Aspects Applicatifs : DEVsimPy

DEVSimpy

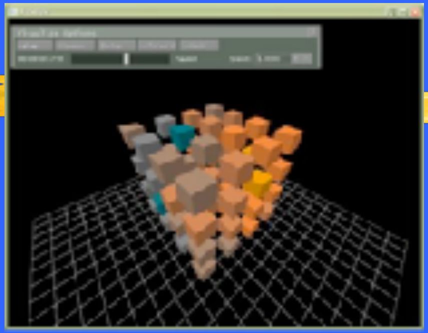
DEVsimPy
SPE

MS4
SYSTEMS

Univ Arizona
RTSync

ATOM3

ATOM3
Mc Gill



CD ++
Carleton

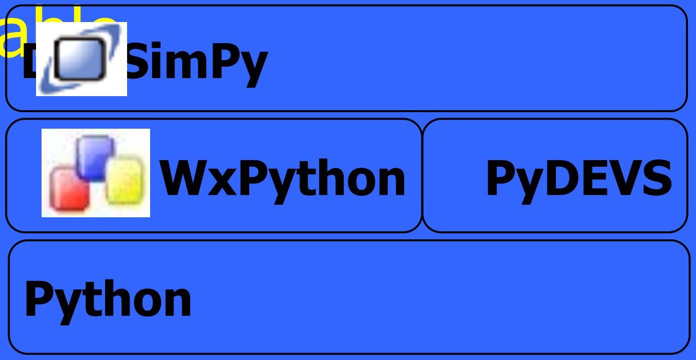
VIRTUAL
LABORATORY
ENVIRONMENT
Complex systems modelling and simulation

VLE
INRA

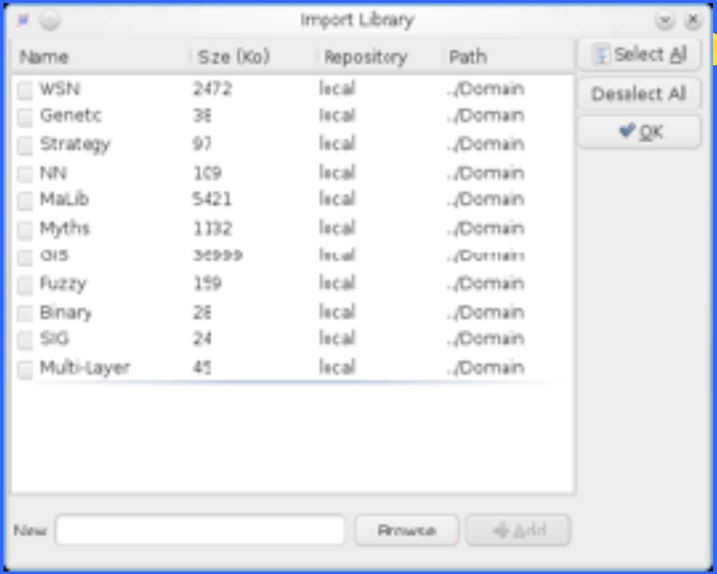
POWERDEVS
Univ Rosario

Aspect Applicatif DEVSimPy

- Logiciel collaboratif : M&S DEVS des systèmes complexes en langage Python.
- Crée en 2008 : proposer une interface graphique au noyau de modélisation et de simulation PyDEVS(Univ. McGill)
- Basé sur le toolkit wxPython
- Sous licence GPL v.3 et téléchargeable

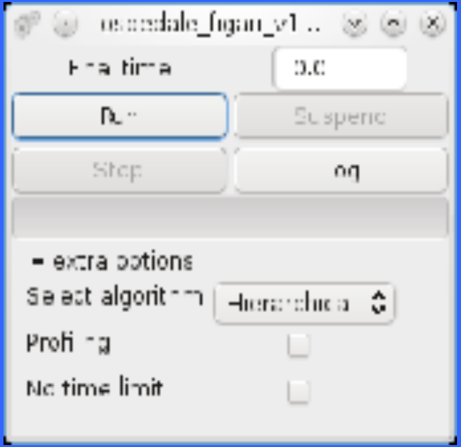


Aspect Applicatif DEVSimPy



Import Library

Name	Size (Ko)	Repository	Path
WSN	2472	local	./Domain
Genetic	38	local	./Domain
Strategy	97	local	./Domain
NN	109	local	./Domain
MaLib	5421	local	./Domain
Myths	1132	local	./Domain
GIS	36999	local	./Domain
Fuzzy	159	local	./Domain
Binary	28	local	./Domain
SIG	24	local	./Domain
Multi-Layer	45	local	./Domain



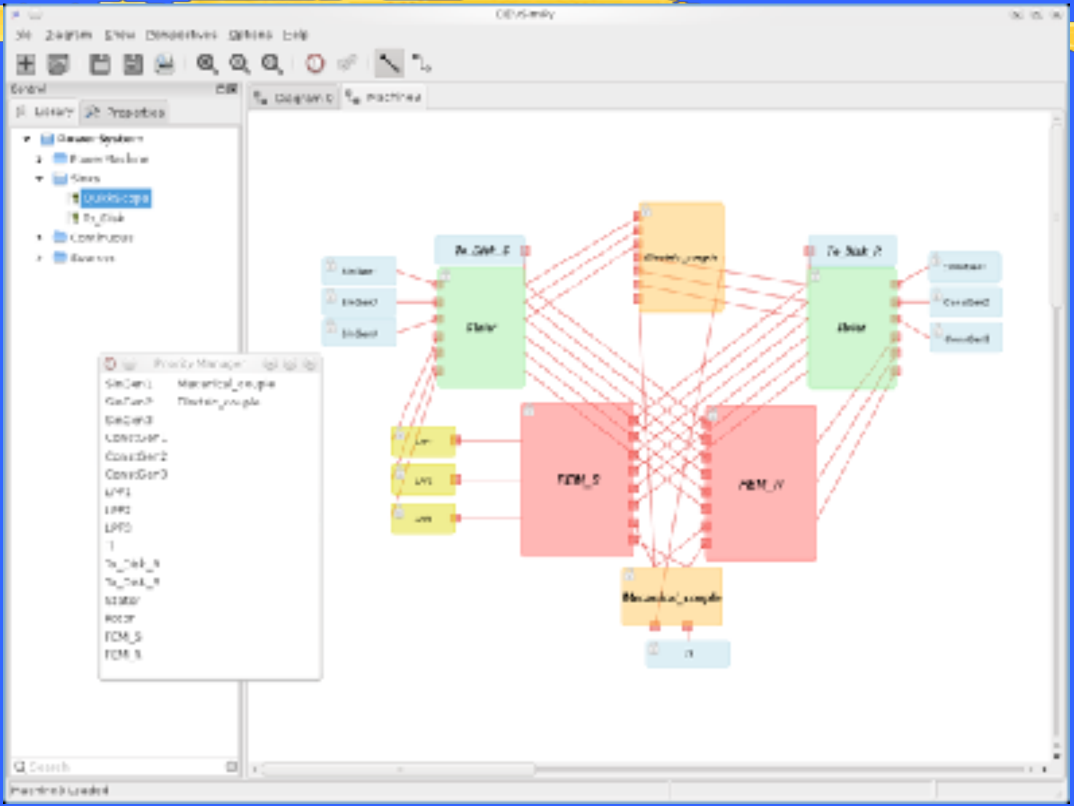
[illegible]

Final time: 0.0

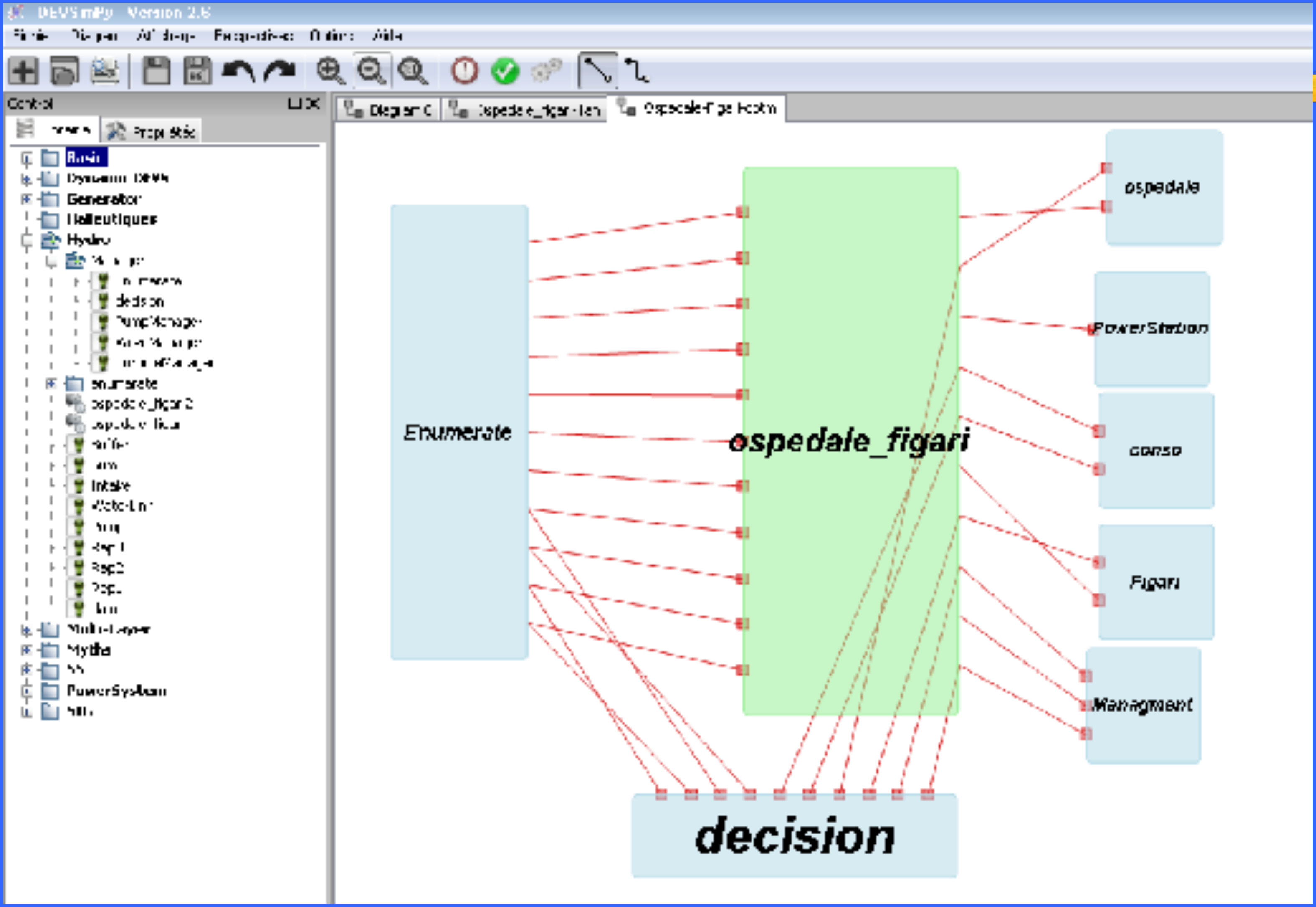
Select algorithm: Hierarchical

Profiling

No time limit



Aspects applicatifs
Exemple 1 : Gestion Barrages



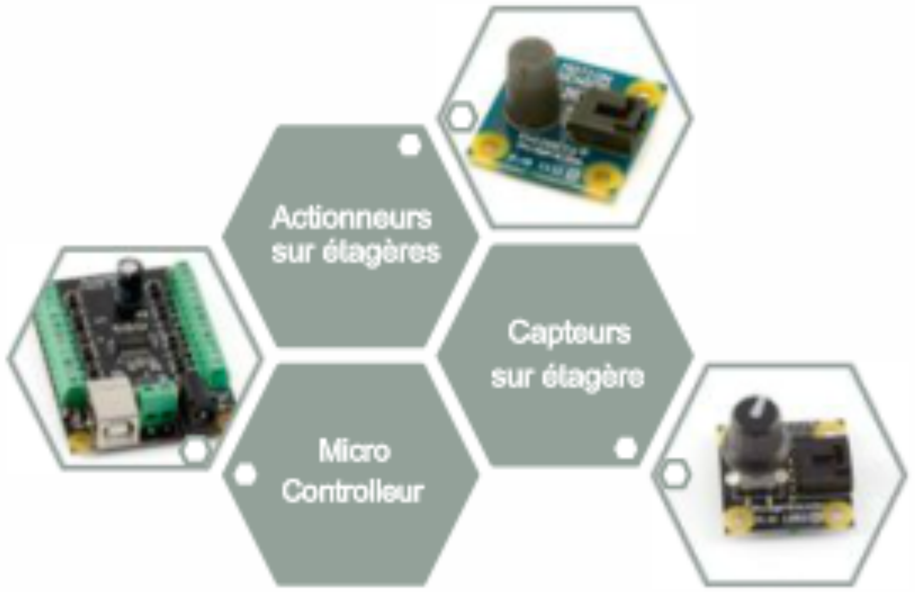
Aide à la gestion d'application mobile et Internet

