MDE: Modelling the DEVS formalism

Yentl Van Tendeloo
Yentl.VanTendeloo@student.ua.ac.be

January 24, 2013

The problem: Difficult to comprehend

```
class Root(CoupledDEVS):
  def __init__(self):
    CoupledDEVS.__init__(self)
    self.gen = self.addSubModel(Generator())
    self.seg1 = self.addSubModel(RoadSegment())
    self.seg2 = self.addSubModel(RoadSegment())
    self.seg3 = self.addSubModel(RoadSegment())
    self.col = self.addSubModel(Collector())
    self.connectPorts(self.gen.Q_send, self.seg1.Q_recv)
    self.connectPorts(self.gen.car_out, self.seg1.car_in)
    self.connectPorts(self.seg1.Q_sack, self.gen.Q_rack)
    self.connectPorts(self.seg1.Q_send, self.seg2.Q_recv)
    self.connectPorts(self.seg1.car_out, self.seg2.car_in)
    self.connectPorts(self.seg2.Q_sack, self.seg1.Q_rack)
    self.connectPorts(self.seg2.Q_send, self.seg3.Q_recv)
    self.connectPorts(self.seg2.car_out, self.seg3.car_in)
    self.connectPorts(self.seg3.Q_sack, self.seg2.Q_rack)
    self.connectPorts(self.seg3.car_out, self.col.car_in)
```

The problem: Difficult to comprehend

```
class Root(CoupledDEVS):
  def __init__(self):
    CoupledDEVS.__init__(self)
    self.gen = self.addSubModel(Generator())
    self.seg1 = self.addSubModel(RoadSegment())
    self.seg2 = self.addSubModel(RoadSegment())
    self.seg3 = self.addSubModel(RoadSegment())
    self.col = self.addSubModel(Collector())
    self.connectPorts(self.gen.Q_send, self.seg1.Q_recv)
    self.connectPorts(self.gen.car_out, self.seg1.car_in)
    self.connectPorts(self.seg1.Q_sack, self.gen.Q_rack)
    self.connectPorts(self.seg1.Q_send, self.seg2.Q_recv)
    self.connectPorts(self.seg1.car_out, self.seg2.car_in)
    self.connectPorts(self.seg2.Q_sack, self.seg1.Q_rack)
    self.connectPorts(self.seg2.Q_send, self.seg3.Q_recv)
    self.connectPorts(self.seg2.car_out, self.seg3.car_in)
    self.connectPorts(self.seg3.Q_sack, self.seg2.Q_rack)
    self.connectPorts(self.seg3.car_out, self.col.car_in)
```

Solution: Graphical representation

The problem: Different simulators

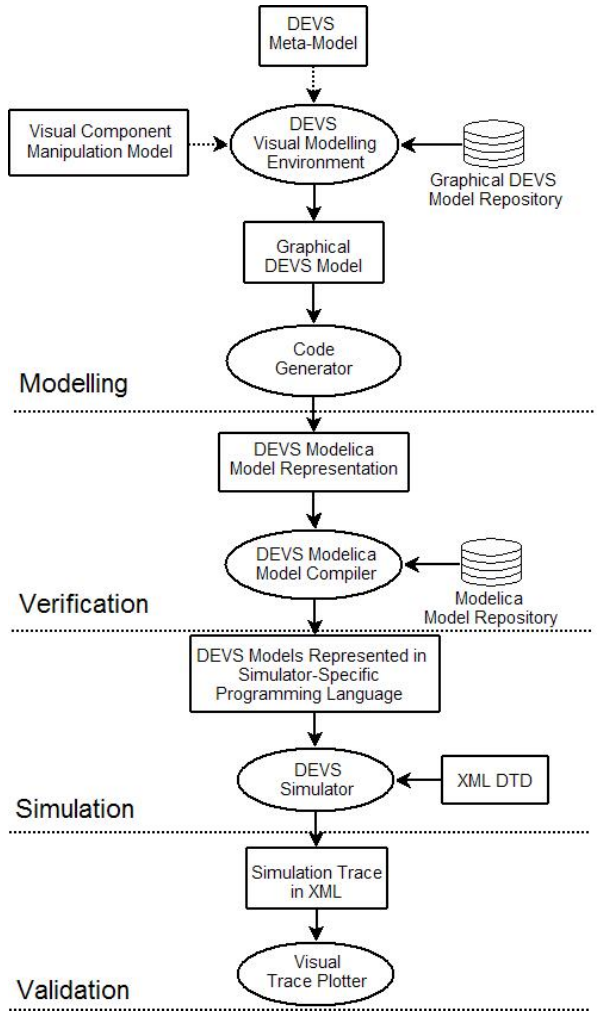
```
int main(int, char** argv){
  adevs::Digraph<Event*> dig;
  Generator* gen = new Generator();
  RoadSegment* seg1 = new RoadSegment();
  RoadSegment* seg2 = new RoadSegment();
  RoadSegment* seg3 = new RoadSegment();
  Collector* col = new Collector();
  dig.add(gen);
  dig.add(seg1);
  dig.add(seg2);
  dig.add(seg3);
  dig.add(col);
  dig.couple(gen, gen->Q_send, seg1, seg1->Q_recv);
  dig.couple(gen, gen->car_out, seg1, seg1->car_in);
  dig.couple(seg1, seg1->Q_sack, gen, gen->Q_rack);
  dig.couple(seg1, seg1->Q_send, seg2, seg2->Q_recv);
  dig.couple(seg1, seg1->car_out, seg2, seg2->car_in);
  dig.couple(seg2, seg2->Q_sack, seg1, seg1->Q_rack);
  dig.couple(seg2, seg2->Q_send, seg3, seg3->Q_recv);
  dig.couple(seg2, seg2->car_out, seg3, seg3->car_in);
  dig.couple(seg3, seg3->Q_sack, seg2, seg2->Q_rack);
  dig.couple(seg3, seg3->car_out, col, col->car_in);
```

