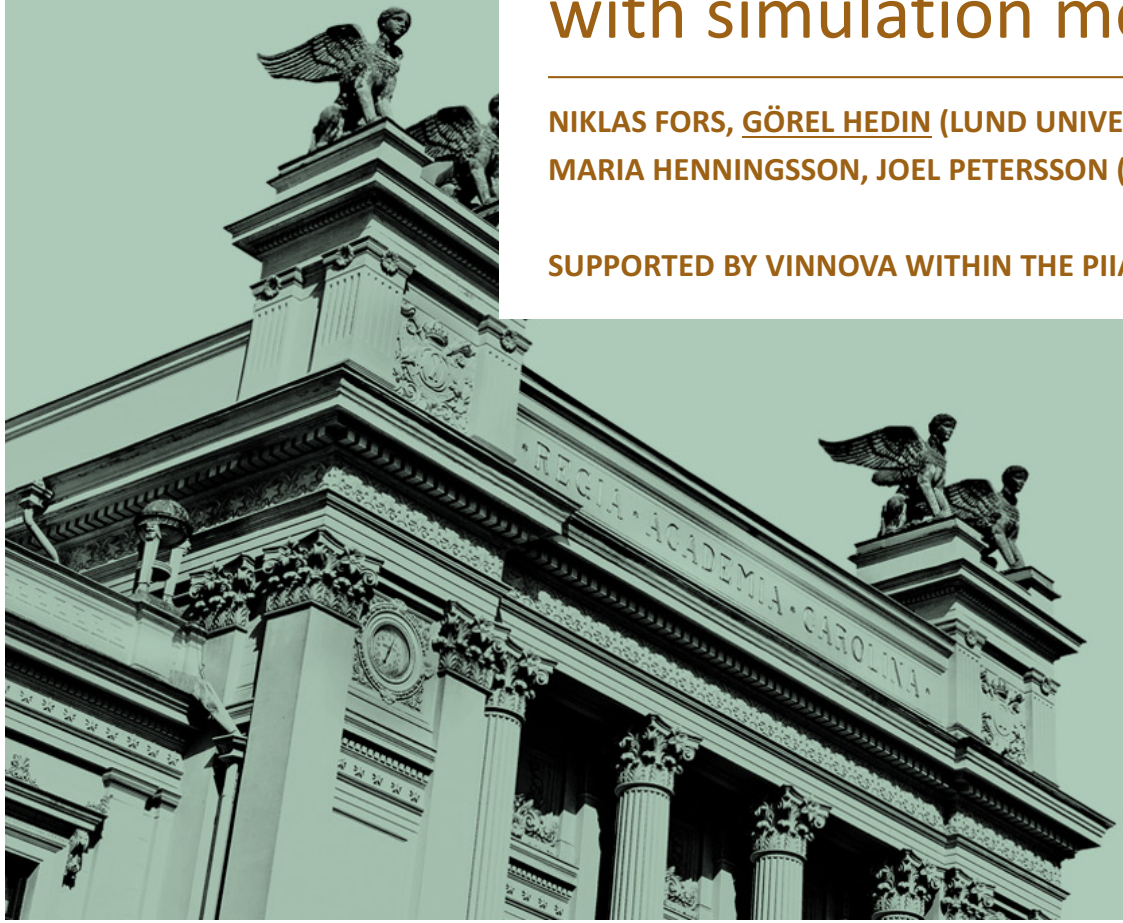
Composing Bloqqi control programs with simulation models

NIKLAS FORS, GÖREL HEDIN (LUND UNIVERSITY),
MARIA HENNINGSSON, JOEL PETERSSON (MODELON)

SUPPORTED BY VINNOVA WITHIN THE PIIA PROGRAMME



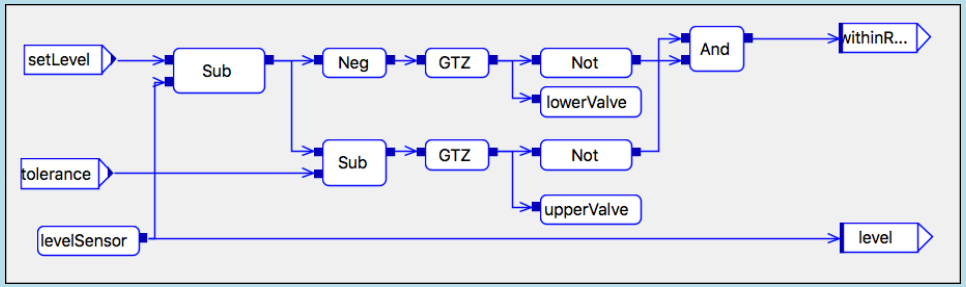
Outline

- Bloqqi – open research language for automation control
  - object-oriented
  - feature-oriented
- Compiling Bloqqi to C
- Wrapping into an FMU (Functional Mockup Unit)
- Composing Bloqqi FMUs with simulation FMUs
- Composition using Modelon's FMI Composer
  - uses new open standard SSP (System Structure and Parameterization)
- Examples

