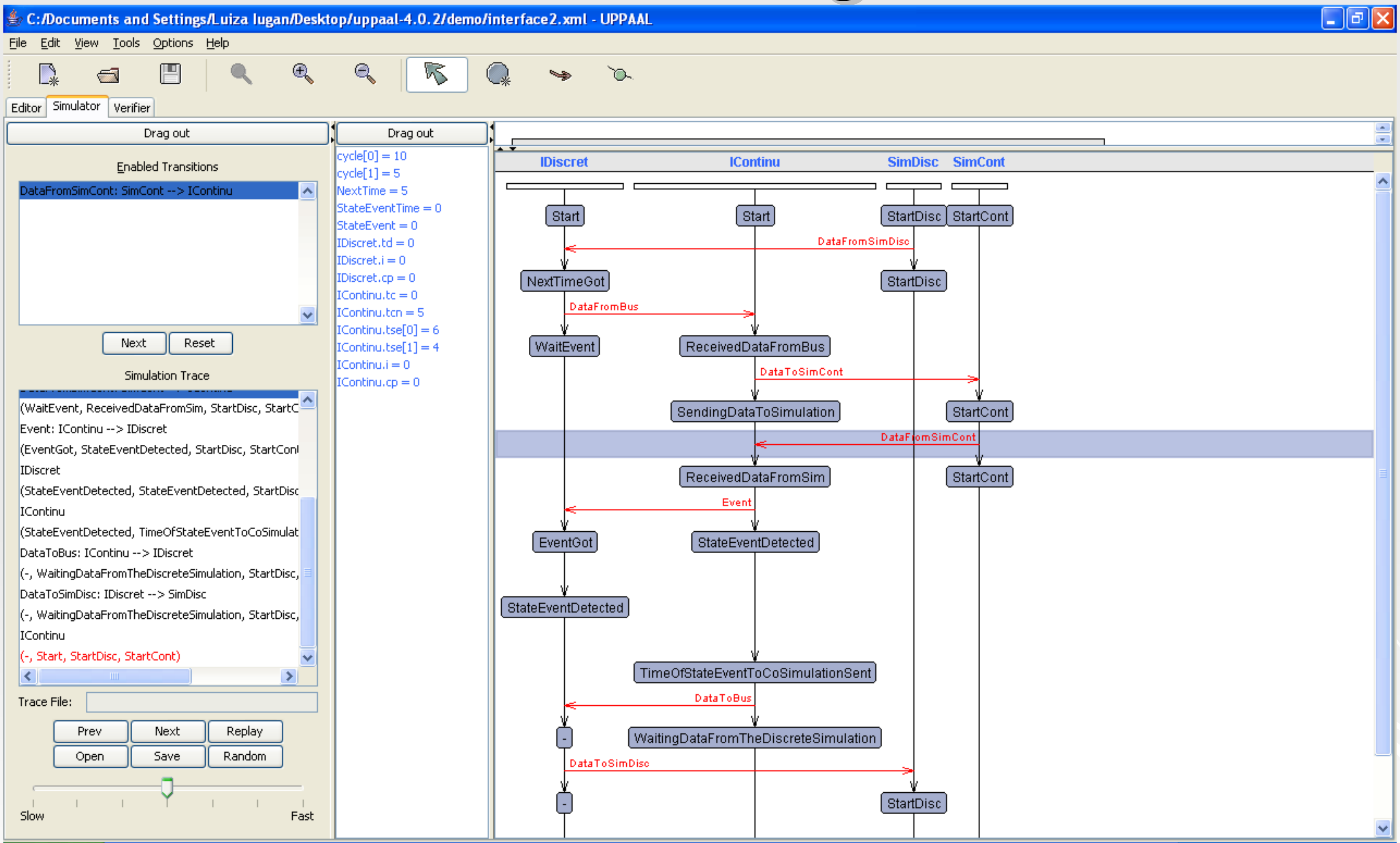
Verification of Co-Simulation interfaces using UPAAL