- Exemple Gestion de capteurs, actionneurs, cartes,etc à partir de le simulation

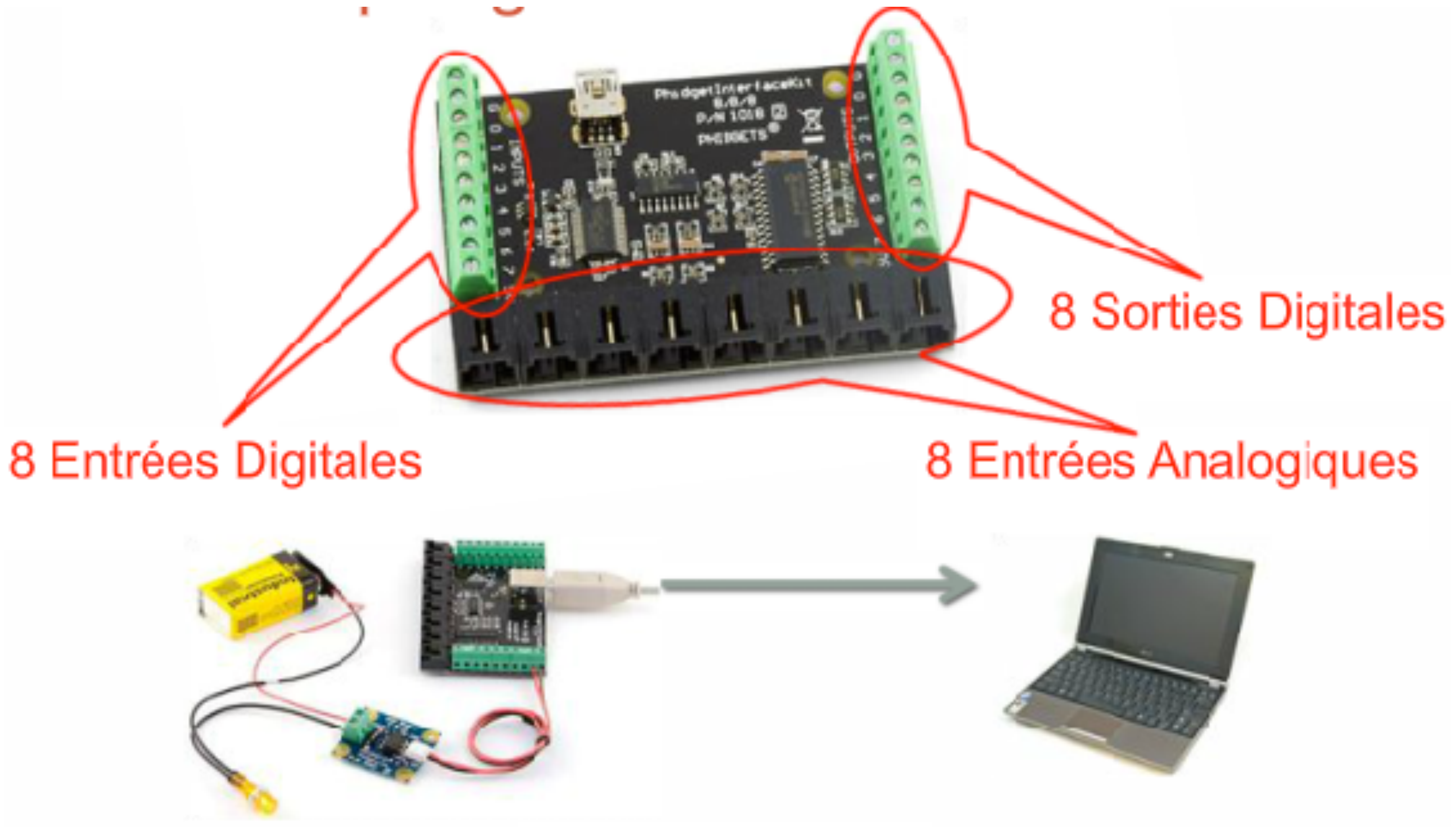
- Pour cela : définir une bibliothèques de modèles DEVS permettant de simuler le comportement des éléments capteurs, actionneurs, cartes.
- En modélisant le comportement d'une carte , on a la possibilité de commander à partir de DEVS des capteurs et actionneurs.

Contexte du travail : Phidgets

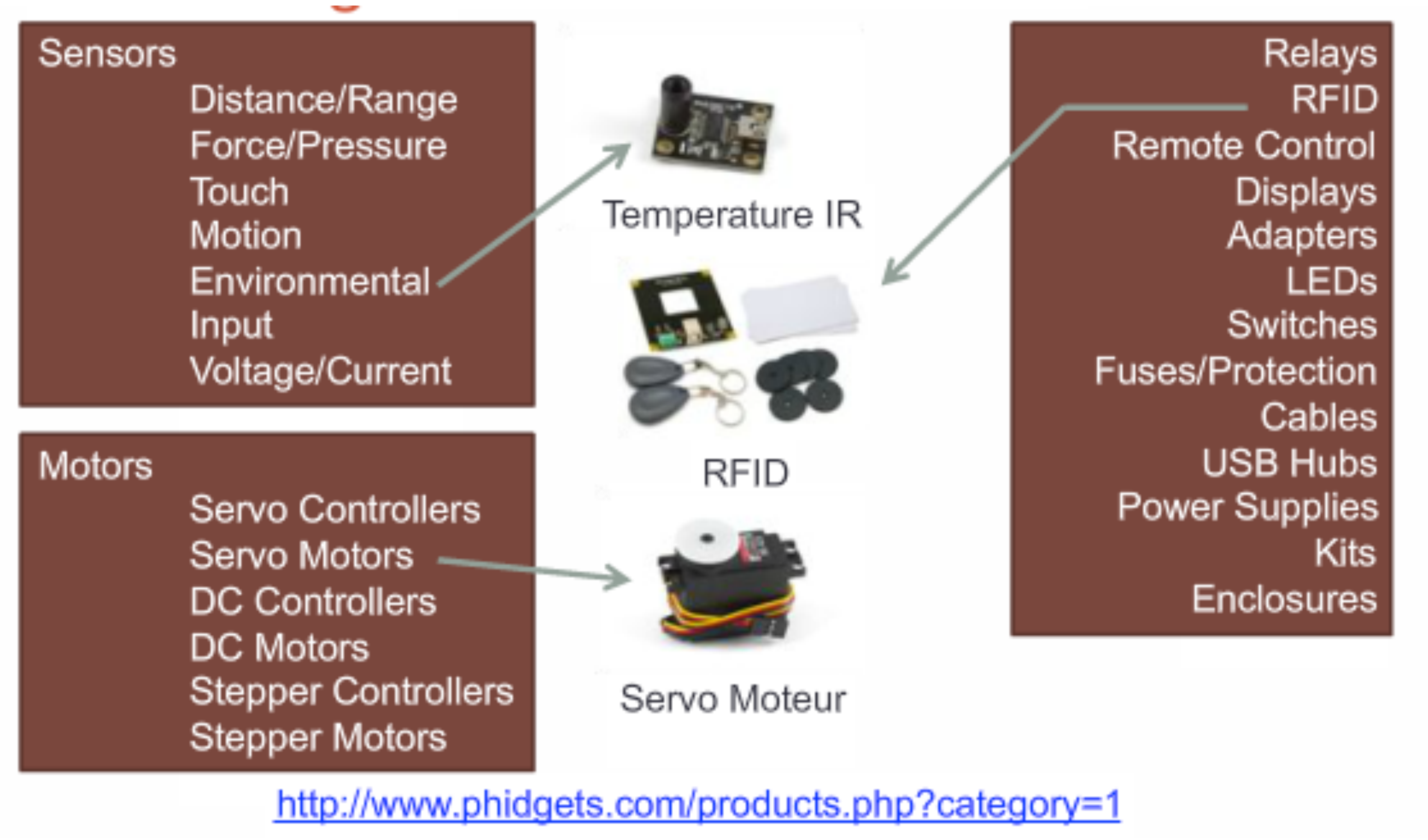
- Capteurs, actionneurs sur étagère



Contexte du travail : Carte SBC



Contexte du travail :les composants



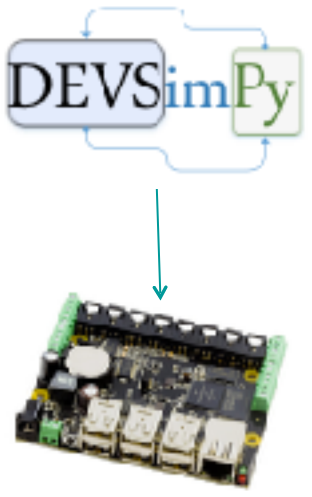
Sensors
- Distance/Range
- Force/Pressure
- Touch
- Motion
- Environmental
- Input
- Voltage/Current

Motors
- Servo Controllers
- Servo Motors
- DC Controllers
- DC Motors
- Stepper Controllers
- Stepper Motors

Relays
RFID
Remote Control
Displays
Adapters
LEDs
Switches
Fuses/Protection
Cables
USB Hubs
Power Supplies
Kits
Enclosures