Bloqqi: Feature-based data-flow programming

Bloqqi program for tank control

Visual view



Textual view

```
diagramtype Tank(setLevel: Int, tolerance: Int
    => level: Int, withinRange: Bool) {
  upperValve: Valve;
  lowerValve: Valve;
  levelSensor: Sensor;
  ...
  connect(setLevel, Sub_1.in1);
  connect(levelSensor.out, Sub_1.in2);
  connect(levelSensor.out, level);
  ...
}
```

Runs in

Real world

1. Read liquid level

3. Open/ close valves

2. Compute control signal

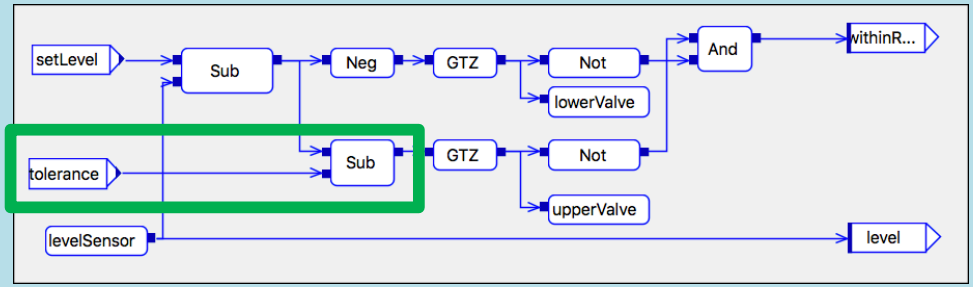
Control system

Bloqqi: Feature-based data-flow programming

Tolerance feature

Bloqqi program for tank control

Visual view



Textual view

```
diagramtype Tank(setLevel: Int, tolerance: Int
    => level: Int, withinRange: Bool) {
  upperValve: Valve;
  lowerValve: Valve;
  levelSensor: Sensor;
  ...
  connect(setLevel, Sub_1.in1);
  connect(levelSensor.out, Sub_1.in2);
  connect(levelSensor.out, level);
  ...
}
```

Runs in

Real world

1. Read liquid level

3. Open/ close valves

2. Compute control signal

Control system

New language constructs for specializing diagrams with features (Niklas Fors, PhD thesis, 2016)

Main mechanisms in Bloqqi

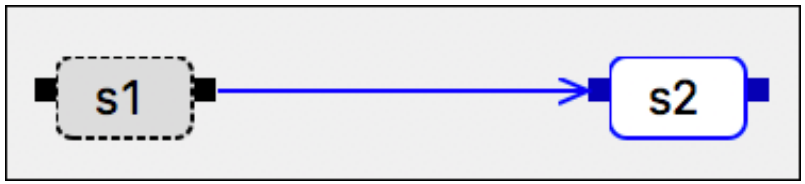
- Data-flow diagrams with ports, blocks, connections, variables, i/o
- Feature support for variability
- Diagram inheritance
- Connection intercept
- Block redeclaration (like Modelica)
- Wirings – *how* to add a feature
- Recommendations – *where* features can be added
- Feature interaction resolution
- Modular feature libraries
- Automatic feature wizards
- Program-by-example for features
  (refactoring to promote to recommendation)

Diagram inheritance

A

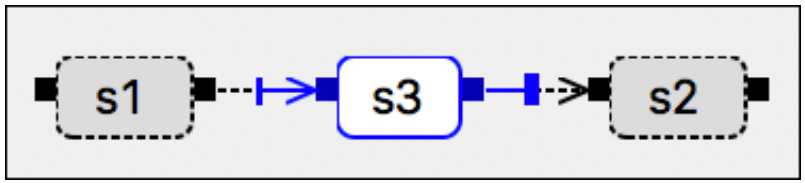

```
diagramtype A {
  s1: S;
}
```

B extends A




```
diagramtype B extends A {
  s2: S;
  connect(s1.out, s2.in);
}
```

C extends B

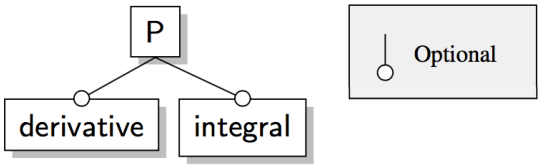



```
diagramtype C extends B {
  s3: S;
  intercept s2.in with s3.in,s3.out;
}
```