Verification of Co-Simulation interfaces using UPAAL

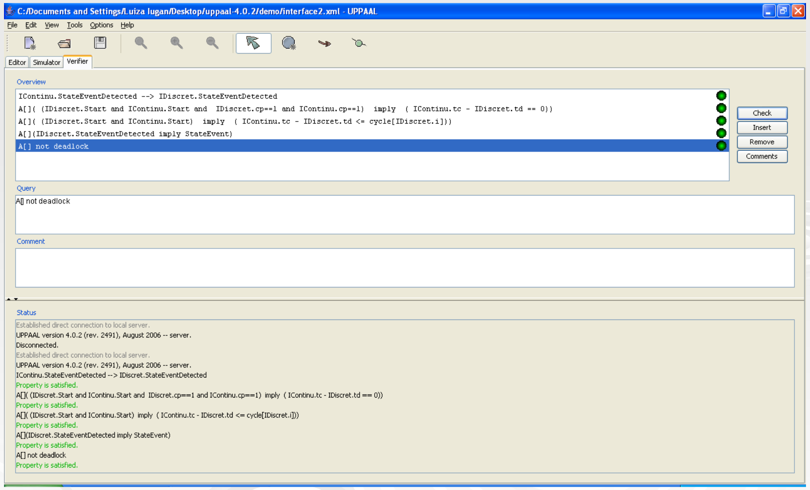


Verification of Co-Simulation interfaces using UPAAL

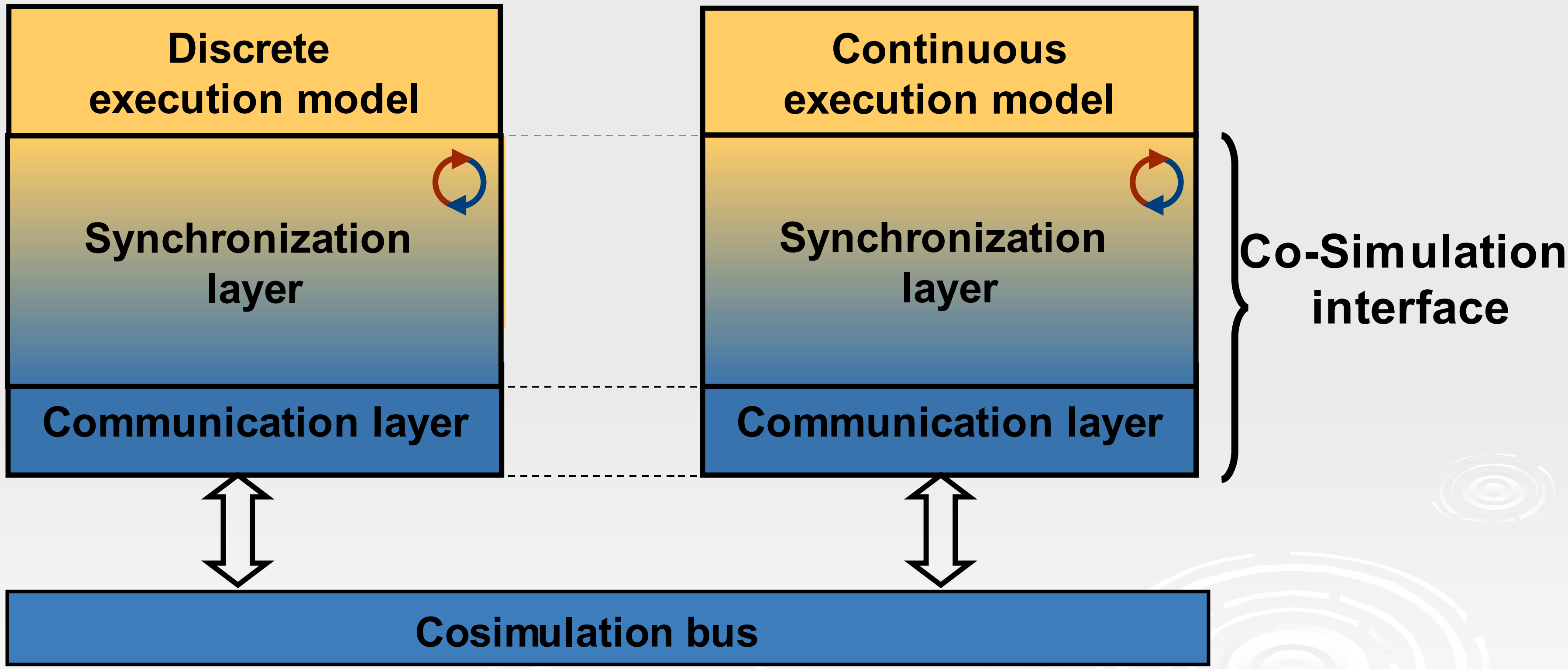