http://www.phidgets.com/products.php?category=1

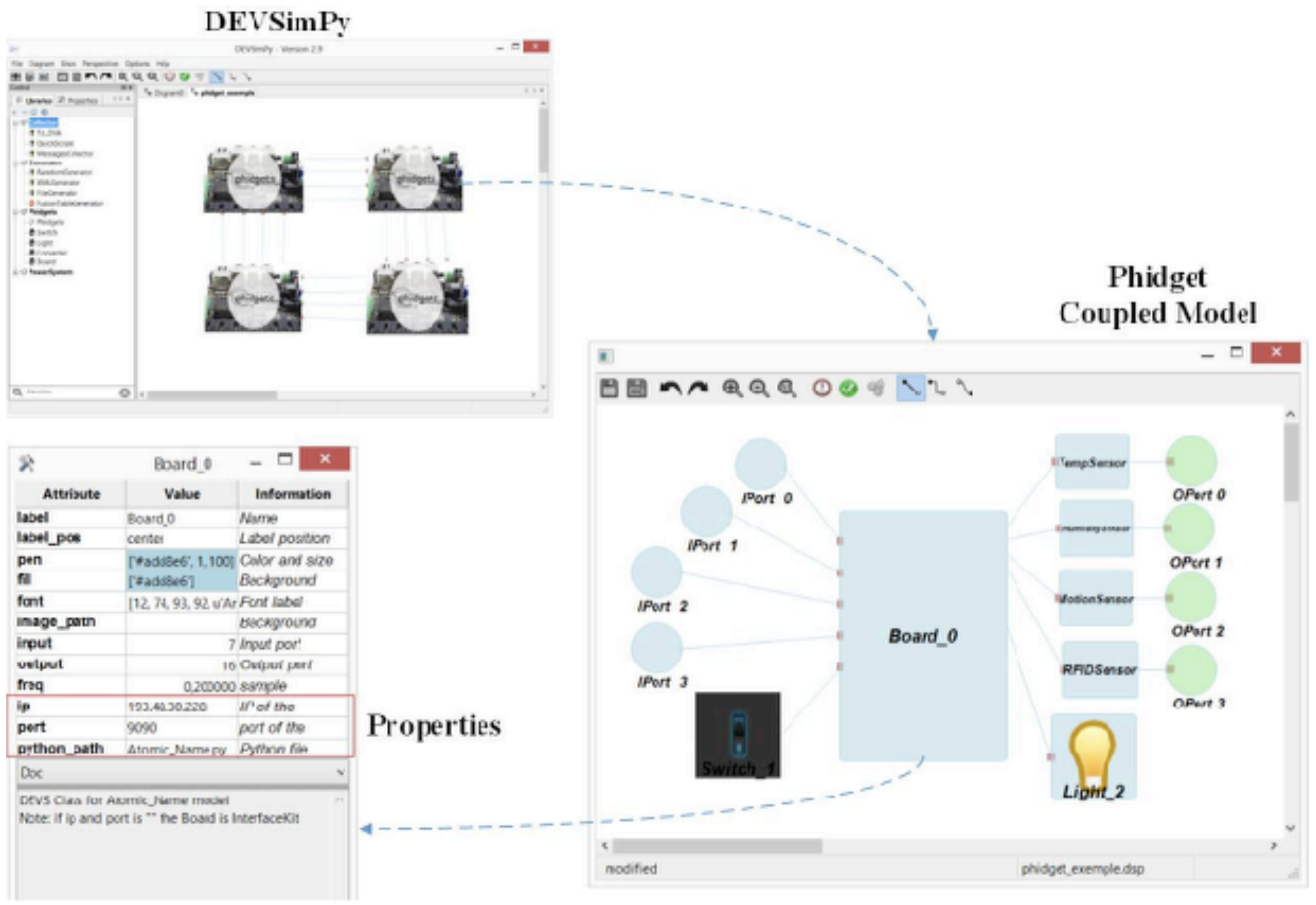
Mise en Oeuvre : Infrastructure Choisie



- Développement d’une bibliothèque de composants Phidget permettant de manipuler une carte Phidget, les capteurs et les actuateurs associés à partir de DEVSimPy

- Problème : comment accéder à une carte Phidget à partir d’une appli mobile avec la possibilité de gérer capteurs et actuateurs à partir de simulations;
- 2 solutions :
  - Embarquer DEVSimPY sur la carte SBC Phidget
  - Utilisation d’un serveur Web

Mise en Oeuvre : Infrastructure Phidget et DEVSimPy



Board_0

Attribute	Value	Information
label	Board_0	Name
label_pos	center	Label position
pen	['#add8e6', 1,100]	Color and size
fill	['#add8e6']	Background
font	[12, 74, 93, 92, u'Ar	Font label
image_path		Background
input	7	Input port
output	16	Output port
freq	0,200000	sample
ip	193.48.30.220	IP of the
port	9090	port of the
python_path	Atomic_Name.py	Python file

Doc

DEVS Class for Atomic_Name model
Note: if ip and port is "" the Board is InterfaceKit

Actuation Conflicts Management : GOAL(1)

- One of challenges in the field of **IoT system Design** is the management of actuation conflicts that may arise as soon as different applications compete for accessing shared actuators (direct conflicts) or shared physical environment properties (indirect conflicts).
- Need of **Actuation Conflict Management (ACM)** software component introduced at the design phase to detect and resolve conflicts by applying resolution rules.
- **Discrete-Event Modeling and Simulation** can be used to validate very early at the Design phase the ACM before the deployment in the physical environment.

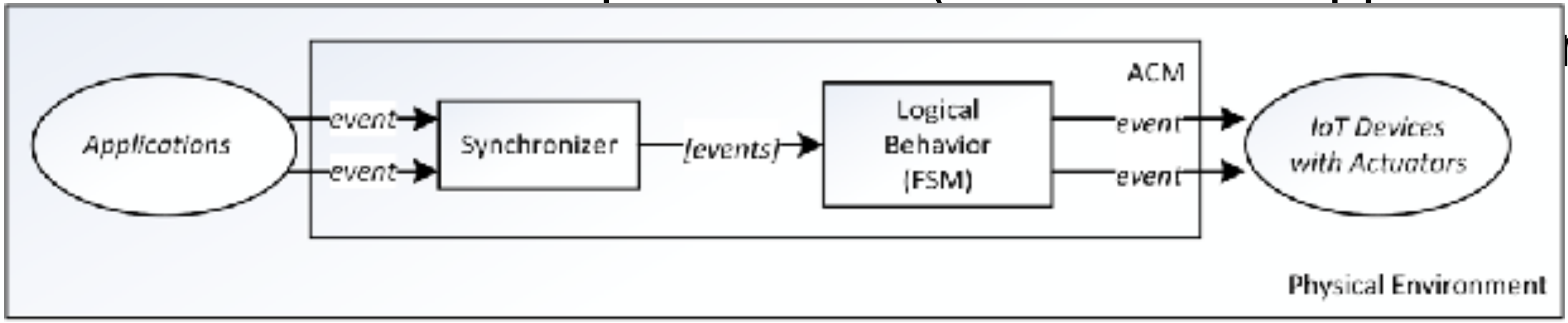
GOAL(2)

- Define DEVS simulation models for IoT systems Design - ACM validation
- Assist the IoT system Designer in the ACM validation phase before the deployment by:
    - Formally specifying conflicts and resolving rules
    - Testing the ACM software component in a simulated environment (with actuators)

ACM (Actuation Conflict Manager) Assisted Design (1)

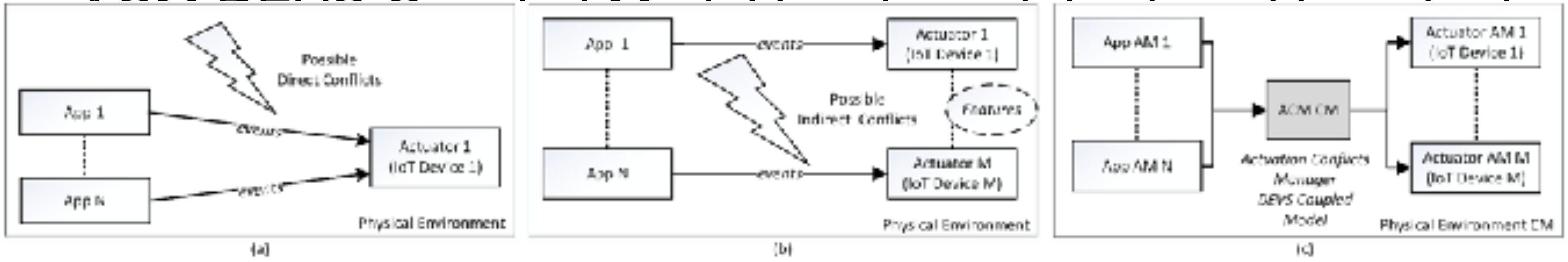
- The ACM component is composed of a **Synchronizer** and a **Logical Behavior.**
- The Logical Behavior executes the resolving rules specified by the designer in a FSM format.
- The Synchronizer component executes a synchronization and serialization of its inputs events (actions from application

n

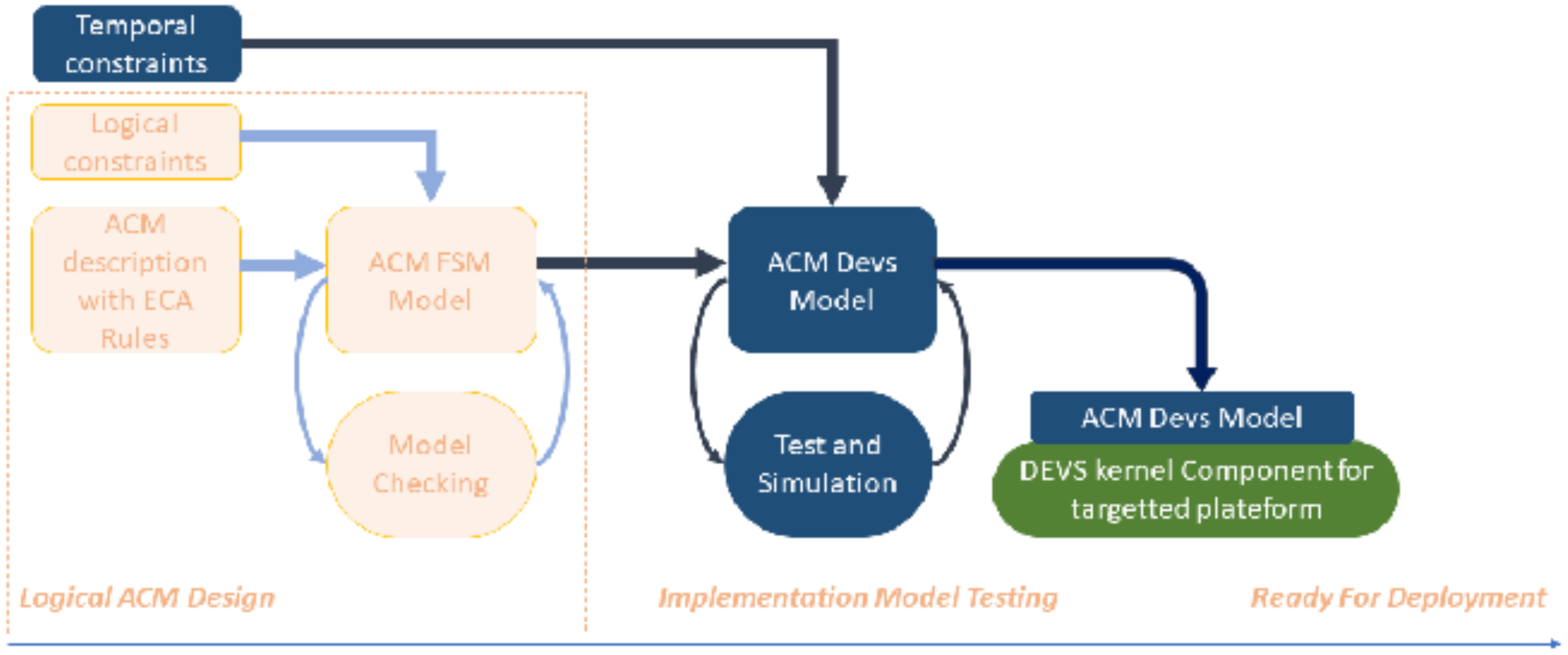


ACM (Actuation Conflict Manager) Assisted Design(2)

- Two type of conflicts in IoT systems are considered:
  - Direct conflicts between N app flows and one IoT device (actuator) which is affected through one of its feature.
  - Indirect conflicts between N app flows and M IoT devices that affect features related to the physical environment (noise for example).
- ACM DEVS Coupled Model [illegible]


Possible Direct Conflicts
App 1
App N
events
events
Actuator 1 (IoT Device 1)
Physical Environment
(a)
App 1
App N
events
events
Actuator 1 (IoT Device 1)
Possible Indirect Conflicts
Features
Actuator M (IoT Device M)
Physical Environment
(b)
App AM 1
App AM N
ACM CM
Actuation Conflicts Manager DEVS Coupled Model
Actuator AM 1 (IoT Device 1)
Actuator AM M (IoT Device M)
Physical Environment CM
(c)

DEVS for IoT systems ACM(1)



DEVS for IoT systems ACM(2)

Formal definition of Conflicts:

(a) Direct conflicts between N application flows and one IoT device (actuator) which is affected through one of its features(properties)