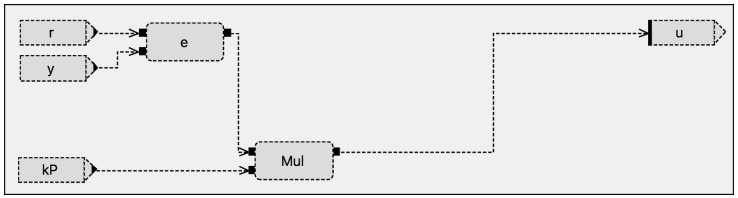
Connection interception

4 control variants

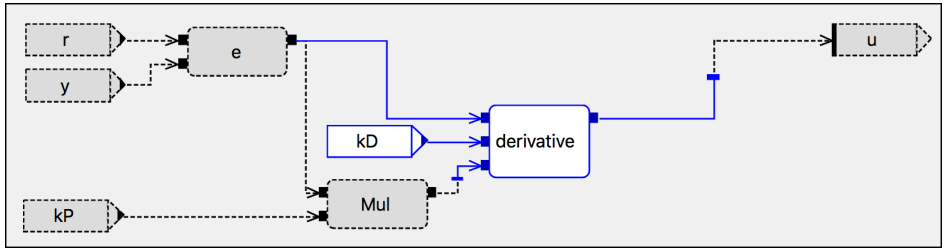
Feature model



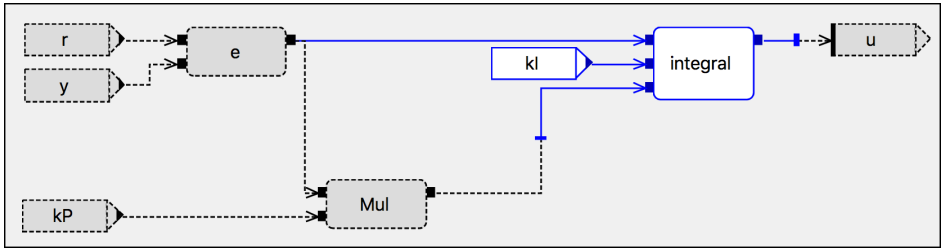
P (base diagram)



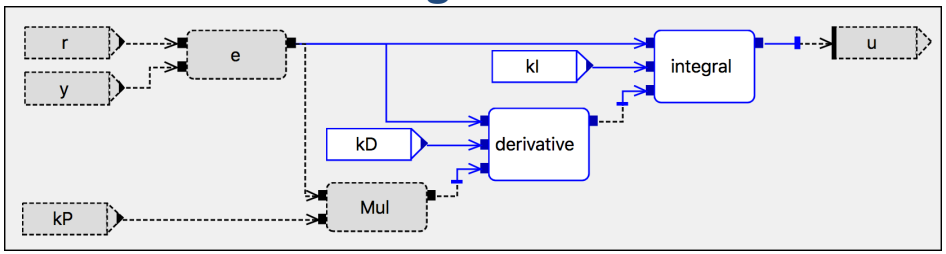
P + derivative



P + integral



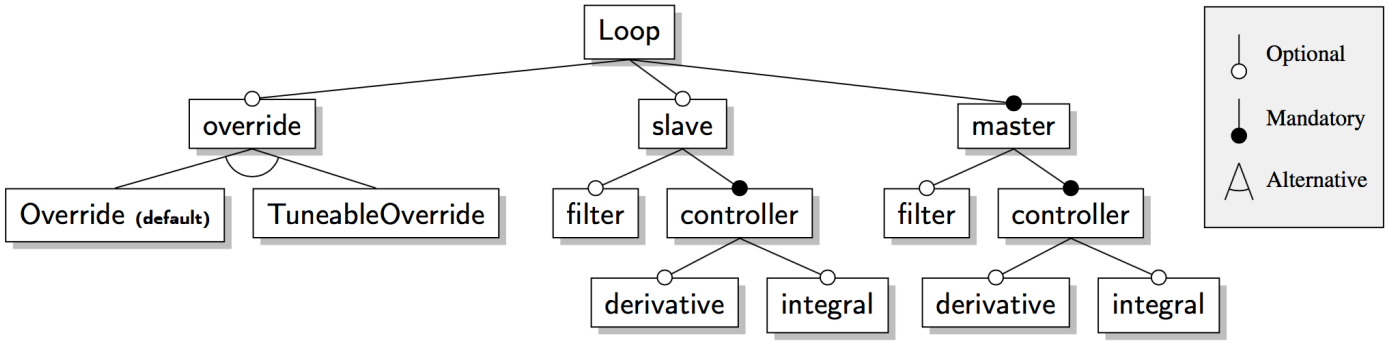
P + derivative + integral

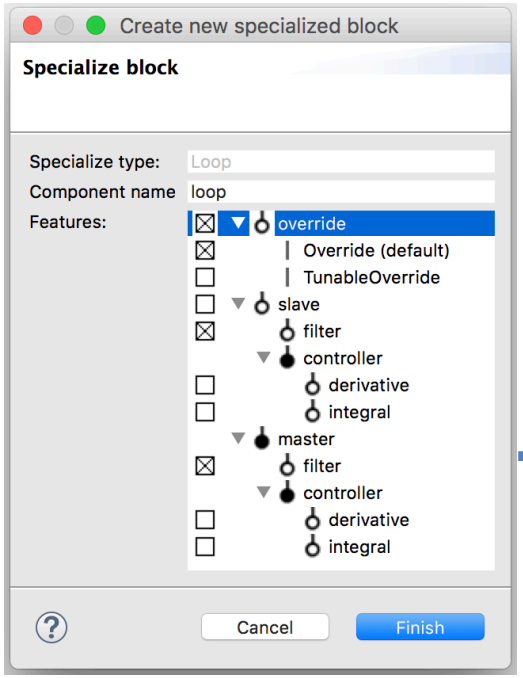


base type + feature selection = anonymous subtype (variant)