The problem: Different simulators

```
int main(int, char** argv){
  adevs::Digraph<Event*> dig;
  Generator* gen = new Generator();
  RoadSegment* seg1 = new RoadSegment();
  RoadSegment* seg2 = new RoadSegment();
  RoadSegment* seg3 = new RoadSegment();
  Collector* col = new Collector();
  dig.add(gen);
  dig.add(seg1);
  dig.add(seg2);
  dig.add(seg3);
  dig.add(col);
  dig.couple(gen, gen->Q_send, seg1, seg1->Q_recv);
  dig.couple(gen, gen->car_out, seg1, seg1->car_in);
  dig.couple(seg1, seg1->Q_sack, gen, gen->Q_rack);
  dig.couple(seg1, seg1->Q_send, seg2, seg2->Q_recv);
  dig.couple(seg1, seg1->car_out, seg2, seg2->car_in);
  dig.couple(seg2, seg2->Q_sack, seg1, seg1->Q_rack);
  dig.couple(seg2, seg2->Q_send, seg3, seg3->Q_recv);
  dig.couple(seg2, seg2->car_out, seg3, seg3->car_in);
  dig.couple(seg3, seg3->Q_sack, seg2, seg2->Q_rack);
  dig.couple(seg3, seg3->car_out, col, col->car_in);
```

Solution: Independent language

Global overview



Source: Hongyan Song. Infrastructure for devs modelling and experimentation. Master's thesis, McGill University, 2006

Overview: Modelling

- In *AToM*^{3}
- Compliant to a metamodel
- Should have a *code generation* feature
  - For intermediate language (*Modelica*)
- Automatically generated environment!

Intermediate Language: Modelica

Modelling language

- Relatively mature
- Language features for both continuous and discrete models
- Standard library
- Re-usability

Modelica Example

```
class Generator
  extends AtomicDEVS;
  output DevsPort p_out;
  GeneratorState state();

  function timeAdvance
    output Integer timespan;
  algorithm
    ...
  end timeAdvance;
end Generator;

class Root
  extends CoupledDEVS;
  output DevsPort p_out;
  Processor ins_2();
equation
  connect( ins_2.p_out , p_out);
end Root;
```

Overview: Verification

- Use the μModelica compiler [1]
- Statically check the model
  - No accesses to inaccessible variables
- Possibly perform optimisations
- Currently (nearly) no verification happens

[1] Weigao Xu. The design and implementation of the modelica compiler. Master's thesis, McGill University, 2005