Verification of Co-Simulation interfaces using UPAAL

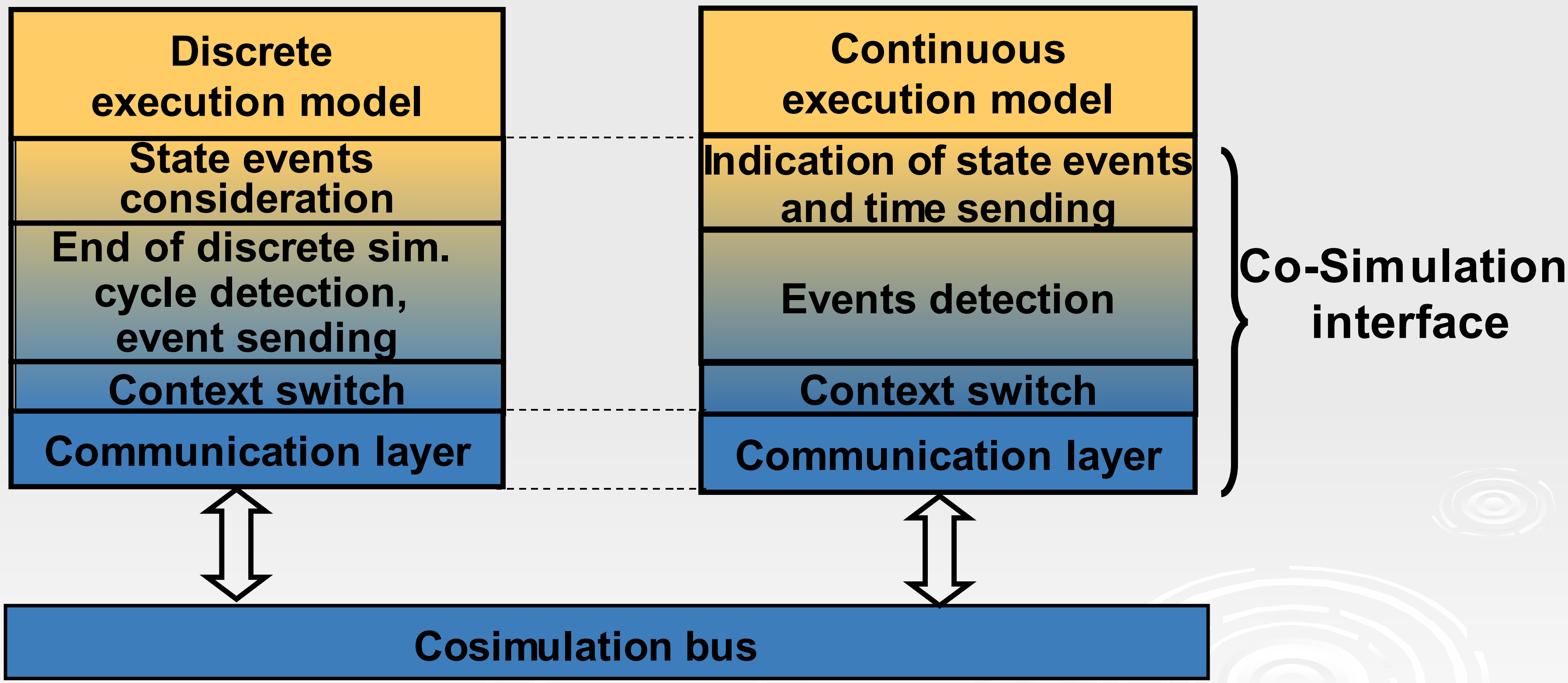
- Properties verification
  - No deadlock
  - Timing synchronization
  - All state events are detected
  - No false state events