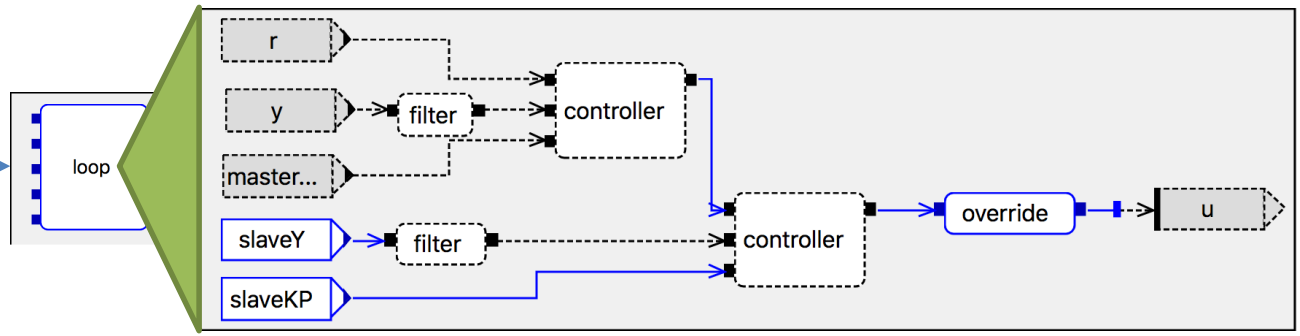
A larger example: control loop

Feature model



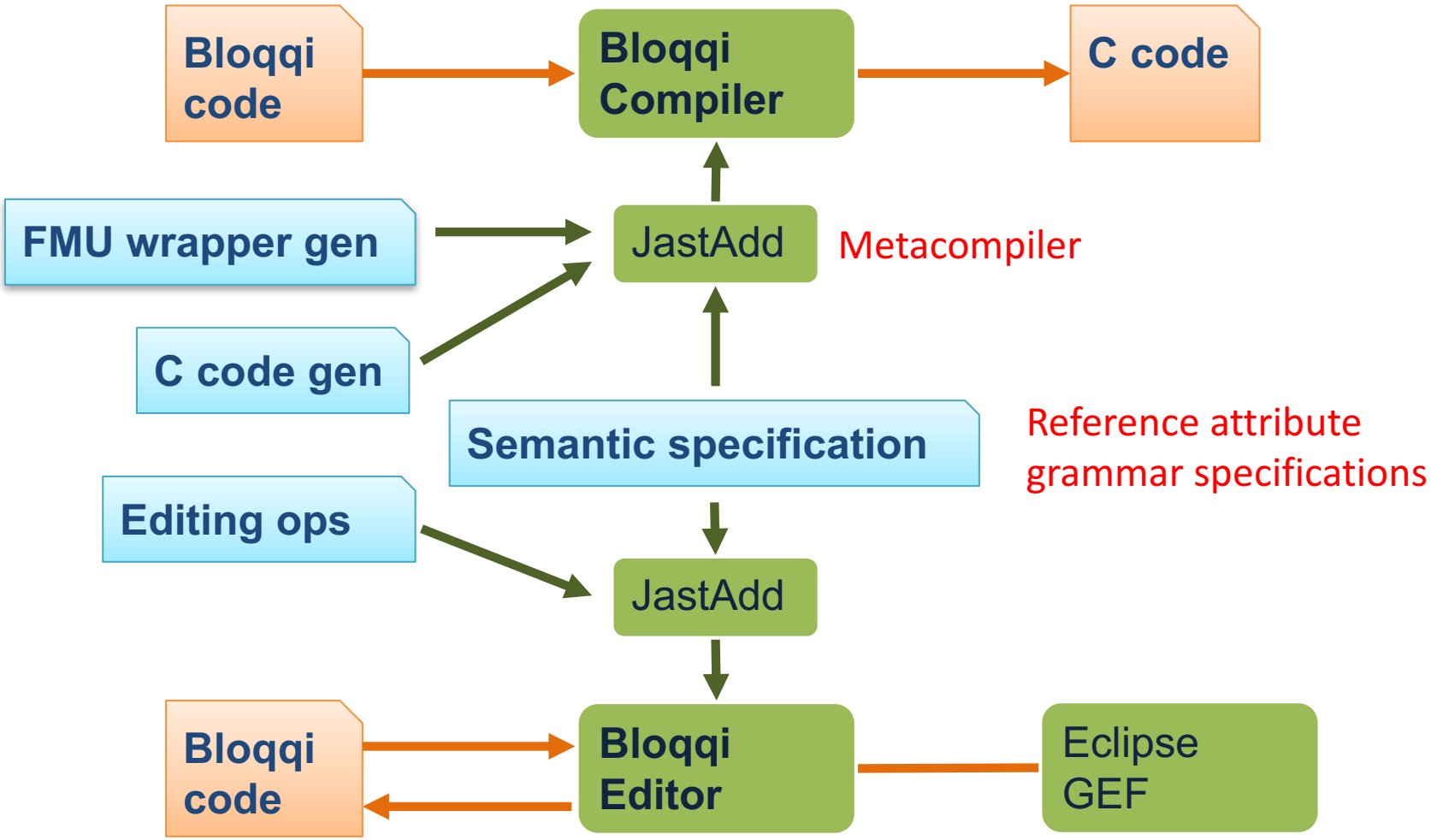


Feature selection



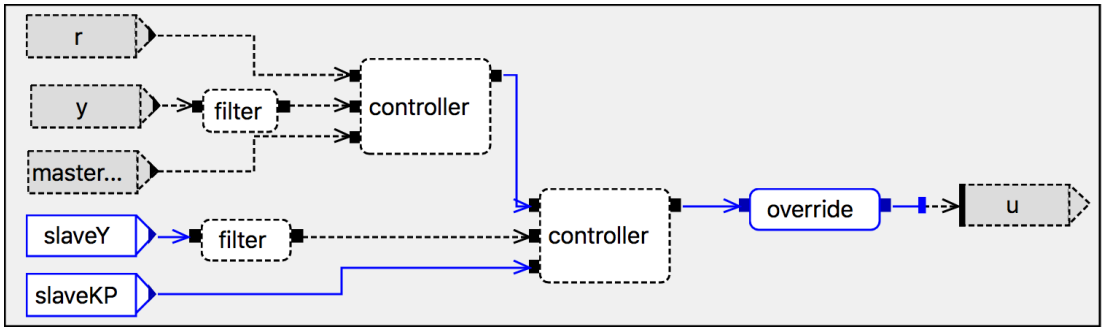
Resulting anonymous subtype

Modular tool implementation



Compiling Bloqqi to C

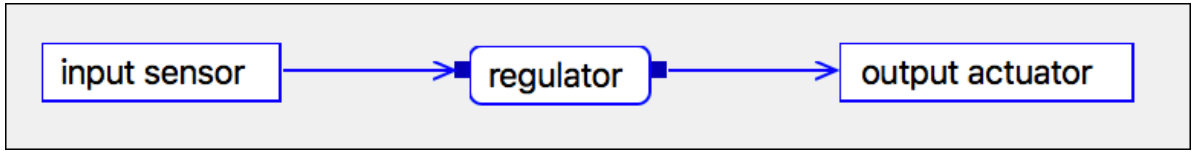
loop : Loop { ... }



Bloqqi	C translation
diagram type	C function
inheritance	flattening
anonymous subtypes	given unique names
block	function call
input ports	parameters
output ports	return value struct
state variables	pointer parameter to C struct
i/o values	pointer parameters to C structs

Example

Main




```
diagramtype Main {
  input sensor: Int;
  output actuator: Int;
  regulator: Regulator;
  connect(sensor, regulator.in);