Overview: Simulation

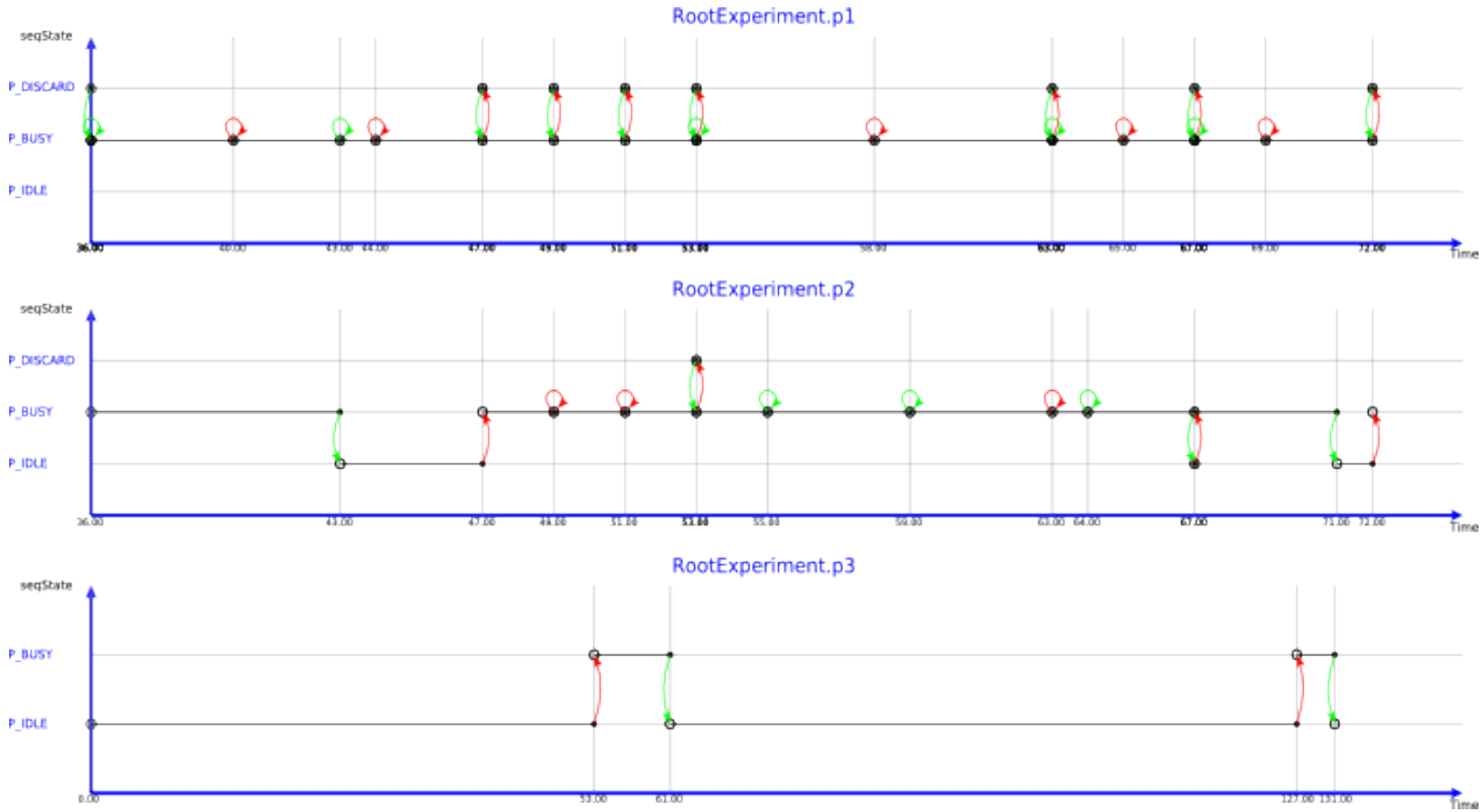
- Use the PythonDEVS simulator [2]
- Save the trace file
  - Text output
  - XML output
  - VCD output

[2] Jean-Sébastien Bolduc and Hans Vangheluwe. The modelling and simulation package pythondevs for classical hierarchical devs. Technical report, MSDL Technical Report, 2001

Overview: Validation

- Compare the output of the simulation to the specifications
- Textual simulation traces are unclear
- Visualize!

Overview: Validation (example)



Source: Hongyan Song. Infrastructure for devs modelling and experimentation. Master's thesis, McGill University, 2006

