Expliciet Modelling of DEVS experiments in AToMPM

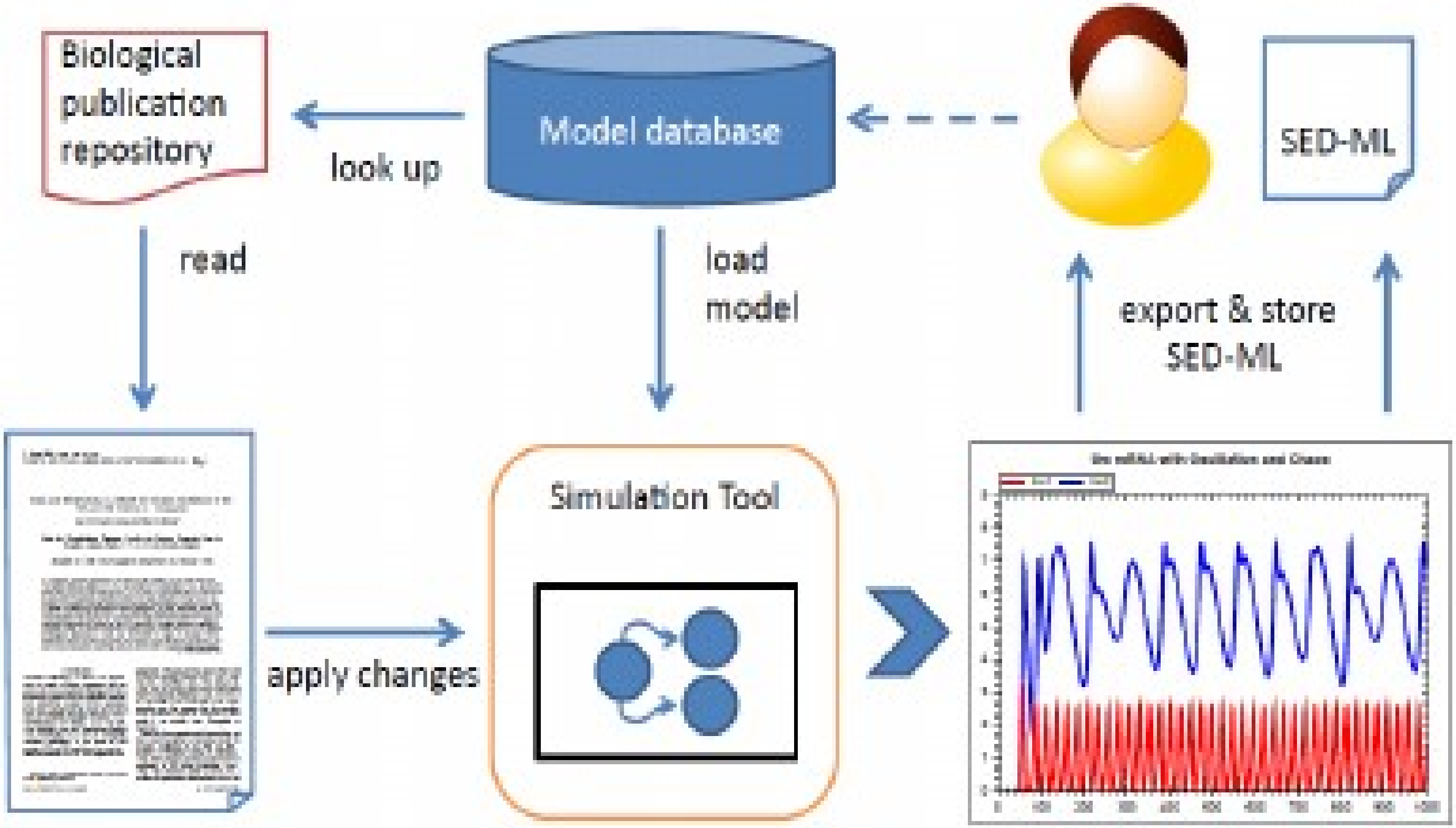
Timmy Nelen

Problem

- Reproduction of simulation experiments: possible
- Modify simulation experiments: possible
- Running modified simulation experiments: ... Not so much
- Knowledge about the system required
- Solution?