  connect(regulator.out, actuator);
}
```

C function

```
void Main(Main_INPUT* _input,
          Main_OUTPUT* _output) {
  Regulator_RES regulator =
    Regulator(_input->sensor);
  _output->actuator = regulator.out;
}
```

Typical driver:

```
Main_VARS v = ...;
while(true) {
  v.input = read_sensors();
  bloqqi(&v);
  write_to_actuators(v.output);
  wait(period)
}
```

Runs on Raspberry PI and Arduino

Wrap Bloqqi as an FMU

Bloqqi	Cosimulation FMU
input values	inputs
output values	outputs
period	parameter `sampling-period`

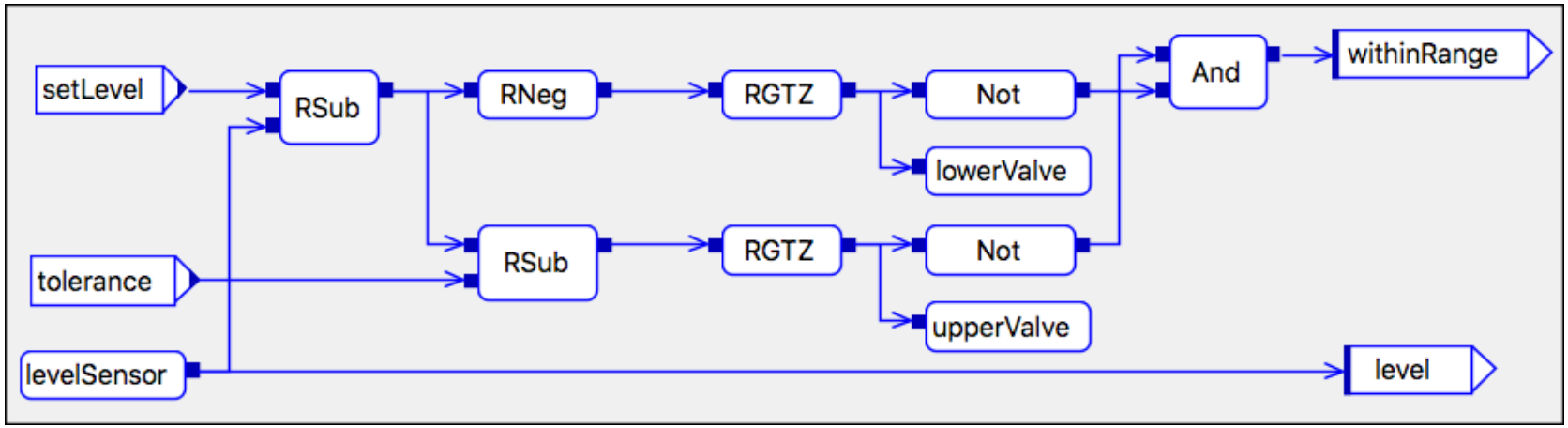
Wrapper code responding to time event:

```
if (timeEvent) {
  v.input = get_inputs();
  bloqqi(&v);
  set_outputs(v.output);
  register_next_time_event();
}
```