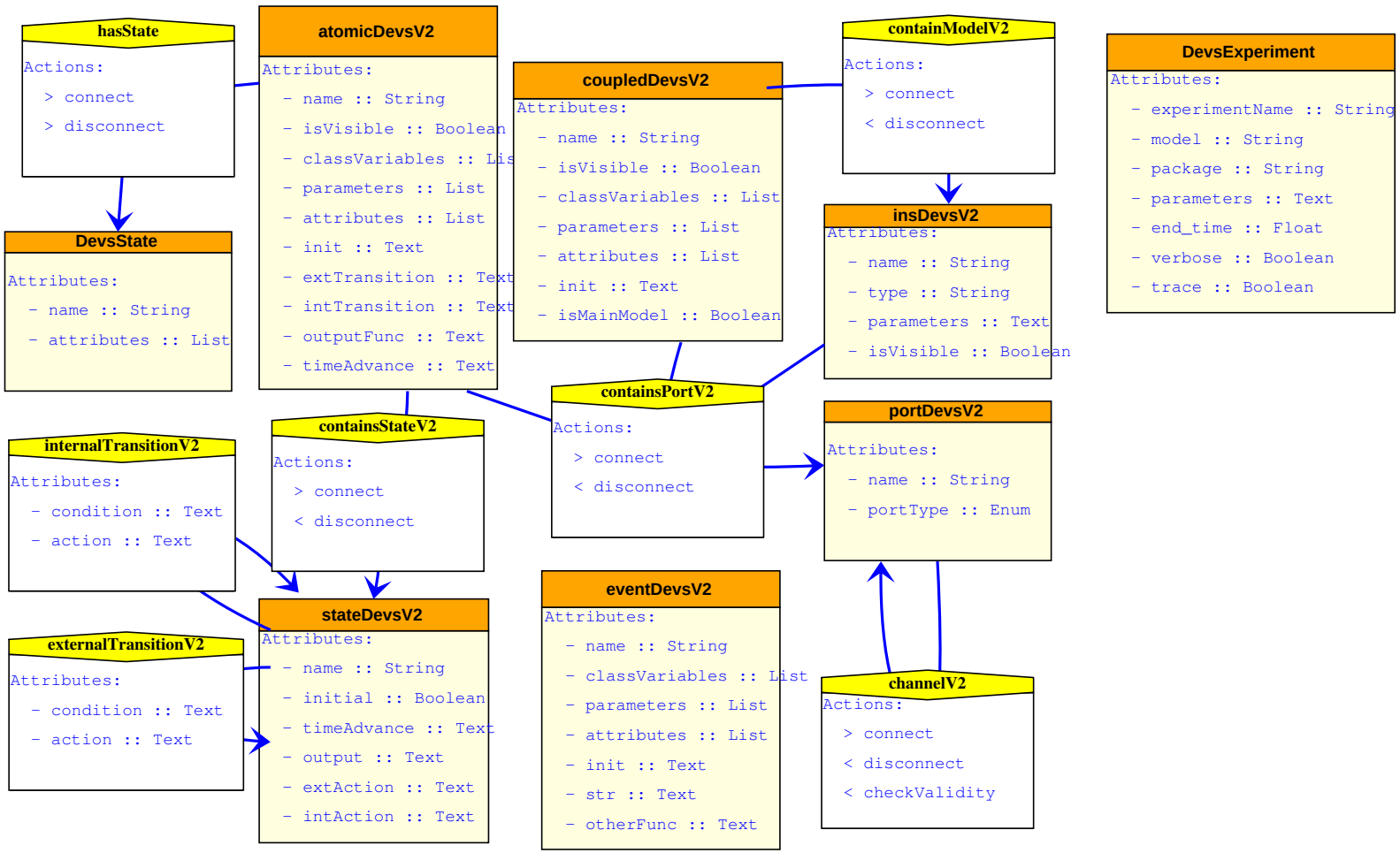
Overview: Features

- Model reuse
  - At *AToM*^{3} level
  - At *Modelica* level
  - At *PythonDEVS* level
- Clear boundaries
- Open structure

