Modeling and Simulation of Complex Systems with DEVSimPy

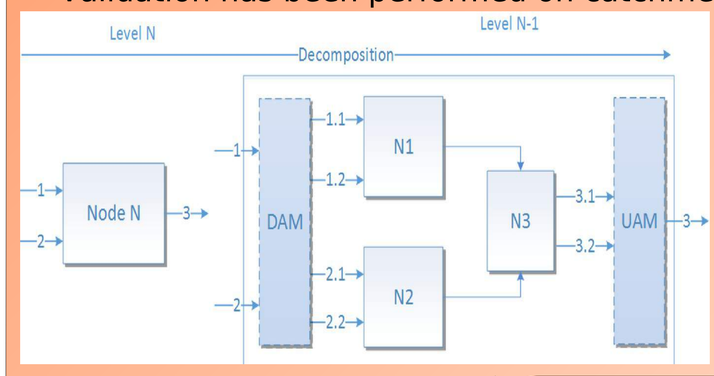
Laurent Capocchi and Jean-François Santucci
University of Corsica
SPE Laboratory UMR CNRS 6134
Quartier Grimaldi, BP 52
20250 Corte, France

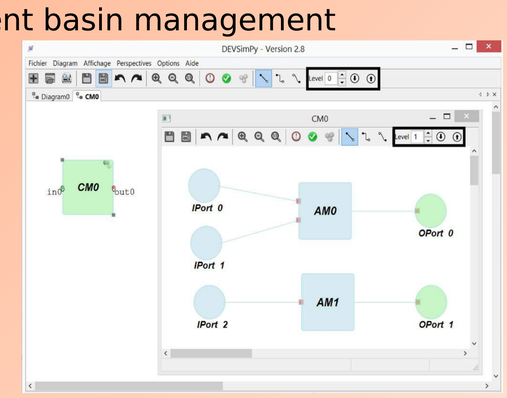
Abstraction Hierarchy

· The abstraction level of a model is a corollary and dependent notion of perspective and determines the amount of information contained therein

· DEVSimPy allows the simulation of models involving several levels of abstraction through the association of downward and upward functions to two new atomic models: DAM (Downward Atomic Model) and UAM (Upward Atomic Model)

· Validation has been performed on catchment basin management



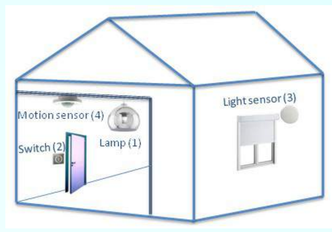


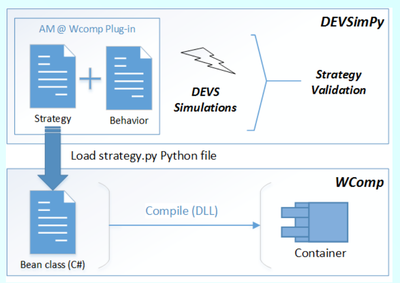
Ubiquitous Systems Modeling

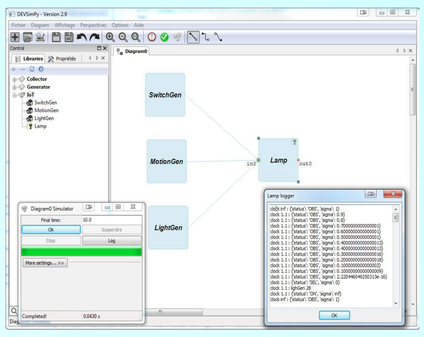
· The main problem is to propose a management adapted to the composition of applications in ubiquitous computing

· We propose the definition of a modeling and simulation scheme based on a discrete-event formalism in order to specify at the very early phase of the design of an ambient system:

- the behavior of the components involved in the ambient system to be implemented;
- the possibility to define a set of strategies which can be implemented in the execution machine

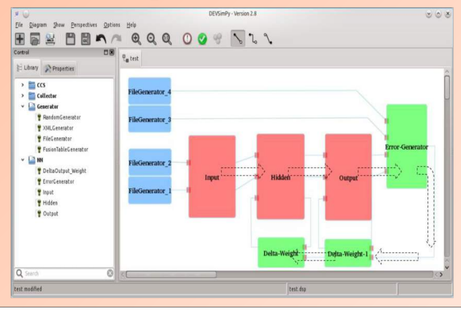






ANN M&S

- A **generic solution** for the Comparative and Concurrent Simulation (CCS) within DEVS formalism
- An Artificial Neural Network (ANN) representation in DEVS that enables a **modular optimization** (architecture, learning algorithms)
- An efficient **ANN configuration** based on comparative and concurrent simulations
- Early efficient **fault detection** in Wound-Rotor Induction Machine (WRIM) using DEVS and ANN