Wrapper code uses a Linux/MacOS port of the QTronic FMU SDK

Tank example

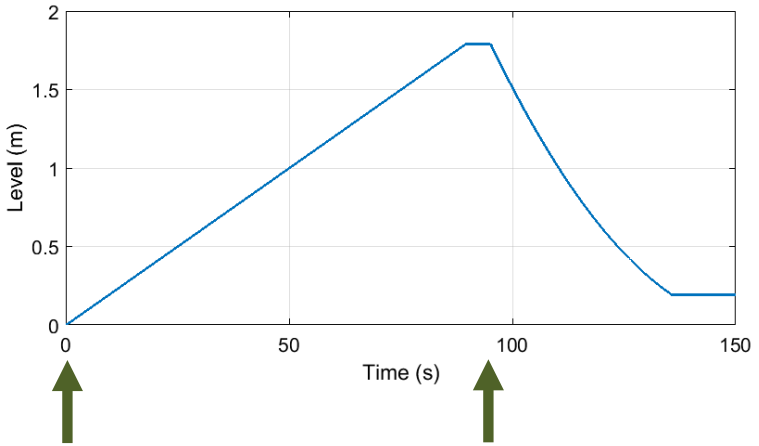
Bloqqi regulator



Tank model equations (in Modelica)

```
der(level) = (inFlow-outFlow)/AREA;
inFlow = if upperValveOpen
  then IN_FLOW
  else 0.0;
outFlow = if lowerValveOpen
  then (OUT_VALVE_AREA)*sqrt(2*9.82*level))
  else 0.0;
```