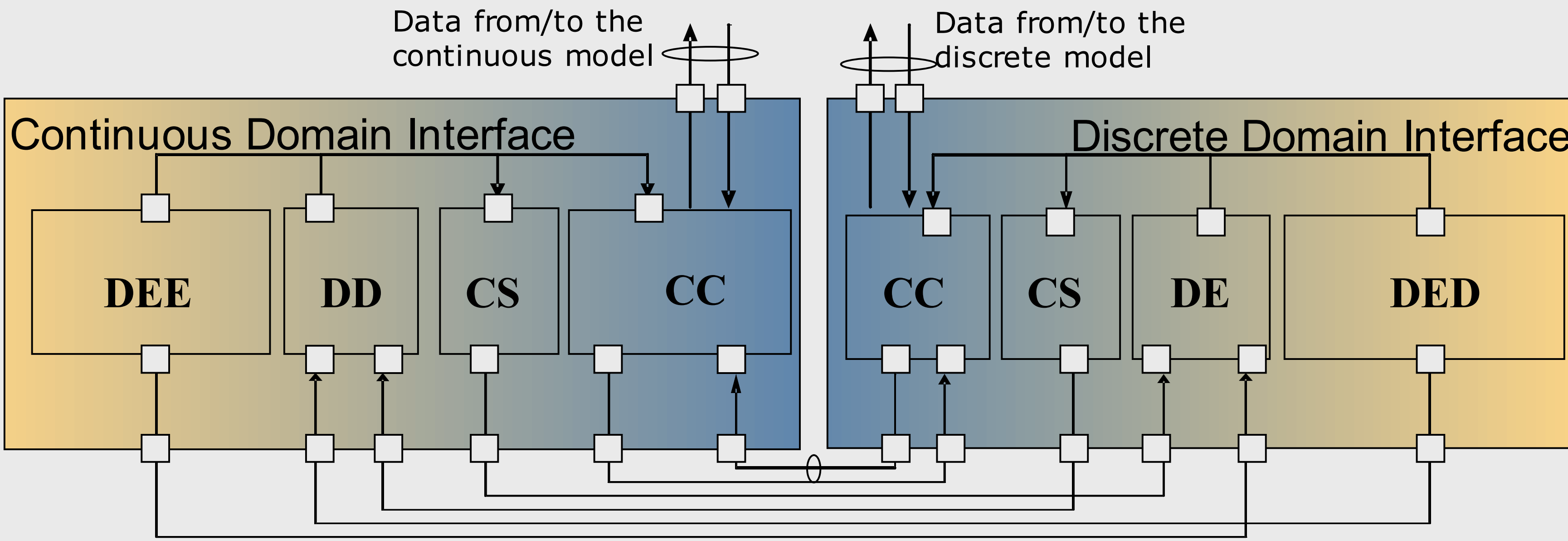
Generic Architecture for Continuous/Discrete Co-Simulation



Generic Architecture for Continuous/Discrete Simulation



Définition architecture interne des intf. et éléments de la bibliothèque



$N=(X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC)$

$X=\{(p, v) | p \in IPorts, v \in X_p\}$ set of *input ports* and *values*

$Y=\{(p, v) | p \in OutPorts, v \in Y_p\}$ set of *output ports* and *values*

D - set of *components names*

$M_d=(X_d, S, Yd, \delta_{int}, \delta_{ext}, \lambda, t_a)$ is a DEVS with

X_d - set of *input ports* and *values*

Y_d - set of *output ports* and *values*

EIC (External Input Coupling)

$EIC=\{((N, ip_N), (d, ip_d)) | ip_N \in IPorts_N, d \in D, ip_d \in IPorts_d\}$

EOC (External Output Coupling)