Abstraction Hierarchy & ANN M&S

DEVSimPy

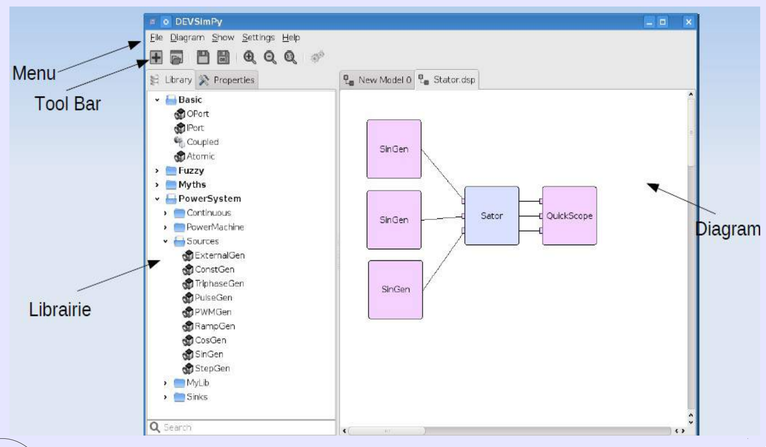
Discrete Event system Specification Python Simulator

DEVSimPy — SPE — Python — DEVS Framework

- Introduced by B.P. Zeigler in the early 70's: formalism for modeling discrete-event systems in a hierarchical and modular way
- With DEVS, a model of a large system can be decomposed into smaller component models with coupling specification between them
- DEVS defines two kinds of models: atomic and coupled models

  (i) *atomic models* represent the basic models providing specifications for the dynamics of a sub-system using function transitions;

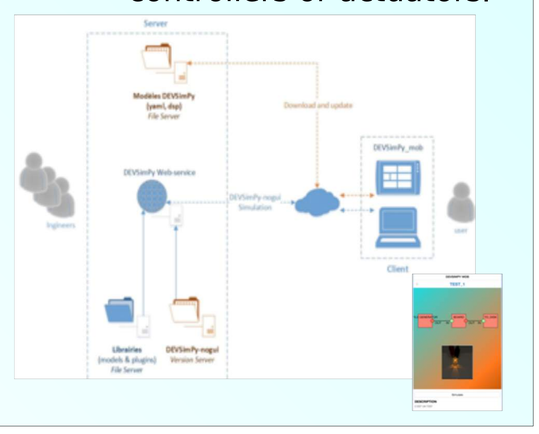
  (ii) *Coupled models* describe how to couple several component models together to form a new model
- DEVS provide an automatic simulation based on time synchronization and message propagation
- DEVSimPy integrates extensions of DEVS allowing to deal with numerous applications



Ubiquitous M&S & DEVSimPy-Mob

DEVSimPy-Mob

Multi-platforme mobile application aimed to manage discrete event simulations obtained from DEVS models associated with connected objects such as board computers, sensors, controllers or actuators.



Continuous System Modeling

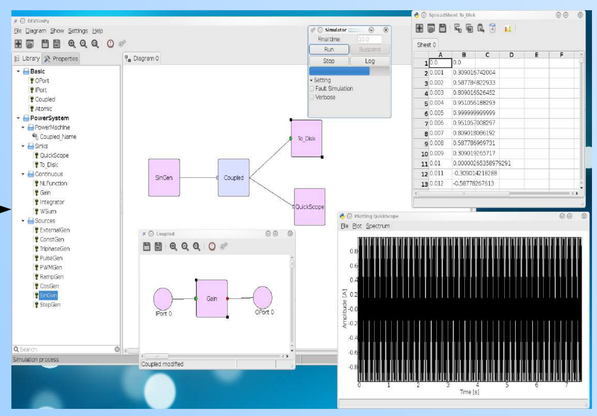
- DEVS Models work with an infinite number of states which is useful for numerical integration
- QSS *(Quantized State System)* use a quantization function to transform a continuous system into a DEVS system with piecewise constant input and output trajectories
- BFS DEVS (Behavioral Fault Simulator for DEVS) use a concurrent and comparative algorithm