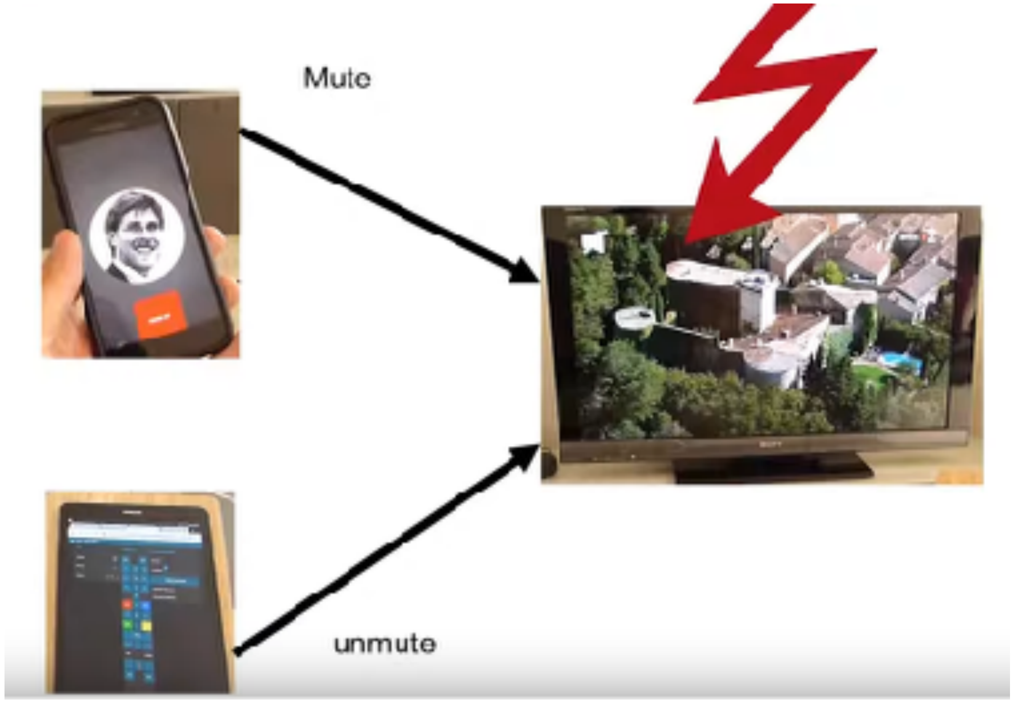
(b) Indirect conflicts between N application flows and M IoT devices that affect features related to the physical environment (noise for example)

Then definition of an ACM DEVS Coupled Model introduced in order to detect and validate by simulation the possible both direct and indirect conflicts resolution.

DEVS ACM Coupled Model interconnects a DEVS Synchronizer atomic model and a DEVS Logical Atomic Model

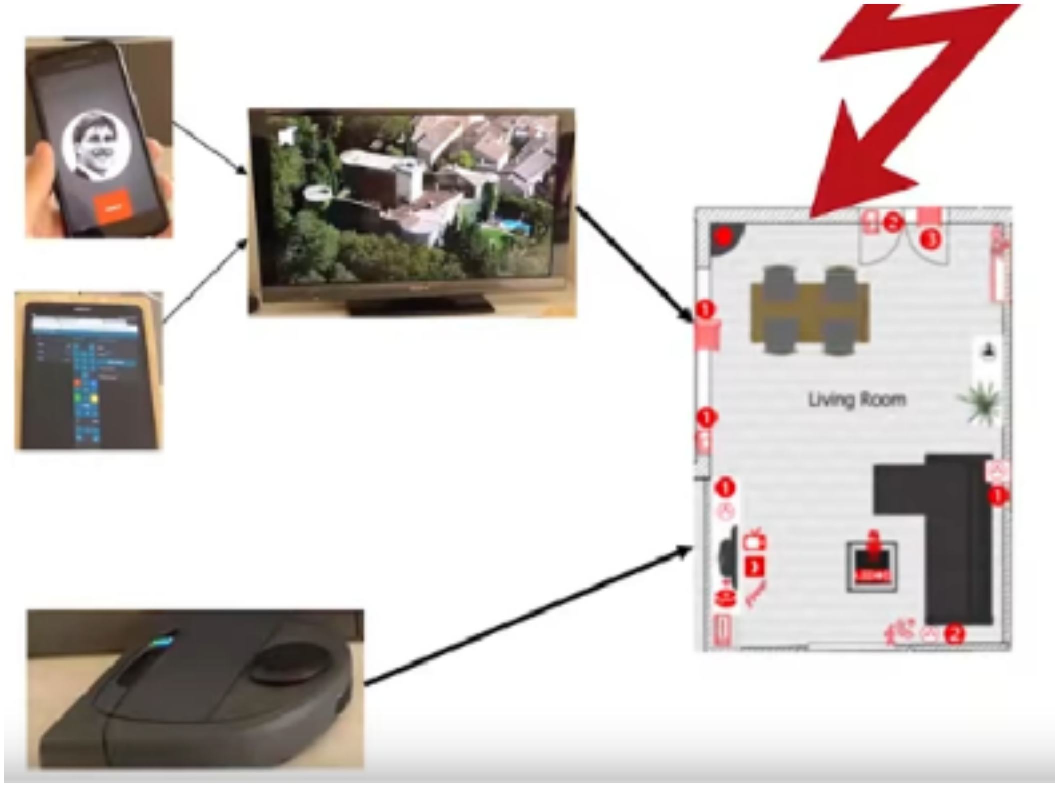
Case Study: Smart home scenario (2)

- Direct Conflict: The Phone app is trying to mute the TV while the remote controller is trying to unmute it.
  - Two app are trying to use the TV at the same time: This is a direct conflict.



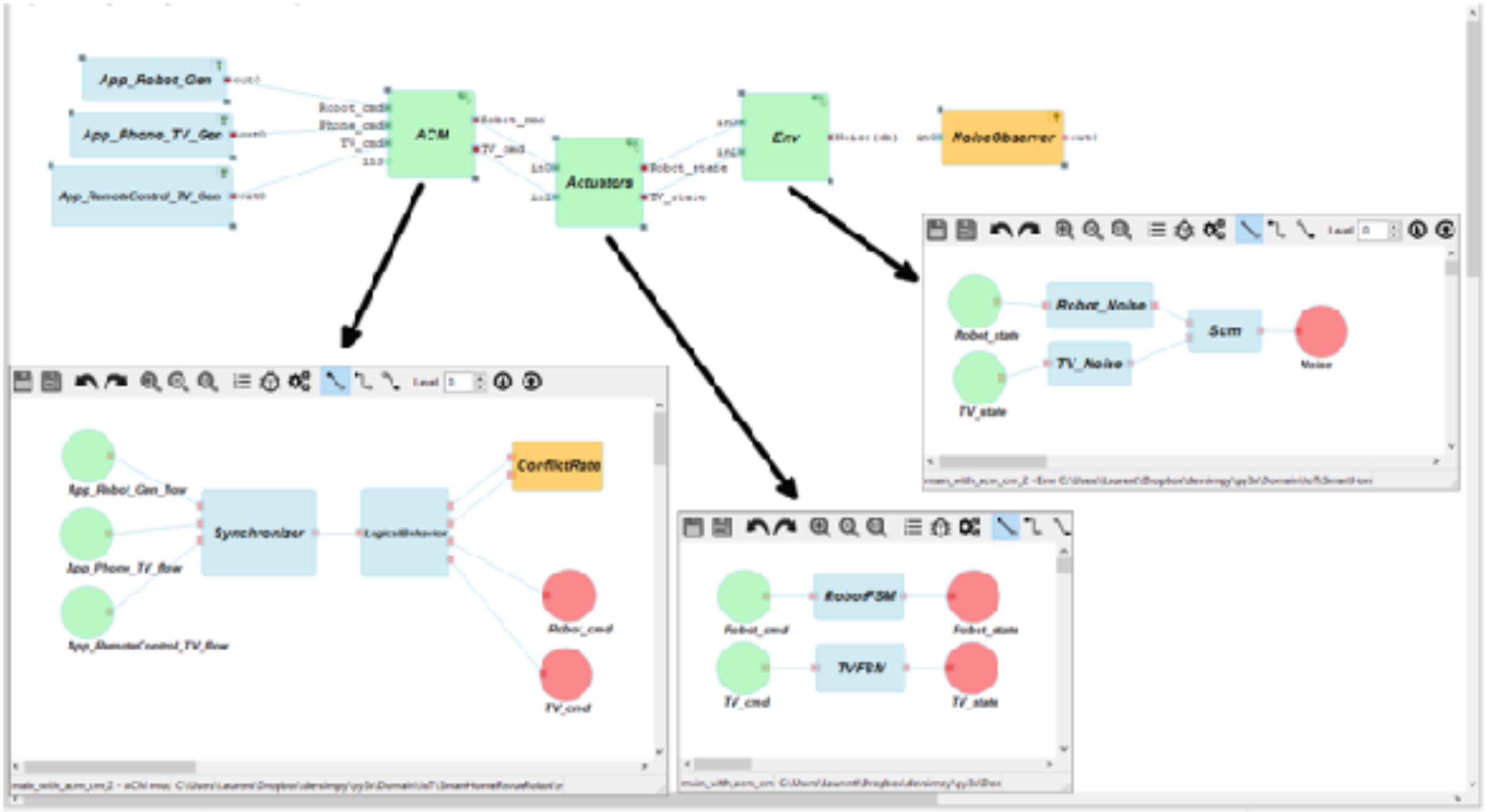
Case Study: Smart home scenario (3)

- Indirect Conflict: the vacuum cleaner is making too much noise while trying to clean the room. Its behavior is disturbing other app such as the phone calls.
  - The vacuum cleaner, the phone and the TV are modifying the sound property of the living room: This is an indirect conflict.

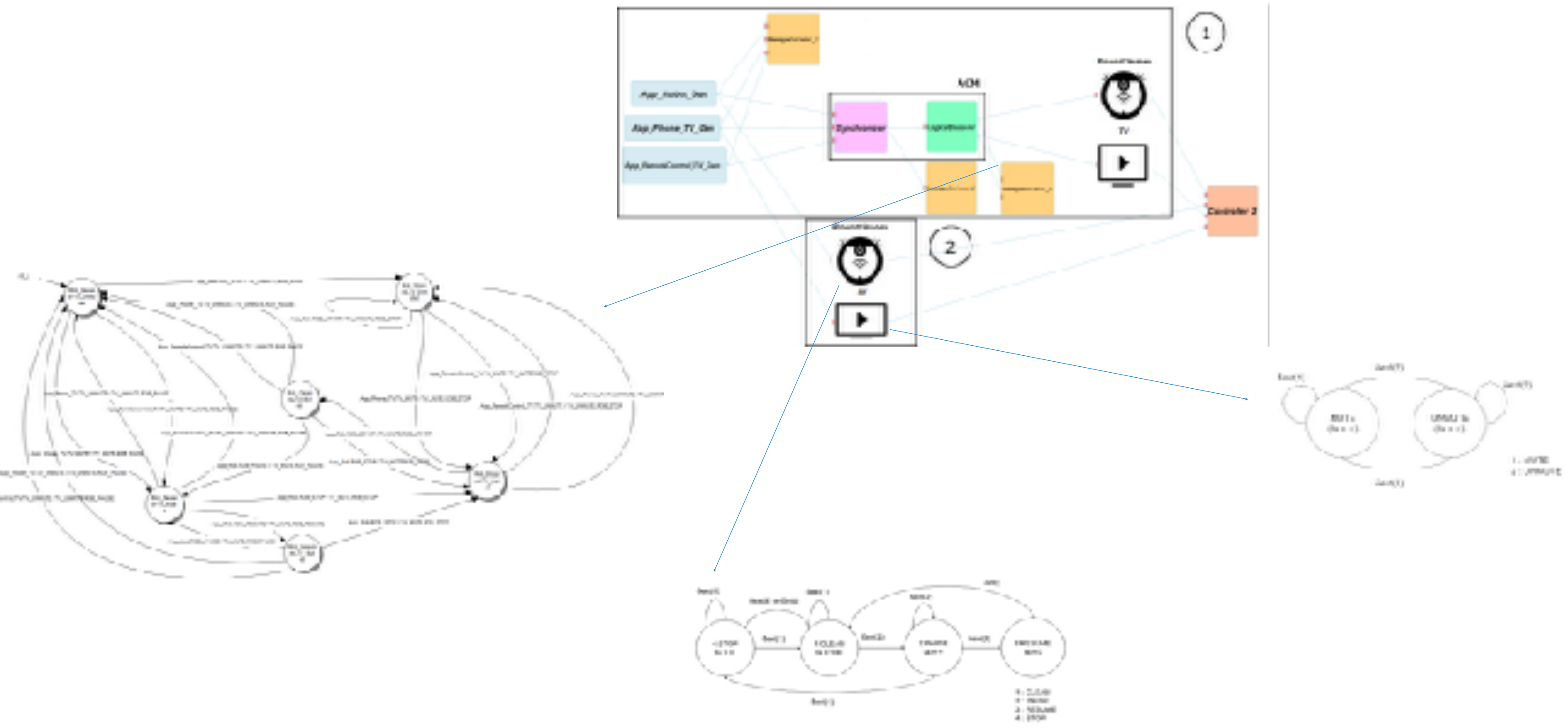


Case Study: Smart home scenario (4)

The *ACM* coupled model contains the *Synchronizer* and the *LogicalBehavior* models which are tracked by an observer atomic model (*ConflictRate* atomic model)dedicated to observe the conflict detection rate during the simulation.