$EOC=\{((d, op_d), (N, op_N)) | d \in D, op_d \in OPorts_d, op_N \in OPorts_N\}$

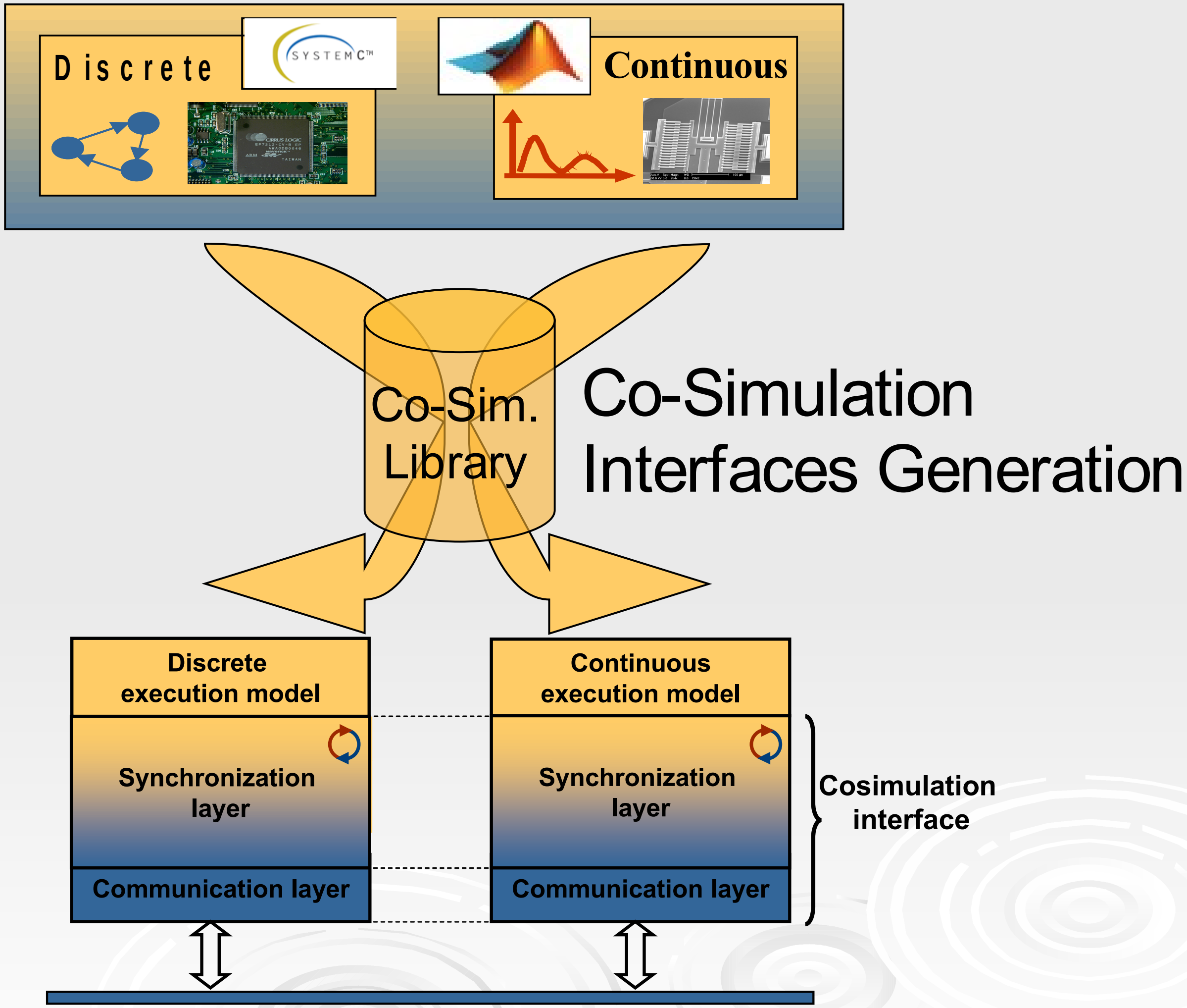
IC (Internal Coupling)