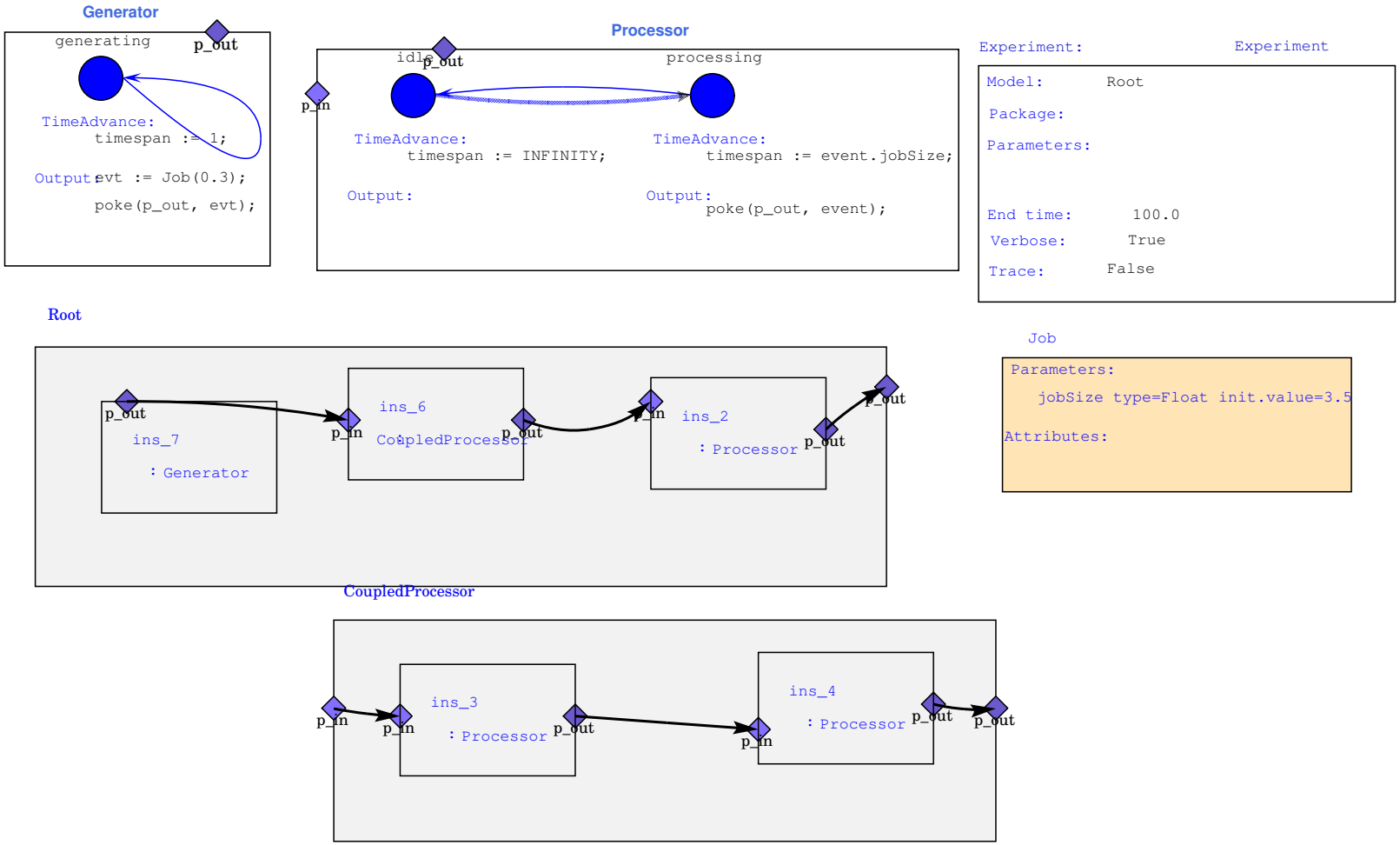
My contribution

- Modelling
  - Updating the existing metamodel
  - Include experiment data
- Verification
  - Update the μModelica compiler
  - Introduce a flattening phase

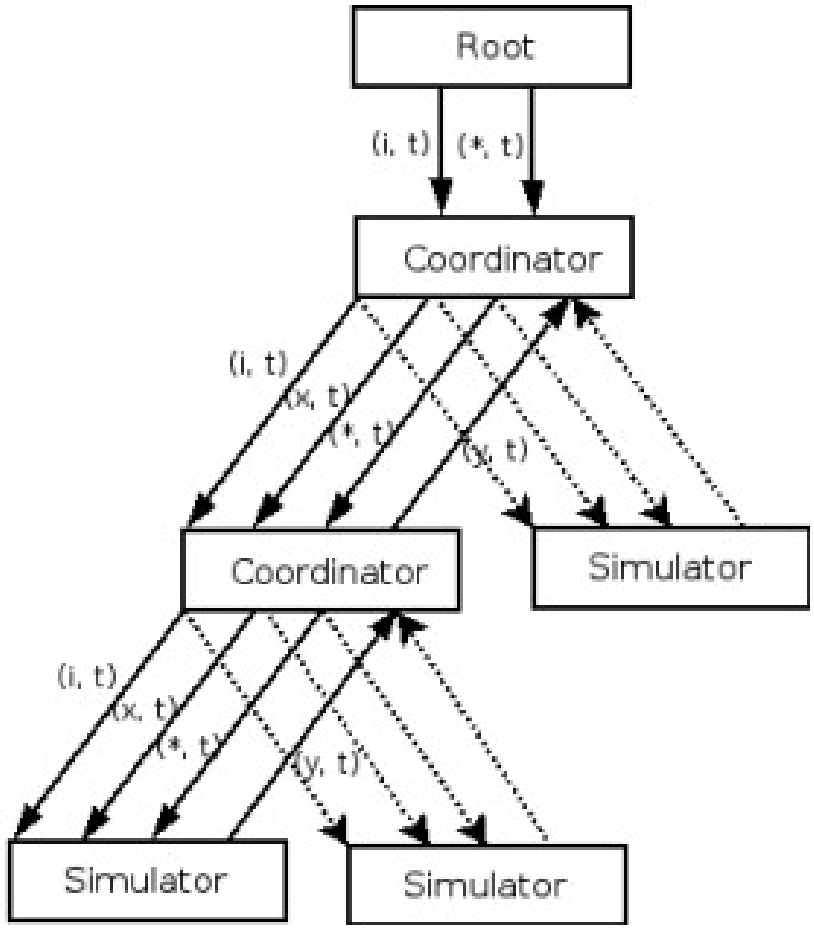
Modelling in *AToM*^{3}: metamodel



Modelling in *AToM*^{3}: example model



DEVS: Messages



Verification: Flattening

- What is flattening?
  - Perform the closure under coupling on a real model
  - Together with *direct connection*
- Use of flattening
  - Increased performance
  - The proof was right!