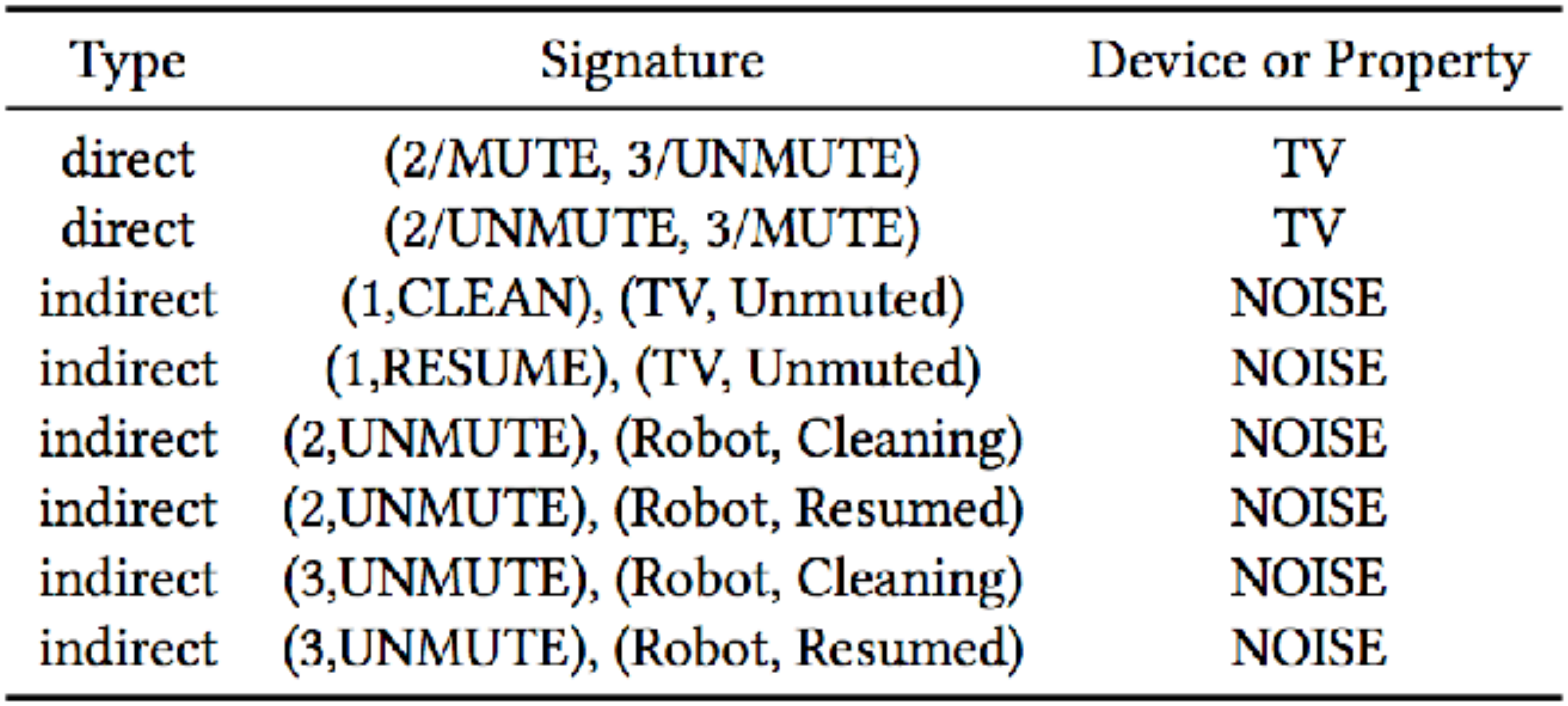


Case Study: Smart home scenario (6)



Case Study: Smart home scenario (7)

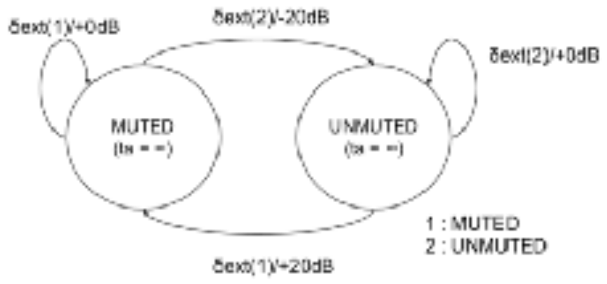
List of direct and indirect conflicts detected by the Logical Behavioral DEVS model



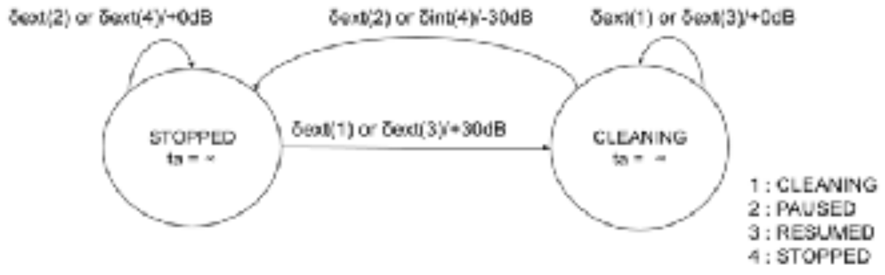
Type	Signature	Device or Property
direct	(2/MUTE, 3/UNMUTE)	TV
direct	(2/UNMUTE, 3/MUTE)	TV
indirect	(1,CLEAN), (TV, Unmuted)	NOISE
indirect	(1,RESUME), (TV, Unmuted)	NOISE
indirect	(2,UNMUTE), (Robot, Cleaning)	NOISE
indirect	(2,UNMUTE), (Robot, Resumed)	NOISE
indirect	(3,UNMUTE), (Robot, Cleaning)	NOISE
indirect	(3,UNMUTE), (Robot, Resumed)	NOISE

Case Study: Smart home scenario (8)

Env coupled Model dedicated to validate the efficiency of the ACM model by observing the generated noise using two atomic models : TV_noise and Robot_noise defined from the following FSM



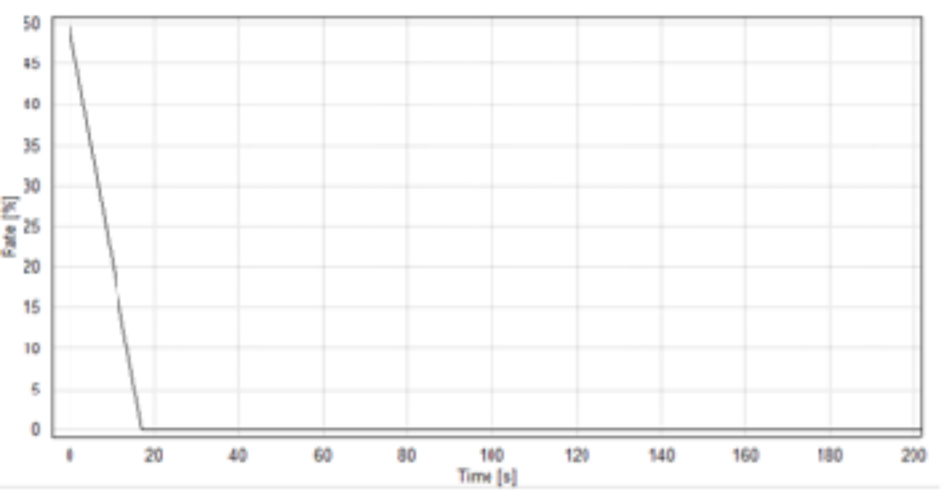
TV_noise



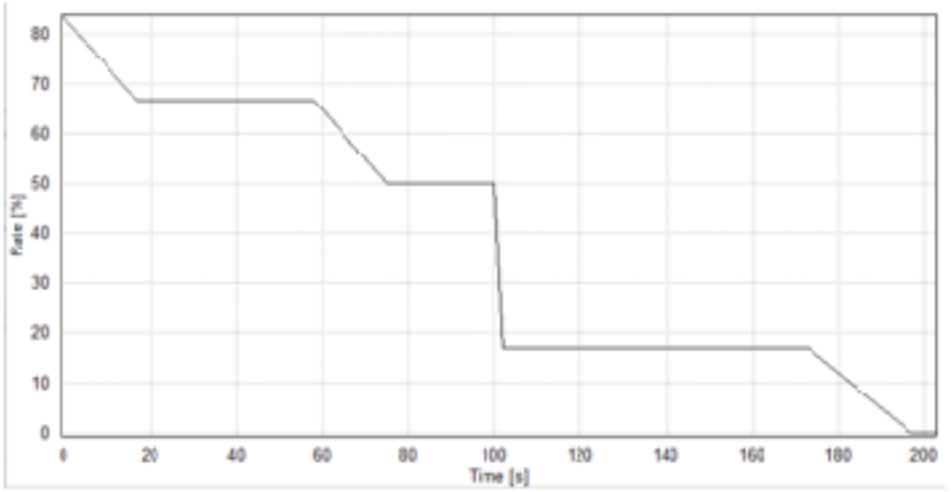
Robot_noise

Case Study: Smart home scenario (10)

Results (a) shows that 50% of the direct conflict are detected at the start of the simulation and the 50 other before the 20 simulation time step. Results (b) shows that indirect conflicts are more difficult t to be raised since we have to wait 200 time steps to obtain a 100% detection. These two results. Point out that the considered test patterns allow to highlight all the direct and indirect conflicts after 200 simulation time steps



(a)