$IC=\{((a, op_a), (b, ip_b)) | a, b \in D, op_a \in OPortsd_a, ip_b \in IPorts_b\}$

CODIS Framework – Simulation Flow –

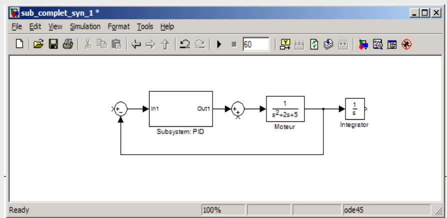
Input Flow

Output Flow

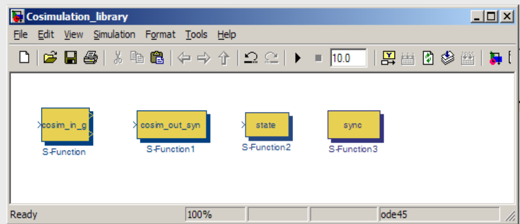


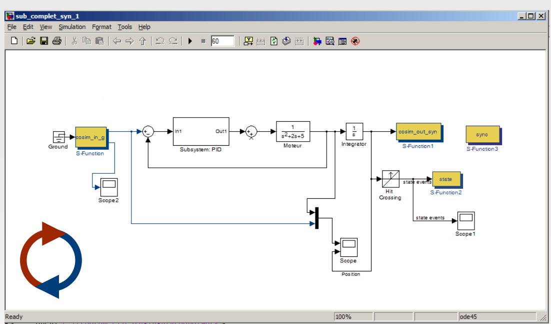
Generation of Continuous Model Interfaces

Simulink input specification



Co-Simulation Library





Simulink Specification with co-simulation interfaces

- Simulink Co-Simulation Library Blocks
  - State-Event detection and signalling
  - Integration Step Adjustment
  - Synchronization with sampling and update events
  - Communication

Generation of Discrete Model Interfaces

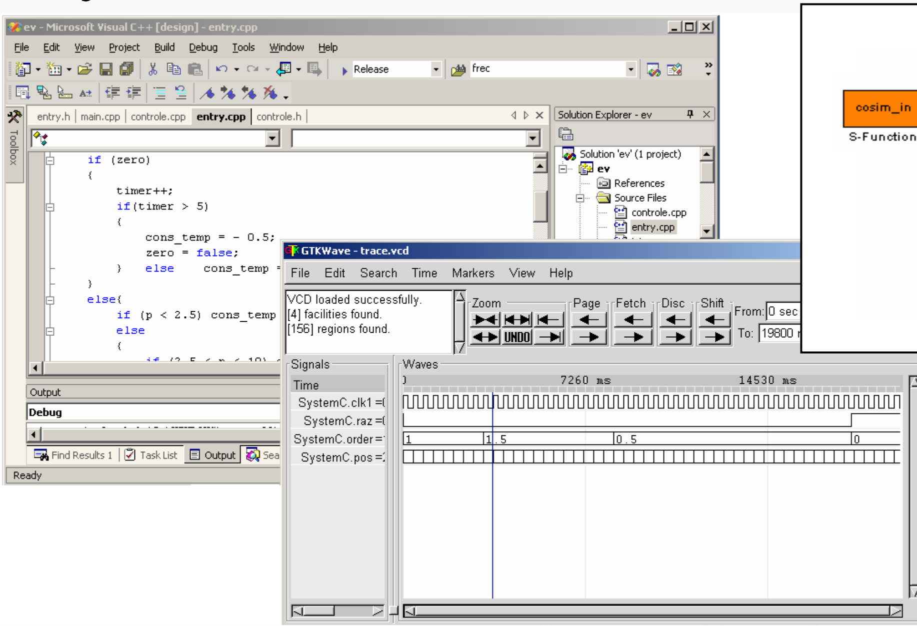
- Based on a SystemC Co-Simulation Library
  - State events management blocks
  - Communication blocks
- Automatic generation of co-simulation interfaces by a code generator that has as input user-defined parameters
  - Data types
  - No. and type of ports
  - Synchronization model