QSS DEVS & BFS DEVS

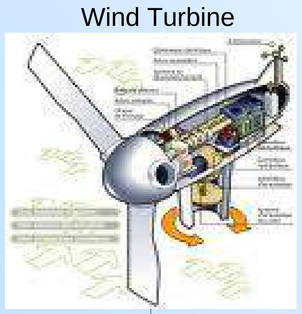
Agile DEVS

Dynamic DEVS & Parallel DEVS

QSS-DEVS
Modeling and Fault Simulation
With DEVSImPy



Wind Turbine



Mathematical Modeling

$$
\begin{cases}
v_{as}(t) &= r_s.i_{as}+L_s\frac{d}{dt}i_{as}-\frac{L_{ms}}{2}\left[\frac{d}{dt}i_{bs}+\frac{d}{dt}i_{cs}\right]+L_{sr}\frac{d}{dt}\left[i_{ar}.\cos(\theta_r(t))+i_{br}.\cos(\theta_r(t)+\frac{2\pi}{3})+i_{cr}.\cos(\theta_r(t)-\frac{2\pi}{3})\right]\\
v_{bs}(t) &= r_s.i_{bs}+L_s\frac{d}{dt}i_{bs}-\frac{L_{ms}}{2}\left[\frac{d}{dt}i_{as}+\frac{d}{dt}i_{cs}\right]+L_{sr}\frac{d}{dt}\left[i_{ar}.\cos(\theta_r(t)-\frac{2\pi}{3})+i_{br}.\cos(\theta_r(t))+i_{cr}.\cos(\theta_r(t)+\frac{2\pi}{3})\right]\\
v_{cs}(t) &= r_s.i_{cs}+L_s\frac{d}{dt}i_{cs}-\frac{L_{ms}}{2}\left[\frac{d}{dt}i_{as}+\frac{d}{dt}i_{bs}\right]+L_{sr}\frac{d}{dt}\left[i_{ar}.\cos(\theta_r(t)+\frac{2\pi}{3})+i_{br}.\cos(\theta_r(t)-\frac{2\pi}{3})+i_{cr}.\cos(\theta_r(t))\right]\\
v_{ar}(t) &= r_r.i_{ar}+L_r\frac{d}{dt}i_{ar}-\frac{L_{mr}}{2}\left[\frac{d}{dt}i_{br}+\frac{d}{dt}i_{cr}\right]+L_{rs}\frac{d}{dt}\left[i_{as}.\cos(\theta_r(t))+i_{bs}.\cos(\theta_r(t)-\frac{2\pi}{3})+i_{cs}.\cos(\theta_r(t)+\frac{2\pi}{3})\right]\\
v_{br}(t) &= r_r.i_{br}+L_r\frac{d}{dt}i_{br}-\frac{L_{mr}}{2}\left[\frac{d}{dt}i_{ar}+\frac{d}{dt}i_{cr}\right]+L_{rs}\frac{d}{dt}\left[i_{as}.\cos(\theta_r(t)+\frac{2\pi}{3})+i_{bs}.\cos(\theta_r(t))+i_{cs}.\cos(\theta_r(t)-\frac{2\pi}{3})\right]\\
v_{cr}(t) &= r_r.i_{cr}+L_r\frac{d}{dt}i_{cr}-\frac{L_{mr}}{2}\left[\frac{d}{dt}i_{ar}+\frac{d}{dt}i_{br}\right]+L_{rs}\frac{d}{dt}\left[i_{as}.\cos(\theta_r(t)-\frac{2\pi}{3})+i_{bs}.\cos(\theta_r(t)+\frac{2\pi}{3})+i_{cs}.\cos(\theta_r(t))\right]\\
T_e(t)-T_l &= J.\frac{d}{dt}\Omega(t)+f.\Omega(t)\\
\Omega(t) &= \frac{d}{dt}\theta_r(t)
\end{cases}
$$

Non linear system

Structural Anthropology: Myth Analysis

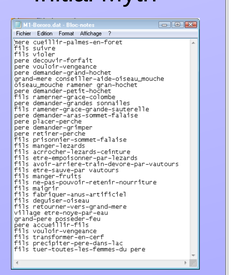
· In 1955 C. Levi Strauss introduced the Structural Anthropology Theory applied to Myth Analysis

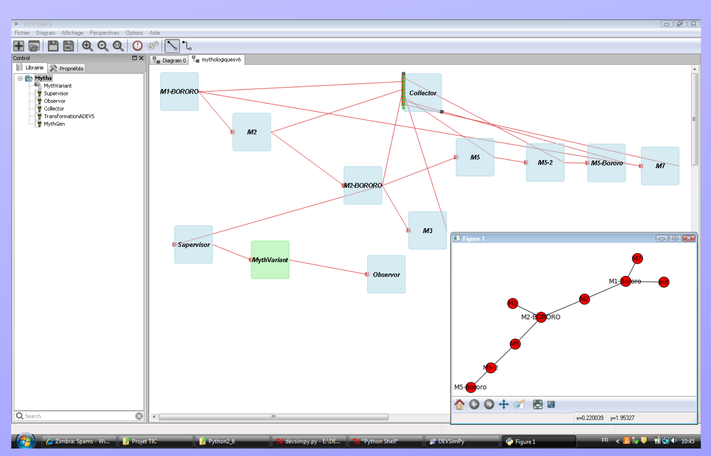
· Claude Levi Strauss explained in his books how the meaning of a given myth can emerge from a set of transformations between myths through generation of myths and interpretation according to different codes

· Development of an object oriented modeling and simulation software based on the DEVS formalism involving dynamic variable structure simulation
The two steps are involved in the analysis of myths :

- the definition of a variable structure DEVS modeling scheme to perform the transformations of myth and generate a set of myths (Dynamic DEVS)
- the simulation of different codes through a given myth already obtained after the first step (Neural Network learning for code interpretations)

Initial Myth





Myth analysis according to Levi-Strauss Structural Anthropology using DEVSIMPY

Examples of generated myths