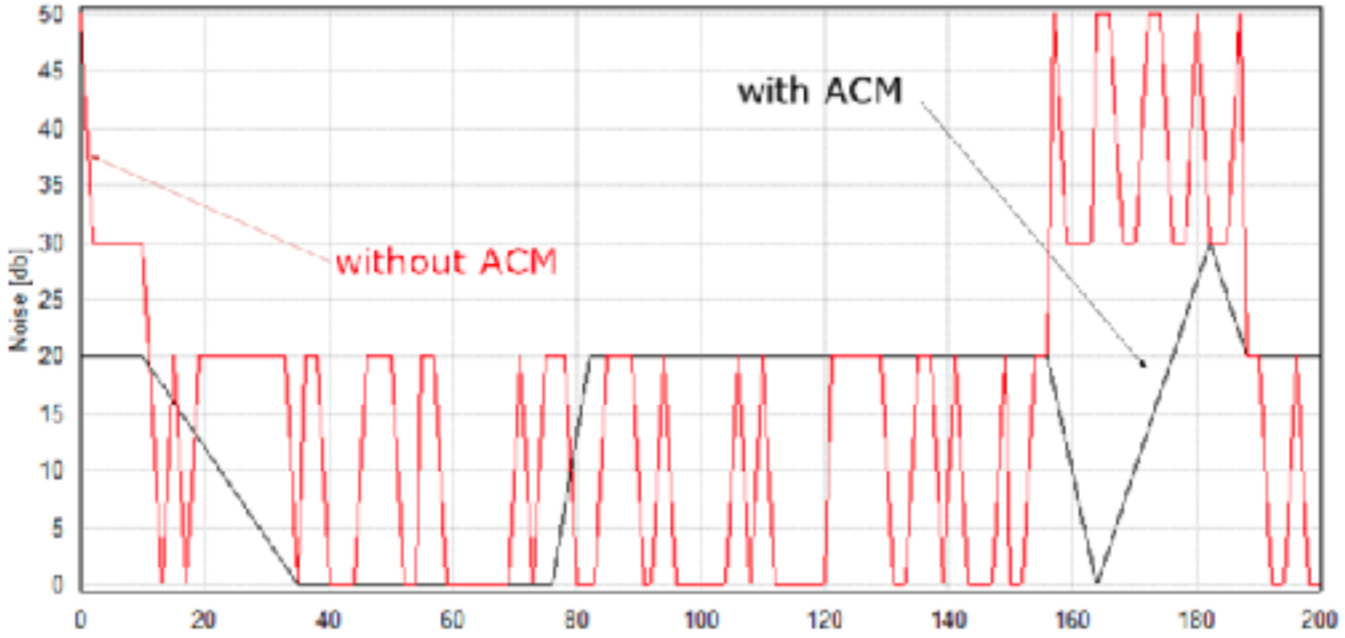


(b)

Case Study: Smart home scenario (11)

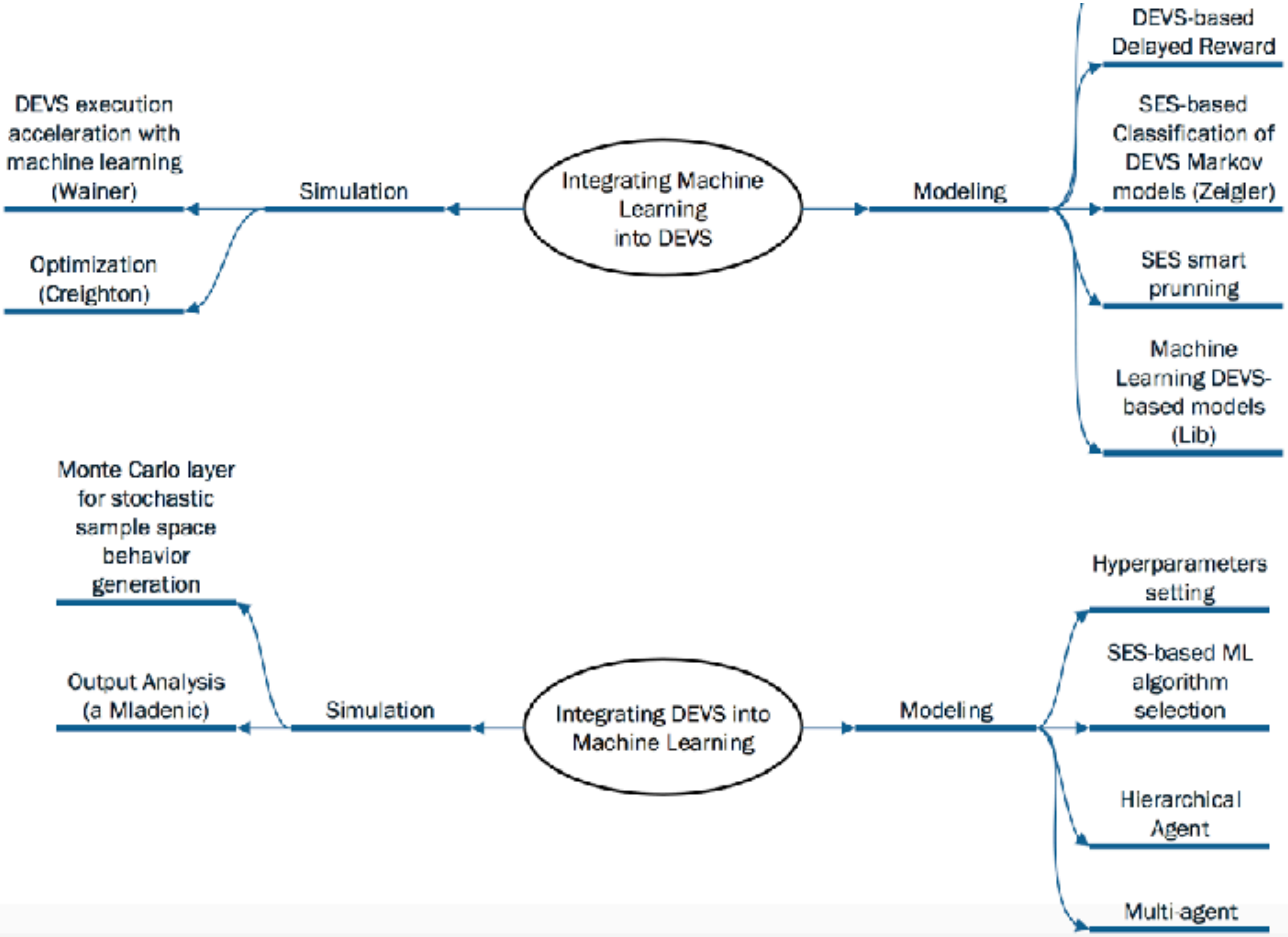
A conflict between the App_RemoteControl_TV and the App_Robot application flows is highlighted by a 50db (20db + 30db) noise level.

We show several conflicts (50db noise level) that appear between 0 and 20 and 160 and 180 simulation time steps when considering the simulation model without the ACM while all the conflicts have been resolved (the curve never reaches 50db) with the ACM component.



Simulation et IA

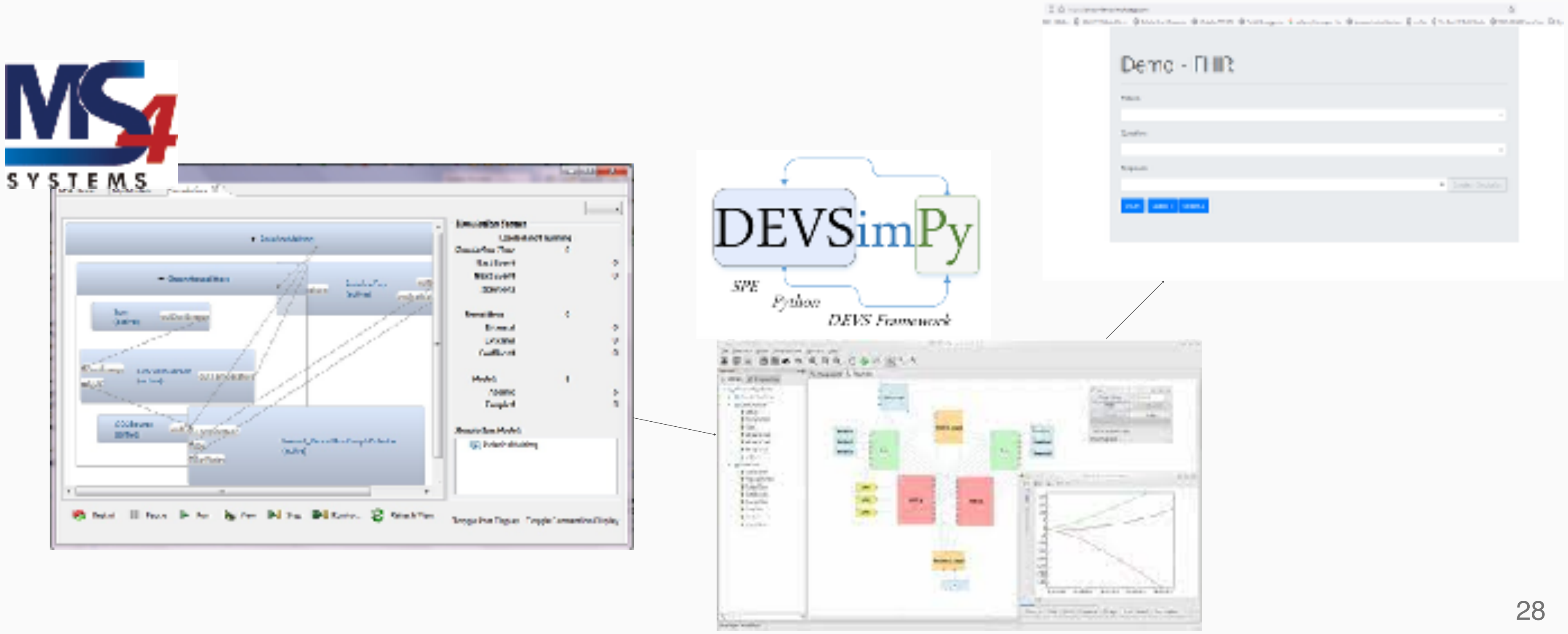
- Aide à la conception (utiliser la simulation pour la mise au point de modèles d’IA).
- Intégration de l’apprentissage au sein de modèles DEVS pour définir des comportements non déterministes.


Integrating Machine Learning into DEVS
Simulation
DEVS execution acceleration with machine learning (Wainer)
Optimization (Creighton)
Modeling
DEVS-based Delayed Reward
SES-based Classification of DEVS Markov models (Zeigler)
SES smart prunning
Machine Learning DEVS-based models (Lib)
Integrating DEVS into Machine Learning
Simulation
Monte Carlo layer for stochastic sample space behavior generation
Output Analysis (a Mladenic)
Modeling
Hyperparameters setting
SES-based ML algorithm selection
Hierarchical Agent
Multi-agent

Overview

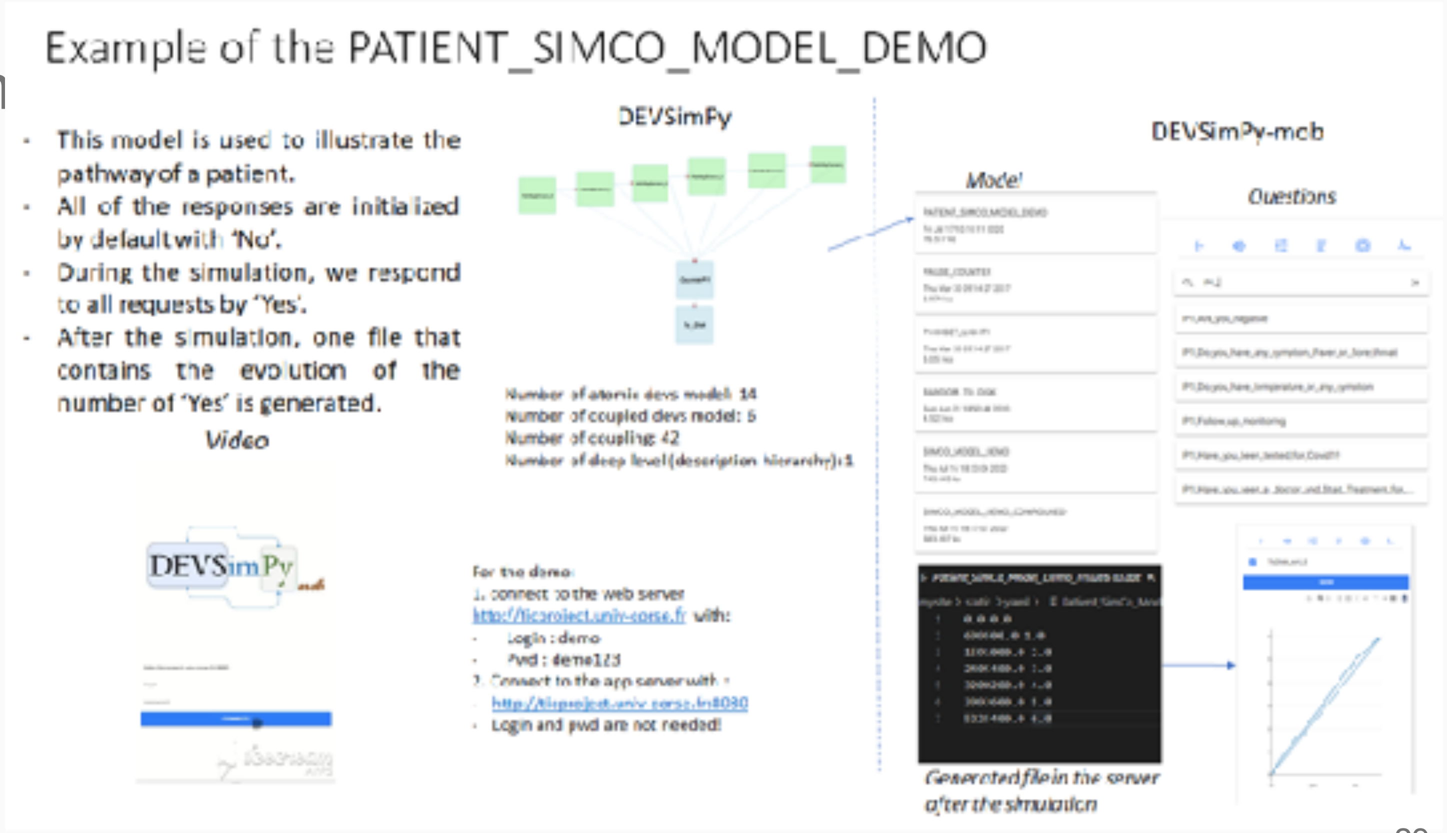
- SimCo collaboration aims to develop M&S projet by combining DEVSimPy and MS4Me softwares.
- MS4Me is used by RTSync in order to model and simulate systems according to industrial/institutional projects.
- DEVSimPy is used to develop simulation models and export them on web.
- **Goal of SimCo is to combine MS4Me and DEVSimPy to offer simulation models accessible by web interfaces.**