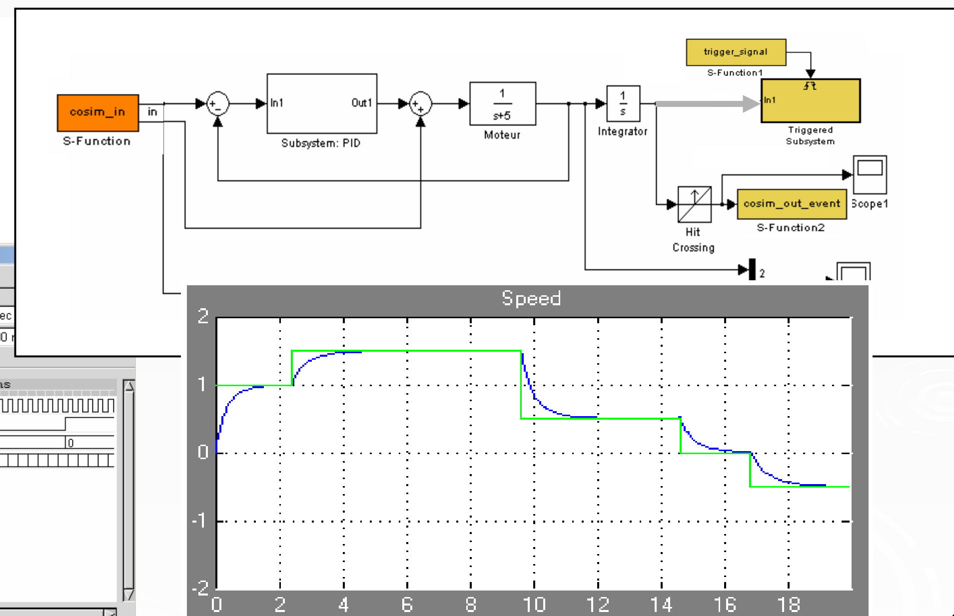
CODIS
- Continuous/Discrete simulation -

- SystemC/Simulink accurate simulation
  - Easy integration, generic library elements

Systemc

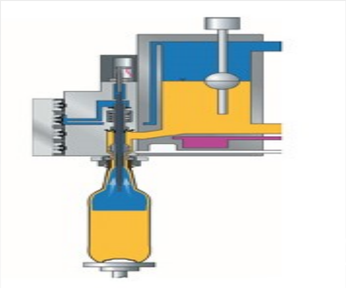
Simulink

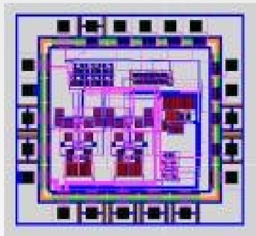


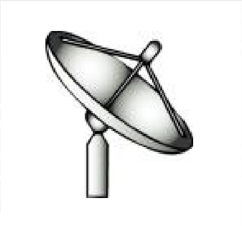


CODIS
- Applications -

- Control applications - Robot arm manipulator, Bottle filling system
- Mixed signal application: Σ/Δ Converter
- Wireless application: Radar system







Conclusions

- Continuous/discrete systems designs requires global validation
  - Co-simulation Technique
- Challenges for continuous/discrete co-simulation
  - Definition of global execution models
  - Automatic generation of co-simulation interfaces
- Methodology for co-simulation tools design
  - Application for SystemC/Simulink

Performances analysis

- Inter-Simulators Communication overhead
  - 20% of the total simulation time
- Overhead caused by the Simulink integration step adjustment
  - max. 5% of total simulation time
- SystemC Synchronization overhead
  - max. 0.2% of the total simulation time

Continuous/Discrete Systems - Synchronization Models -

Synchronization model	**Synchronization Step**	**Advantages**	**Inconvenient**
Full synchronisation mode	Each discrete step and/or state event occurrence	General	Synchronization overhead
Predictible events mode	Each update/sampling events and/or state event occurrence	Improve performances	The prediction of update/sample events is required
Unpredictable events mode	Each update/sampling events and/or state event occurrence	Non-periodic update/sample events	Rollback may be required