SED-ML

- Simulation Experiment Description Markup Language
- Encode the description of a SE in XML
- Exchangeable, reusable, readable
- Ideally provided together with publications
- Allows easy modification of experiments without breaking everything



Biological publication repository

look up

Model database

read

load model

export & store SED-ML

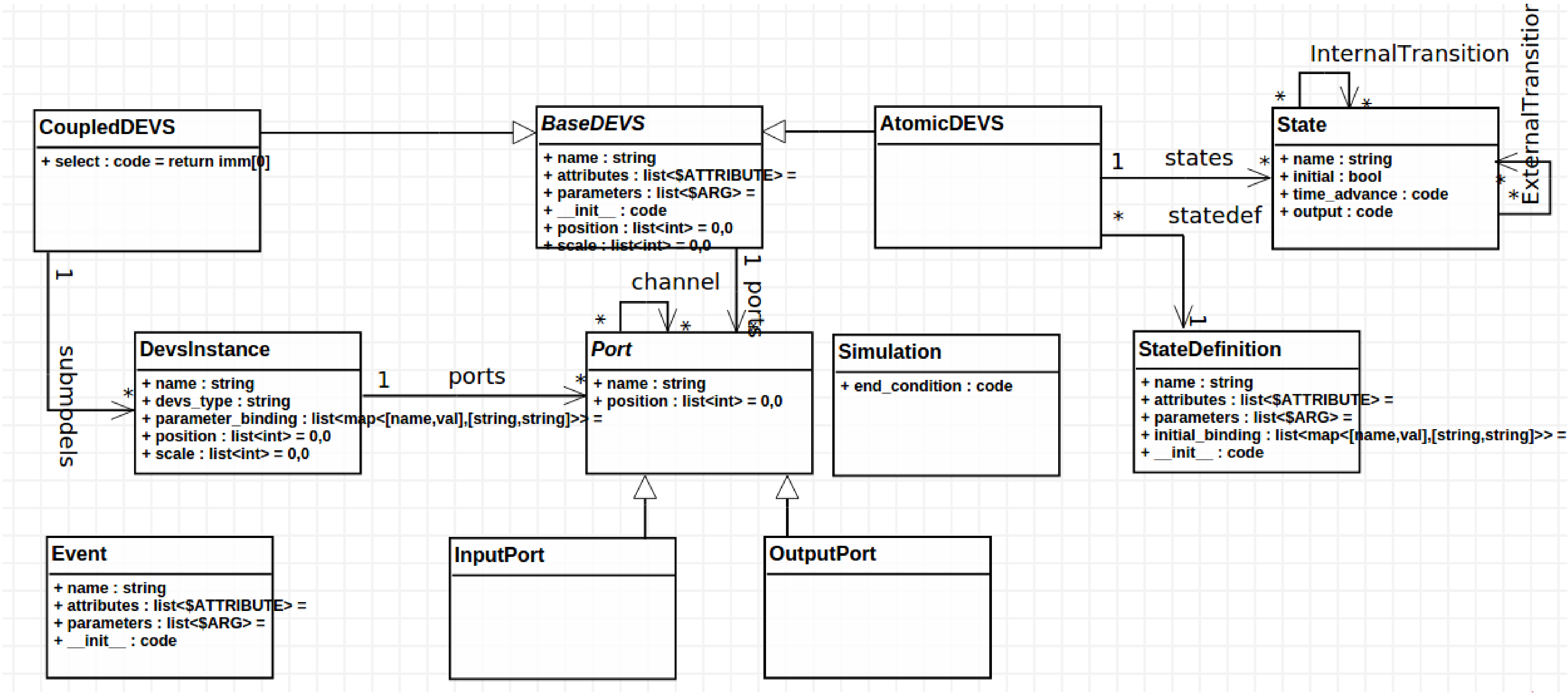
SED-ML

apply changes

Simulation Tool

DEVS

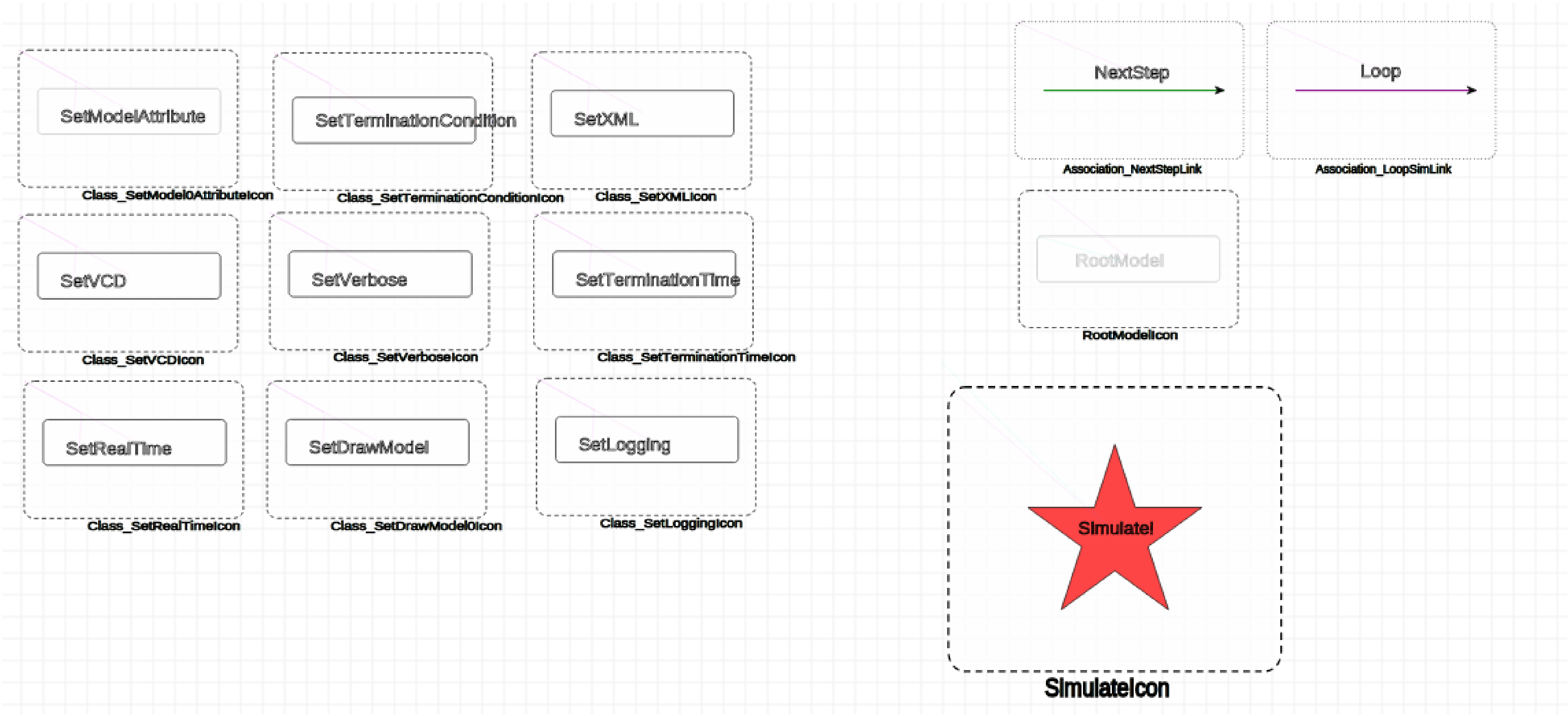
- Discrete Event System Specification
- Textual, low-level
- Very powerful, but...
- Customizing experiments without knowledge of the system can be hard


CoupledDEVS
+ select : code = return imm[0]
BaseDEVS
+ name : string
+ attributes : list<$ATTRIBUTE> =
+ parameters : list<$ARG> =
+ __init__ : code
+ position : list<int> = 0,0
+ scale : list<int> = 0,0
AtomicDEVS
State
+ name : string
+ initial : bool
+ time_advance : code
+ output : code
InternalTransition
ExternalTransitior
states
statedef
1
*
channel
ports
submodels
DevsInstance
+ name : string
+ devs_type : string
+ parameter_binding : list<map<[name,val],[string,string]>> =
+ position : list<int> = 0,0
+ scale : list<int> = 0,0
Port
+ name : string
+ position : list<int> = 0,0
Simulation
+ end_condition : code
StateDefinition
+ name : string
+ attributes : list<$ATTRIBUTE> =
+ parameters : list<$ARG> =
+ initial_binding : list<map<[name,val],[string,string]>> =
+ __init__ : code
Event
+ name : string
+ attributes : list<$ATTRIBUTE> =
+ parameters : list<$ARG> =
+ __init__ : code
InputPort
OutputPort