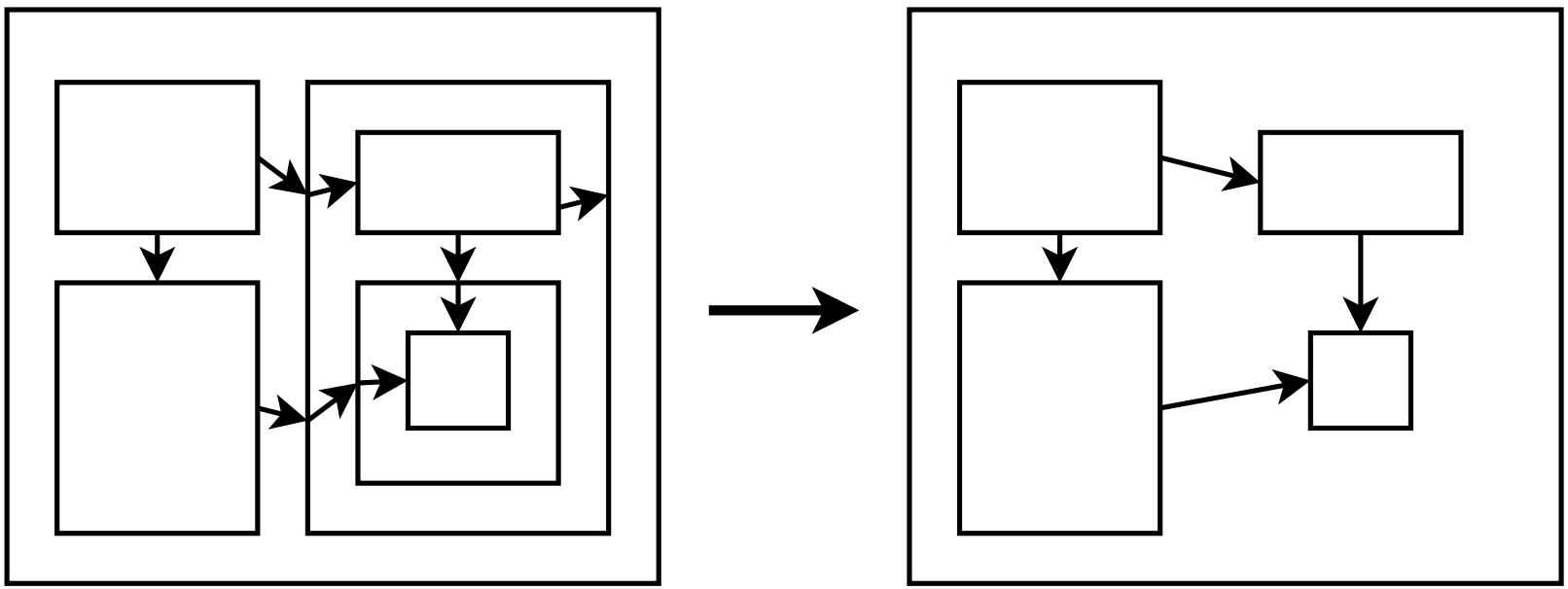
Verification: Direct connection

- What is direct connection?
  - Remove the complete hierarchy
  - Just 1 coupled model
  - Connections become 'one-step'
- Problems! [3]
  - Z functions need to be rewritten
  - select functions need to be rewritten

---

[3] Bin Chen and Hans Vangheluwe. Symbolic flattening of devs models. In *2010 Summer Simulation Multiconference*, SummerSim '10, pages 209–218, San Diego, CA, USA, 2010. Society for Computer Simulation International