Workflow



Background (1)

DEVSimPy-m



Example of the PATIENT_SIMCO_MODEL_DEMO

- This model is used to illustrate the pathway of a patient.
- All of the responses are initialized by default with 'No'.
- During the simulation, we respond to all requests by 'Yes'.
- After the simulation, one file that contains the evolution of the number of 'Yes' is generated.

Video

DEVSimPy

Number of atomic devs model: 14
Number of coupled devs model: 5
Number of coupling: 42
Number of deep level (description hierarchy): 1

For the demo:
1. connect to the web server http://ticproject.univ-corse.fr with:
- Login : demo
- Pwd : demo123
2. Connect to the app server with :
- http://ticproject.univ-corse.fr:8080
- Login and pwd are not needed!

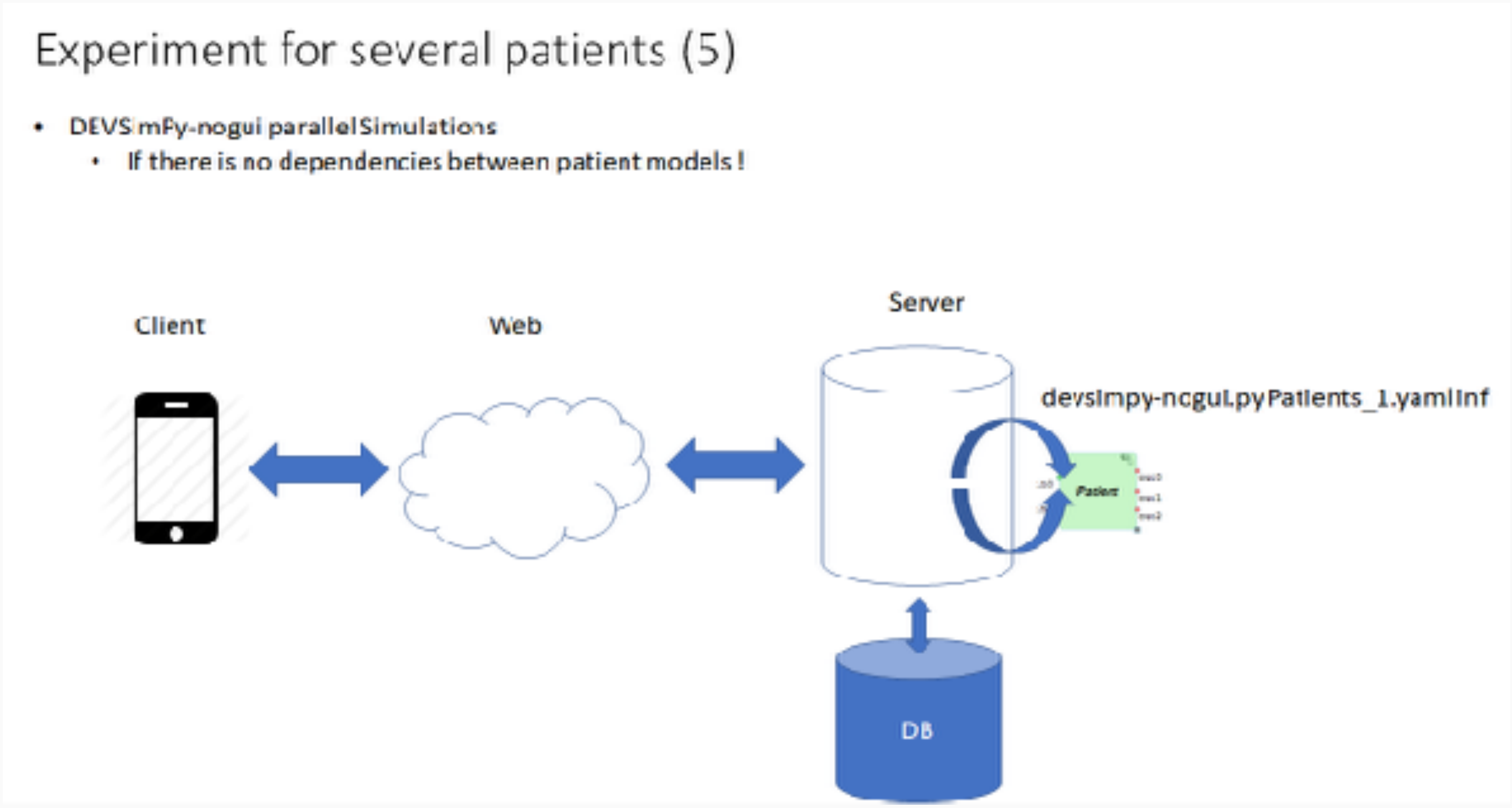
DEVSimPy-mob

Model

Questions

Generated file in the server after the simulation

Background (2)



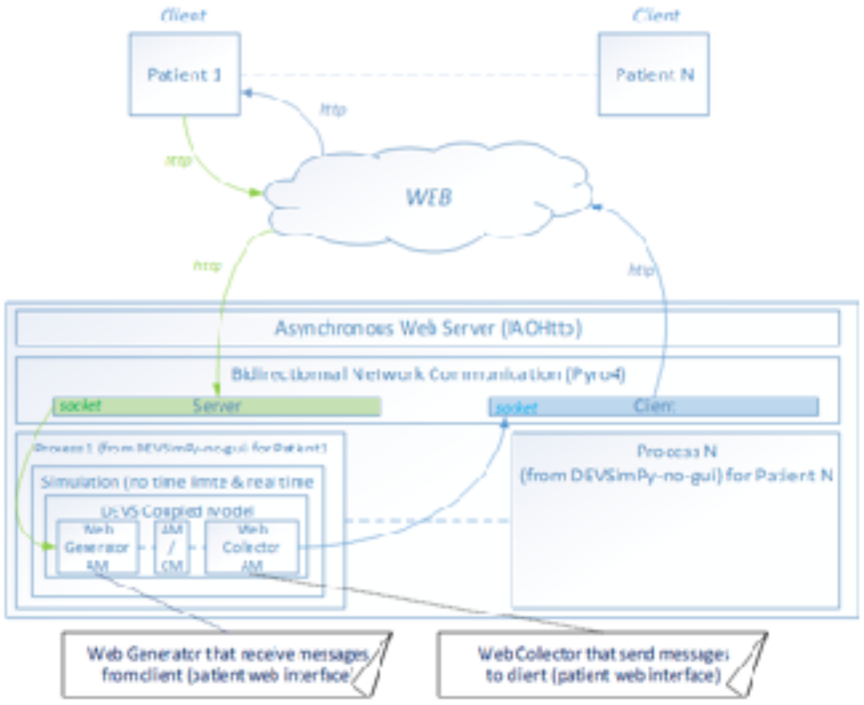
Experiment for several patients (5)

- DEVSimPy-nogui parallelSimulations
    - If there is no dependencies between patient models !

Background (3)

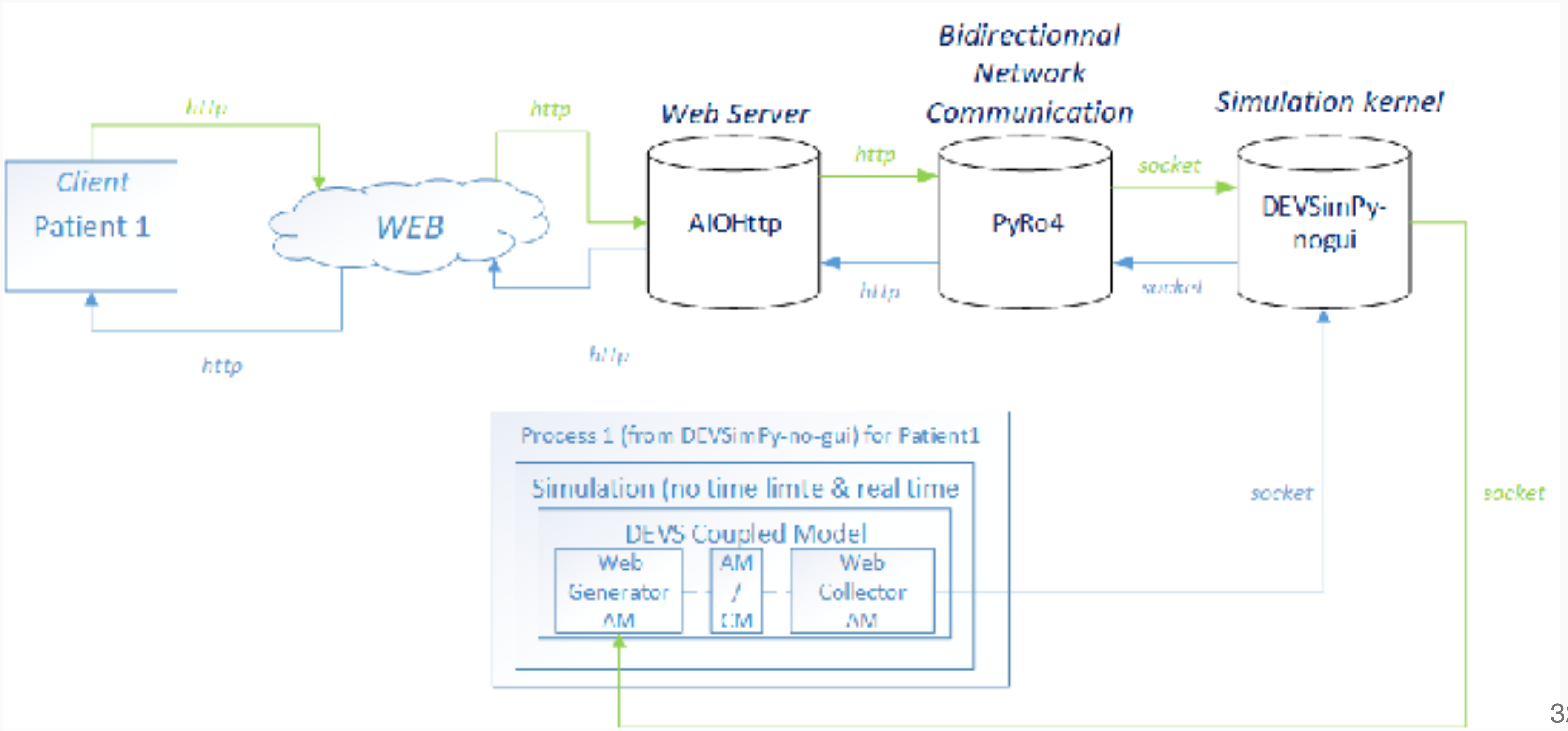
Idea for PGCHD Demo Architecture (1)

- Real/No Time Limite DEVS Simulations interact with HTTP clients in efficient and asynchronous way.
- Based on:
  - Asynchronous HTTP Client/Server for asyncio and Python (aiohttp)
  - Object network communication (PyRo)
  - DEVSimPy-nogui framework
- Proposed approach:
  - Patient have a web interface (mobile) to populate the DEVS simulation model based on questions and/or informations from database....
  - For each patient, one real time DEVS simulation is performed and communication in biderctionnal way with the client is possible by using a WebGeneratorand WebCollectoratomic models.



