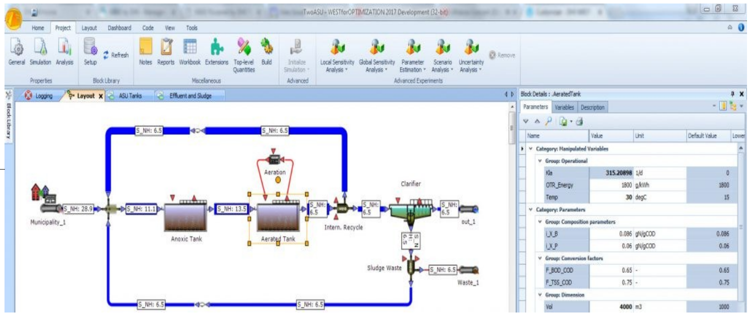
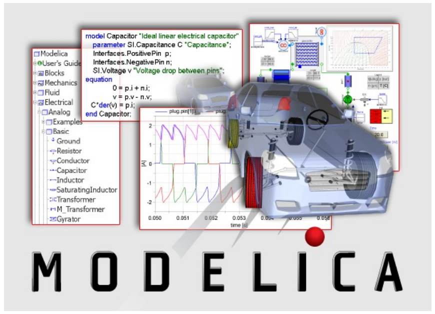
# Hans Vangheluwe   Modelling, Simulation and Design Lab (MSDL)

## (Domain-Specific) Modelling



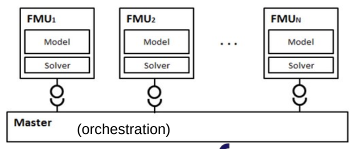
```
CLASS Capacitor
"
 Ideal Electrical Capacitor
"
EXTENDS ElectricalTwoTerminal WITH
{:
 parameters <-
  {OBJ C "capacitance of the capacitor": Capacitance};
 equations  <-
  {parameters.C*DERIV(state.v, [independent.t]) = state.i};
:};
```

MSL−USER



```
model Capacitor "Ideal linear electrical capacitor"
  parameter SI.Capacitance C "Capacitance";
  Interfaces.PositivePin p;
  Interfaces.NegativePin n;
  SI.Voltage v "Voltage drop between pins";
equation
  0 = p.i + n.i;
  v = p.v - n.v;
  C*der(v) = p.i;
end Capacitor;
```

## (co-)Simulation / Execution



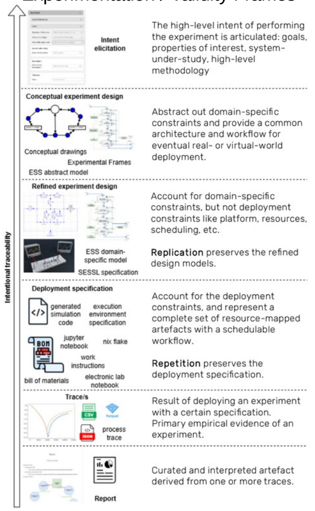
(multi-component)

FUNCTIONAL MOCK-UP INTERFACE

```
void BouncingBallClass :: ComputeState(void)
{
  _D_x_  = _v_hor_;
  _D_y_  = _vy_;
  _D_vy_ = (_y_ > _R_) ? -_g_ : -_K_/_M_*(_y_ - _R_) -_D_/_M_*_vy_;
}
```

MSL−EXEC

## Experimentation / Validity Frames



The high-level intent of performing the experiment is articulated: goals, properties of interest, system-under-study, high-level methodology

Abstract out domain-specific constraints and provide a common architecture and workflow for eventual real- or virtual-world deployment.

Account for domain-specific constraints, but not deployment constraints like platform, resources, scheduling, etc.

**Replication** preserves the refined design models.

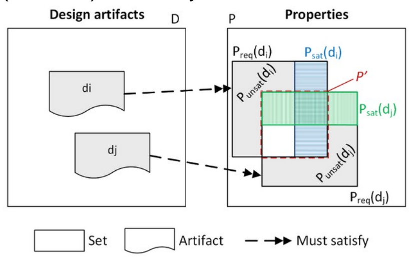
Account for the deployment constraints, and represent a complete set of resource-mapped artefacts with a schedulable workflow.

**Repetition** preserves the deployment specification.

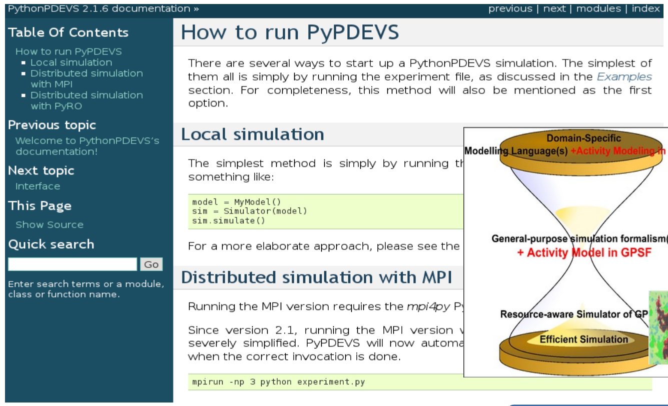
Result of deploying an experiment with a certain specification. Primary empirical evidence of an experiment.