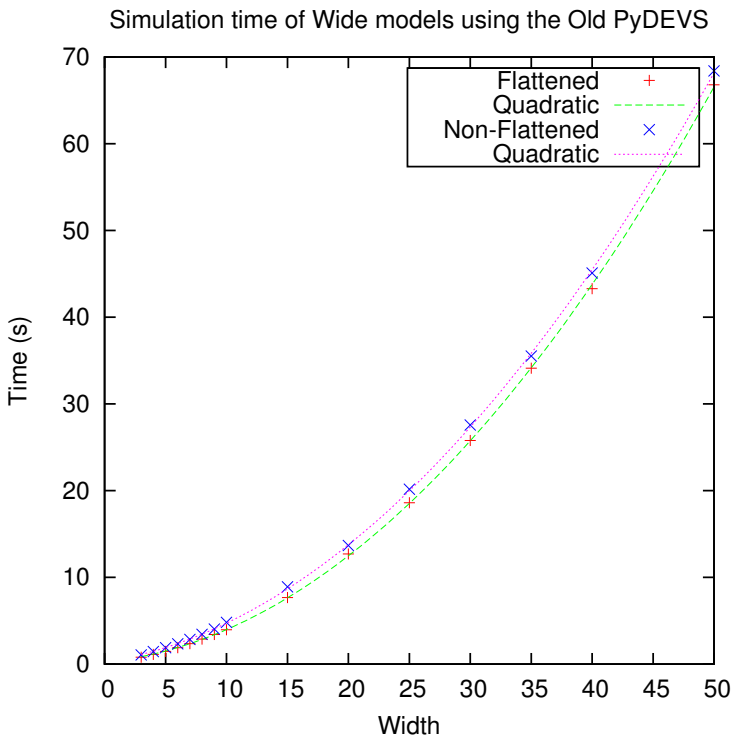
Verification: Direct connection

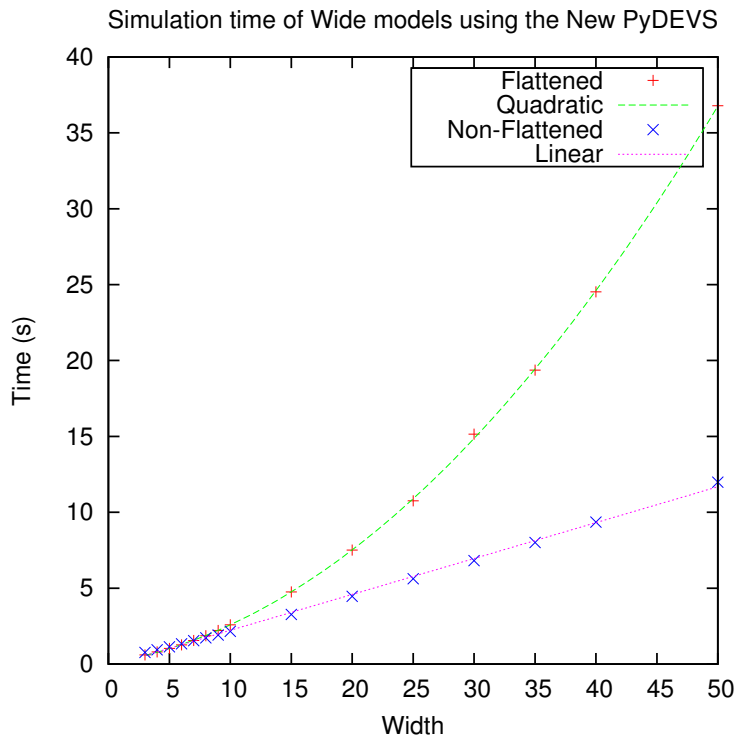


Verification: Compilation

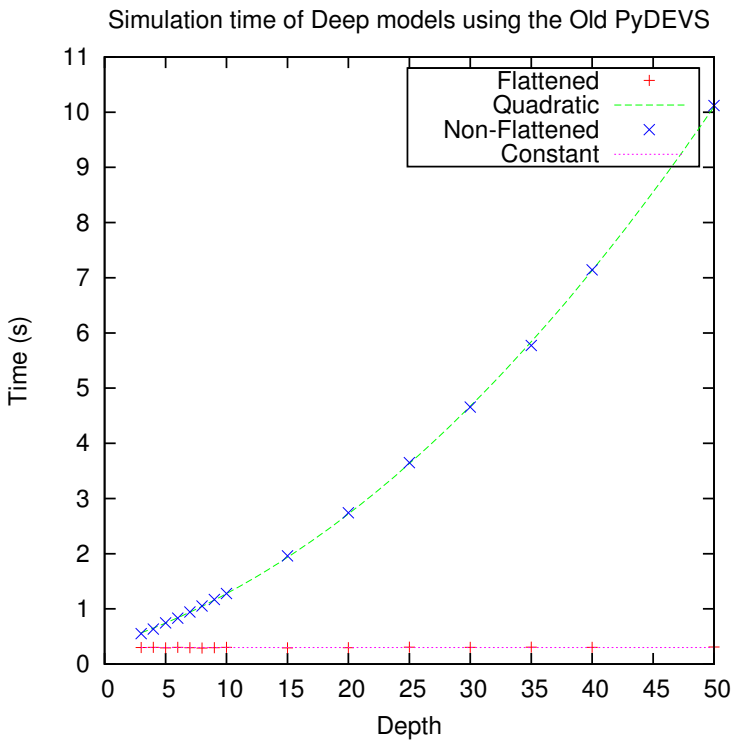
- Still need to compile the generated modelica code
- Some slight changes were done here