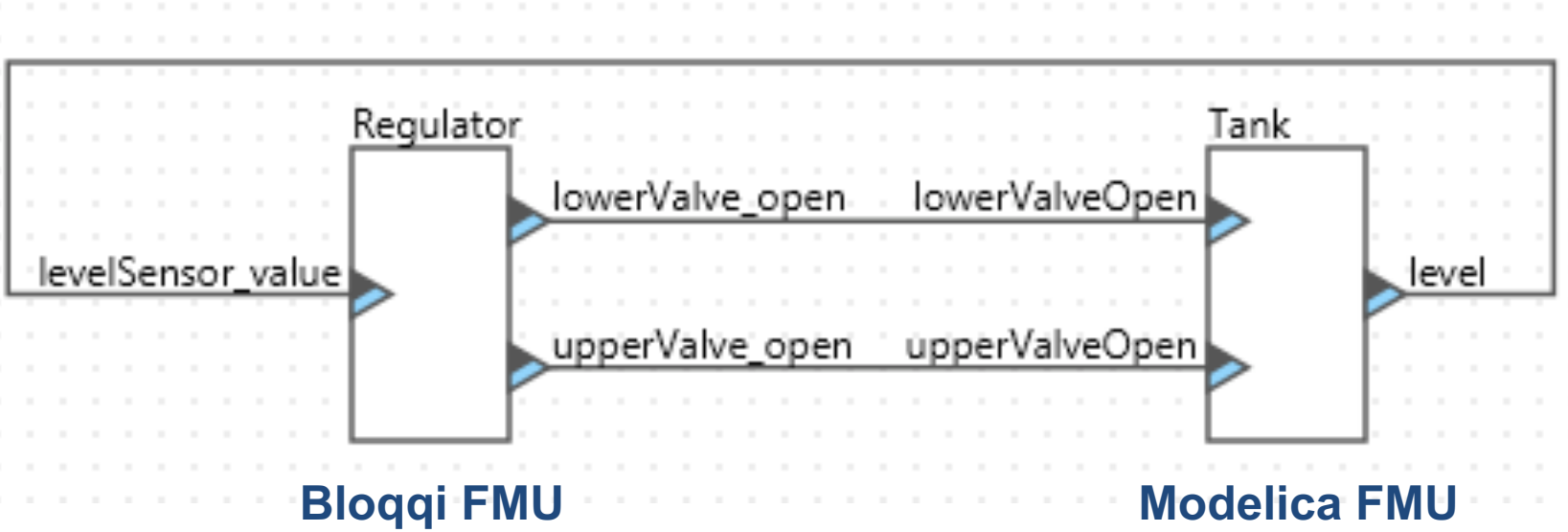
Tank level simulation result



Setpoint changes: 1.8 0.2

Composition

using Modelon's tool FMI composer

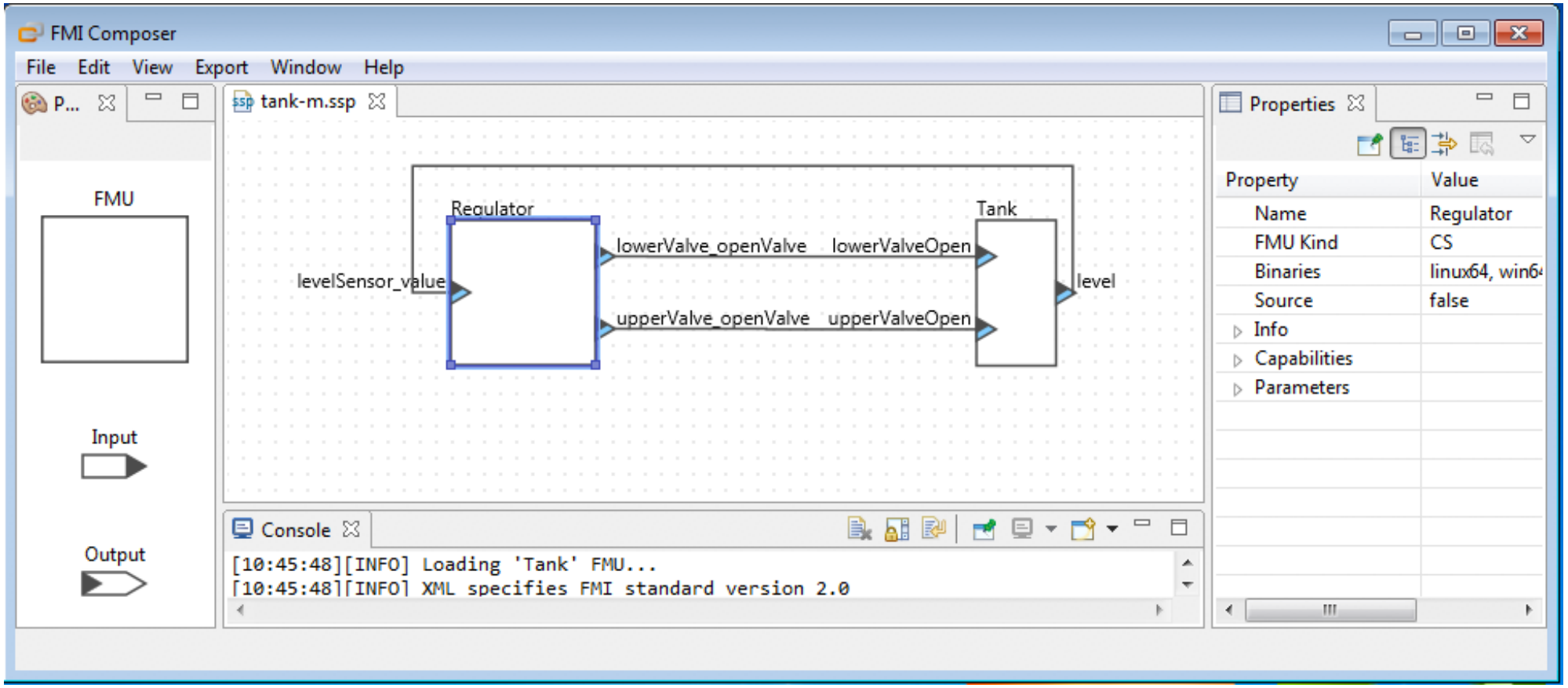


Result: a new (composite) FMU

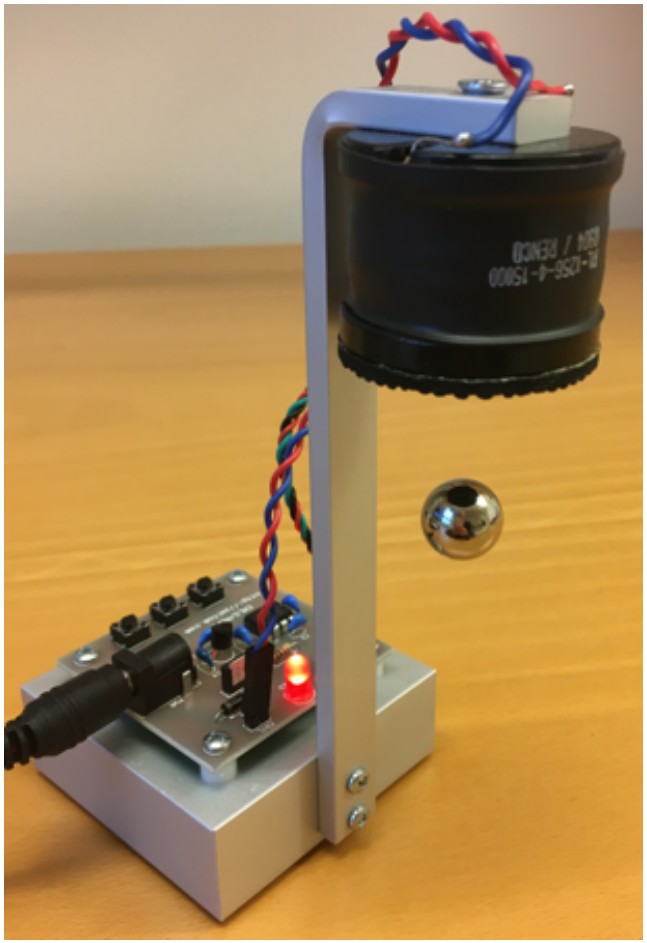
Composition stored as **SSP** (System Structure and Parameterization)