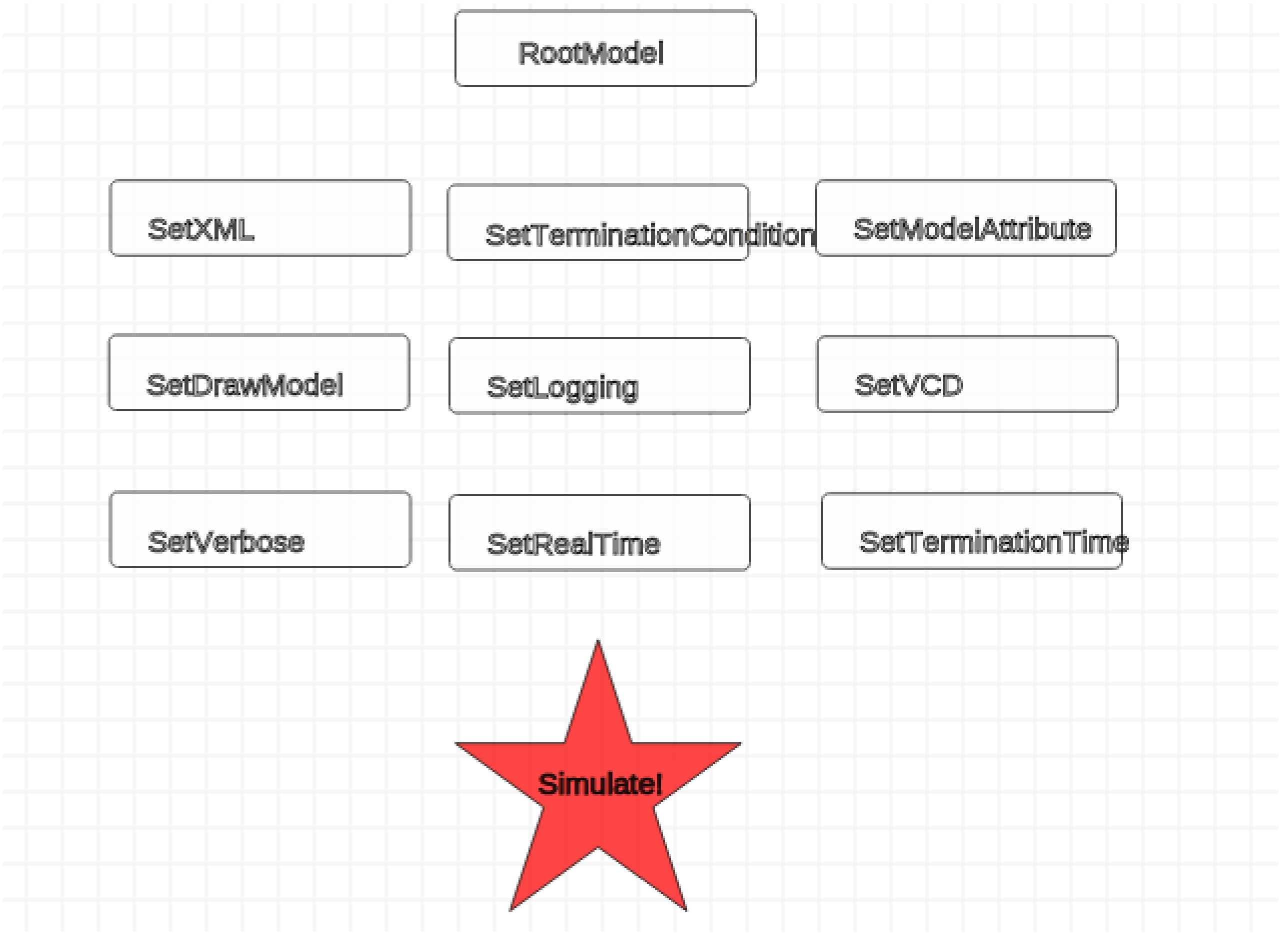
Approach

- Create a new language in AtomPM
- Visual approach
- Easy to use even with little knowledge
- Transform that language into usable Python code
- No low-level coding required

Visual language



Example simulation



Transforming into Python code

- 2 steps:
  - Transform visual representation into MetaDepth code (exporter was provided)
  - Transform that code into Python using EGL

Small demo