Client
Client
Patient 1
Patient N
http
http
WEB
http
http
Asynchronous Web Server (AOHttp)
Bidirectionnal Network Communication (Pyro4)
socket
Server
socket
Client
Process 1 (from DEVSimPy-no-gui for Patient1
Simulation (no time limite & real time
DEVS Coupled Model
Web Generator AM
AM / CM
Web Collector AM
Process N (from DEVSimPy-no-gui) for Patient N
Web Generator that receive messages from client (patient web interface)
WebCollector that send messages to client (patient web interface)

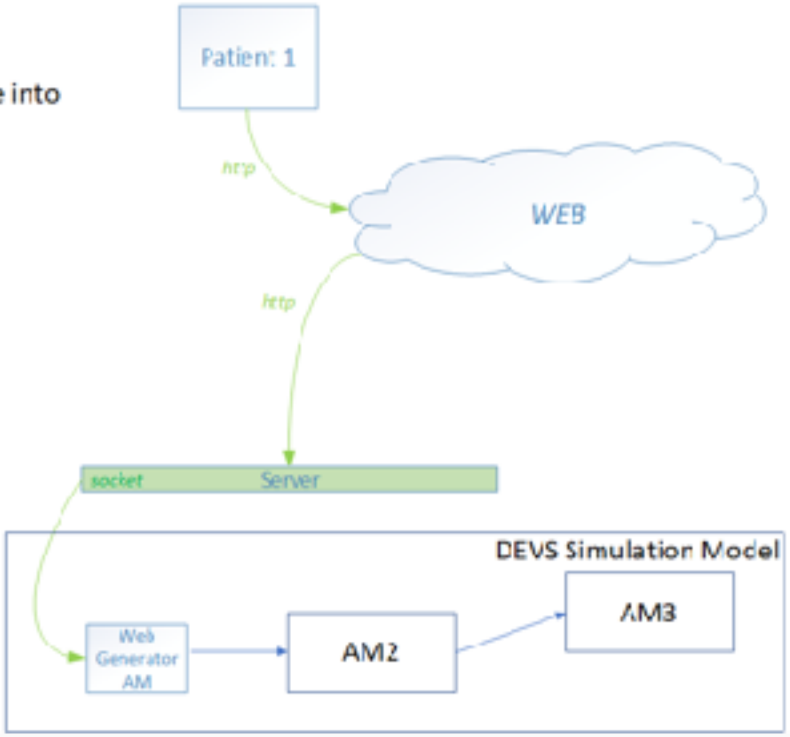
Background (4)



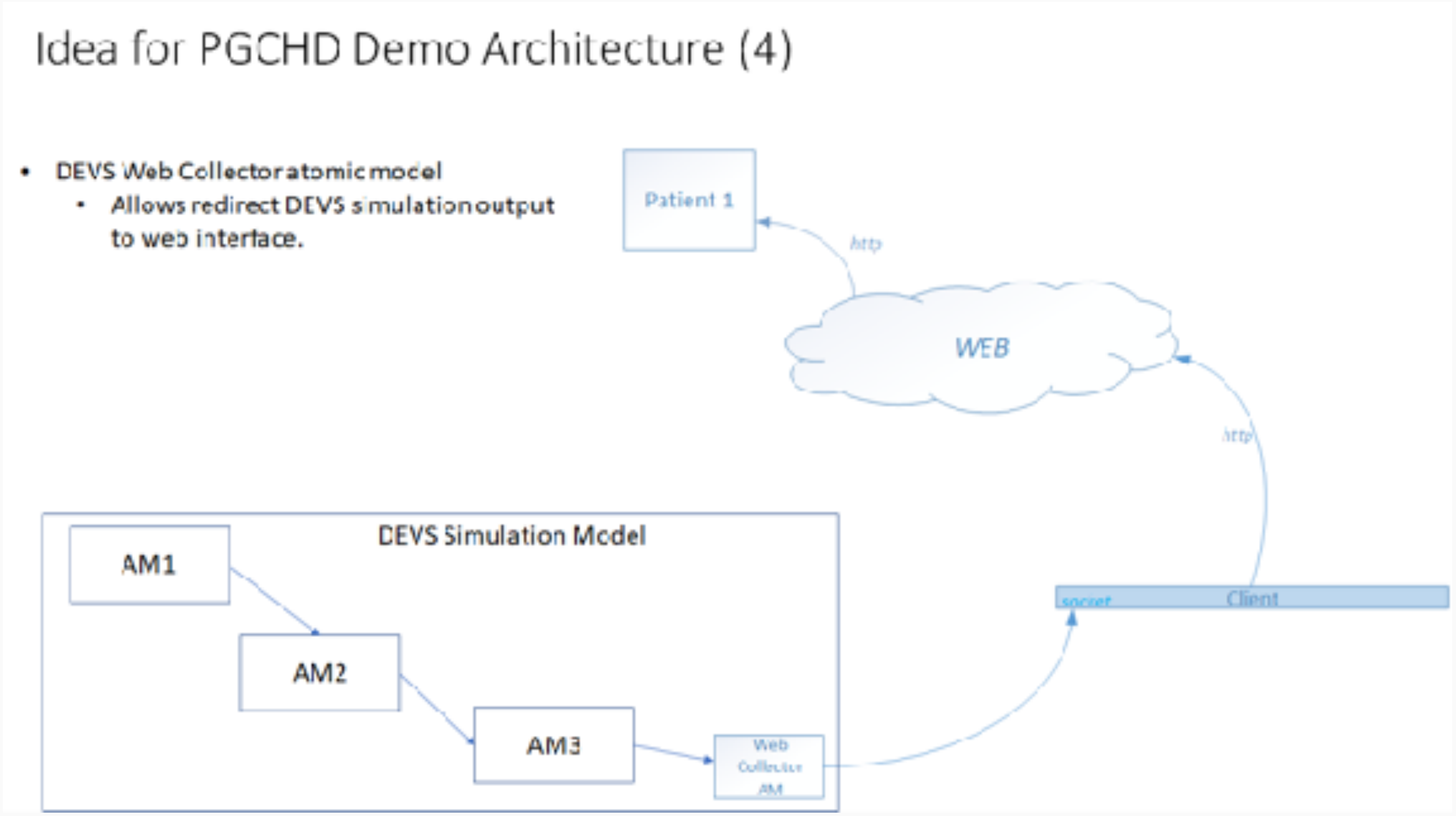
Background (5)

Idea for PGCHD Demo Architecture (3)

- DEVS Web Generator atomic model
  - Allows to inject data from web interface into DEVS simulation model.



Background (6)



Idea for PGCHD Demo Architecture (4)

- DEVS Web Collector atomic model
  - Allows redirect DEVS simulation output to web interface.

Autre perspective de Recherche : Web-Based DEVS

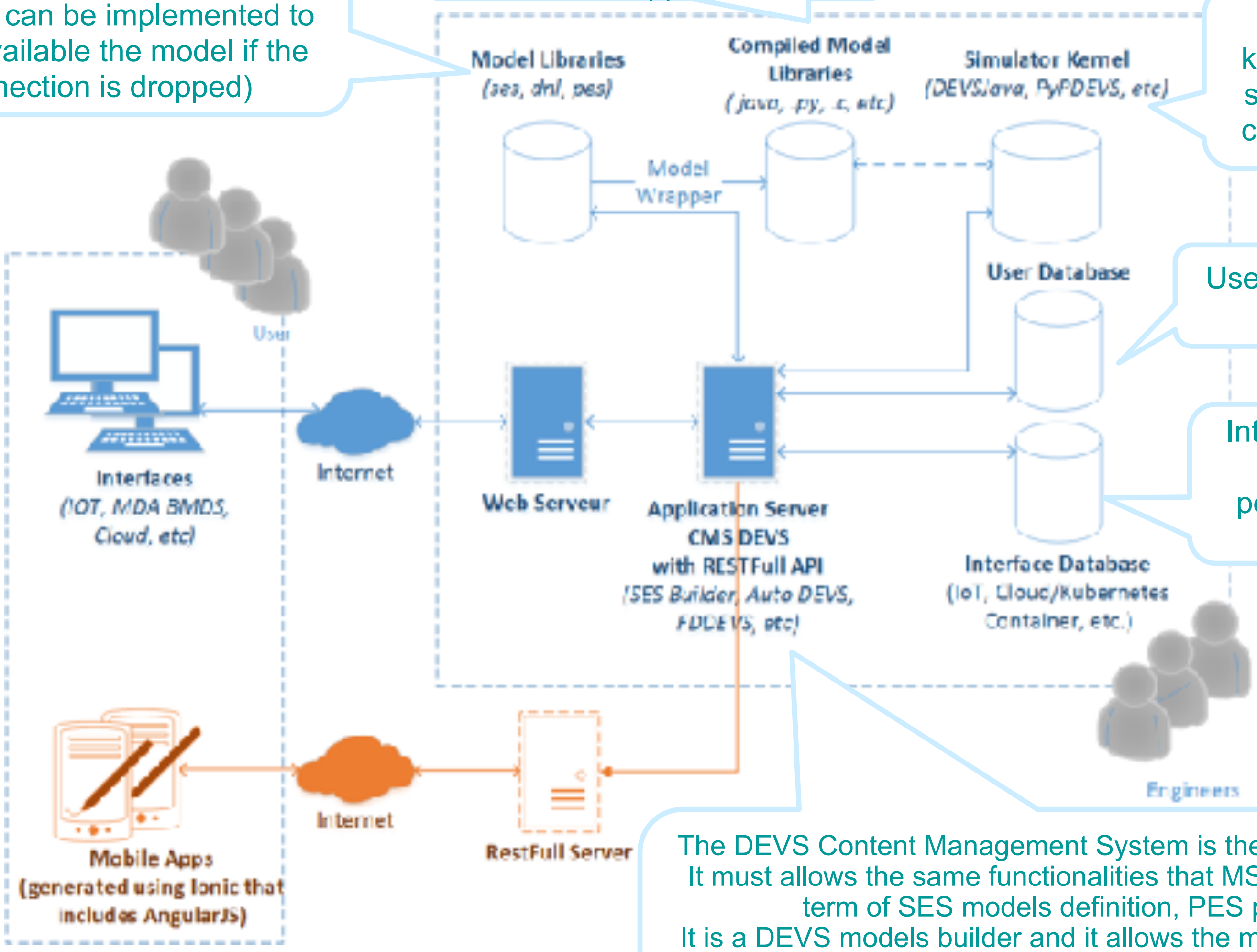
Ses, dnl, pes models have been defined using the DEVS CMS (a cache can be implemented to make available the model if the connection is dropped)

Compiled models are generated using a model wrapper

The simulation kernels allow the simulation of the compiled models

User profile can be store on db

Interface view can be stored to personalized the common GUI



The DEVS Content Management System is the key of the architecture. It must allows the same functionalities that MS4Me and DEVSimPy in term of SES models definition, PES possibility, etc.
It is a DEVS models builder and it allows the models management, the generation of interfaces to configure and interact with the simulation models, the generation of the mobile app to test models in real case with sensors and actuators, and more over.