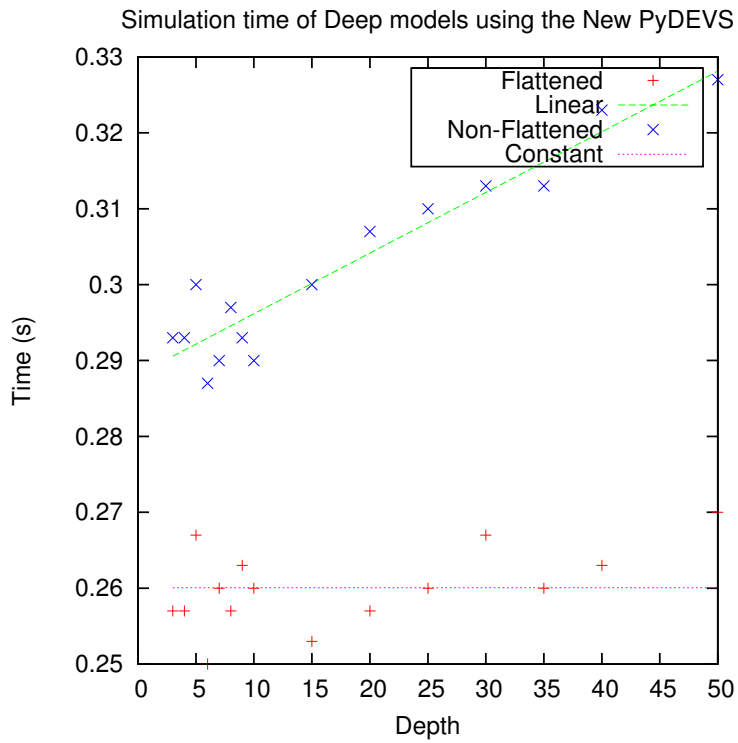
Evaluation


Simulation time of Wide models using the Old PyDEVS
Time (s)
Width
Flattened Quadratic
Non-Flattened Quadratic
0
10
20
30
40
50
60
70
0
5
10
15
20
25
30
35
40
45
50


Simulation time of Wide models using the New PyDEVS
Time (s)
Width
Flattened Quadratic
Non-Flattened Linear
0
5
10
15
20
25
30
35
40
0
5
10
15
20
25
30
35
40
45
50

Evaluation


Simulation time of Deep models using the Old PyDEVS
Time (s)
Depth
Flattened
Quadratic
Non-Flattened
Constant
0
1
2
3
4
5
6
7
8
9
10
11
0
5
10
15
20
25
30
35
40
45
50


Simulation time of Deep models using the New PyDEVS
Time (s)
Depth
Flattened
Linear
Non-Flattened
Constant
0.25
0.26
0.27
0.28
0.29
0.3
0.31
0.32
0.33
0
5
10
15
20
25
30
35
40
45
50

Evaluation: Gain

These optimisations are *simulator-independent*!

- Optimisations will propagate
- Difference depends on the simulator

This comparison was with an *optimised* simulator [4]

[4] Yentl Van Tendeloo. Research internship 1: Optimizing pydevs. Technical report, MSDL, 2013

Future work

Different dimensions

- Optimise flattening
- Implement Verifications
- Implement other back-ends
- Some 'details' were ignored
  - Z and select function

Demo

Time for a demo!


Jean-Sébastien Bolduc and Hans Vangheluwe.
The modelling and simulation package pythondevs for classical hierarchical devs.
Technical report, MSDL Technical Report, 2001.


Bin Chen and Hans Vangheluwe.
Symbolic flattening of devs models.
In *2010 Summer Simulation Multiconference*, SummerSim '10, pages 209–218, San Diego, CA, USA, 2010. Society for Computer Simulation International.


Hongyan Song.
Infrastructure for devs modelling and experimentation.
Master's thesis, McGill University, 2006.


Yentl Van Tendeloo.
Research internship 1: Optimizing pydevs.
Technical report, MSDL, 2013.


Weigao Xu.
The design and implementation of the modelica compiler.

Master's thesis, McGill University, 2005.