- standard format for defining **systems of FMUs**
- Defines **connections**, **parameterizations**, etc.
- New vendor-neutral **open standard**
- Developed as a project within Modelica Association

FMI composer tool



Magnetic levitation example



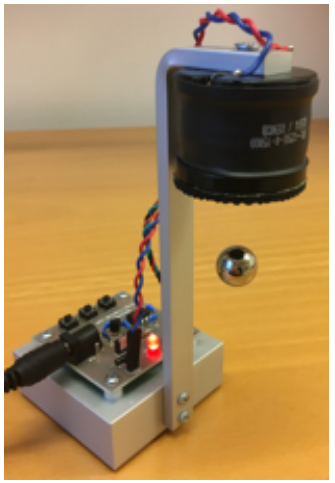
Bloqqi PD regulator

Modelica model

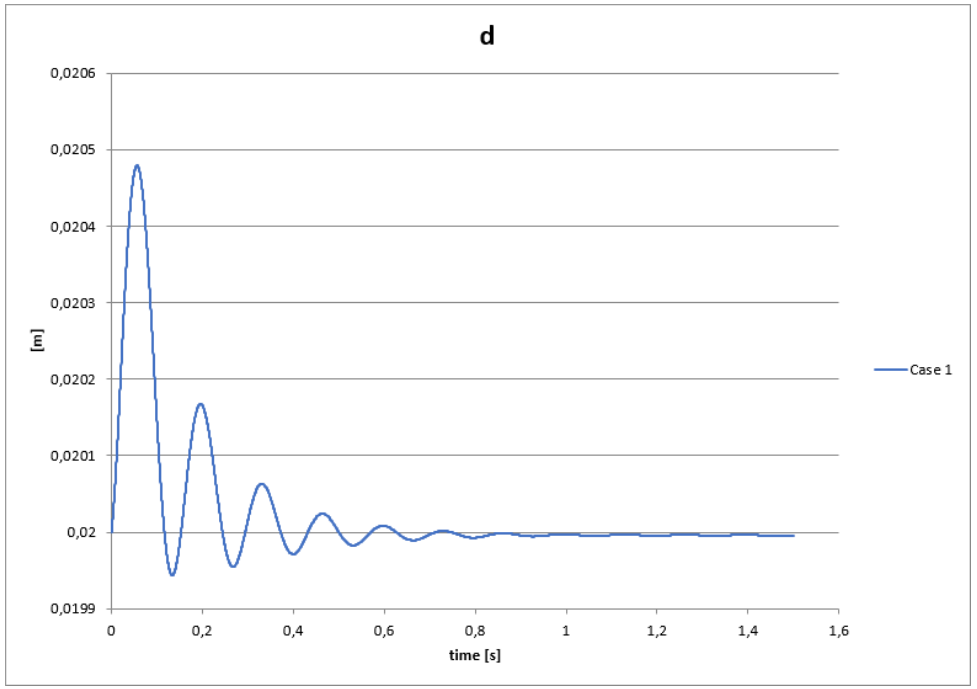
Composed FMU

Levitation model and PD regulator algorithm by Bernhard Thiele

Running the composed FMU in FMI add-in for Excel



1	FMI Add-In for Excel™						
2	*FMU Simulation sheet*						
6		Sheet version	*Generated by Modelon FMI Add-In for Excel version 2.0*				
7		Model name	*maglevSys*				
8		Generation tool	*FMI Composer*				
9		FMU kind	*CoSimulation*				
10		Checksum	*388b91bf5f098382e51efedfb791abe5*				
11		Expiry date					
14	Experiment setup						
17						*Default*	Case 1
19					Status		Success
20		Simulation settings					
22					FMU path	C:\Work\GIT\maglevSys.fmu	
23					Start time	0	
24					Stop time	1,5	
25					Output points	5000	
26					Timeout	0	
27					Log level	All	
28					Enable	TRUE	
30		Parameterization					
32		*Exact initials*	Name	Unit		*Default*	Case 1
33			pi_dPart_h			0,0005	
34			pi_dPart_Nd			5	
35			pi_dPart_Td			0,05	
36			pi_Kp			5,5	
37			r			0	
38			n	V		0	
39			PID-con.sampling-period			0,0005	
41		*Guess initials*	Name	Unit		*Default*	Case 1
43	Results						
45		Final Values					
47			Name	Unit	Plot		Case 1
48			d	m	TRUE		0,019996



Conclusions and Future Work

- Test control programs
  - translation to FMU
  - composition with simulation FMU
  - run in any tool with FMI driver support
- Future work on composition
  - larger examples
  - support automated tests
- Future work on Bloqqi
  - combine data-flow with sequential control
  - control scenarios with interlocked startup sequences