Curated and interpreted artefact derived from one or more traces.

(multi-view) consistency



(multi-component)



PythonPDEVS 2.1.6 documentation » previous | next | modules | index

Table Of Contents
- How to run PyPDEVS
  - Local simulation
  - Distributed simulation with MPI
  - Distributed simulation with PyRO

Previous topic
Welcome to PythonPDEVS's documentation!

Next topic
Interface

This Page
Show Source

Quick search
Go
Enter search terms or a module, class or function name.

How to run PyPDEVS

There are several ways to start up a PythonPDEVS simulation. The simplest of them all is simply by running the experiment file, as discussed in the *Examples* section. For completeness, this method will also be mentioned as the first option.

Local simulation

The simplest method is simply by running th something like:

```
model = MyModel()
sim = Simulator(model)
sim.simulate()
```

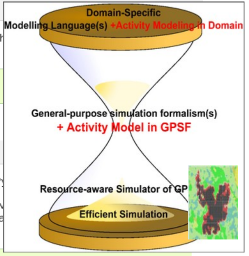
For a more elaborate approach, please see the

Distributed simulation with MPI

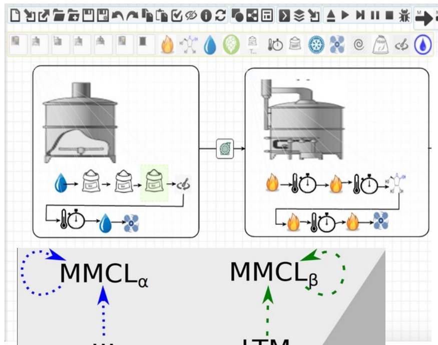
Running the MPI version requires the *mpi4py* P

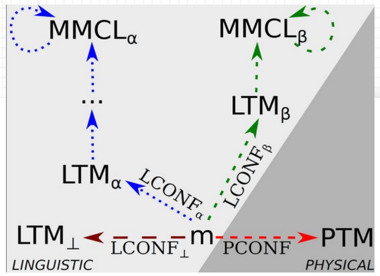
Since version 2.1, running the MPI version severely simplified. PyPDEVS will now automa when the correct invocation is done.

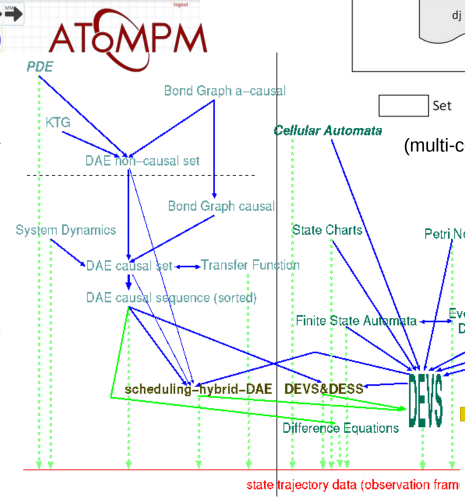
```
mpirun -np 3 python experiment.py
```


Domain-Specific
Modelling Language(s) +Activity Modelling in Domain
General-purpose simulation formalism(s)
+ Activity Model in GPSF
Resource-aware Simulator of GP
Efficient Simulation

## Meta-Modelling and Model (graph) Transformation



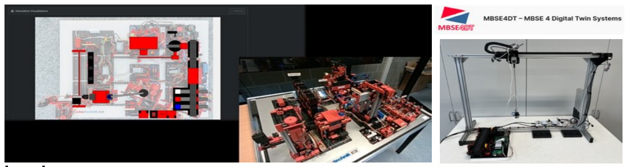


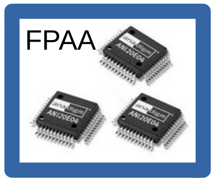


Formalism Transformation Graph (FTG) + PM

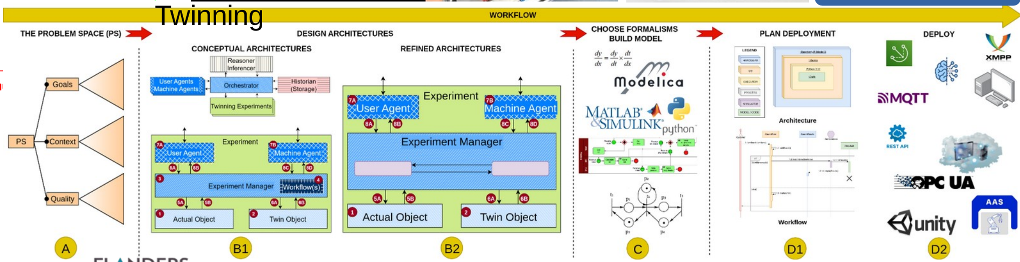
Multiverse / Modelverse
(model management: operation-based versioning, live modelling, etc)

# Multi-Paradigm Modelling (MPM)





Twinning



Universiteit Antwerpen
Ansymo Antwerp Systems & Software Modelling University of Antwerp
Nexor Cyber-Physical Systems University of Antwerp
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
McGill
MPM4CPS
cost EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY