FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK

Nexor
Cyber-Physical Systems
University of Antwerp

Ansymo
Antwerp Systems & Software Modelling
University of Antwerp

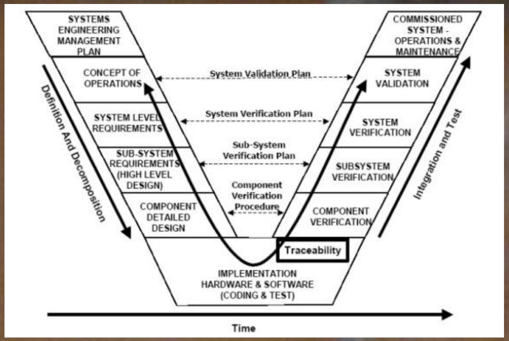
Universiteit Antwerpen
Department of Mathematics and Computer Science

McGill
School of Computer Science

MSDL
Modelling, Simulation and Design Lab

http://msdl.cs.mcgill.ca/

Pieter Mosterman



MODEL EVERYTHING!

at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling processes

Enabler: (domain-specific) modelling language engineering,
including model transformation

Simulation in Europe

SiE

ESPRIT Basic Research Working Group 8467
Simulation for the Future: New Concepts, Tools and Applications

Keywords:
simulation technologies, multi-paradigm modelling, solvers, standards, interoperability, industrial deployment, demonstrators, user-simulator interfaces

MPM4CPS

FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK

cost
EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

Validation, Verification, Testing and Accreditation

Analysis and Verification of Model Transformations, Debugging, Instrumentation, Tracing, etc.

Language Engineering

Domain-Specific Languages, Model Transformation, (web-based) Visual and Textual Modelling Environments, etc.

Simulation

Co-Simulation, Discrete-event, DEVS, continuous time, acausal, Modelica, etc.

Deployment & Resource-optimized Execution

Platforms (e.g. AUTOSAR, CAN, etc.), Design-Space Exploration, Virtualization, Models@run-time, Efficient execution of model transformations, etc.

Model Management & Process

FTG+PM, Safety (ISO 26262, Railway, etc,), Agile Modelling, Consistency management, Experimental frames, etc.

The Modelverse:

A Foundation for Multi-Paradigm Modelling

Yentl Van Tendeloo

Yentl.VanTendeloo@uantwerpen.be

School of Computer Science, McGill

Ansymo
Antwerp Systems & Software Modelling
University of Antwerp

Summary

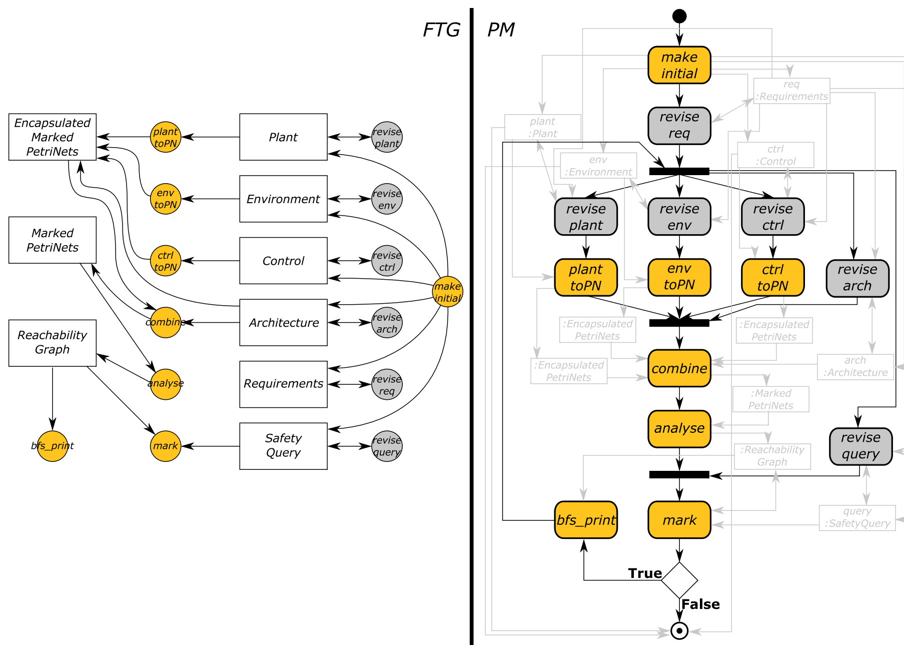
- What? Multi-Paradigm Modelling kernel and repository
- Why? Support the use of Multi-Paradigm Modelling
- How? Using Multi-Paradigm Modelling techniques
- Maturity? Academic tool

[1] Y. Van Tendeloo and H. Vangheluwe. The Modelverse: a Tool for Multi-Paradigm Modelling and Simulation. In Proceedings of the 2017 Winter Simulation Conference, 2017 (accepted).

[2] Y. Van Tendeloo. Foundations of a Multi-Paradigm Modelling Tool. In ACM Student Research Competition at MoDELS, 2015.

What?

Multi-Paradigm Modelling kernel and repository

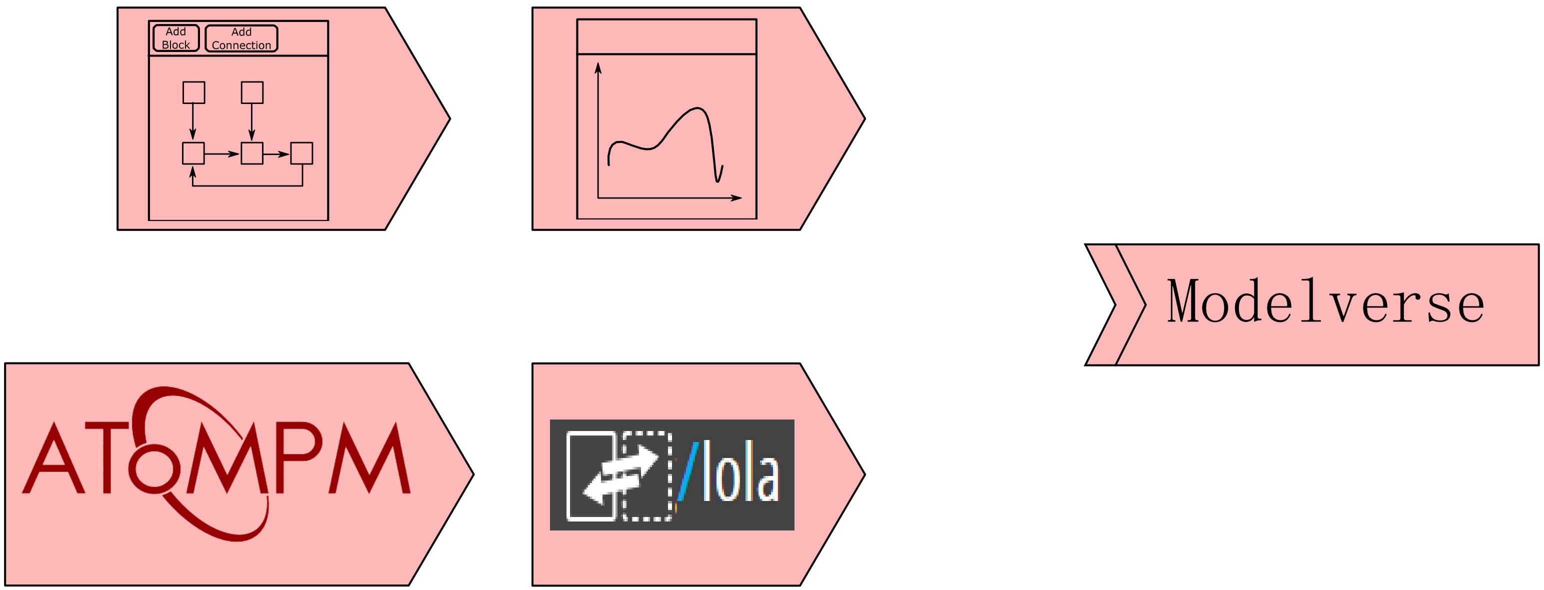


[3] L. Lucio, S. Mustafiz, J. Denil, H. Vangheluwe, and M. Jukss. FTG+PM: An Integrated Framework for Investigating Model Transformation Chains. In SDL 2013: Model-Driven Dependability Engineering, Volume 7916 of Lecture Notes in Computer Science, 182-202, 2013.
[4] P. Mosterman, and H. Vangheluwe. Computer Automated Multi-Paradigm Modeling: An Introduction. SIMULATION 80(9): 433-450, 2004.

McGill School of Computer Science
NECSIS
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

What?

Multi-Paradigm Modelling kernel and repository

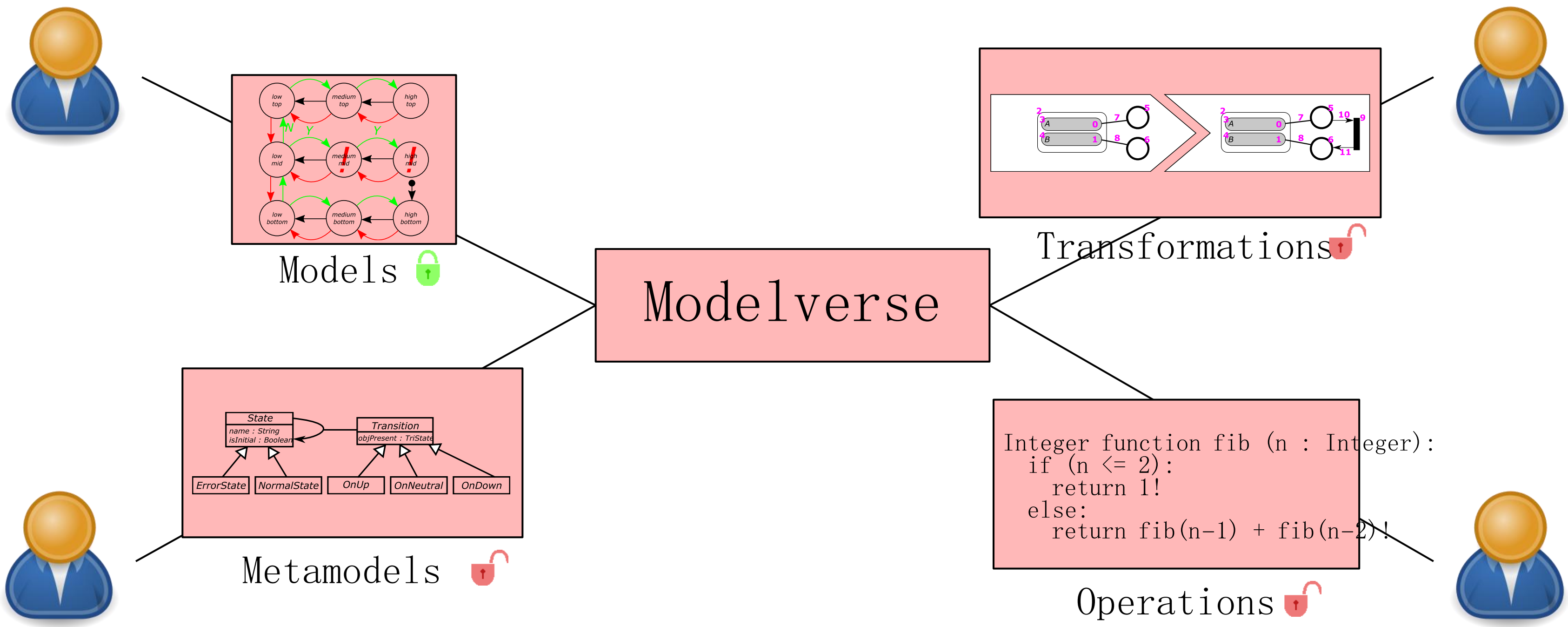


[5] E. Syriani, H. Vangheluwe, R. Mannadiar, C. Hansen, S. Van Mierlo, and H. Ergin. AToMPM: A Web-based Modeling Environment. In Proceedings of MODELS'13 Demonstration Session, 21-25, 2013.

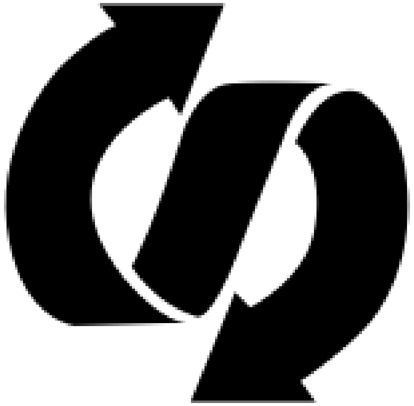
[6] K. Schmidt. LoLA: a low level analyser. In Proceedings of the 21st international conference on Application and theory of petri nets, 465-474, 2000.

What?

Multi-Paradigm Modelling kernel and repository

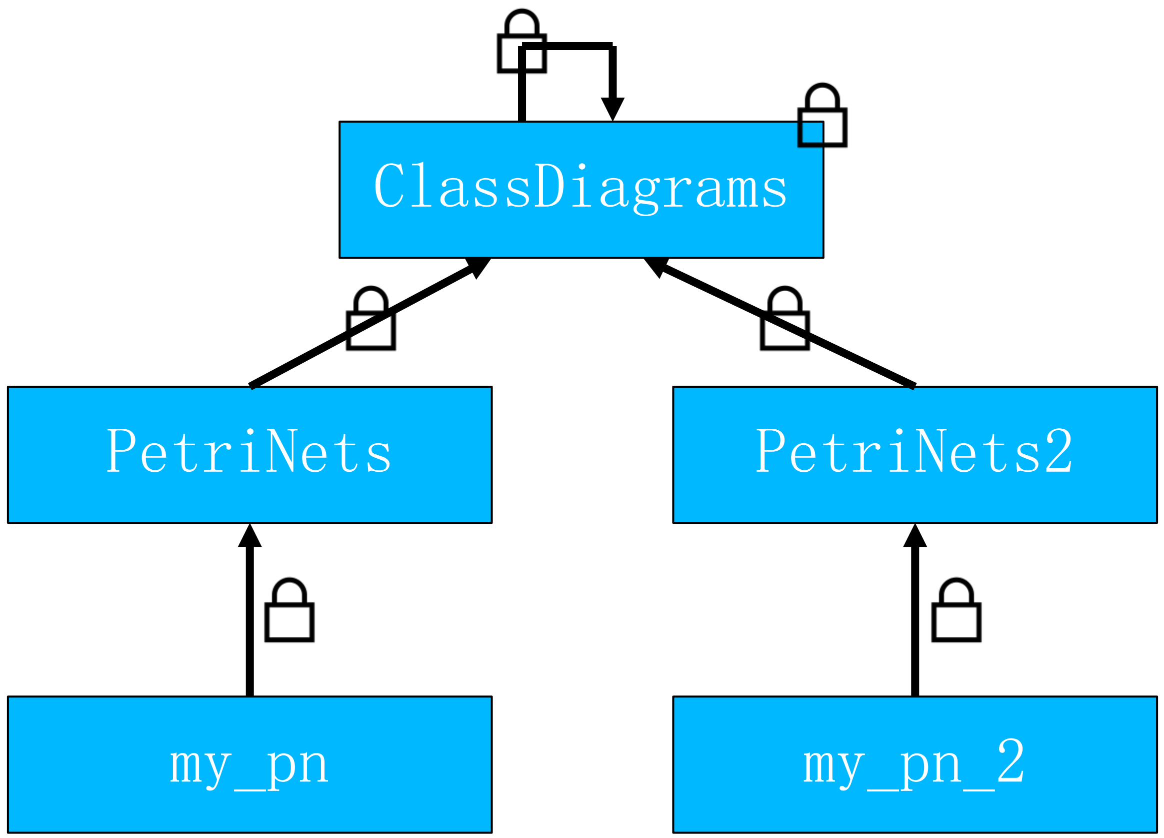


Why?

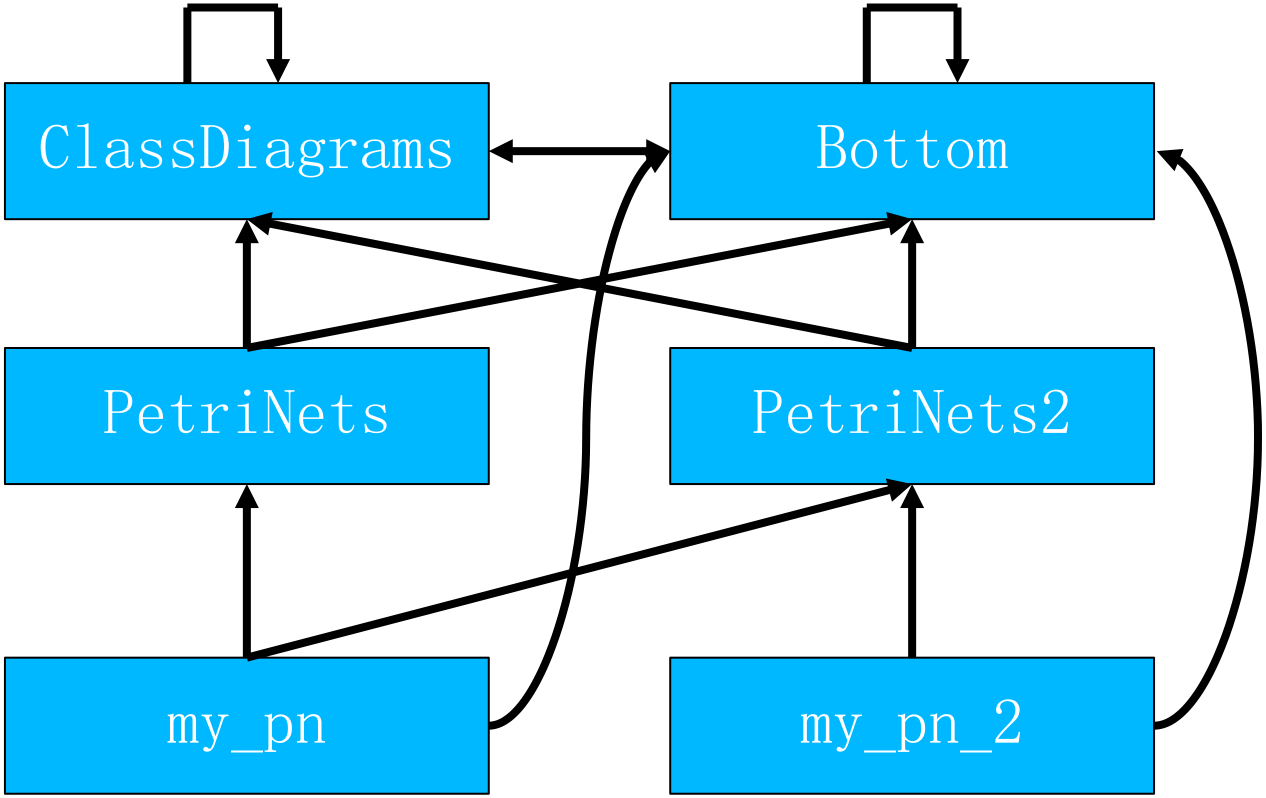


Flexibility

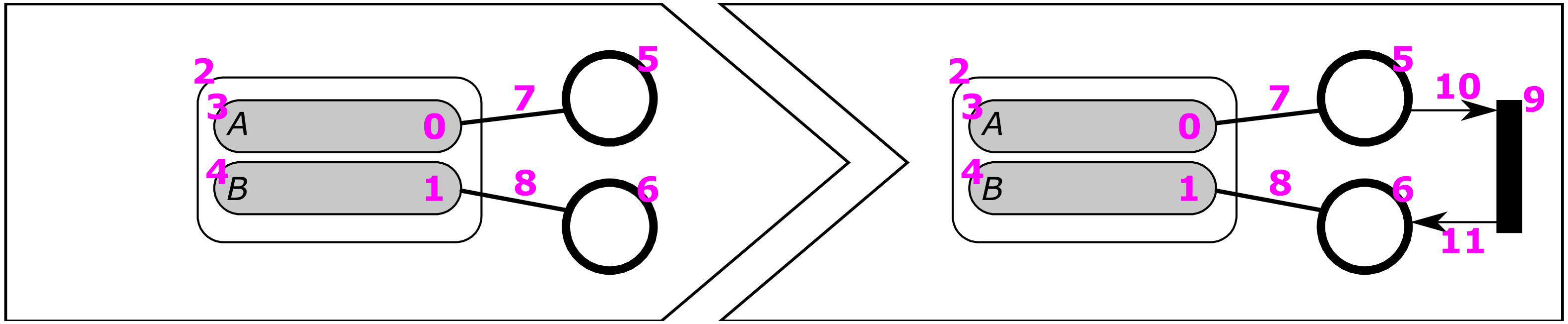
Why?



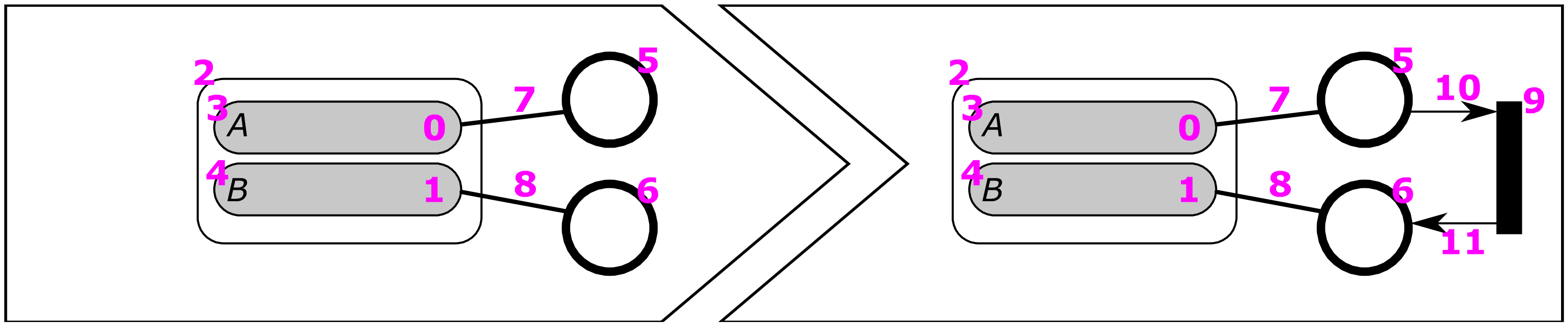
Why?



Why?



Why?



Algorithm 1 Strongly Connected Component Algorithm.

```
topSort(graph)
rev_graph ← reverse_edges(graph)
for all node ∈ rev_graph do
    node.visited ← False
end for
while rev_graph ≠ ∅ do
    start_node ← highest_orderNumber(rev_graph)
    component ← dfsCollect(start_node, rev_graph)
    strong_components.append(component)
    rev_graph.remove(component)
end while
```

Why?

"I need to hack the transformation server for that."

Why?

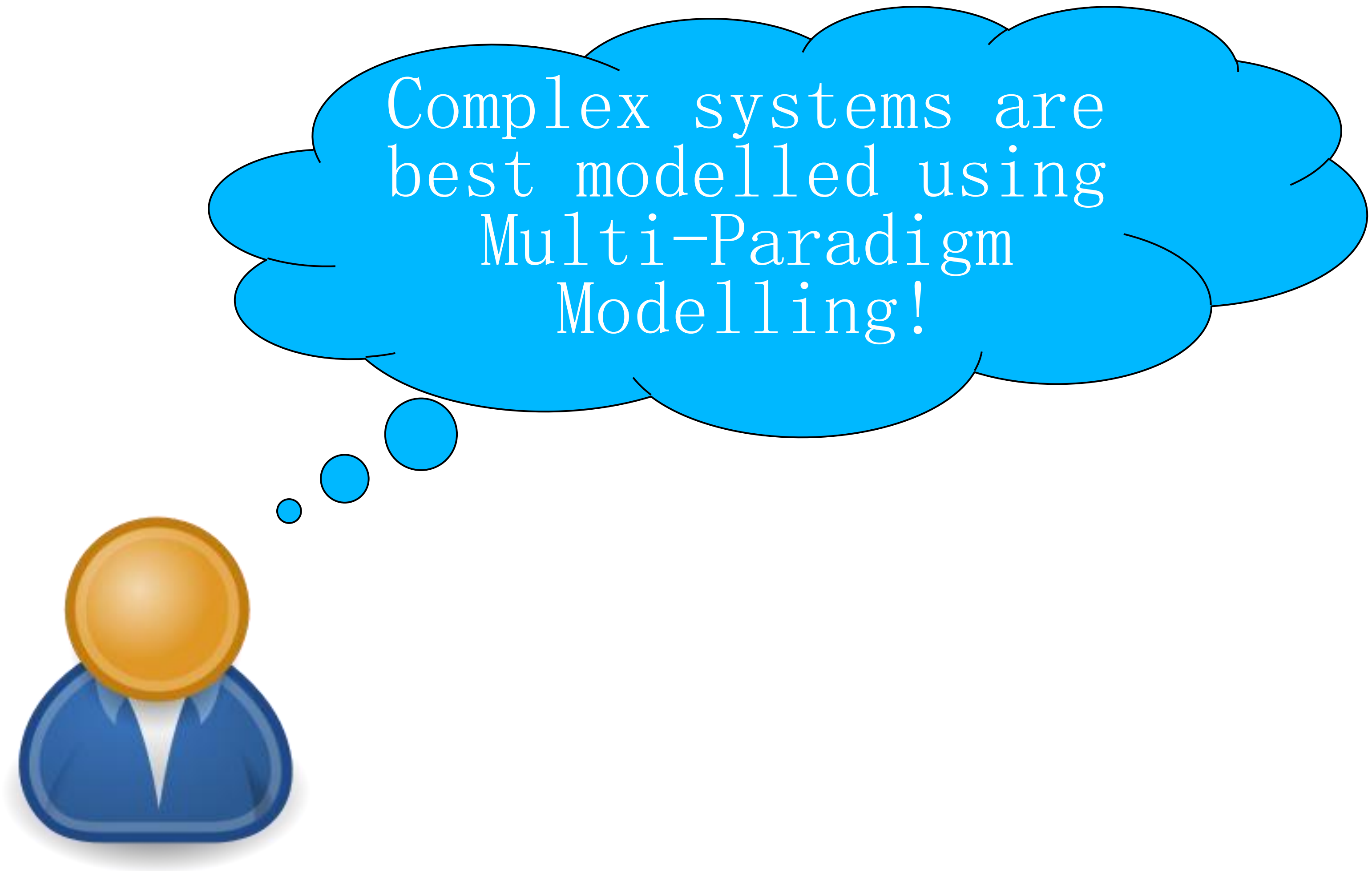
Storage

Manipulation

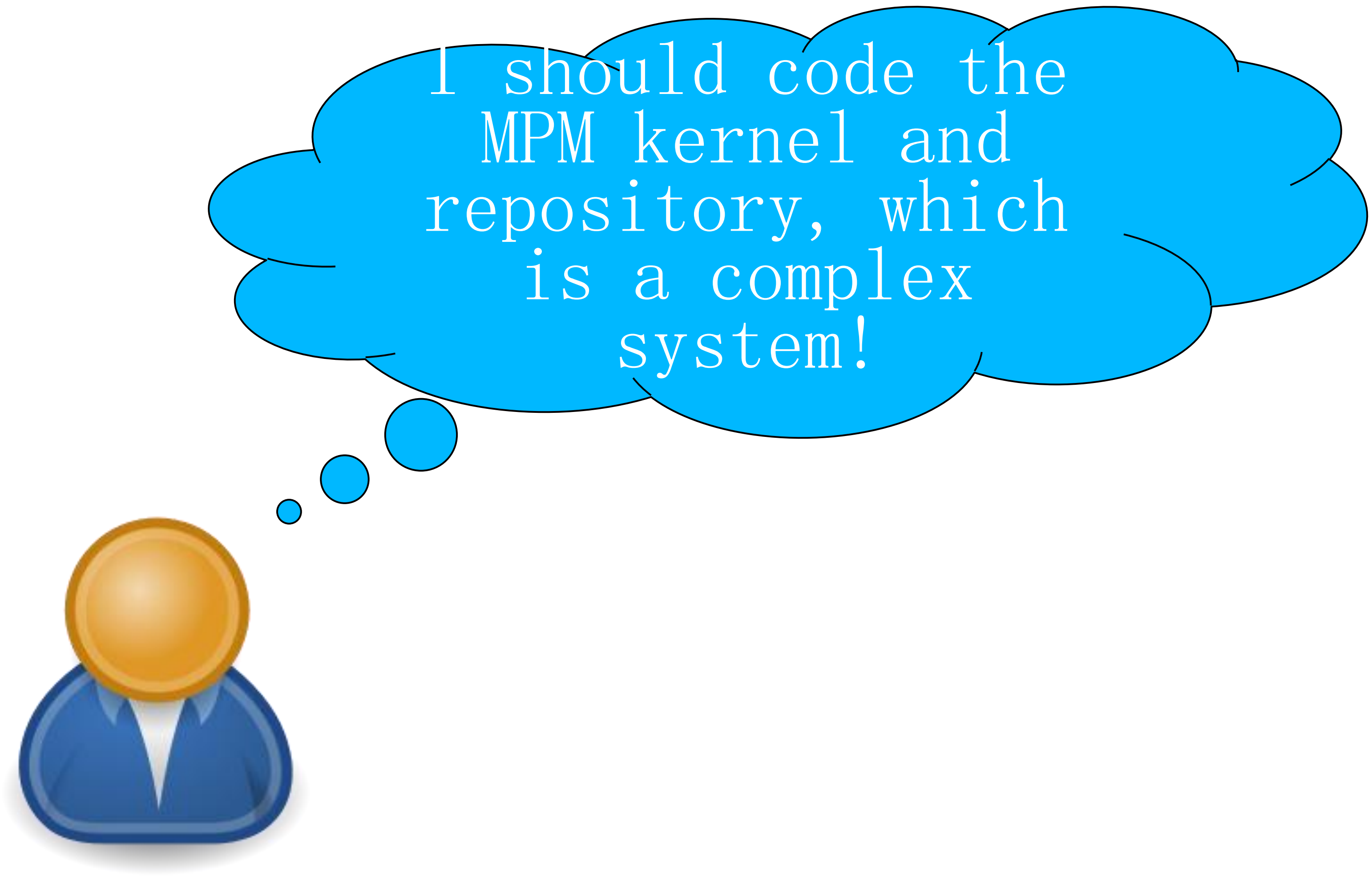
[7] R. France, J. Biemand, and B. H. C. Cheng. Repository for Model Driven Development. In Proceedings of the International Conference on Model Driven Engineering Languages and Systems (MoDELS), 311-317, 2006.

[8] F. Basciani, J. Di Rocco, D. Di Ruscio, A. Di Salle, L. Iovino, and A. Pierantonio. MDEForge: an extensible web-based modeling platform. In Proceedings of the Workshop on Model-Driven Engineering on and for the Cloud (CloudMDE), 66-75, 2014.

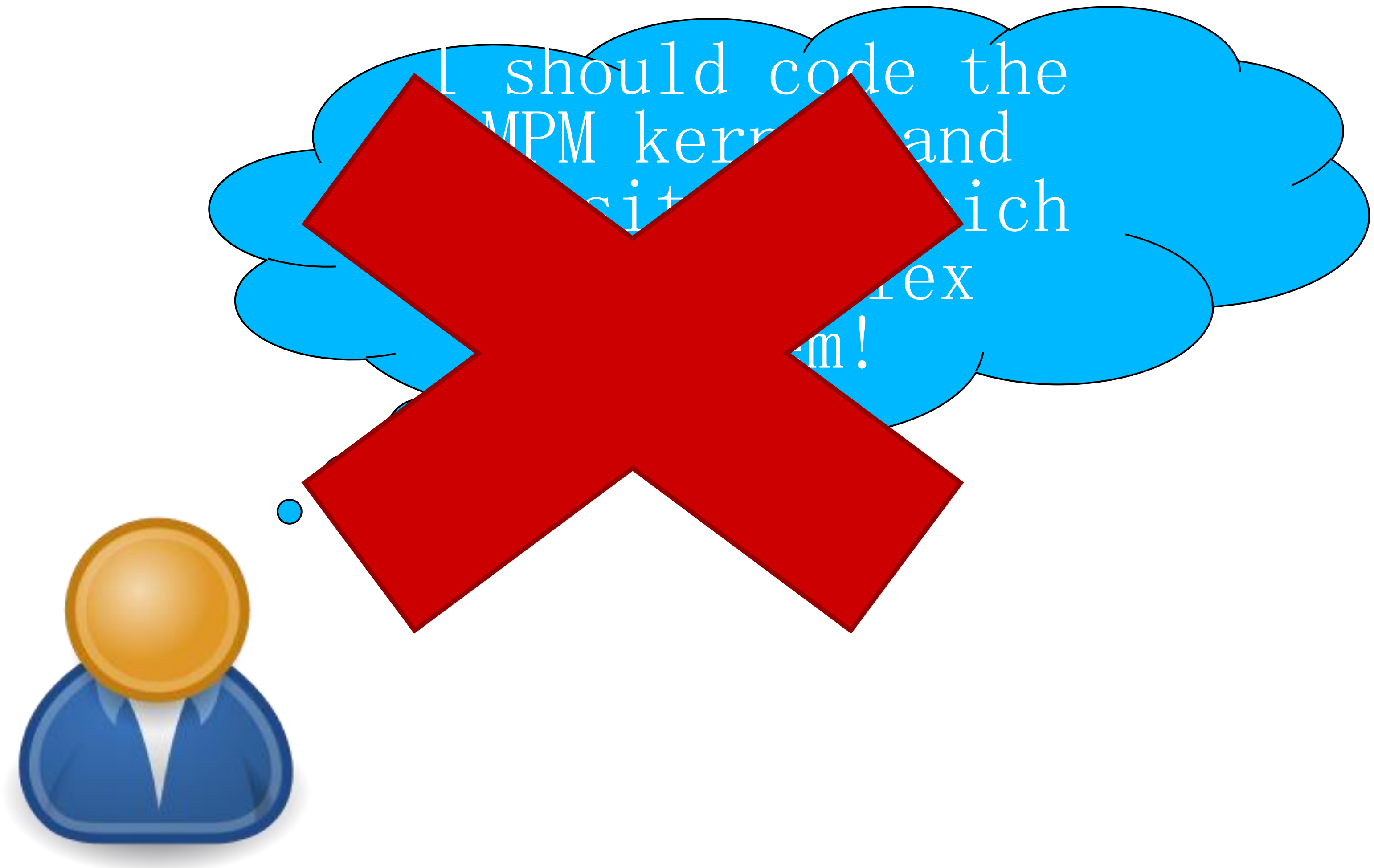
How?


Complex systems are best modelled using Multi-Paradigm Modelling!

How?


I should code the MPM kernel and repository, which is a complex system!

How?



How?

MODEL EVERYTHING!

- Protocols
- Performance
- Task management
- Action Language
- Conformance
- Services
- Concrete syntax
- Data
- Transformations

Roadmap

Modelling

- Performance
- Client-side code
- Conformance bottom (applications)
- Multi-conformance (applications)

Coupling

- Interfaces (graphical?)
- Services

Usability

- Performance
- Action language syntax
- Compiler

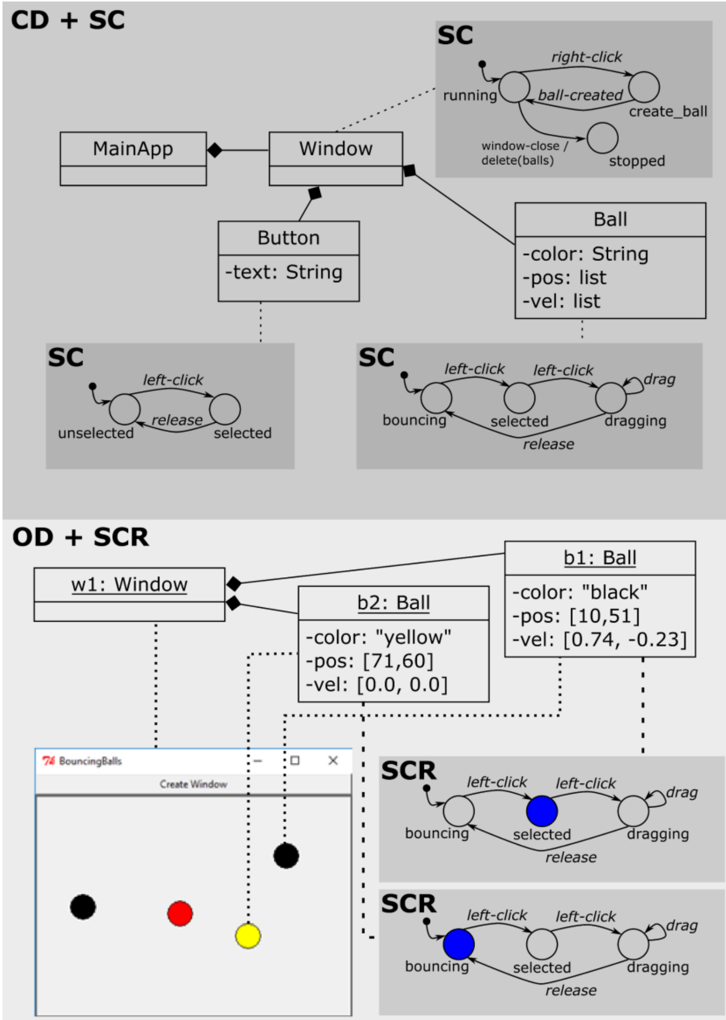
SCCD: A Statecharts and Class Diagrams Hybrid

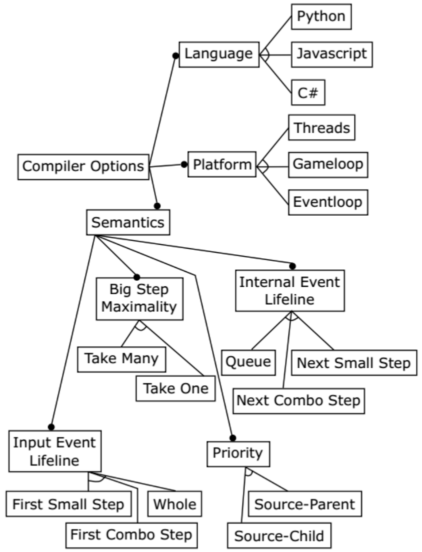
Simon Van Mierlo

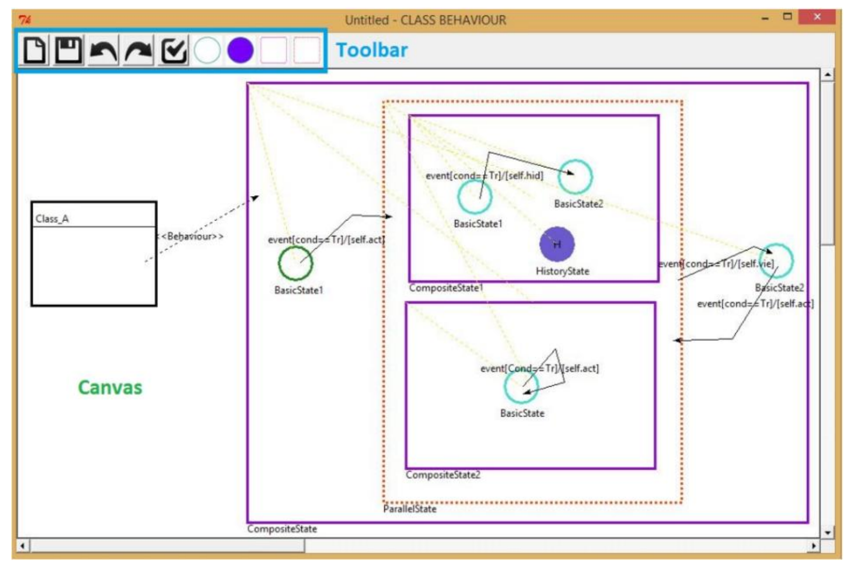
Universiteit Antwerpen

simon.vanmierlo@uantwerpen.be

Summary

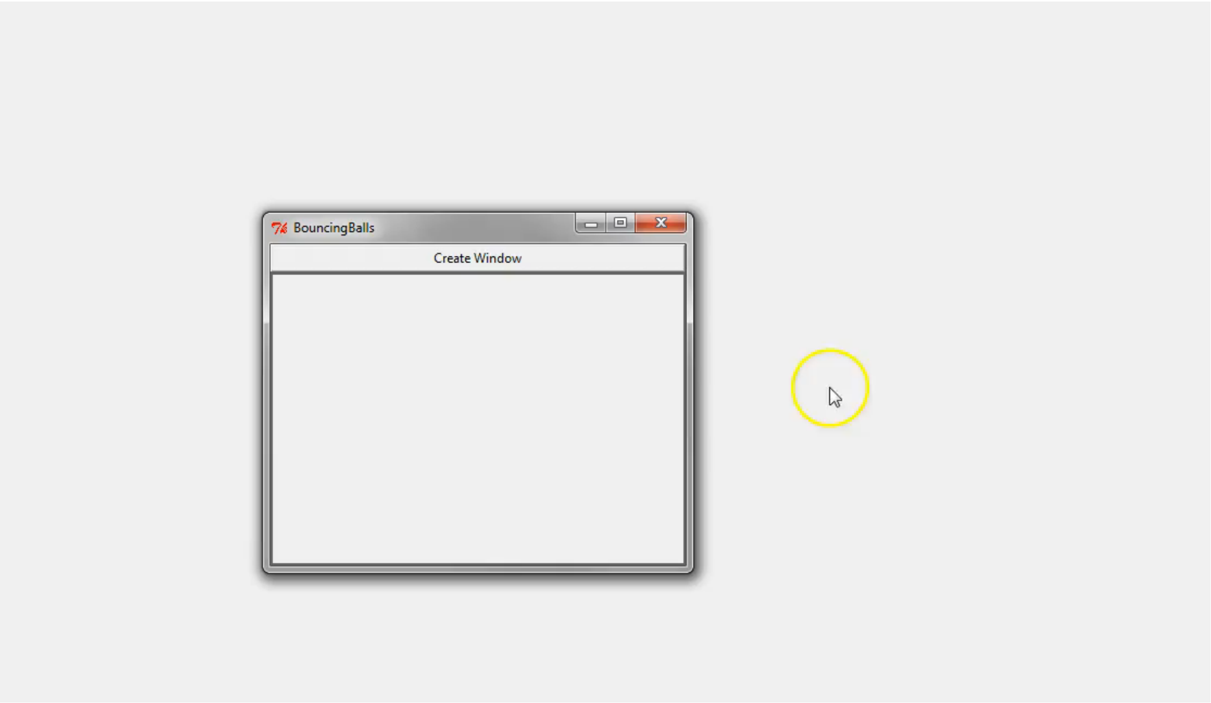






Simon Van Mierlo, Yentl Van Tendeloo, Bart Meyers, Joeri Exelmans, and Hans Vangheluwe. **SCCD: SCXML extended with class diagrams**. In *3rd Workshop on Engineering Interactive Systems with SCXML, part of EICS 2016*, 2016

Motivation



Behavior

- Timed
- Autonomous
- Interactive
- Hierarchical

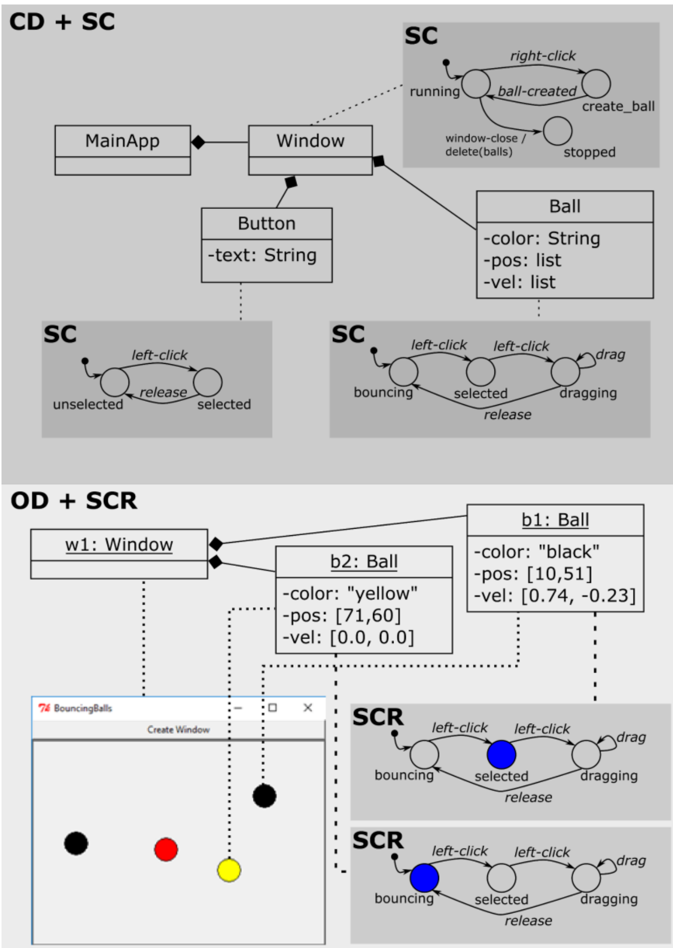
Structure

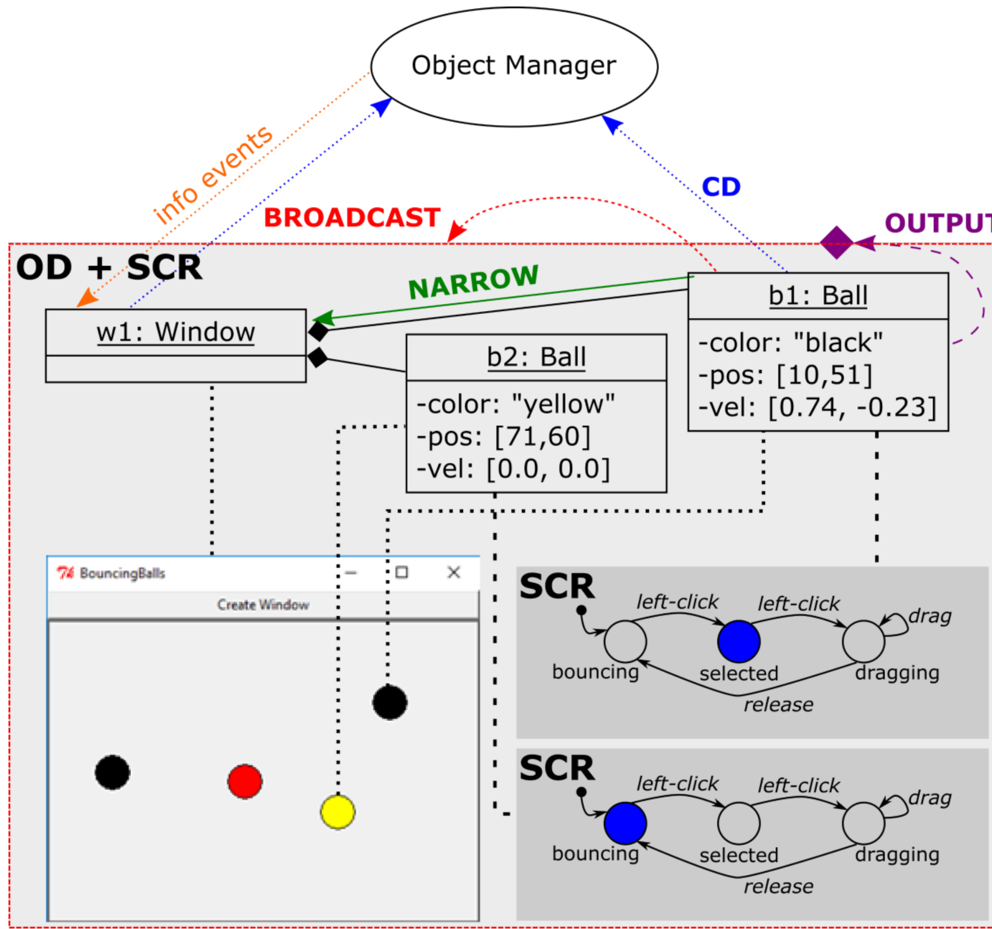
- Dynamic
- Hierarchical

Design? Statecharts + Class Diagrams = SCCD(XML)

Modelling Complex, Timed, Autonomous,

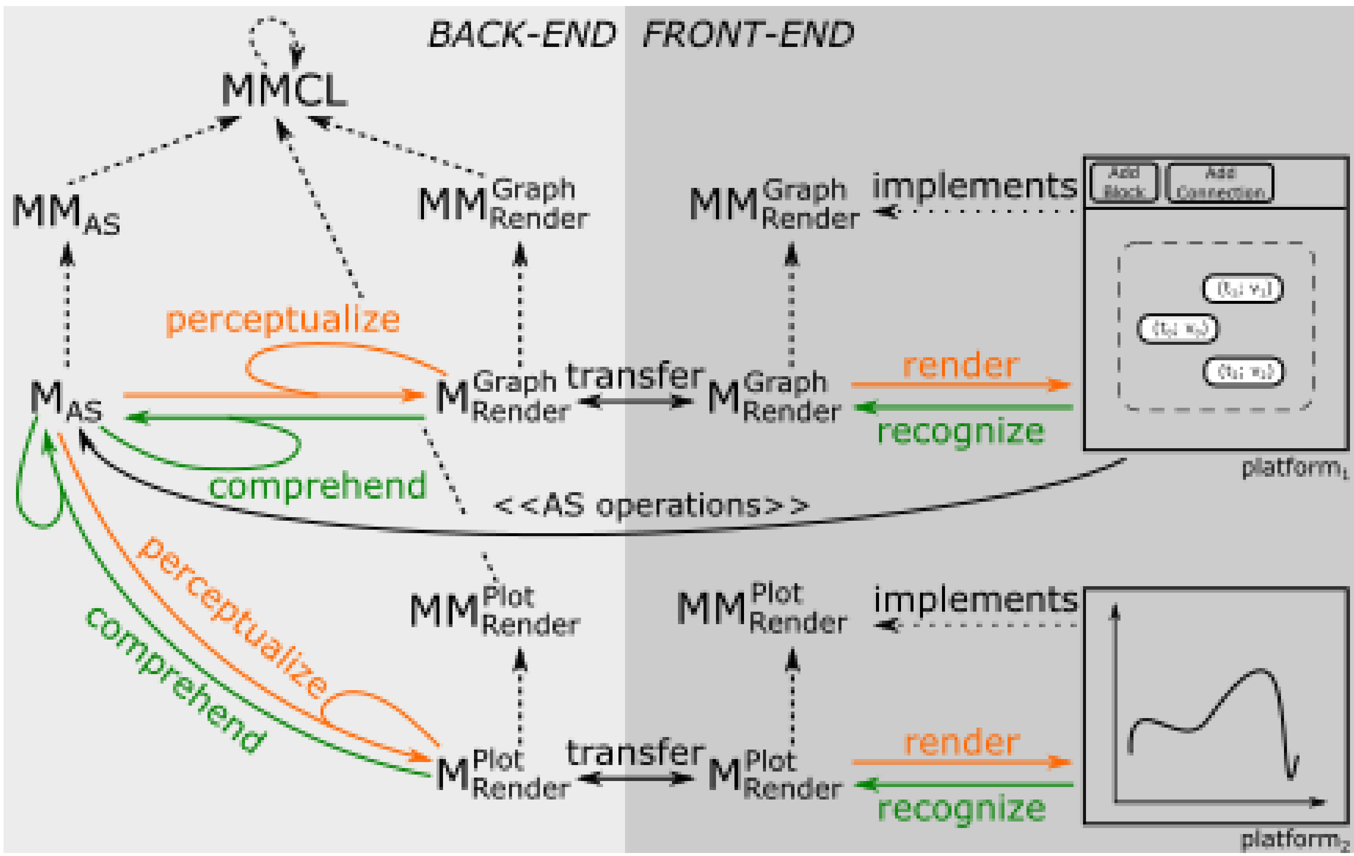
Dynamic-Structure Systems





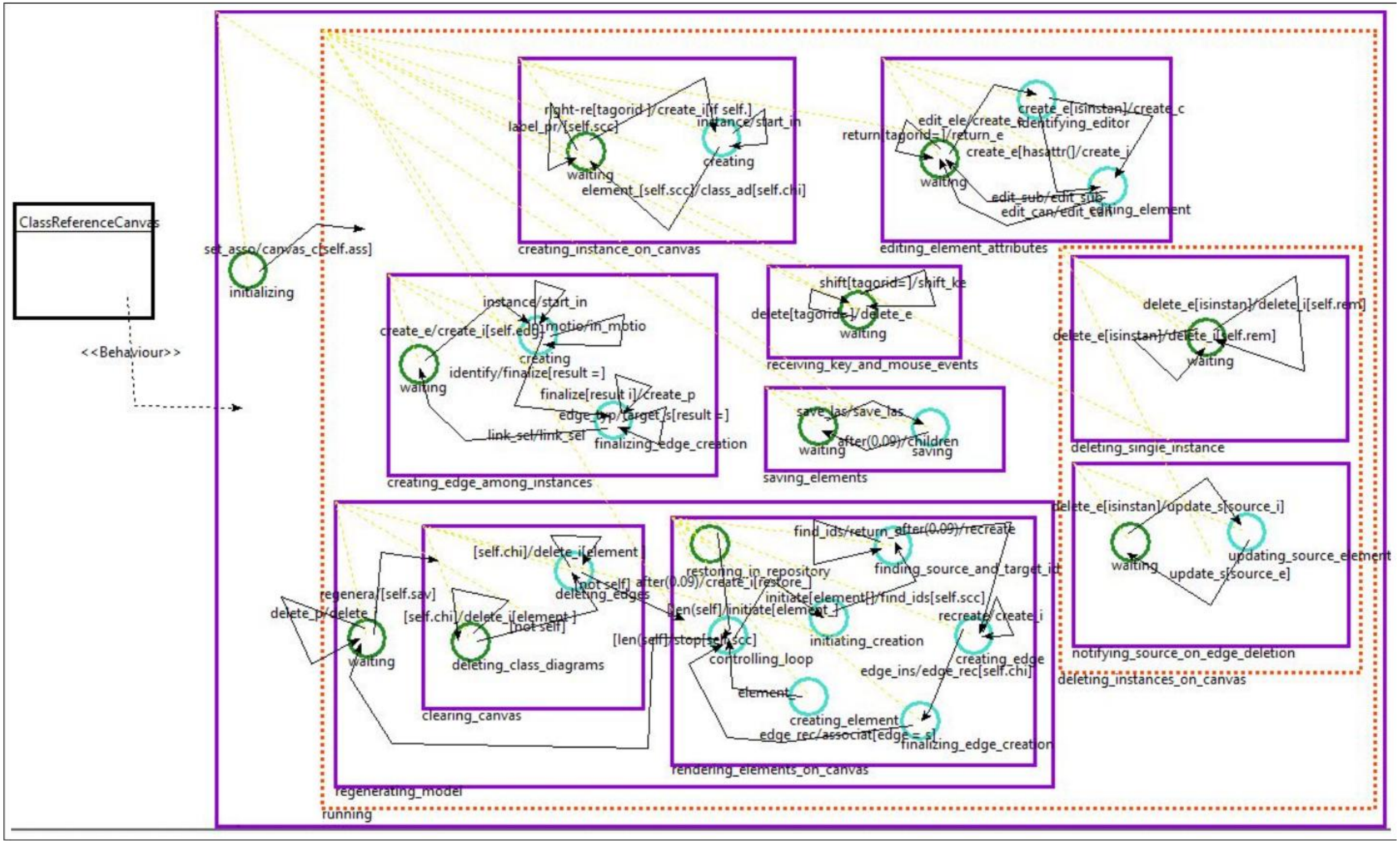
McGill School of Computer Science · nessis · FLANDERS MAKE · Ansymo Antwerp Systems & Software Modelling University of Antwerp

Visual Modelling Interface Behaviour: Concrete Syntax



McGill School of Computer Science
NECSIS
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Visual Modelling Interface Behaviour: User Interaction



Roadmap

- SCCD Language: Syntax and Semantics
  - Conformance
    - Initialization/Destruction
    - Exceptions
  - Dynamic Loading of SCCD Models
  - Interfaces/Contracts: Protocol Machine
  - Subtyping
    - Events as Objects
    - Behavior
  - Object Creation Decoupled from Associations
- Model user interaction in DSL
  - (see Vasco Sousa's work)
- Concrete Syntax: separation of AS/CS modification operations

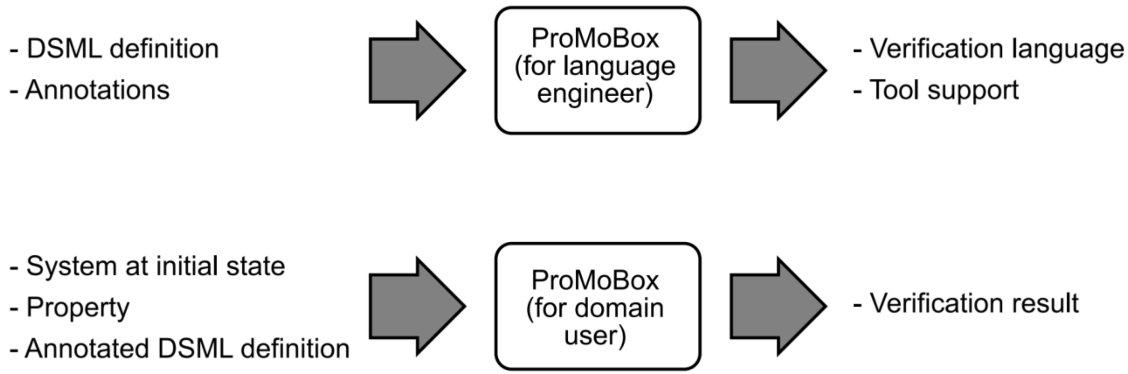
Verification of Domain-Specific Models with ProMoBox

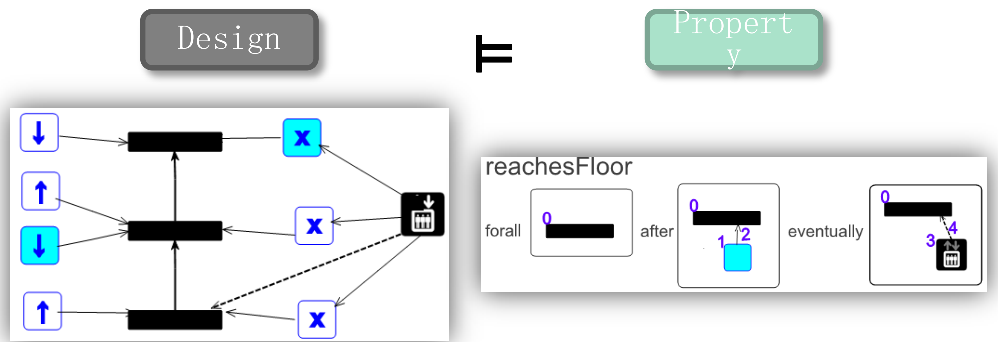
Bart Meyers

Universiteit Antwerpen

bart.meyers@uantwerpen.be

Summary



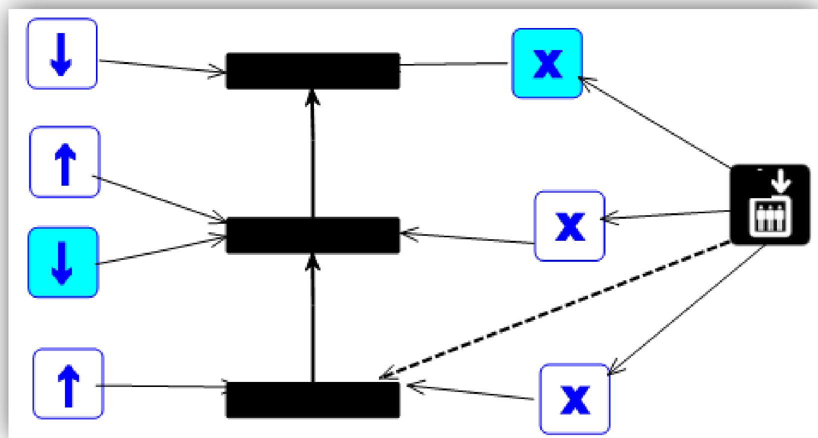


- specification of properties at DS level
- fully automatic generation of languages from annotated metamodel
- fully automatic verification of properties
- application to model checking, testing, DSE, …

- Bart Meyers, Romuald Deshayes, Levi Lucio, Eugene Syriani, Manuel Wimmer and Hans Vangheluwe. ProMoBox: A Framework for Generating Domain-Specific Property Languages. In "Proceedings of the 7th International Conference on Software Languages Engineering (SLE 2014)", Lecture Notes on Computer Science, vol. 8706, p. 1-20, 2014.
- Bart Meyers, Joachim Denil, István Dávid, and Hans Vangheluwe. Automated Testing Support for Reactive Domain-Specific Modelling Languages. In "Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering". ACM digital library, p. 181-194, 2016.

Properties for DSMLs: State of the Art

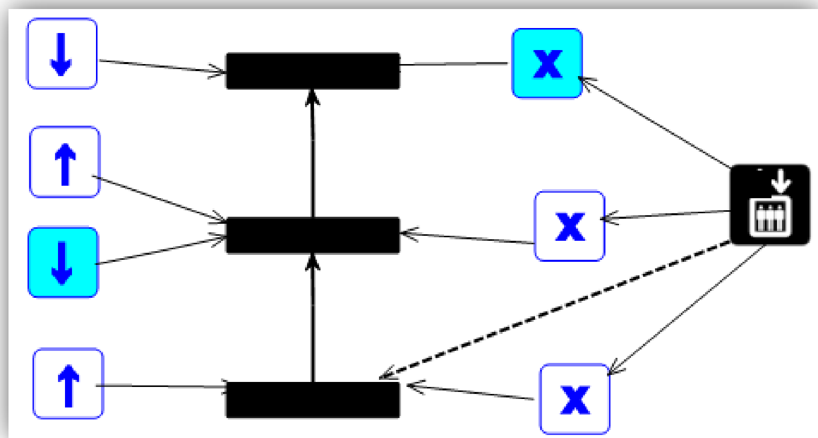
Design $\models$ Property

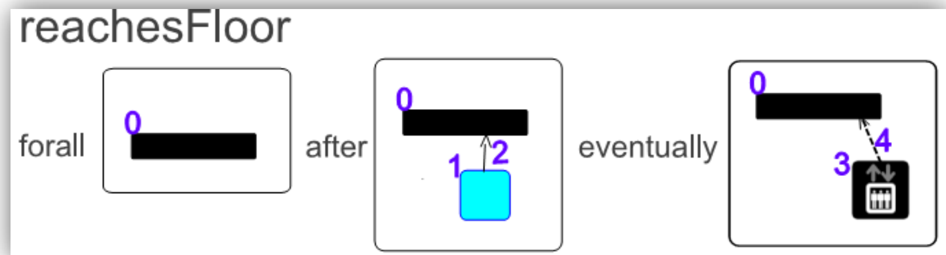


$$\Box(((go0 \wedge up0) \vee \Diamond(floor0 \vee idle)) \rightarrow ((\neg(floor0) \vee \neg(floor0 \vee idle))\mathcal{U}((floor0 \vee idle) \wedge (((floor0) \vee \neg(floor0 \vee idle))\mathcal{U}((floor0 \vee idle) \wedge ((\neg(floor0) \vee \neg(floor0 \vee idle))\mathcal{U}((floor0 \vee idle) \wedge (((floor0) \vee \neg(floor0 \vee idle))\mathcal{U}((floor0 \vee idle) \wedge (\neg(floor0)\mathcal{U}(floor0 \vee idle)))))))))))) \vee \Box(((go1 \wedge up1 \wedge down1) \vee \Diamond(floor1 \vee idle)) \rightarrow ((\neg(floor1) \vee \neg(floor1 \vee idle))\mathcal{U}((floor1 \vee idle) \wedge (((floor1) \vee \neg(floor1 \vee idle))\mathcal{U}((floor1 \vee idle) \wedge ((\neg(floor1) \vee \neg(floor1 \vee idle))\mathcal{U}((floor1 \vee idle) \wedge (((floor1) \vee \neg(floor1 \vee idle))\mathcal{U}((floor1 \vee idle) \wedge (\neg(floor1)\mathcal{U}(floor1 \vee idle))))))))))) \vee \Box(((go2 \wedge down2) \vee \Diamond(floor2 \vee idle)) \rightarrow ((\neg(floor2) \vee \neg(floor2 \vee idle))\mathcal{U}((floor2 \vee idle) \wedge (((floor2) \vee \neg(floor2 \vee idle))\mathcal{U}((floor2 \vee idle) \wedge ((\neg(floor2) \vee \neg(floor2 \vee idle))\mathcal{U}((floor2 \vee idle) \wedge (((floor2) \vee \neg(floor2 \vee idle))\mathcal{U}((floor2 \vee idle) \wedge (\neg(floor2)\mathcal{U}(floor2 \vee idle))))))))))$$

Properties for DSMLs: Property DSML

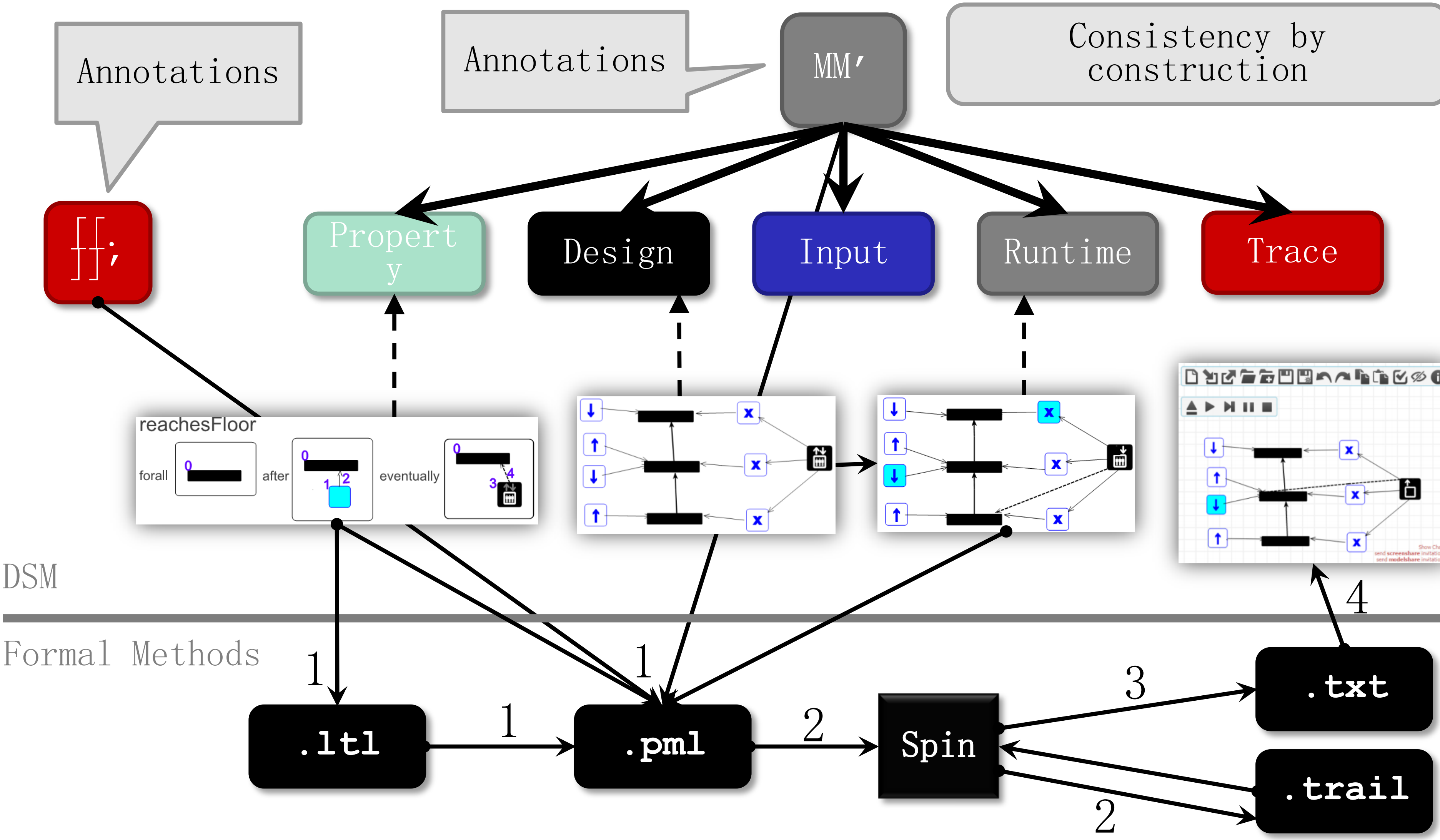
Design ⊨ Property





- Bart Meyers, Romuald Deshayes, Levi Lucio, Eugene Syriani, Manuel Wimmer and Hans Vangheluwe. ProMoBox: A Framework for Generating Domain-Specific Property Languages. In "Proceedings of the 7th International Conference on Software Languages Engineering (SLE 2014)", Lecture Notes on Computer Science, vol. 8706, p. 1-20, 2014.

Properties for DSMLs: Consistency



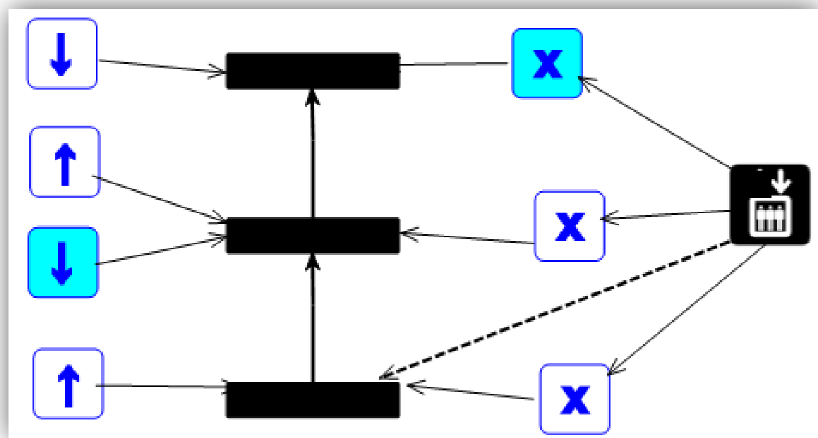
Evaluation (TSE paper)

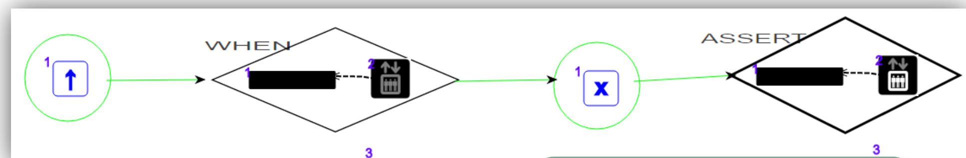
- Modelling effort
  - comparison LOC and complexity
- Correctness + Usability study
  - 6 participants, qualitative study + SUS
- Model checking performance
  - better than adapted Elevator example from literature
- Expressiveness
  - Exhaustive comparison with Promela language constructs
- Customisability
  - added patterns to property language and replaced Spin backbone with Groove

Properties for DSMLs: Testing

Design

Test Case

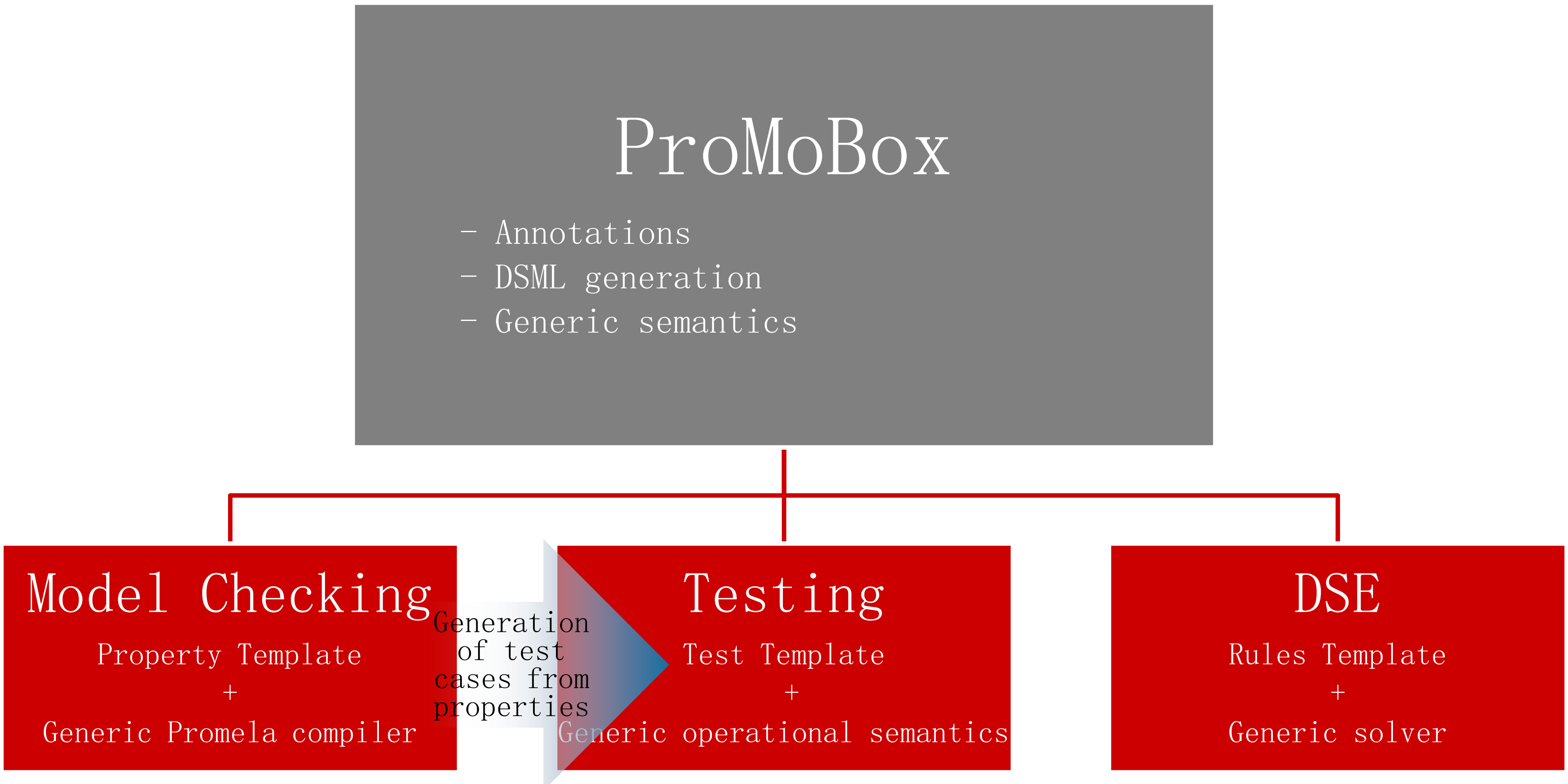




Test DSML

- Bart Meyers, Joachim Denil, István Dávid, and Hans Vangheluwe. Automated Testing Support for Reactive Domain-Specific Modelling Languages. In "Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering". ACM digital library, p. 181-194, 2016.

Roadmap


ProMoBox
- Annotations
- DSML generation
- Generic semantics
Model Checking
Property Template
+
Generic Promela compiler
Generation of test cases from properties
Testing
Test Template
+
Generic operational semantics
DSE
Rules Template
+
Generic solver

Semantic Adaptation for FMI Co-simulation*

Cláudio Gomes, Bart Meyers, Joachim Denil,
Casper Thule, Kenneth Lausdahl,
Hans Vangheluwe, Paul De Meulenaere

* Journal paper submitted to SIMULATION

McGill School of Computer Science

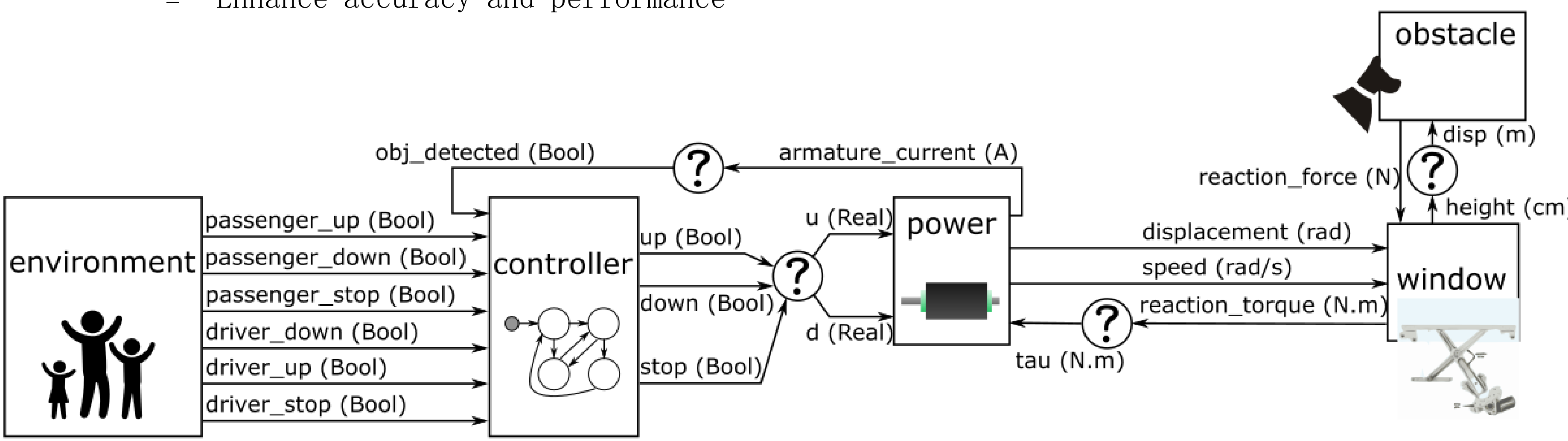
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Summary

- Why? There is a need for quick (but sound) changes to the behavior of simulators.
- What? We developed a DSL for that…
- How? …using hierarchical co-simulation principles.
- Maturity: Set of techniques and tool, applied to academic cases.

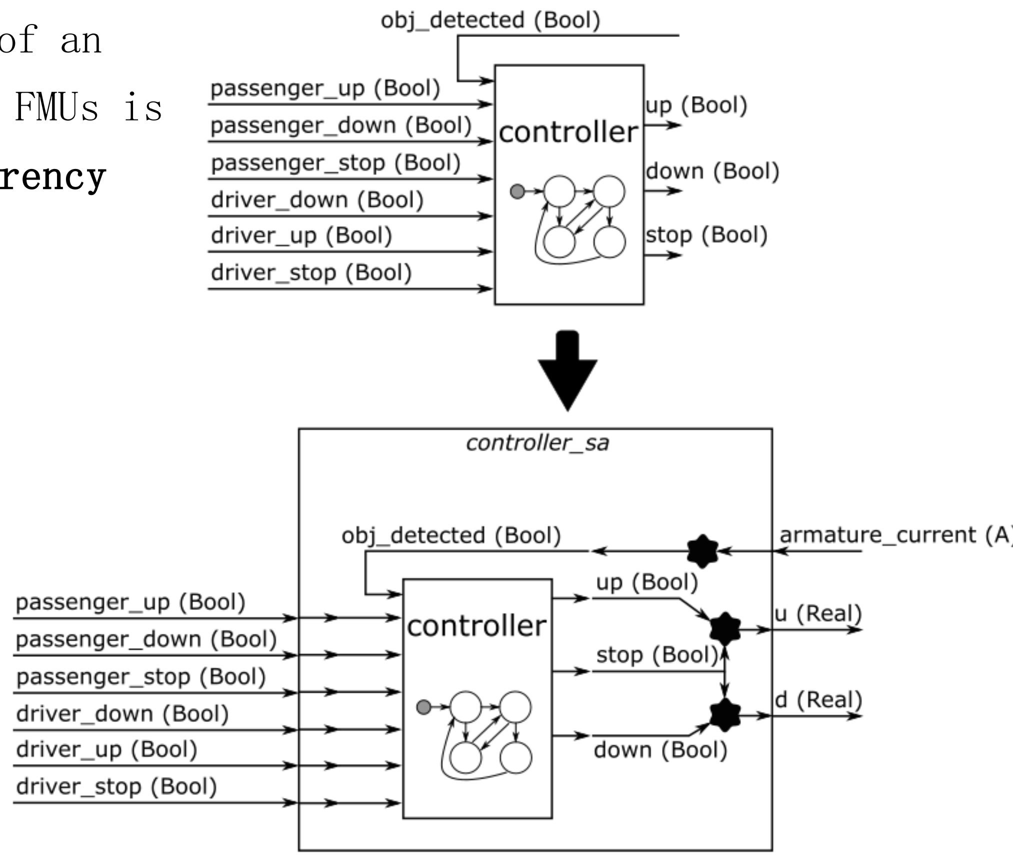
Motivation

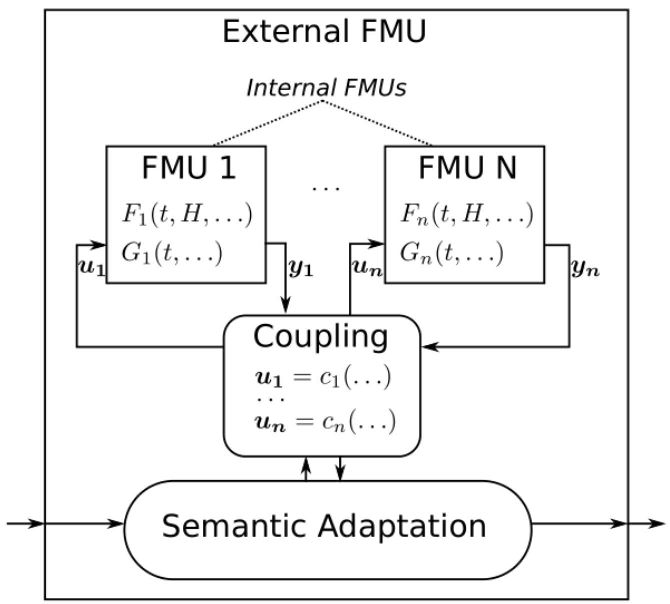
- Quick and sound way of adapting the behaviour of an interconnected set of FMUs
  - Unit conversion
  - Interaction protocol modification
  - Enhance accuracy and performance



Semantic Adaptation

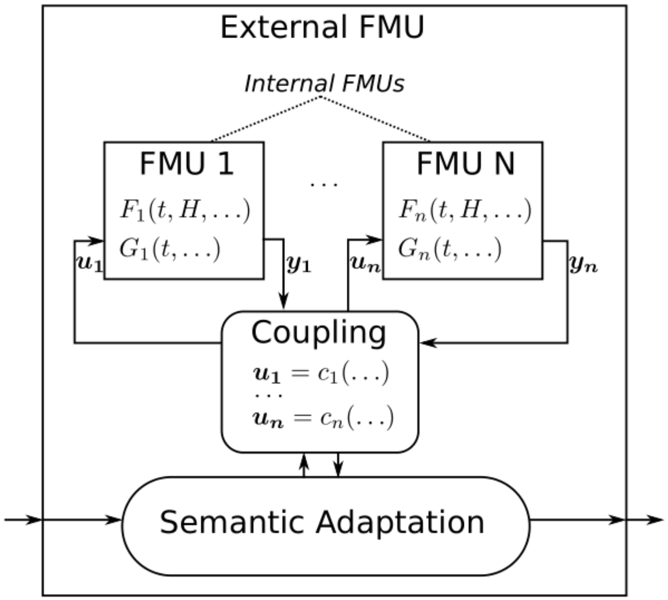
– Actions by which the **behavior** of an original set of interconnected FMUs is **altered**, following the **transparency** and **modularity** principles.

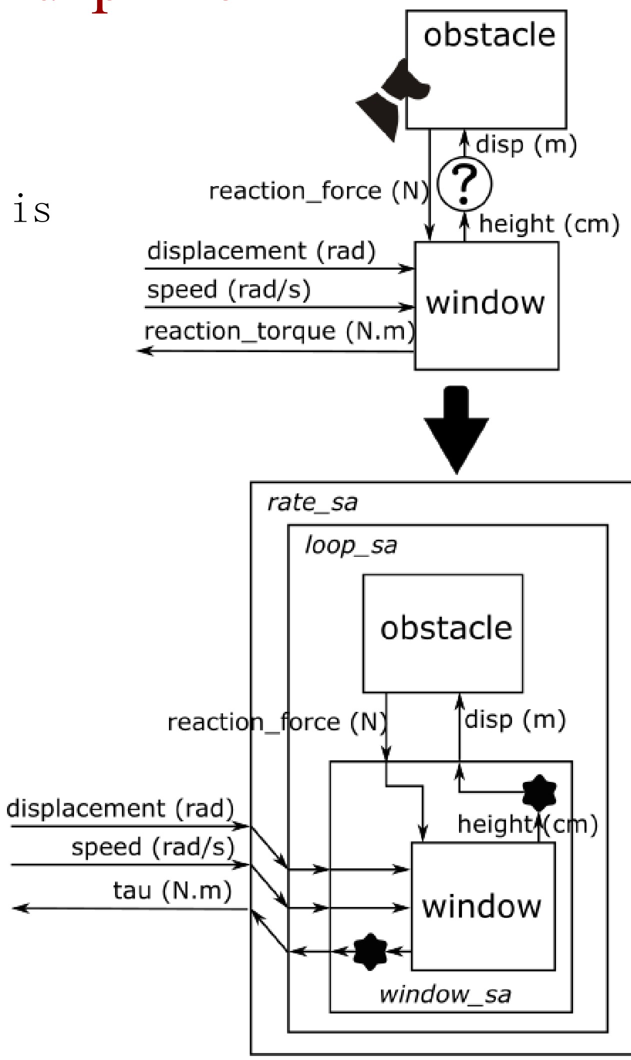




Semantic Adaptation

– Actions by which the **behavior** of an original set of interconnected FMUs is **altered**, following the **transparency** and **modularity** principles.





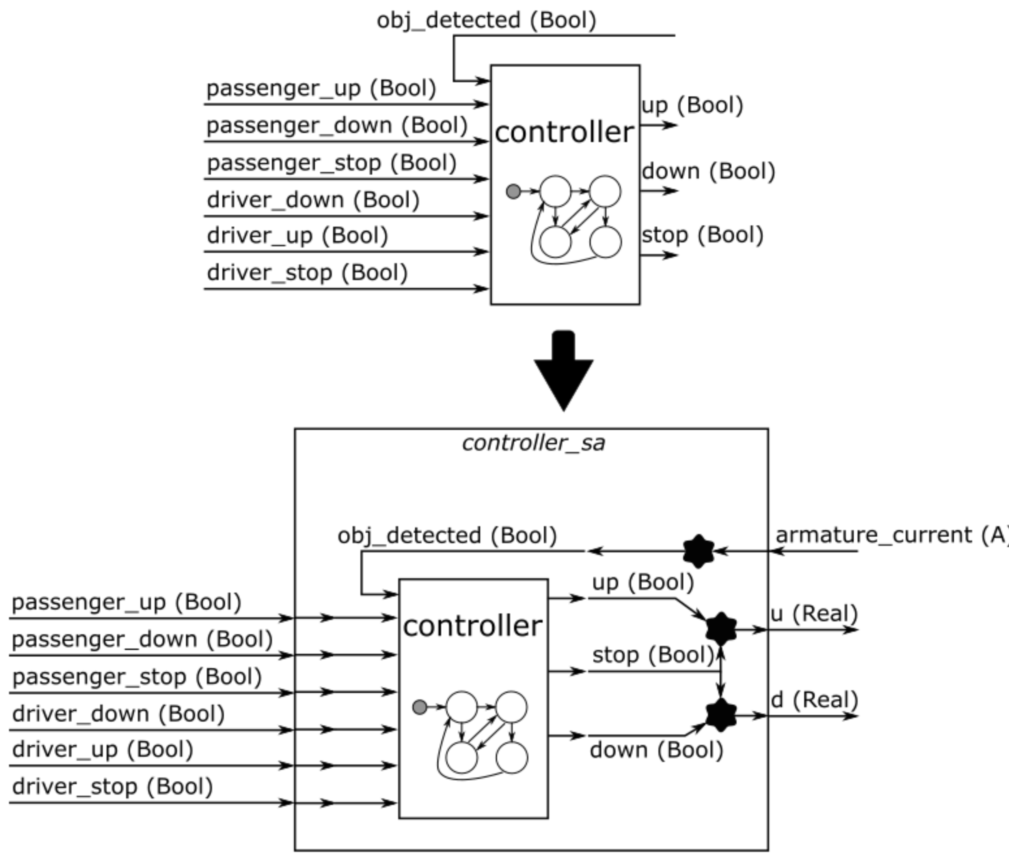
A DSL for Semantic Adaptation

```
semantic adaptation reactive moore ControllerSA controller_sa
at "./path/to/ControllerSA.fmu"

    for inner fmu Controller ctrl
    at "./path/to/LazySA.fmu"
    with input ports obj_detected, passenger_up, passenger_d
    with output ports up, down, stop

input ports armature_current -> ctrl.obj_detected,
            passenger_up -> ctrl.passenger_up,
            passenger_down -> ctrl.passenger_down,
            passenger_stop -> ctrl.passenger_stop,
            driver_up -> ctrl.driver_up,
            driver_down -> ctrl.driver_down,
            driver_stop -> ctrl.driver_stop

output ports    u, d
```



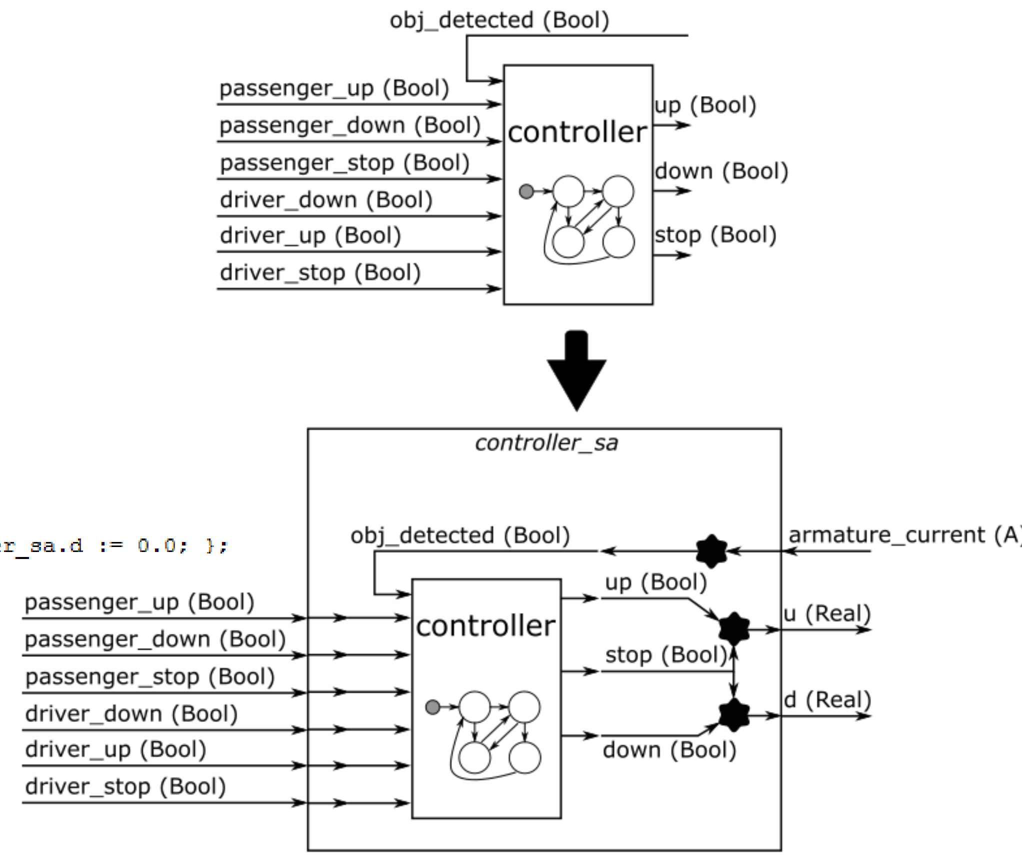
A DSL for Semantic Adaptation

```
in var  f_v := INIT_V;
in rules {
    true -> {
        f_v := controller_sa.armature_current;
    } --> {
        ctrl.obj_detected := c;
    };
}

out rules {
    ctrl.up -> { } --> {controller_sa.u := 1.0; };
    not ctrl.up -> { } --> {controller_sa.u := 0.0; };

    ctrl.down -> { } --> {controller_sa.d := 1.0; };
    not ctrl.down -> { } --> {controller_sa.d := 0.0; };

    ctrl.stop -> { } --> {controller_sa.u := 0.0 ; controller_sa.d := 0.0; };
}
```



McGill School of Computer Science

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

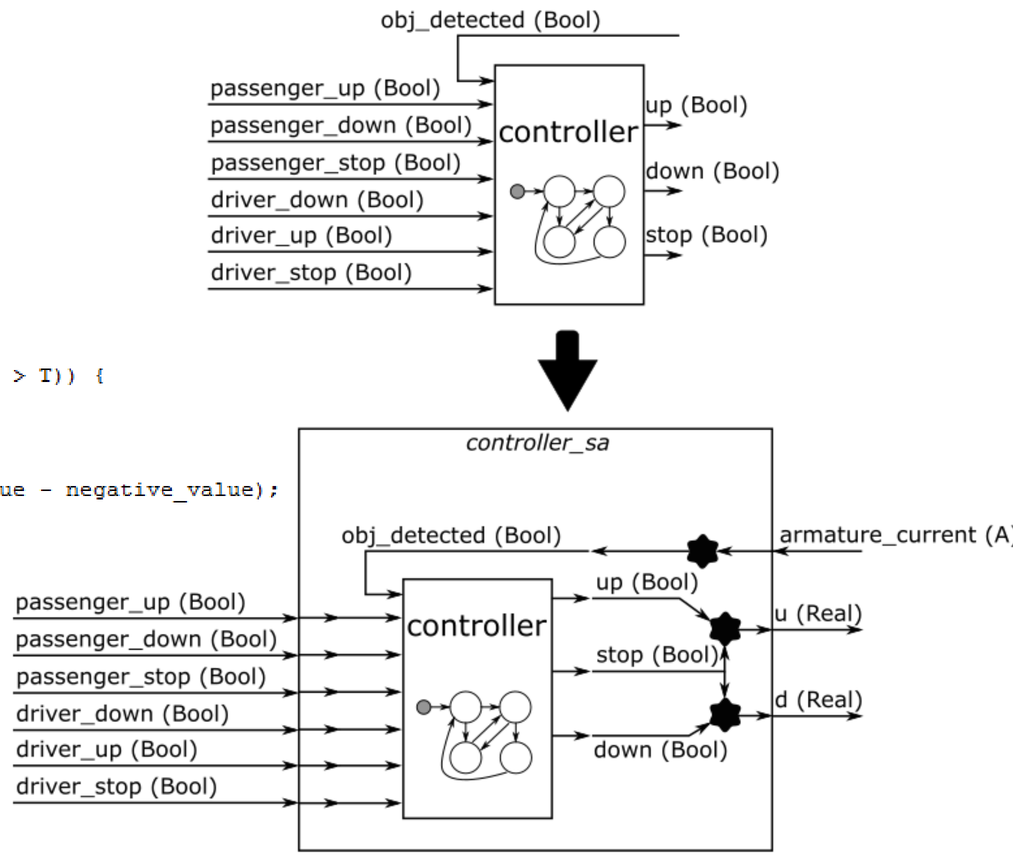
Ansymo Antwerp Systems & Software Modelling University of Antwerp

A DSL for Semantic Adaptation

```
control rules {
    var step_size := H;
    var aux_obj_detected := false;
    var crossedTooFar := false;
    if ((not is_close(p_v, T, RTOL, ATOL) and p_v < T)
            and (not is_close(f_v, T, RTOL, ATOL) and f_v > T)) {
        crossedTooFar := true;
        var negative_value := p_v - T;
        var positive_value := f_v - T;
        step_size := (H * (- negative_value)) / (positive_value - negative_value);
    } else {
        if ((not is_close(p_v, T, RTOL, ATOL) and p_v < T)
                and is_close(f_v, T, RTOL, ATOL )) {
            c := true;
        }
    }

    if (not crossedTooFar){
        step_size := do_step(ctrl, t, H);
    }

    if (is_close(step_size, H, RTOL, ATOL)) {
        p_v := f_v;
    }
    return step_size;
}
```



Hierarchical Co-simulation

$$\left\langle \boldsymbol{x}(t+\tilde{H}), \tilde{H} \right\rangle = F(t, H, \boldsymbol{x}(t), \boldsymbol{u}_{ext}(t+H))$$
$$\boldsymbol{y}(t) = G(t, \boldsymbol{x}(t), \boldsymbol{u}_{ext}(t))$$
$$\boldsymbol{x}(0) = Init(\boldsymbol{u}_{ext}(0))$$

Function $Init(\boldsymbol{u}_{ext})$
  for $i = 1, \ldots, n$ **do**
    $\boldsymbol{x_i} := \boldsymbol{up_i} := \boldsymbol{y_i} := \boldsymbol{0}$;
  end
  for $j \in (1, \ldots, n)$ **do**
    $\boldsymbol{up_{\sigma(j)}} :=$
    $c_{\sigma(j)}(\boldsymbol{u}_{ext}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_{\sigma(j)-1}}, \boldsymbol{y_{\sigma(j)+1}}, \ldots, \boldsymbol{y_n})$;
    $\boldsymbol{x_{\sigma(j)}} := Init_{\sigma(j)}(\boldsymbol{up_{\sigma(j)}})$ or $Init_{\sigma(j)}()$;
    $\boldsymbol{y_{\sigma(j)}} := G_{\sigma(j)}(0, \boldsymbol{x_{\sigma(j)}}, \boldsymbol{up_{\sigma(j)}})$
      or $G_{\sigma(j)}(0, \boldsymbol{x_{\sigma(j)}})$;
  end
  return $[\boldsymbol{up_1}, \ldots, \boldsymbol{up_n}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_n}]^T$;
end

Function
$F(t, H, [\boldsymbol{x_{in}}, \boldsymbol{x_{ctrl}}, \boldsymbol{x_{out}}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_n}]^T, \boldsymbol{u}_{ext})$
  $\tilde{\boldsymbol{x}}_{in} := In([\boldsymbol{x_{in}}, \boldsymbol{x_{ctrl}}, \boldsymbol{x_{out}}]^T, \boldsymbol{u}_{ext})$;
  $\left\langle \tilde{\boldsymbol{x}}_{ctrl}, \tilde{\boldsymbol{x}}_{out}, [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n]^T, \tilde{H} \right\rangle :=$
  $Ctrl(t, H, [\tilde{\boldsymbol{x}}_{in}, \boldsymbol{x_{ctrl}}, \boldsymbol{x_{out}}]^T, [\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}]^T)$;
  return $\left\langle [\tilde{\boldsymbol{x}}_{in}, \tilde{\boldsymbol{x}}_{ctrl}, \tilde{\boldsymbol{x}}_{out}, \tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n]^T, \tilde{H} \right\rangle$;
end

Function $G(t, [\boldsymbol{x_{in}}, \boldsymbol{x_{ctrl}}, \boldsymbol{x_{out}}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_n}]^T, \boldsymbol{u}_{ext})$
  $\tilde{\boldsymbol{x}}_{in} := In([\boldsymbol{x_{in}}, \boldsymbol{x_{ctrl}}, \boldsymbol{x_{out}}]^T, \boldsymbol{u}_{ext})$;
  if σ *is defined* **then**
    for $i = 1, \ldots, n$ **do**
      $\boldsymbol{uc_i} := \boldsymbol{y_i} := \tilde{\boldsymbol{y}}_i := \boldsymbol{0}$;
    end
    for $j \in (1, \ldots, n)$ **do**
      $[\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_n]^T :=$
      $MapIn([\tilde{\boldsymbol{x}}_{in}, \boldsymbol{x_{ctrl}}, \boldsymbol{x_{out}}]^T, 0, 0)$;
      $\boldsymbol{uc_{\sigma(j)}} :=$
      $c_{\sigma(j)}(\tilde{\boldsymbol{u}}_{\sigma(j)}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_{\sigma(j)-1}}, \boldsymbol{y_{\sigma(j)+1}}, \ldots, \boldsymbol{y_n})$;
      $\boldsymbol{y_{\sigma(j)}} := G_{\sigma(j)}(t, \boldsymbol{x_{\sigma(j)}}, \boldsymbol{uc_{\sigma(j)}})$
        or $G_{\sigma(j)}(t, \boldsymbol{x_{\sigma(j)}})$;
      $\tilde{\boldsymbol{x}}_{out} :=$
      $MapOut([\tilde{\boldsymbol{x}}_{in}, \boldsymbol{x_{ctrl}}, \boldsymbol{x_{out}}]^T, [\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}]^T, 0, 0)$;
    end
  else
    $\tilde{\boldsymbol{x}}_{out} := \boldsymbol{x_{out}}$;
  end
  $\boldsymbol{y} := Out([\tilde{\boldsymbol{x}}_{in}, \boldsymbol{x_{ctrl}}, \tilde{\boldsymbol{x}}_{out}]^T)$;
  return $\boldsymbol{y}$;
end

Roadmap

- Industrial Case Study with AgroIntelli
- Raise level of abstraction

```
import PowerWindowModel

semantic adaptation reactive moore RateLoopSA rate_loop
at "./path/to/RateLoopSA.fmu"
for fmu WindowSA windowSA, Obstacle obstacle
successive substitution starts at height with absolute tolerance = 1e-8 and
        relative tolerance = 0.0001
multiply rate 10 times with first order interpolation
```

```
semantic adaptation reactive moore RateSA rate_sa
at "./path/to/RateSA.fmu"

    for inner fmu LoopSA loop_sa
        at "./path/to/LoopSA.fmu"
        with input ports displacement (rad), speed (rad/s)
        with output ports tau (N.m)

input ports speed
output ports tau <- loop_sa.tau

param   RATE := 10;

control var previous_speed := 0;
control rules {
    var micro_step := H/RATE;
    var inner_time := t;

    for (var iter in 0 .. RATE) {
        do_step(loop_sa,inner_time,micro_step);
        inner_time := inner_time + micro_step;
    }

    previous_speed := current_speed;
    return H;
}

in var current_speed := 0;
in rules {
    true -> {
        current_speed := speed;
    } --> {
        loop_sa.speed := previous_speed + (current_speed - previous_speed)*(dt + h);
    };
}
```

McGill School of Computer Science

Ansymo Antwerp Systems & Software Modelling University of Antwerp

Stability Analysis for Adaptive Co-simulation*

Cláudio Gomes, Benoît Legat,

Raphaël M. Jungers, Hans Vangheluwe

* Paper accepted in IUTAM Symposium on Co-simulation and solver coupling – Recent developments in theory and application, September, Darmstadt

McGill School of Computer Science

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

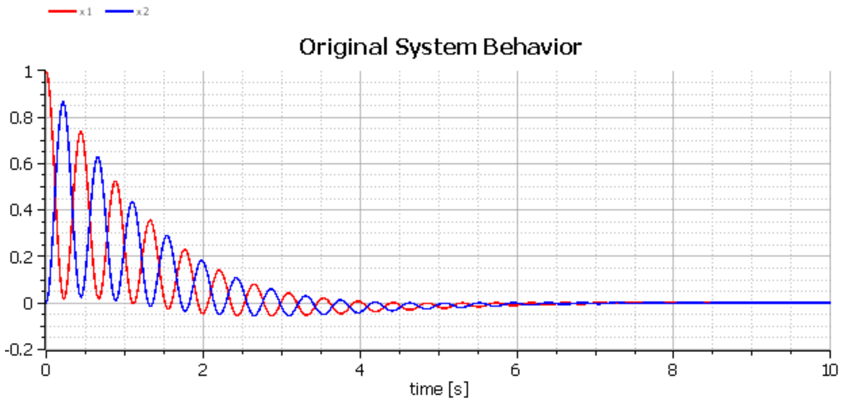
Ansymo Antwerp Systems & Software Modelling University of Antwerp

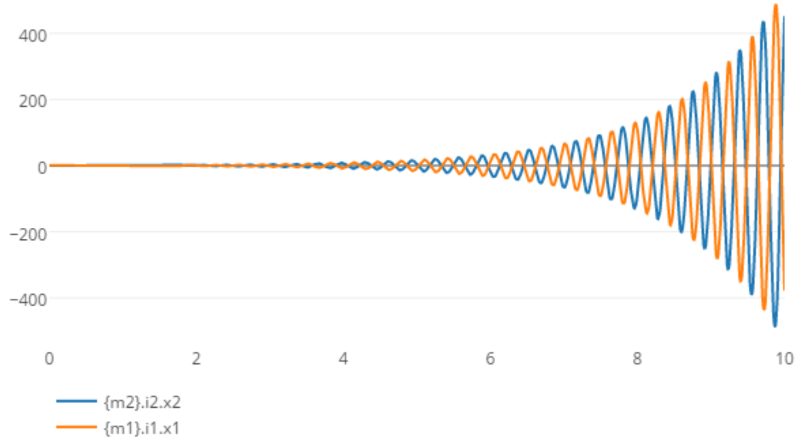
Summary

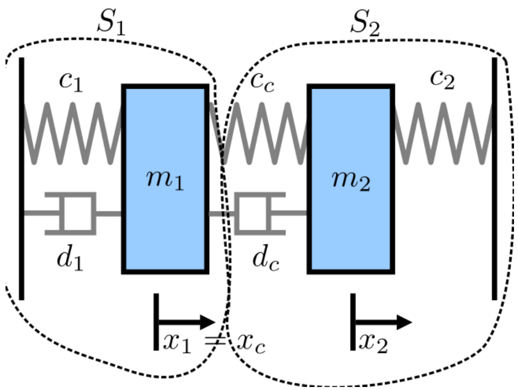
- Stability of adaptive master algorithms is seldom taken into account, but can increase performance/accuracy tradeoff.
- We apply the Joint Spectral Radius theory to study the stability of such orchestration algorithms for linear co-simulation scenarios.

Stability – Non-Adaptive Numerical Solver

$$\dot{x} = \hat{A}x \quad = \quad x_{i+1} = Ax_i$$







Stability:

$$\lim_{k \to \infty} \left\| \underbrace{AA \cdots A}_{k \text{ times}} x_0 \right\| = 0$$

Stability – Adaptive Numerical Solver

$$x_{i+1} = \begin{cases} A_1 x_i & \text{if } \; g_1(x_i) \\ A_2 x_i & \text{if } \; g_2(x_i) \\ \dots \end{cases} \quad \Rightarrow \quad x_{i+1} \in \{A x_i : A \in \Sigma\}$$

Example in co-simulation: adapt the step size

$$H \in \{0.001, 0.002, \dots, 0.01\}$$

Stability:

$$\lim_{k \to \infty} \left\| A_{s(k)} A_{s(k-1)} \cdots A_{s(0)} x_0 \right\| = 0 \quad \text{for all } A_{s(0)}, A_{s(1)}, \cdots A_{s(k)} \in \Sigma$$

Stability Analysis

Non adaptive solver (spectral radius):

$$\rho(A) = \lim_{k\to\infty} \max_{||x||=1} \left\|A^k x\right\|^{1/k} \qquad \rho(A) < 1 \Leftrightarrow \lim_{k\to\infty} \left\| \underbrace{AA\cdots A}_{k \text{ times}} x_0 \right\| = 0$$

Adaptive solver (joint spectral radius):

$$\hat{\rho}_m(\Sigma) = \sup \left\{ \max_{||x||=1} ||A_m A_{m-1} \cdots A_1 x|| : A_1, \ldots, A_m \in \Sigma \right\}$$

$$\hat{\rho}(\Sigma) = \limsup_{m\to\infty} \hat{\rho}_m(\Sigma)^{1/m}$$

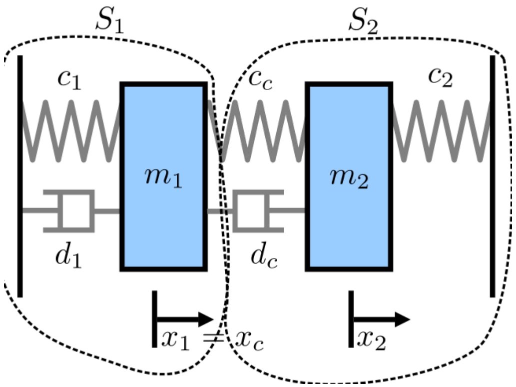
Adaptive Co-simulation Master

– Given a co-simulation scenario, and a specification of the master algorithm, one can compute Sigma

– Example:

$$\begin{bmatrix} x_2^{(n+1)} \\ v_2^{(n+1)} \end{bmatrix} = A_2^{k_2} \begin{bmatrix} x_2^{(n)} \\ v_2^{(n)} \end{bmatrix} + \left(\sum_{m=0}^{k_2-1} A_2^{m} B_2 \right) u_2^{(n)}$$

$$y_2^{(n)} = \begin{bmatrix} c_k & d_k \end{bmatrix} \begin{bmatrix} x_2^{(n)} \\ v_2^{(n)} \end{bmatrix} + \begin{bmatrix} -c_k & -d_k \end{bmatrix} u_2^{(n)}$$



$$\begin{bmatrix} x_1^{(n+1)} \\ v_1^{(n+1)} \\ x_2^{(n+1)} \\ v_2^{(n+1)} \end{bmatrix} = A_{\text{euler}} \begin{bmatrix} x_1^{(n)} \\ v_1^{(n)} \\ x_2^{(n)} \\ v_2^{(n)} \end{bmatrix}$$

$$A_{\text{euler}} = \begin{bmatrix} A_1^{k_1} & \bar{0} \\ \bar{0} & A_2^{k_2} \end{bmatrix} + \begin{bmatrix} \sum_{m=0}^{k_1-1} A_1^{m} & \bar{0} \\ \bar{0} & \sum_{m=0}^{k_2-1} A_2^{m} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ -h_1 c_k & -h_1 d_k & h_1 c_k & h_1 d_k \\ 0 & 0 & 0 & 0 \\ h_2 \frac{1}{m_2} c_k & h_2 \frac{1}{m_2} d_k & 0 & 0 \end{bmatrix}$$

Roadmap

- Address scalability by using adaptive master specifications based on state machines.
- Identify conditions for which JSR can be computed directly. (Example: a repeating sequence of matrices)

Stability Analysis for Hybrid Co-simulation*

Cláudio Gomes, Paschalis Karalis,

Eva M. Navarro-López, Hans Vangheluwe

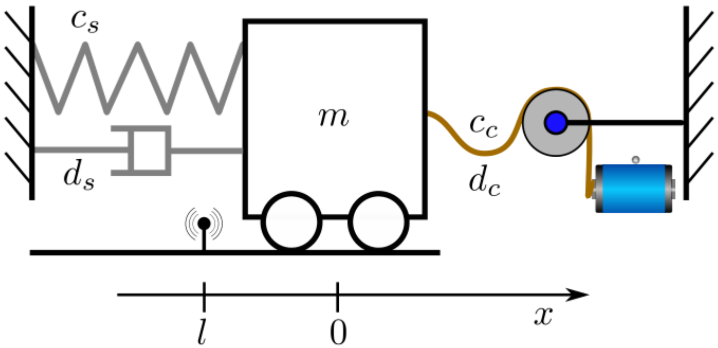
* Paper accepted in Workshop on Formal Co-Simulation of Cyber-Physical Systems, September, Trento

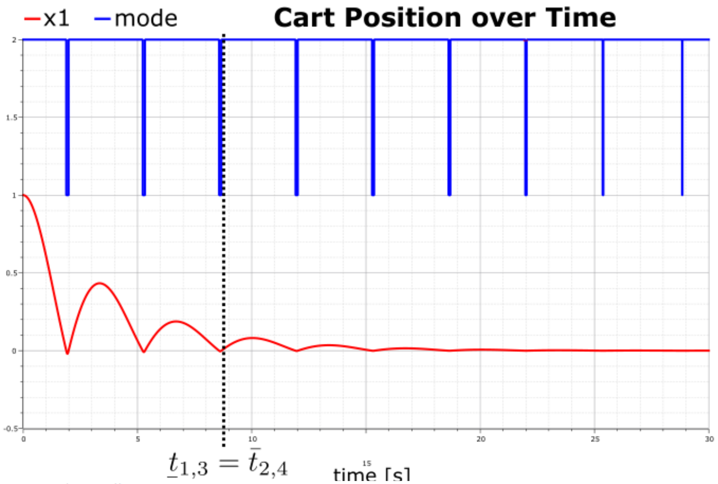
McGill School of Computer Science
necsis
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

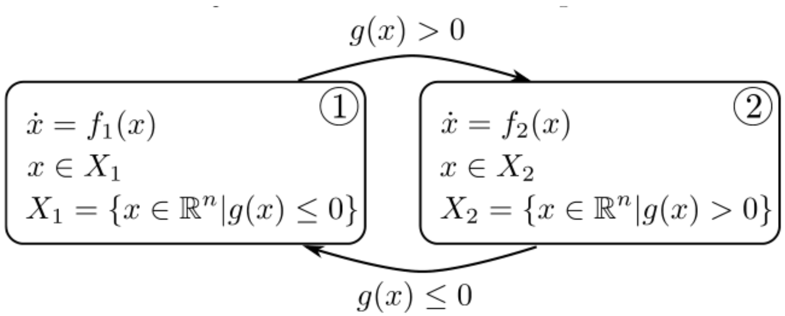
Summary

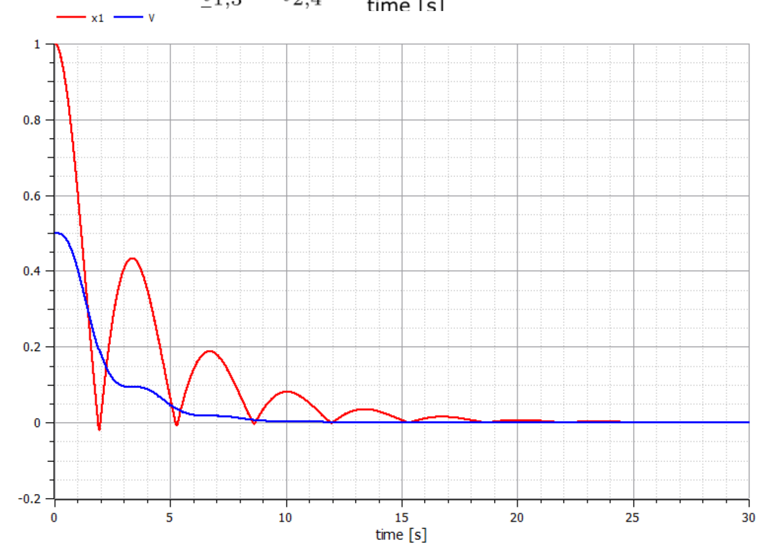
- A co-simulation of a hybrid system must preserve the stability properties of the later, so that the results can be trustworthy.
- We analyze the range of communication frequencies between simulators that ensure those properties are kept.

Example: Hybrid System

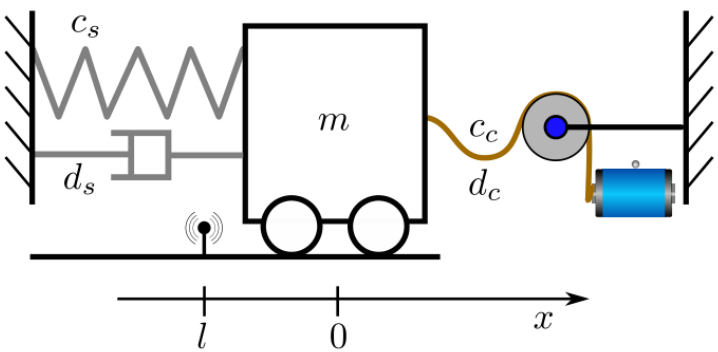


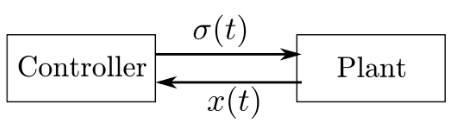


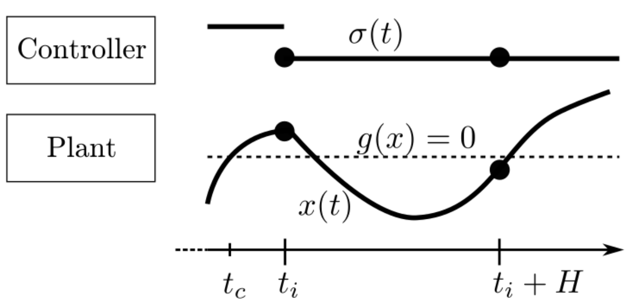


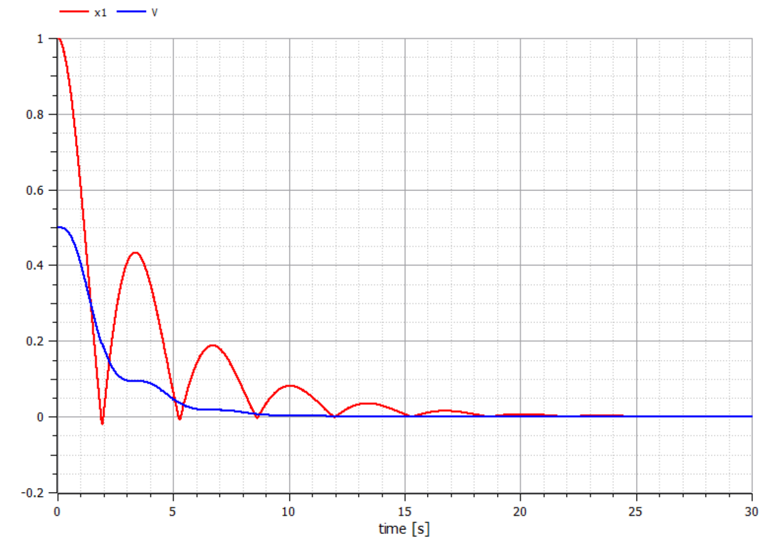


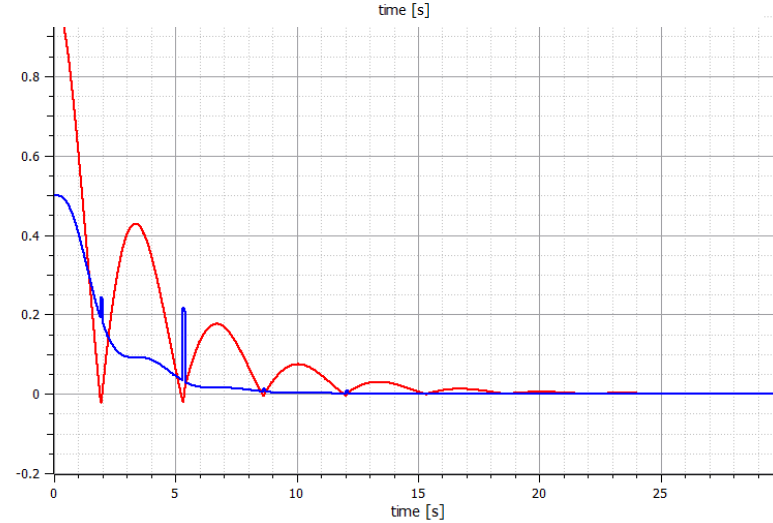
Example: Hybrid Co-simulation Scenario



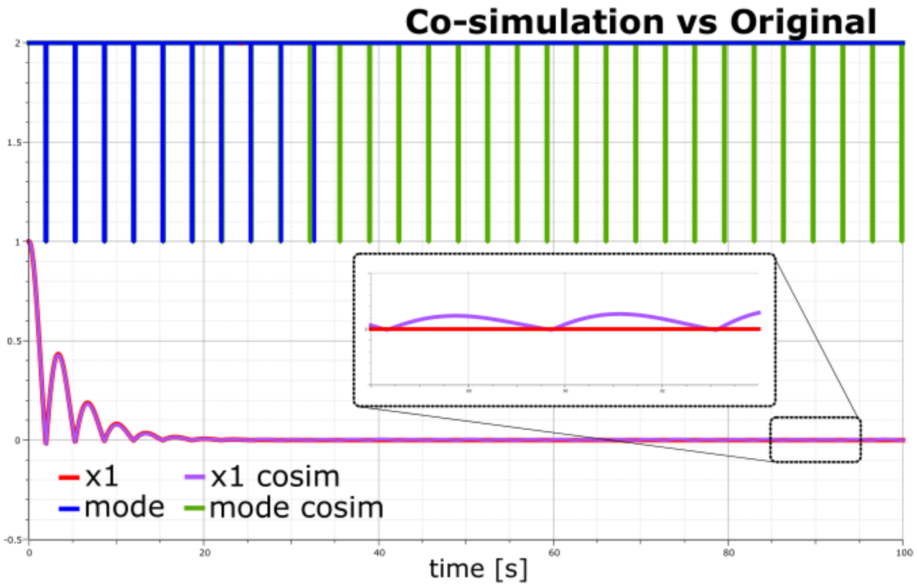




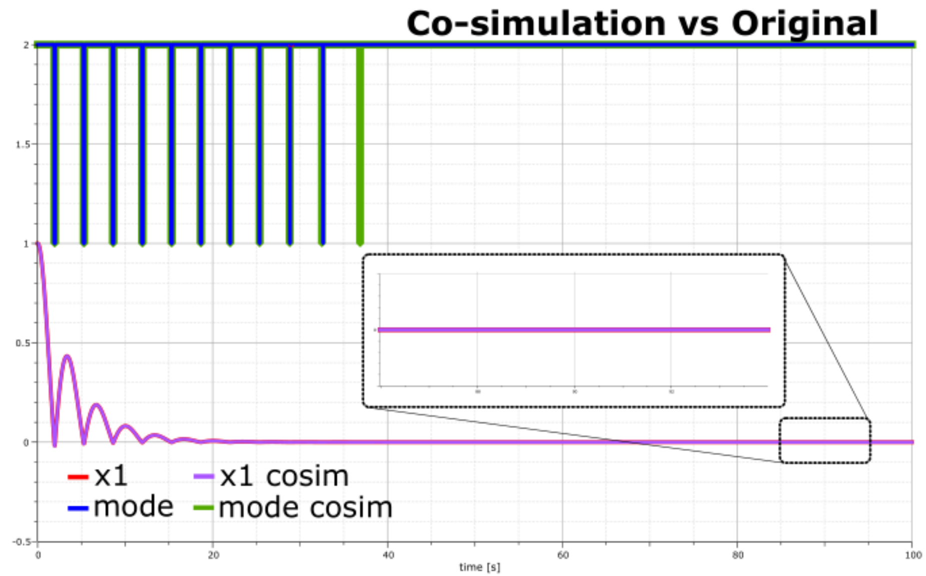




Question: How much delay can be tolerated?

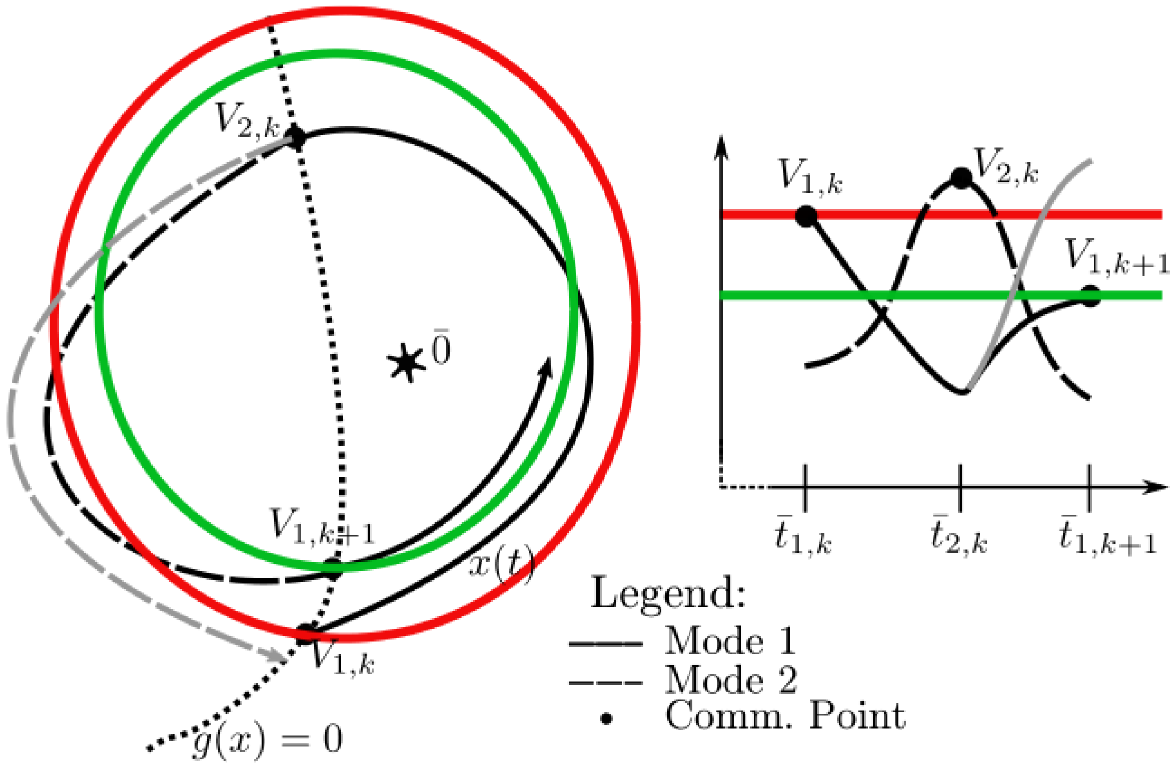


H = 0.05

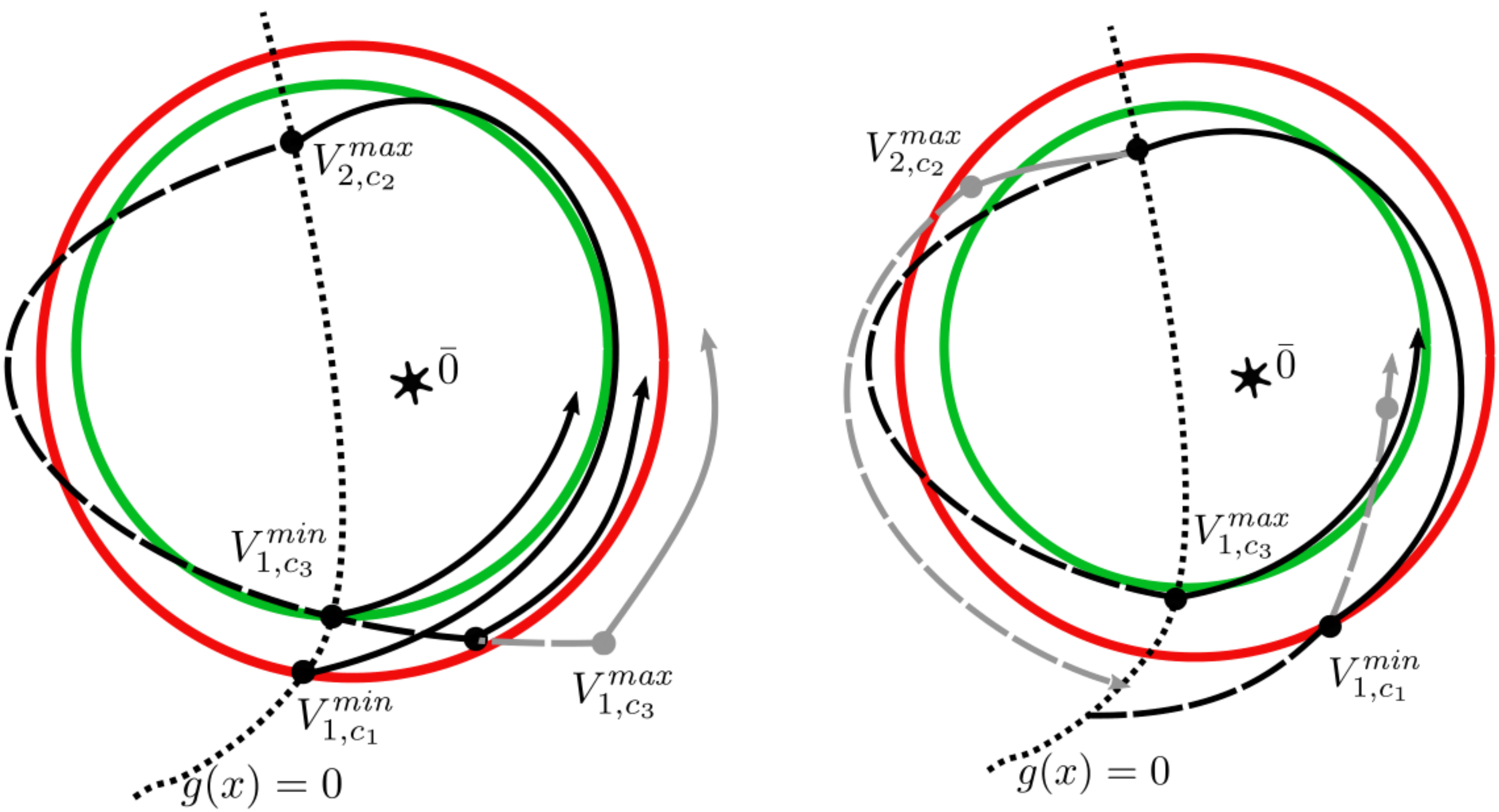


H = 0.001

(Lyapunov) Stability of Hybrid Systems



(Lyapunov) Stability of Hybrid Co-simulation



Roadmap

- Generalize the approach for many modal systems (not just two), and systems with resets.
- Use a more relaxed Lyapunov stability theorem, developed by Paschalis and Eva

Hybrid System Simulation with Dirac Deltas*

Cláudio Gomes, Yentl Van Tendeloo,

Joachim Denil, Paul De Meulenaere,

Hans Vangheluwe

* Paper accepted in Symposium on Theory of Modeling and Simulation, April, Virginia Beach

Summary

– We compare two different approaches for the simulation of impulsive differential equation, and formulate their differences.

McGill School of Computer Science NECSIS

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK Ansymo Antwerp Systems & Software Modelling University of Antwerp

Impulse-based Modeling

Bouncing ball dynamics:

$$y'' = -g + F_c(t)$$

Around a collision:

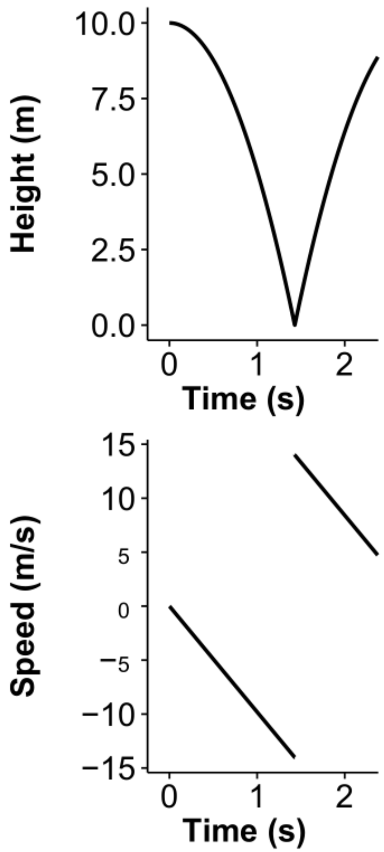
$$y'(t_c^+) = y'(t_c^-) + \int_{t_c^-}^{t_c^+} -g + F_c(\tau) d\tau$$

(Momentum) Conservation dictates:

$$y'(t_c^+) = -y'(t_c^-)$$

Hence, <u>whatever the shape of Fc</u>,

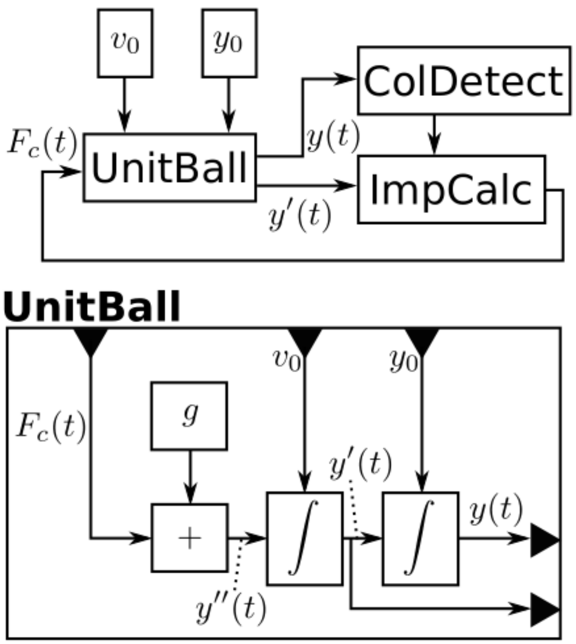
$$\int_{t_c^-}^{t_c^+} F_c(\tau) d\tau = -2y'(t_c^-)$$



Impulse-based Modeling

Bouncing ball dynamics:

$$y'' = -g + F_c(t)$$



Let δ be a function abstraction, such that:

$$\int_{0^-}^{0^+} \delta(\tau) d\tau = 1$$

Then:

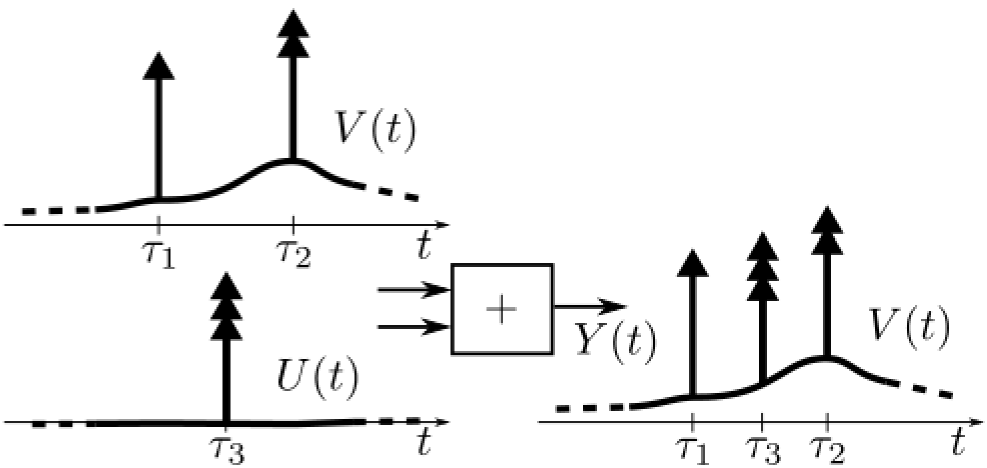
$$F_c(\tau) = -2y'(t_c^-)\delta(t - t_c)$$

Symbolic Simulation of Impulses

Manipulate signals with impulses encoded

$$S(t) = s(t) + \sum_{i=0}^{n} \sum_{\tau_j \in \{\tau_j\}} a_{ij}\delta^{(i)}(t - \tau_j)$$

$$S(t_i) \in \mathbb{R}^2 \times \mathbb{R}^m$$

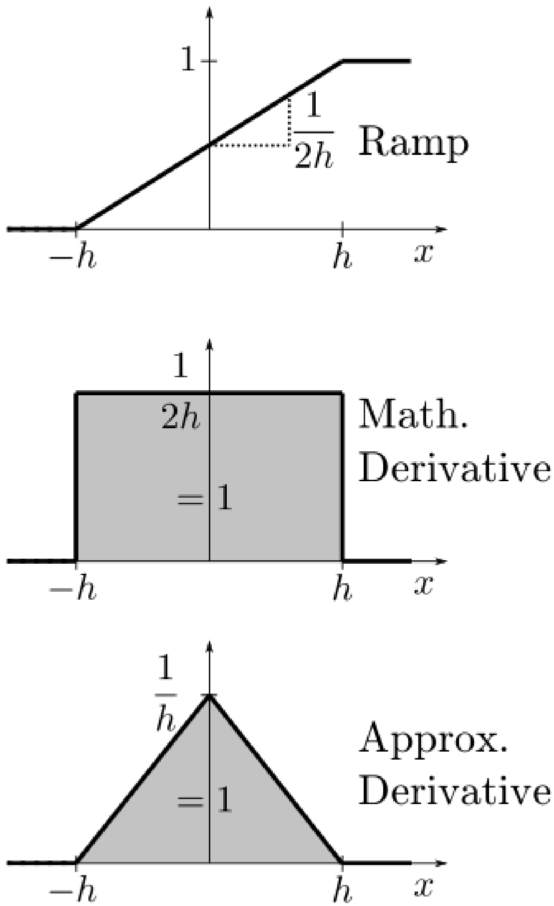


Numerical Simulation of Impulses

Numerically approximate an impulse as the derivative of a steep ramp.

$$\delta(t - \tau_d) \approx \frac{1}{h}$$

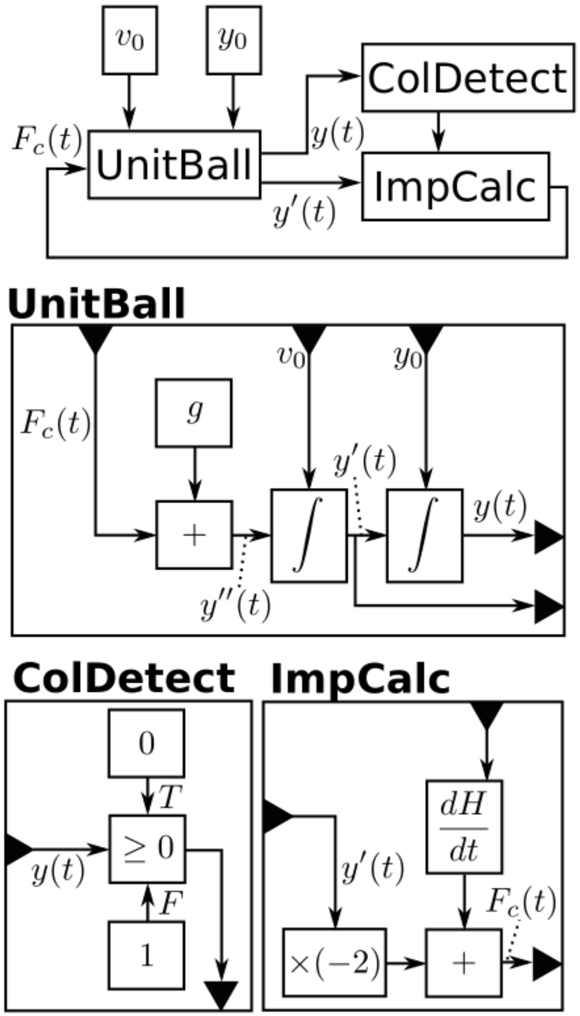
$$\int_{0^-}^{0^+} \delta(\tau) d\tau = 1$$

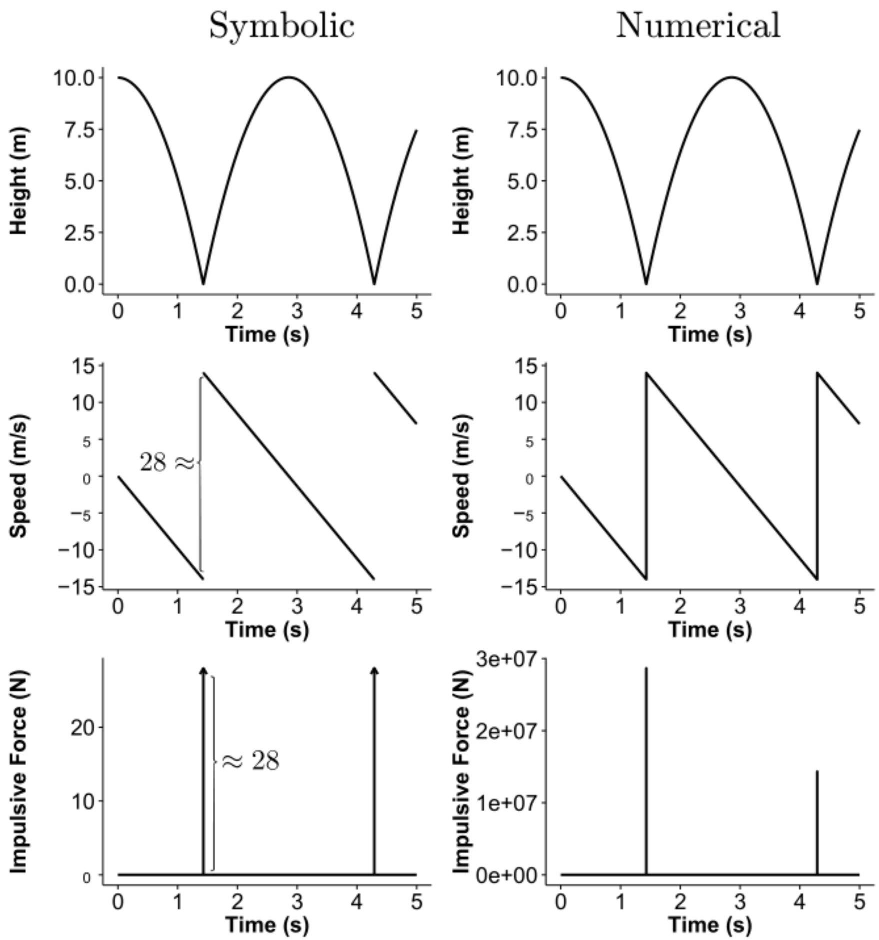


McGill School of Computer Science

Ansymo Antwerp Systems & Software Modelling University of Antwerp

Example: Bouncing ball

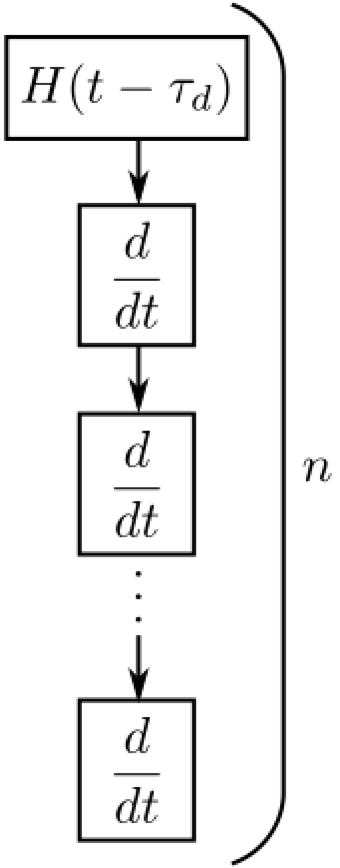




Comparison: Symbolic vs Numerical

- Numerical approach shifts the solution n.h time units
- Maximum magnitude for a discontinuity D:

$$D\binom{n-1}{k-1}/h^k \quad \text{where} \quad k = floor\left(\frac{n}{2}\right)$$



Conclusion: Symbolic approach is more accurate for models that manipulate impulse derivatives.

Roadmap

- Find models that require impulse derivatives.

Coffee Break

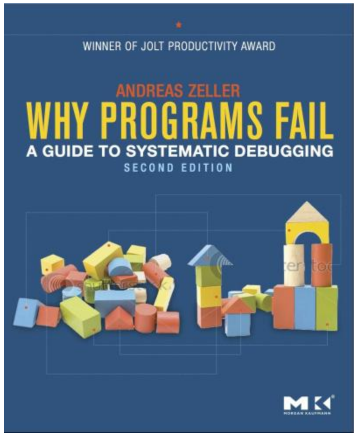
Model Debugging

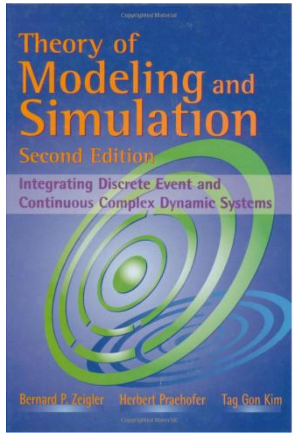
Simon Van Mierlo

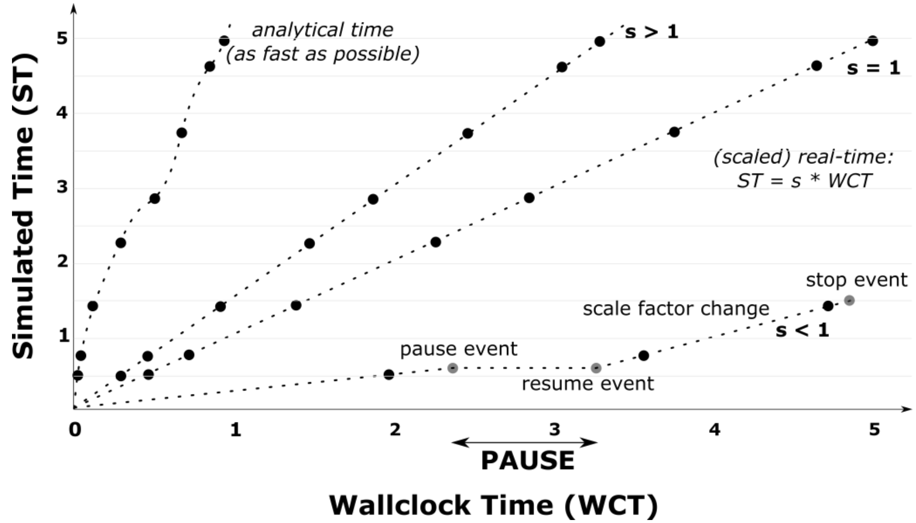
Universiteit Antwerpen

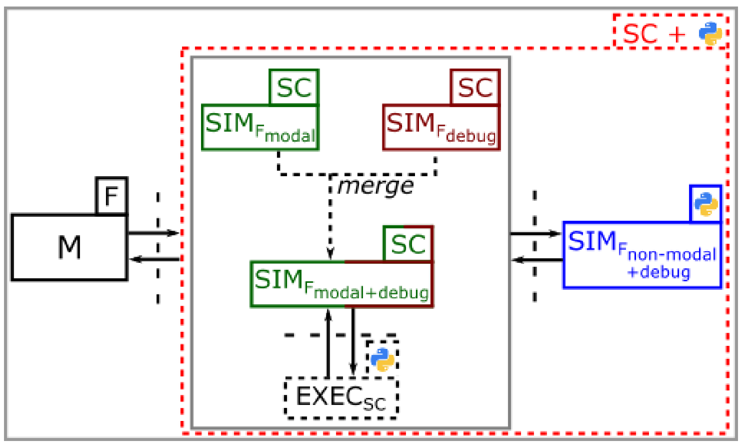
simon.vanmierlo@uantwerpen.be

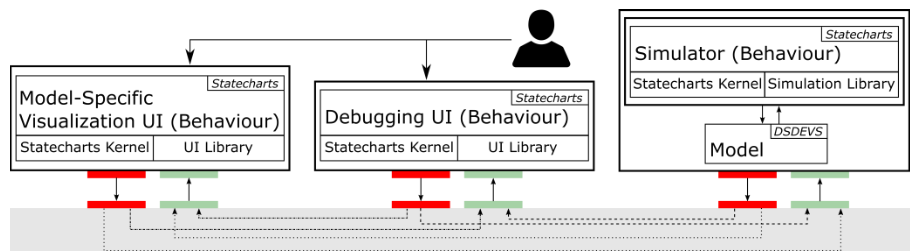
Summary









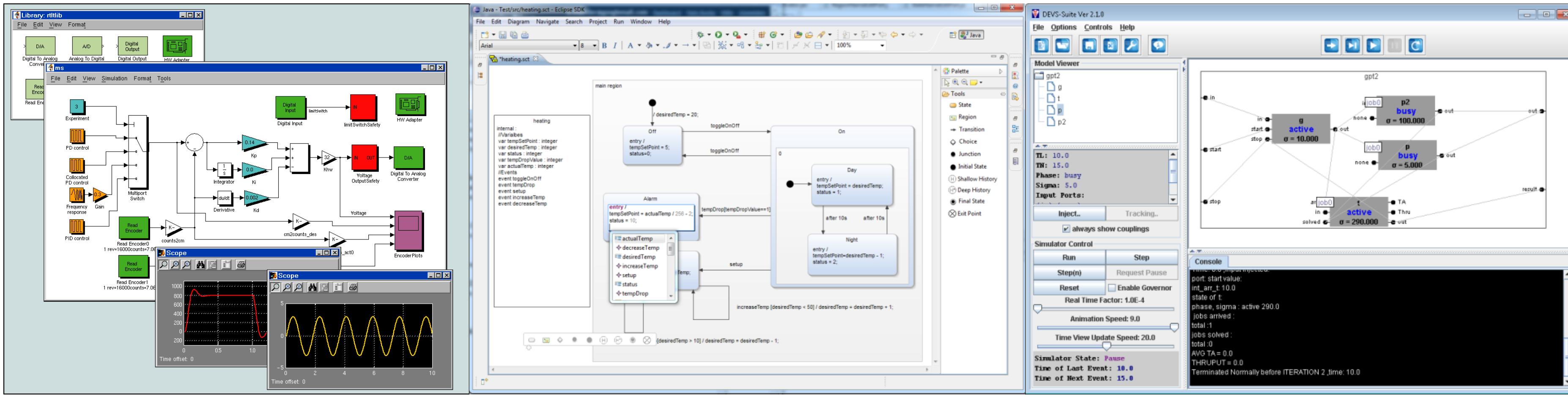


→ Causal-Block Diagrams, Parallel DEVS, Statecharts, Petrinets, Dynamic-Structure DEVS, Hybrid TFSA-CBD, Action Language

Simon Van Mierlo, Yentl Van Tendeloo, and Hans Vangheluwe. **Debugging Parallel DEVS**. *SIMULATION*, 93(4):285-306, 2017

Simon Van Mierlo, Cláudio Gomes, and Hans Vangheluwe. **Explicit Modelling and Synthesis of Debuggers for Hybrid Simulation Languages**. In *Proceedings of the 2017 Symposium on Theory of Modeling and Simulation - DEVS (TMS/DEVS)*, pages 1013-1024, 2017

Motivation



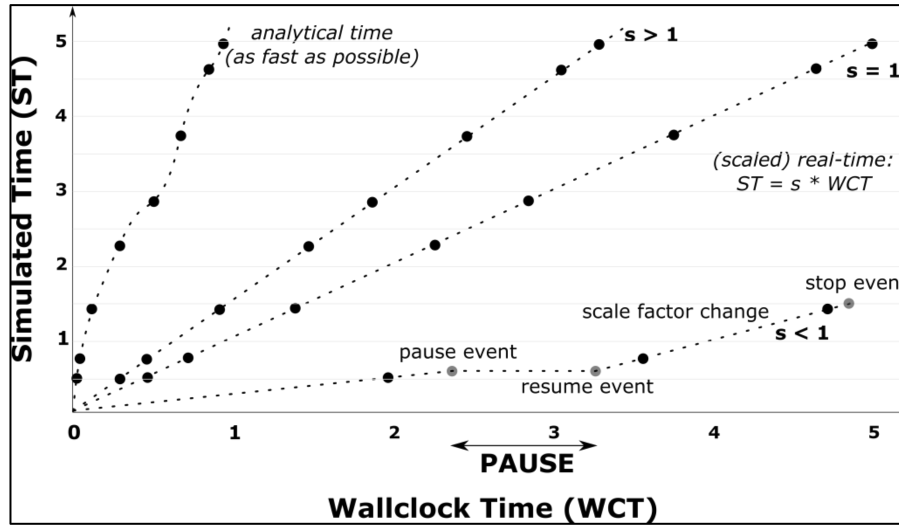
Usable M&S Environments:

- Fidelity (w.r.t. formalism's syntax and semantics)
- Accuracy (in simulation results)
- Resuse (model libraries)
- Performance
- **Debugging**

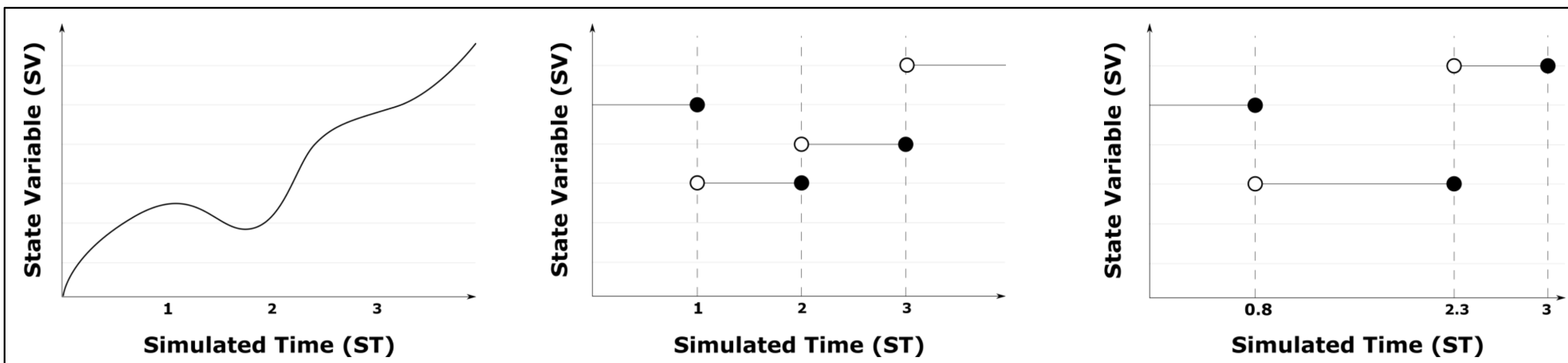
Building Language-Specific Debugging Environments

- Operations:
  - Pause/Resume
  - Stepping
  - Breakpoints
  - State Tracing (Visual)
  - Manual State Changes
  - Omniscient Debugging

+

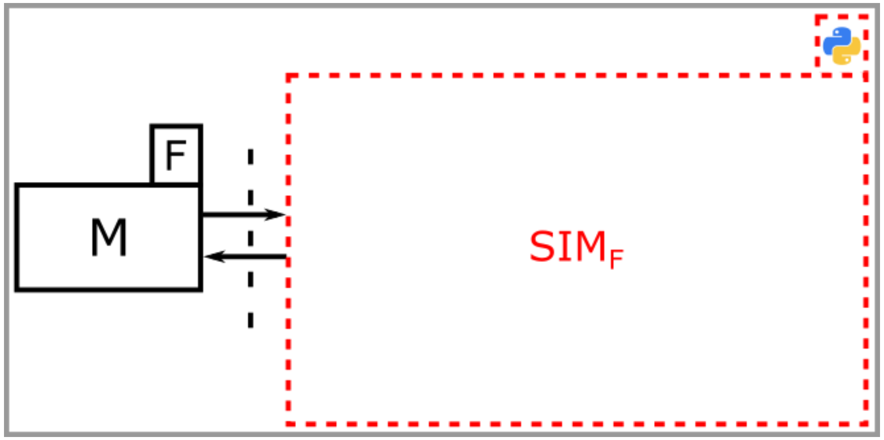


+

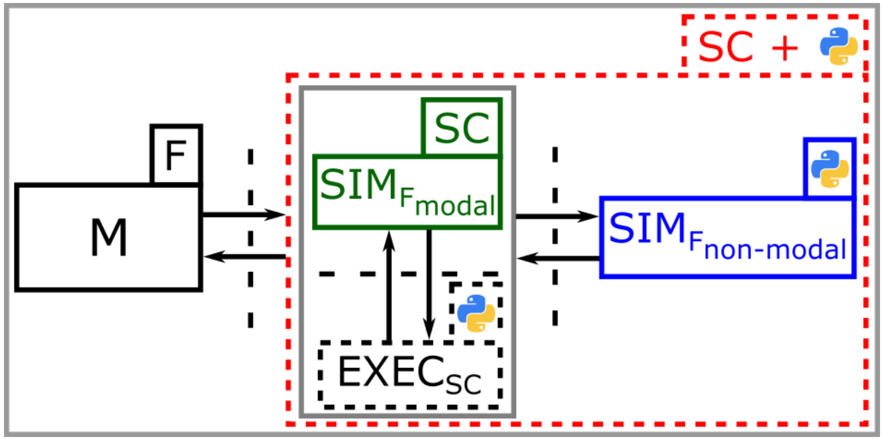


De- and Reconstruction

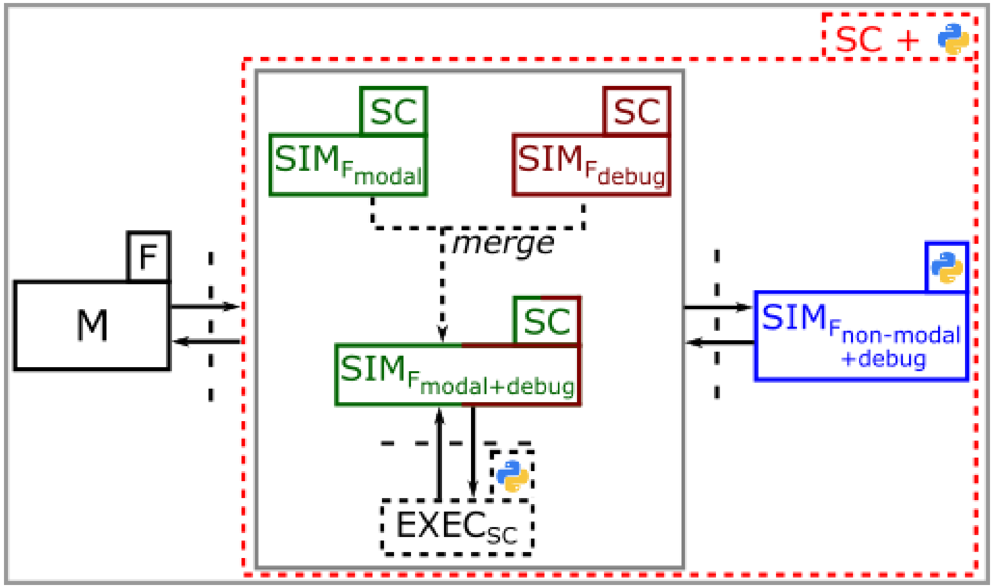
1.



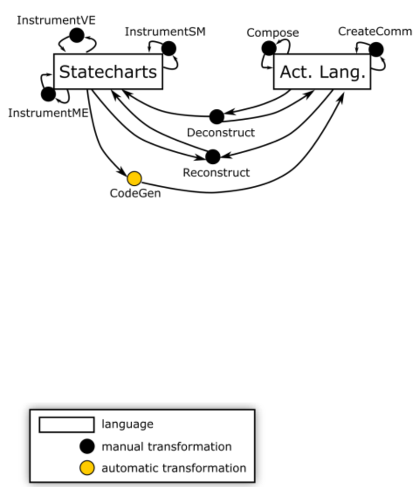
2.

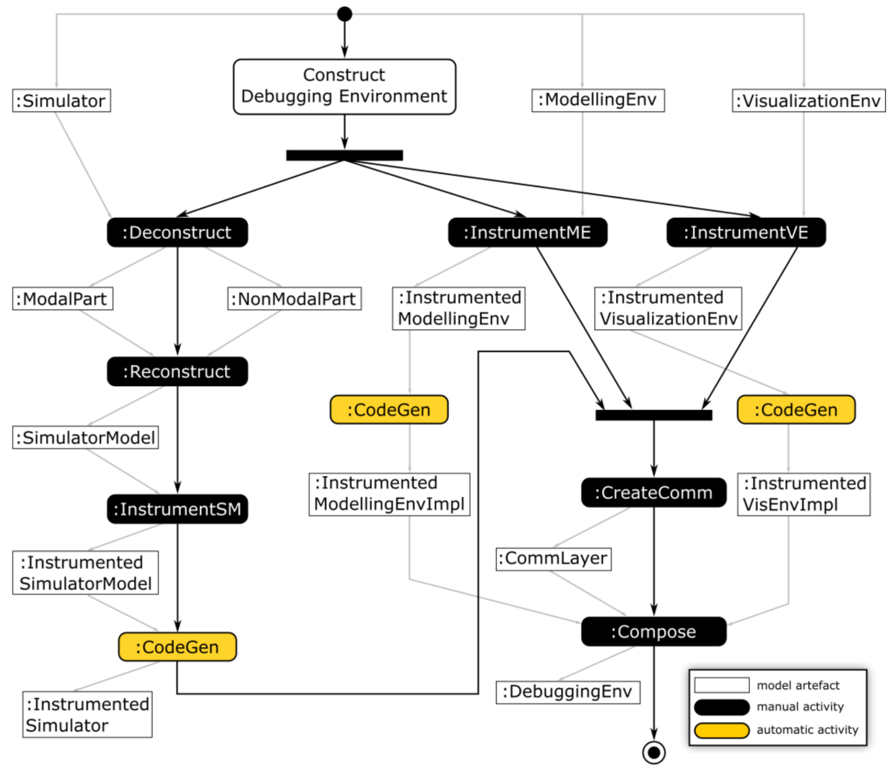


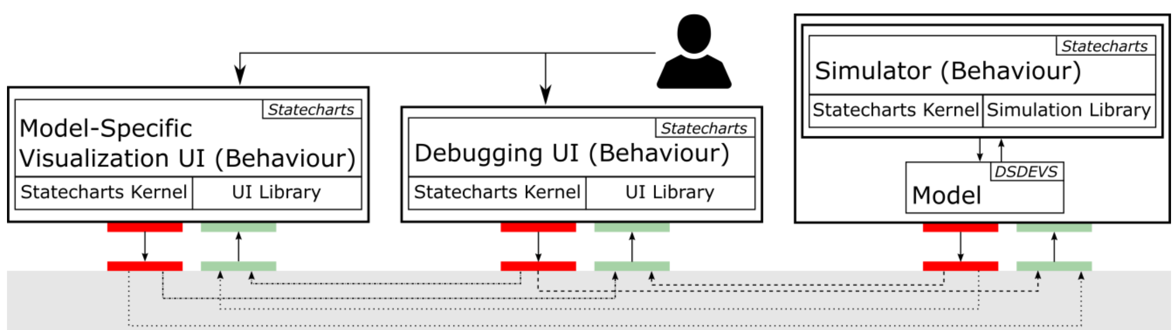
3.



Architecture and Workflow



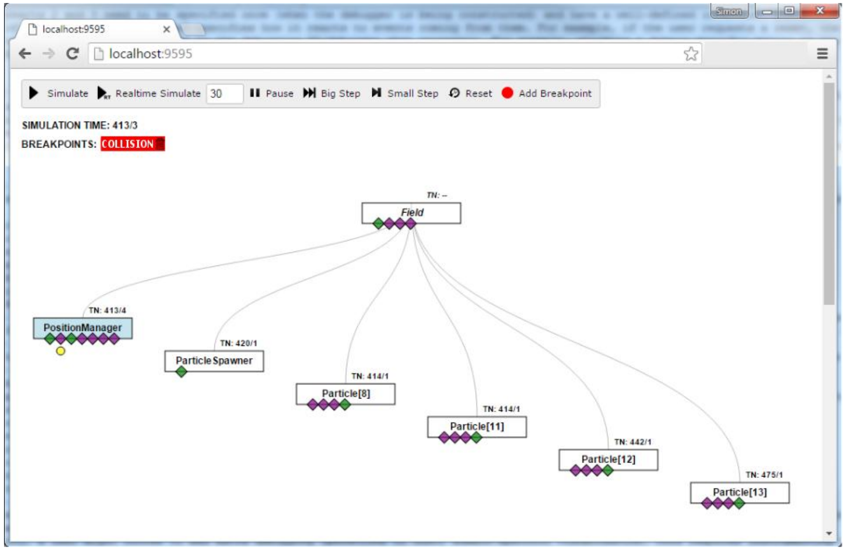




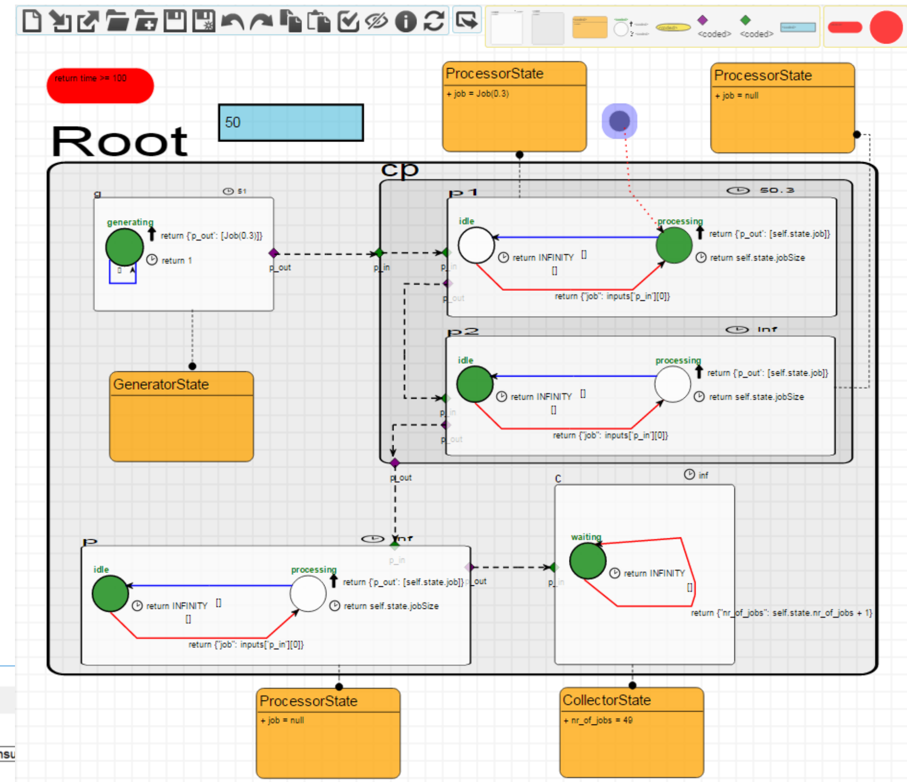
McGill School of Computer Science | NECSIS | FLANDERS MAKE MANUFACTURING INNOVATION NETWORK | Ansymo Antwerp Systems & Software Modelling University of Antwerp

Examples

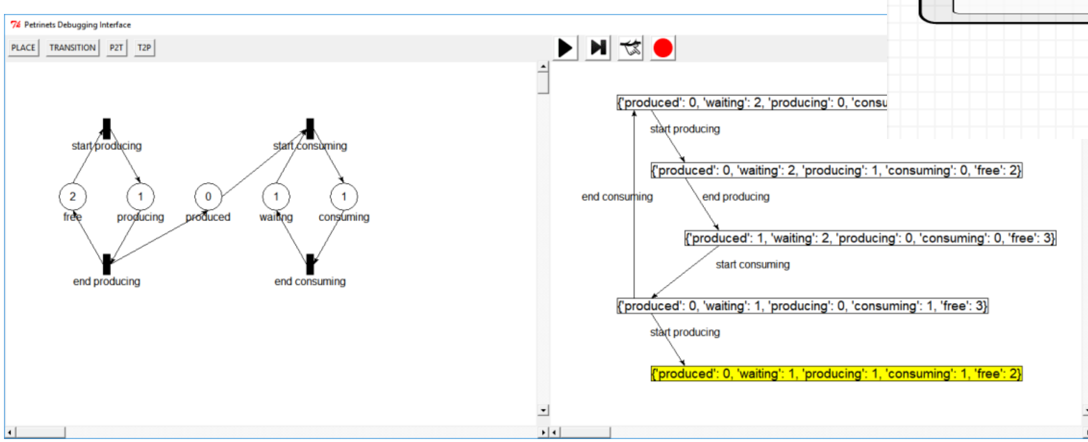
Dynamic-Structure DEVS



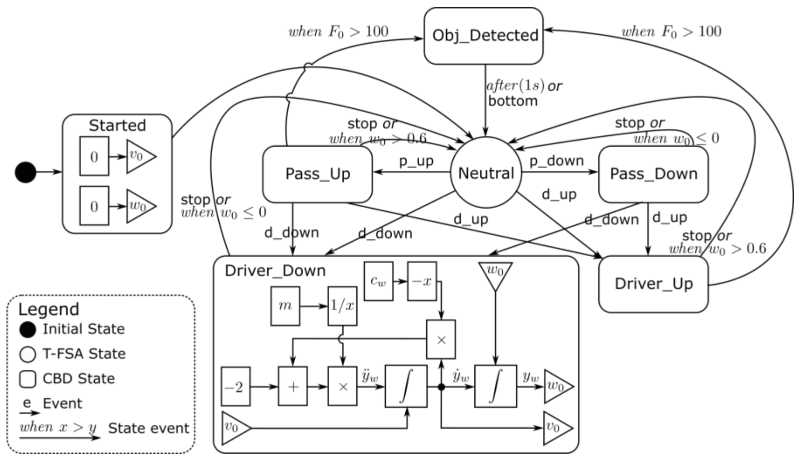
Parallel DEVS

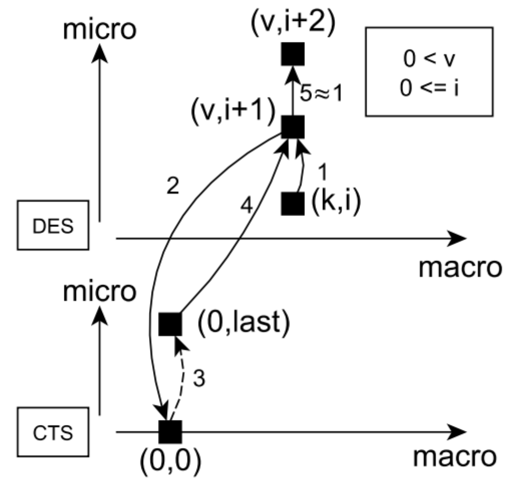


Petrinets



Debugging Hybrid Formalisms





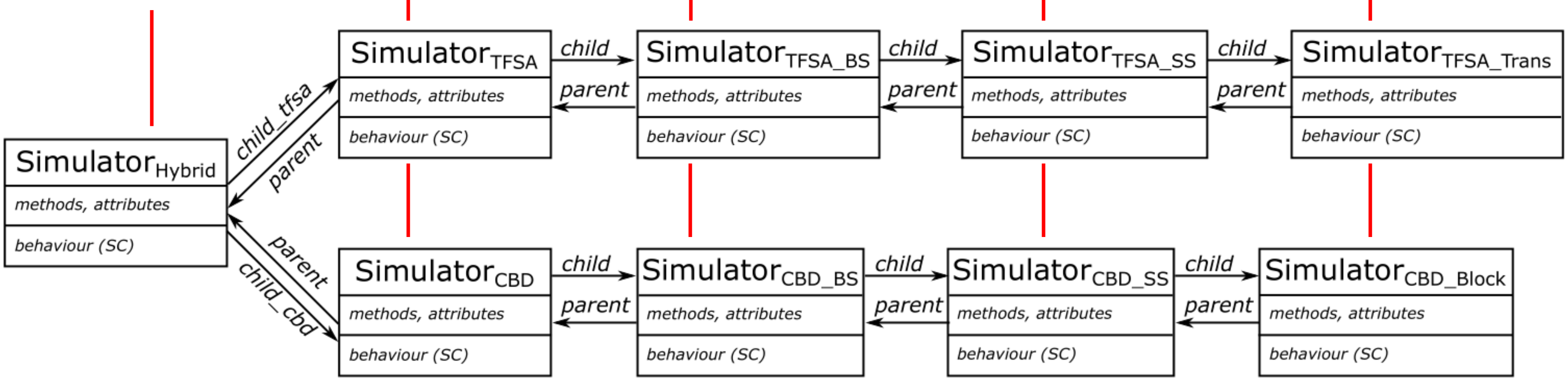
Continuous
Breakpoints
Pause/Resume
Stepping
Realtime Simulation

Continuous
Stepping
Event Injection

Continuous
Stepping

Continuous
Stepping

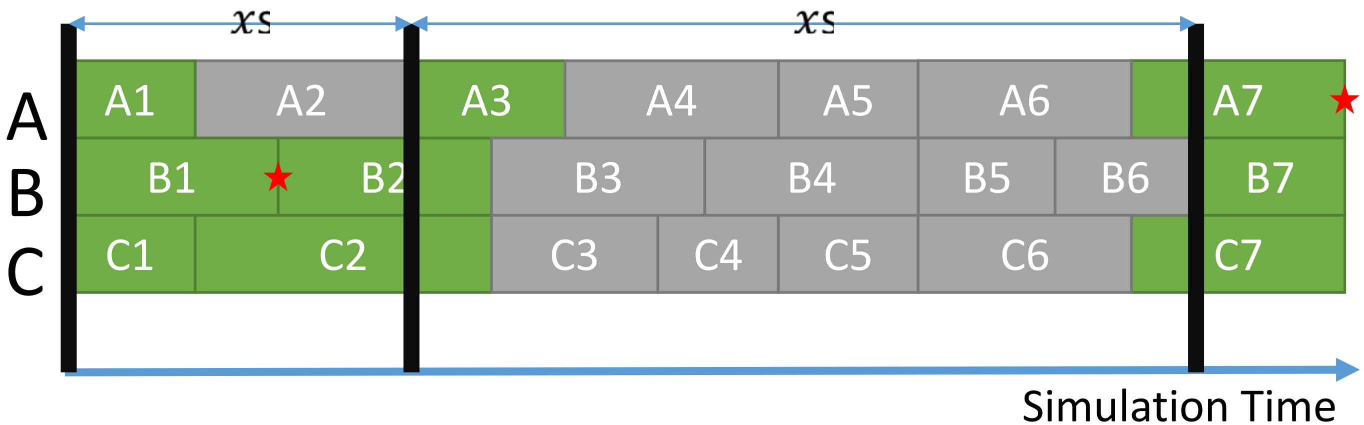
Continuous



McGill School of Computer Science
necsis

FLANDERS MAKE
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Efficient Omniscient Debugging (PDEVS)



Periodic State Saving

Optimizations:

- *xs* -> *2xs*
- Disk I/O
- Compression

Roadmap

- Denotational (vs. Operational) Semantics
- Language Engineering
  - "weaving" debugging language
- Simulators
  - black- or grey-box (see FMI)
  - hybrid: canonical form (moving away from SCCD)
- Architecture
  - automatic artefact generation/instrumentation
- Advanced Breakpoint Conditions
  - (see ProMoBox)

Fully Verifying Graphical Contracts on Model Transformations

Bentley James Oakes

McGill University

bentley.oakes@mail.mcgill.ca

July 26, 2017

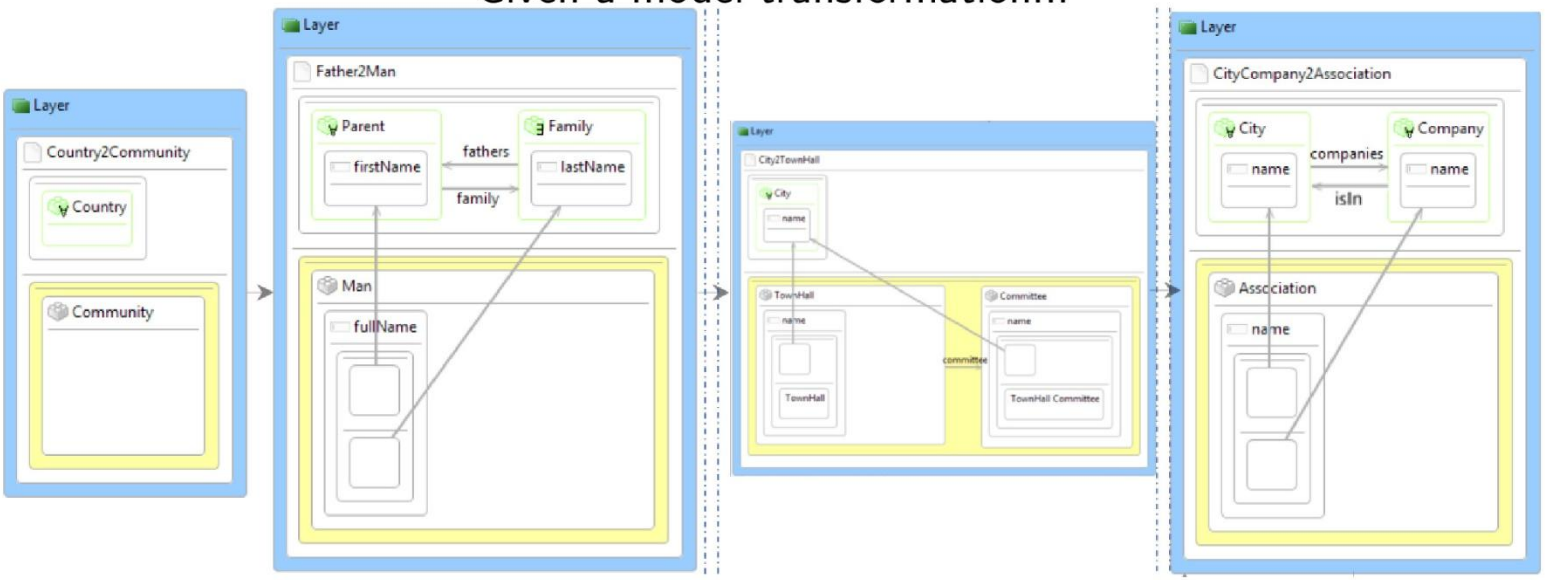
McGill School of Computer Science

Ansymo
Antwerp Systems & Software Modelling
University of Antwerp

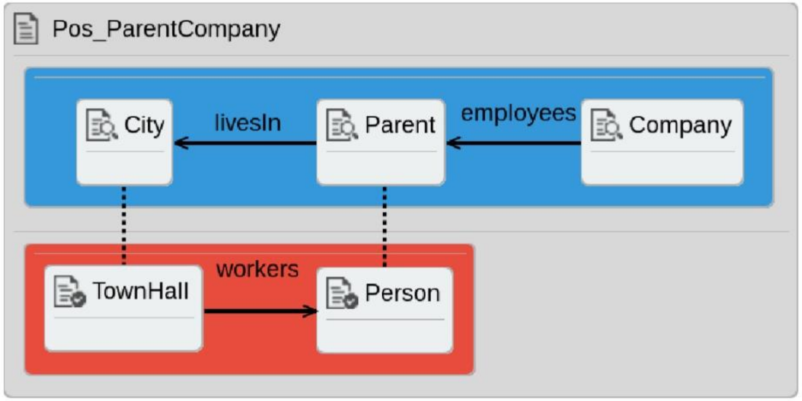
Problem Statement

Model transformations are at the heart and soul of model-based engineering[1]

Given a model transformation...



Does the following *structural contract* hold on all input/output model pairs?



[1] S. Sendall and W. Kozaczynski. Model Transformation: The Heart and Soul of Model-driven Software Development. IEEE Software, 20(5):4245, Sep 2003.

Our Approach

Step 1 - Generate Path Conditions

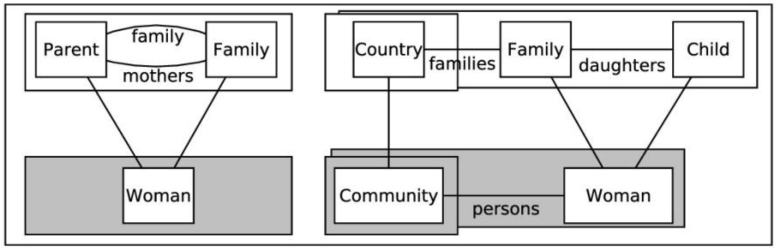
- We build representations of rule interactions - *path conditions*
  - Represent elements present in input and output models
- Through *abstraction relation*, represent all possible transformation executions

Step 2 - Contract Proving by Matching

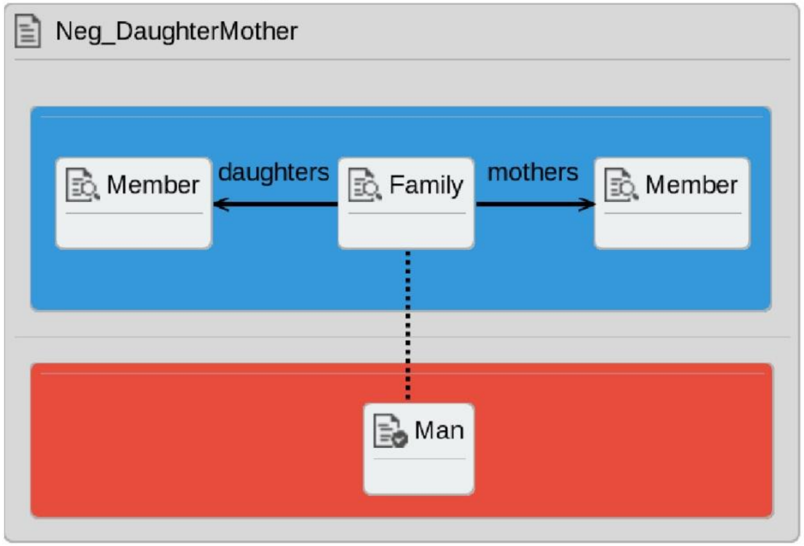
- Contract statement: "If pre-condition matches on input model, then post-condition must match on output model"
- If this does not hold, the path condition is a counter-example

L. Lucio, B. Barroca, V. Amaral. "A Technique for the Verification of Model Transformations" Proceedings of MODELS, 2010.

Path condition representing execution of Daughter2Woman, Mother2Woman, Country2Community rules:



A contract that will not hold on the above path condition:



Interpretation: *Families with Daughters and Mothers will produce a Man element*

Applications

Verification that proprietary General Motors model for Vehicle Control Software is properly translated to industry-standard AUTOSAR:

Pattern Contracts: *(Properties that relate source and target metamodel elements)*

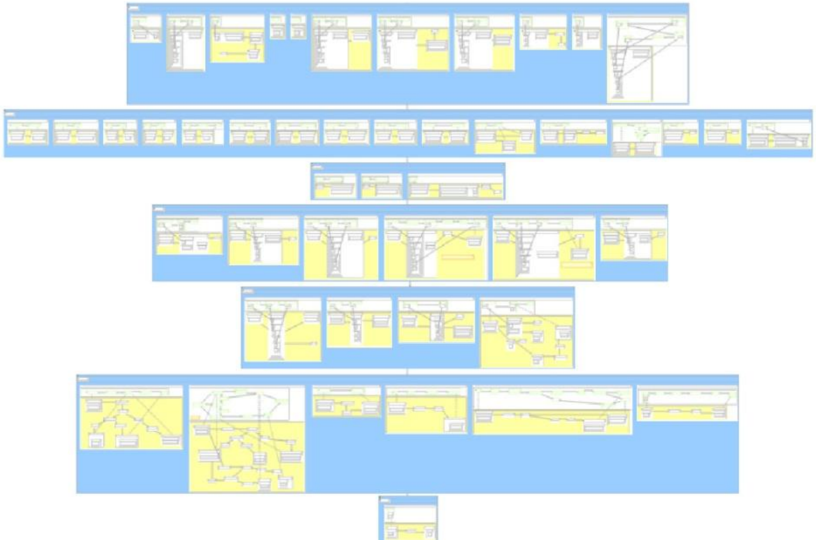
- ($P1$) If a *PhysicalNode* is connected to a *Service* through the *provided* association (in the input), then the corresponding *CompositionType* will be connected to a *PPortPrototype* (in the output).
- ($P2$) If a *PhysicalNode* is connected to a *Service* through the *required* association (in the input), then the corresponding *CompositionType* will be connected to a *RPortPrototype* (in the output).

G. Selim, L. Lúcio, J. Cordy, J. Dingel, B. Oakes. "Specification and Verification of Graph-Based Model Transformation Properties". ICGT 2014.

Verification of translation from UML-RT state machine diagrams into Kiltera (language for timed, event-driven, mobile and distributed simulation):

G. Selim, L. Lúcio, J. Cordy, J. Dingel, B. Oakes. "Specification and Verification of Graph-Based Model Transformation Properties". ICGT 2014.
G. Selim, J. Cordy, J. Dingel, L. Lúcio, B. Oakes. "Finding and Fixing Bugs in Model Transformations with Formal Verification: An Experience Report" MODELS 2015.

- Verification of mbeddr
  - Designed to aid the development of embedded C software by providing a higher-level language

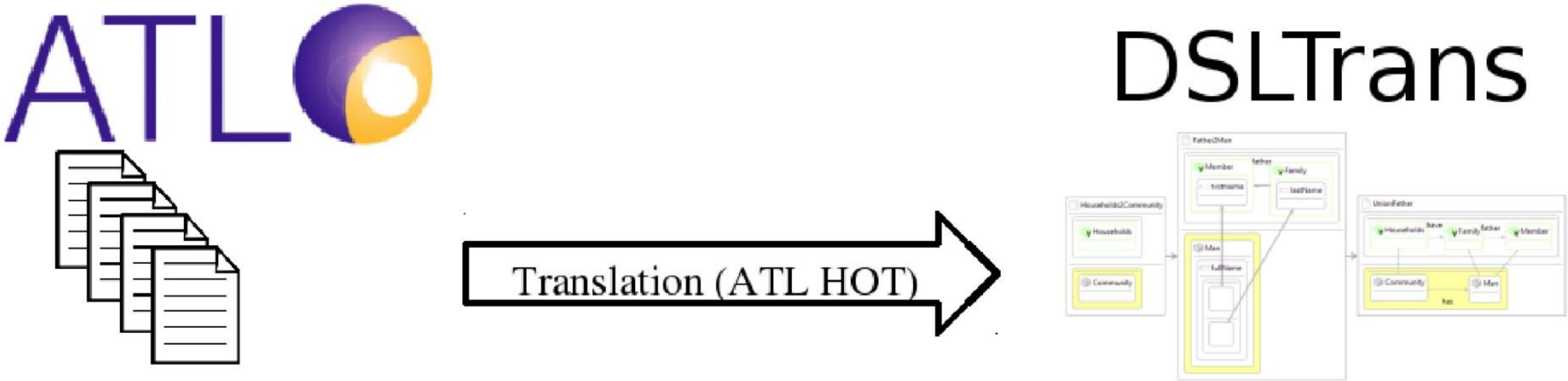


Building Transformations and Contracts

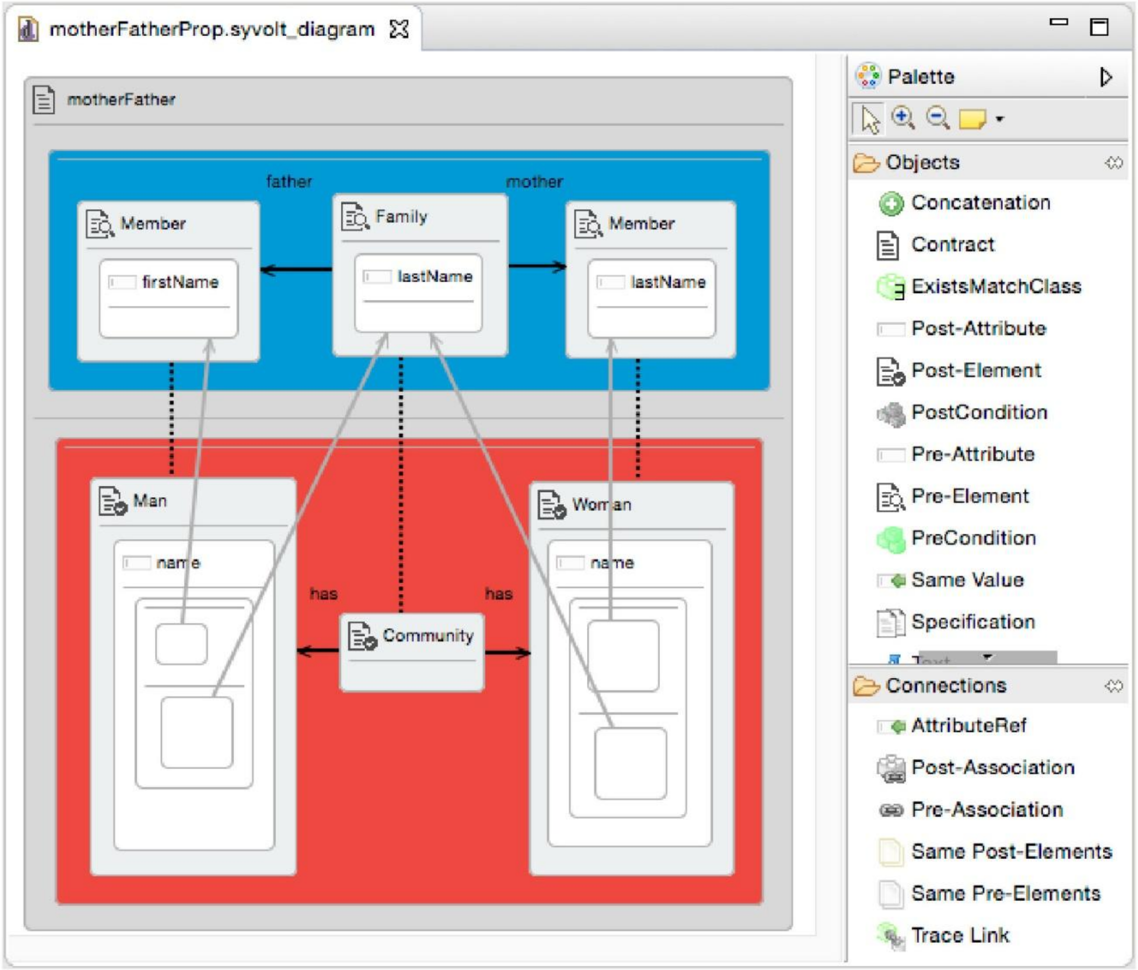
Currently there are three approaches to build transformation and contracts:

1. Translating declarative ATL transformations into DSLTrans
2. Eclipse plug-in
3. Meta-Programming System plug-in

ATL Translation



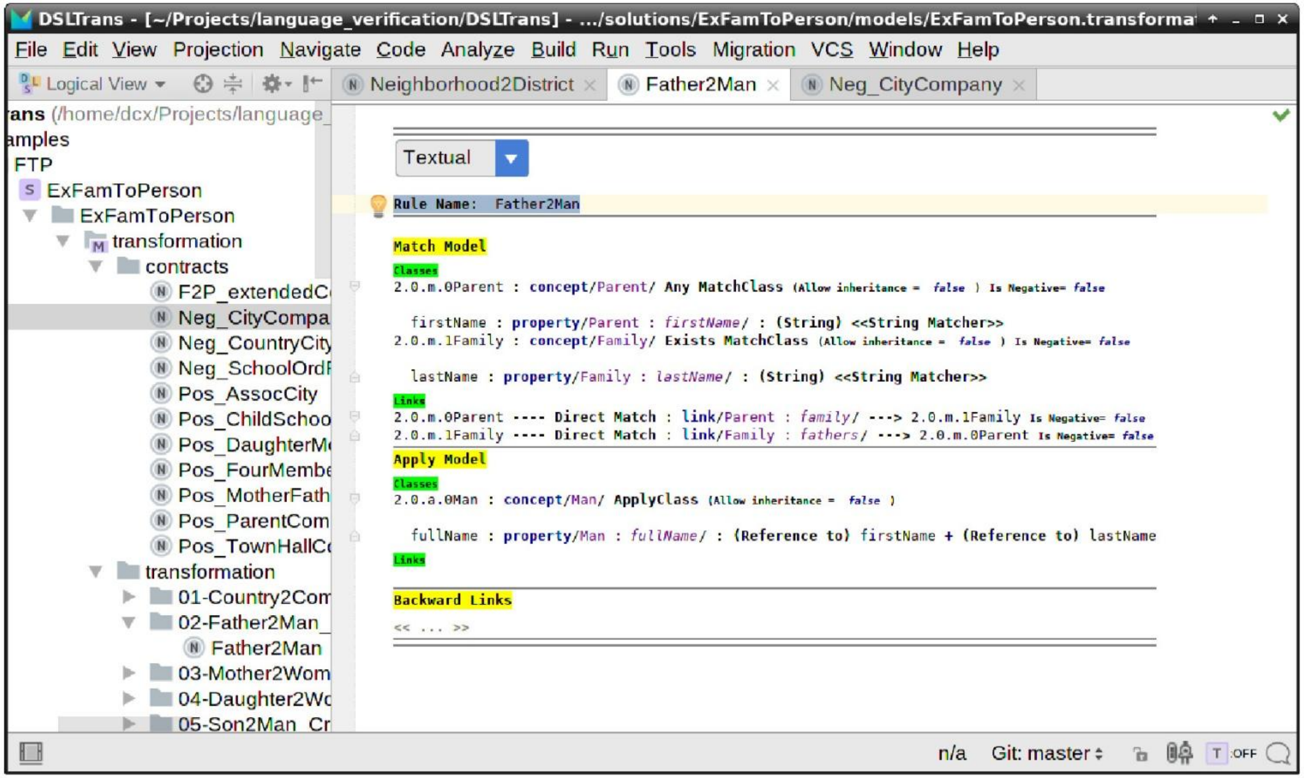
- *Atlas Transformation Language* is heavily used in industry and academia
- We translate ATL transformations using a higher-order transformation into our language DSLTrans for contract proving
- Approach covers all declarative ATL model transformations
- B. Oakes, J. Troya (Universidad de Sevilla), L. Lúcio, M. Wimmer (TU Wien). “Fully Verifying Transformation Contracts for Declarative ATL” MODELS 2015.
- Expanded to journal article: “Full Contract Verification for ATL using Symbolic Execution” SoSyM 2016.



Eclipse plug-in to build transformation and perform contract verification

- L. Lúcio, B. Oakes, C. Gomes, G. Selim, J. Dingel, J. Cordy, H.Vangheluwe. "SyVOLT: Full Model Transformation Verification Using Contracts" MODELS 2015.
- Collaboration with University of Antwerp and Queen's University

MPS Tool Integration



Integration into the Meta-Programming System from Jetbrains

- MPS is designed to easily create *domain-specific languages*
- Projectional editor for creation of the DSLTrans transformation and contracts
  - Provides auto-complete and syntax checking
- Can execute DSLTrans transformations on a model, or verify the transformations using contracts

Roadmap

Current Work

- Formalizing all DSLTrans constructs of DSLTrans [1]
  - Includes indirect links and negative elements
- A detailed and more formal approach to the contract-proving technique [2]
  - Includes negative elements and the representation of multiple rule application

[1] B. Oakes, L. Lúcio, C. Gomes, H. Vangheluwe.
"Complete Semantics for the DSLTrans Transformation Language".
[2] B. Oakes, L. Lúcio, C. Gomes, H. Vangheluwe.
"Expressive Symbolic-Execution Contract Proving for the DSLTrans Transformation Language".

Future Work

Direction 1: Application of contract-proving

Overview: Apply proving to different transformation languages and (industrial) case studies

- Technique applicable to most transformation languages with restrictions?
- Further work on mbeddr case studies and investigate more ATL transformations

Direction 2: Integration of contract-proving approach

Overview: Add contract-proving ability to model transformation tools such as AtomPM/ModelVerse

Components can be extracted from existing SyVOLT prover

- Creation of matchers operating on transformation rules
- Graph non-isomorphism matching
- Slicing/pruning optimizations

Frames

--= Enabling Reuse in MBSE =--

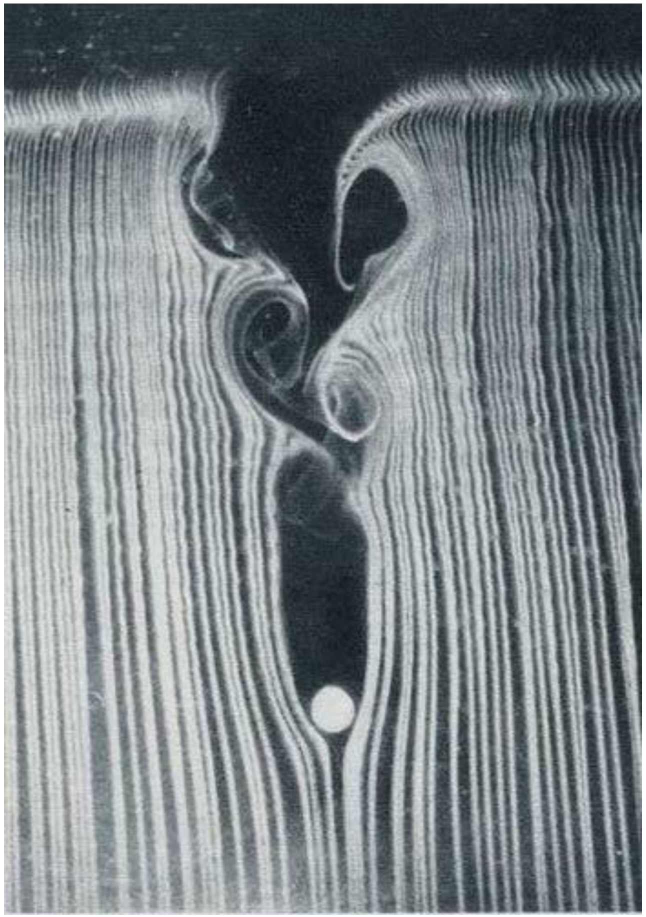
Joachim Denil

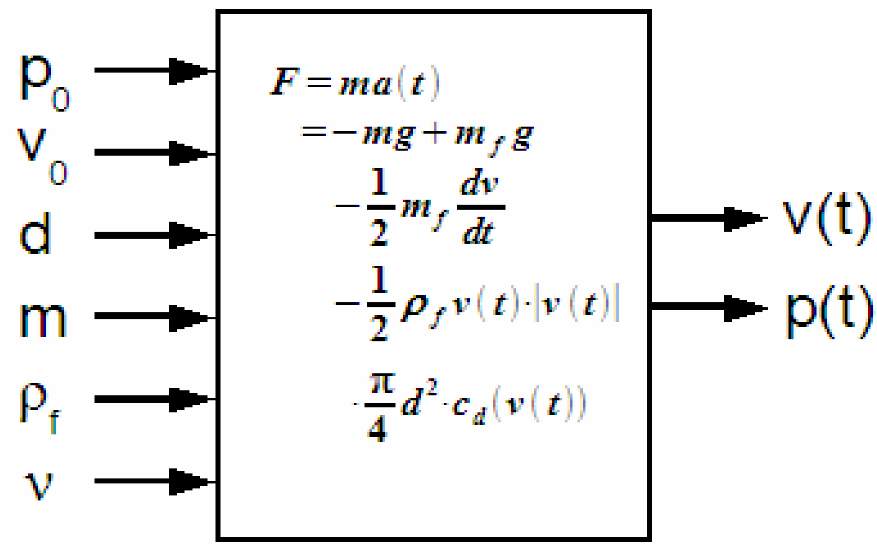
Joachim.Denil@uantwerpen.be

Summary

- Reuse of (Simulation) Models is important
- Modelers make assumptions during all stages of a M&S process:
  - Model Construction
  - Model Calibration
  - Etc.
- Frames record information to allow such reuse!
- Information needs to be stored for meaningful reuse
  - Selection from catalogue of models
- Information needs to be modelled for automation
  - Test that Frame works on Model
  - Test that Models works with Frame

Reuse of Models Example





1. Invariant Constraints

1.a Sphere Attributes

1. Sphere Property - The body is a sphere and it remains spherical.
2. Smooth Property - The body is smooth and it remains smooth.
3. Impermeable Property - The body is completely impermeable.
4. Initial Velocity - The body has an initial velocity of v_0 that has no horizontal component of motion.
5. Angular Velocity - The body has no initial angular velocity.
6. Constant Mass - The mass of the body remains constant over time. The body does not experience ablation or accretion.
7. Constant Diameter - The diameter of the body remains constant over time.
8. Distribution of Mass - The body has a centrally symmetric mass distribution that remains constant over time.
9. Uncertainty Principle - The diameter of the body is much greater than the Plank length.
10. Brownian Motion - The mass and diameter of the body are large enough such that Brownian motion of the fluid has negligible impact on the body.
11. General Relativity - The mass of the body is low enough to ignore the gravitational curvature of space-time.

1.b Fluid Attributes

12. Fluid Density - The fluid density is constant. The fluid is incompressible.
13. Fluid Pressure - The fluid pressure is constant.
14. Fluid Temperature - The fluid temperature is constant.
15. Kinematic Viscosity - The kinematic viscosity is constant. The medium is a Newtonian fluid.
16. Stationary Fluid - The fluid is stationary apart from being disturbed by the falling body.
17. Infinite Fluid - The volume of the fluid is large enough to completely envelope the sphere. The movement of the fluid is not restricted by a container such as a pipe or tube.

1.c Earth Attributes

18. Flat Terrain - The ground does not have terrain and remains flat for all $t > 0$.
19. Coriolis Effect - The Earth is not rotating. We ignore the Coriolis effect.

2. Dynamic Constraints

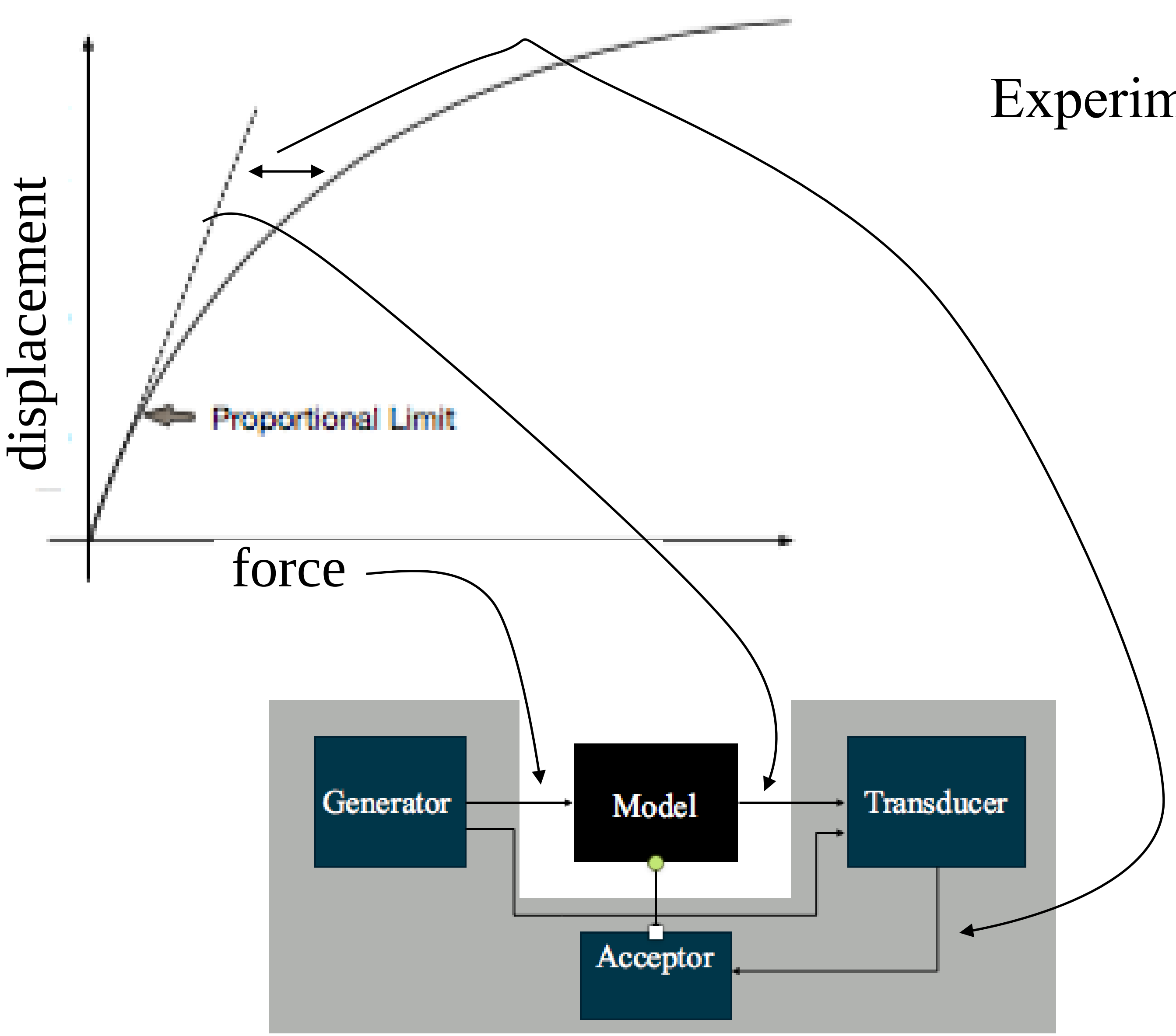
20. Mach Speed - The velocity of the body is sufficiently less than the speed of sound for that medium.
21. Special Relativity - The velocity of the body is sufficiently less than the speed of light for that medium.
22. Reynolds Number - The Reynolds number remains between 10^{-2} and 10^{7} for all $t > 0$. The Reynolds number is a function of velocity.

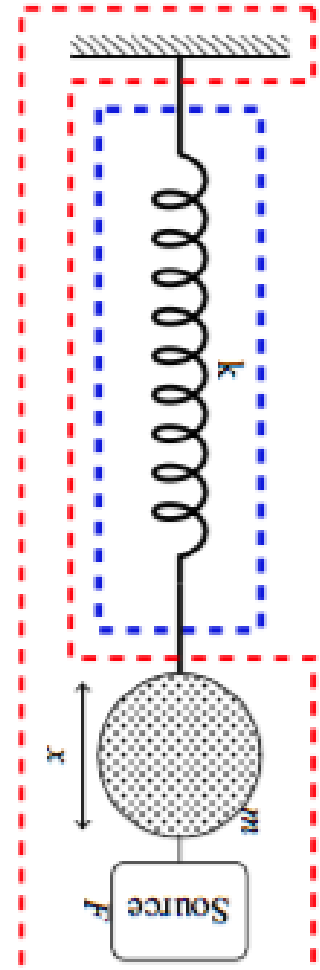
3. Inter-Object Constraints

23. Sphere/Fluid Interaction - The body and the fluid interact only through buoyancy and drag. For example, the body cannot dissolve in the fluid, nor can the body transfer heat to the fluid.
24. Sphere/Earth Interaction - The body and the eart interact only through the gravitational force.
25. Fluid/Earth Interaction - The fluid and the eart do not interact.
26. Closed System - The Earth, sphere, and fluid d not interact with any other objects.
27. Simple Gravity - Gravity is a constant downwar force of 9.8 m/s^2.
28. One-Sided Gravity - The mass of the body i much less than the mass of the Earth. The Eart is not affected by the gravitational pull of th body.
29. Inelastic Collision - The collision between th sphere and the ground is perfectly inelastic.

Spiegel, Michael, Paul F. Reynolds Jr, and David C. Brogan. "A case study of model context for simulation composability and reusability." Proceedings of the 37th conference on Winter simulation. Winter Simulation Conference, 2005.

1st attempt: Zeigler's Experimental Frame

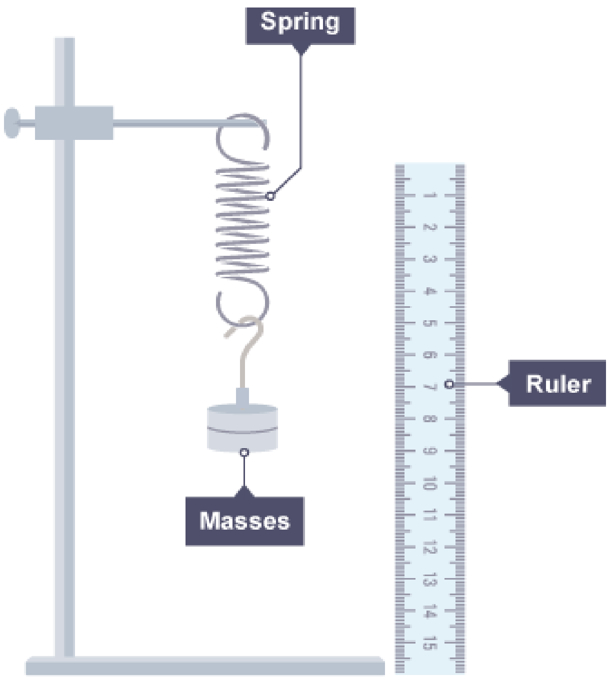


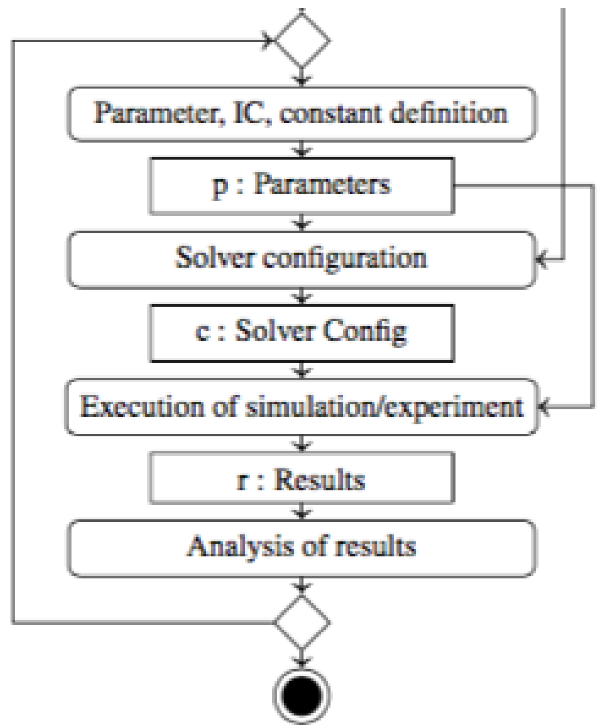


Give Procedure that can be enacted and automated and allows for reuse!

Experimental spring results, with mass m in kg and displacement x (± 0.0001) in cm

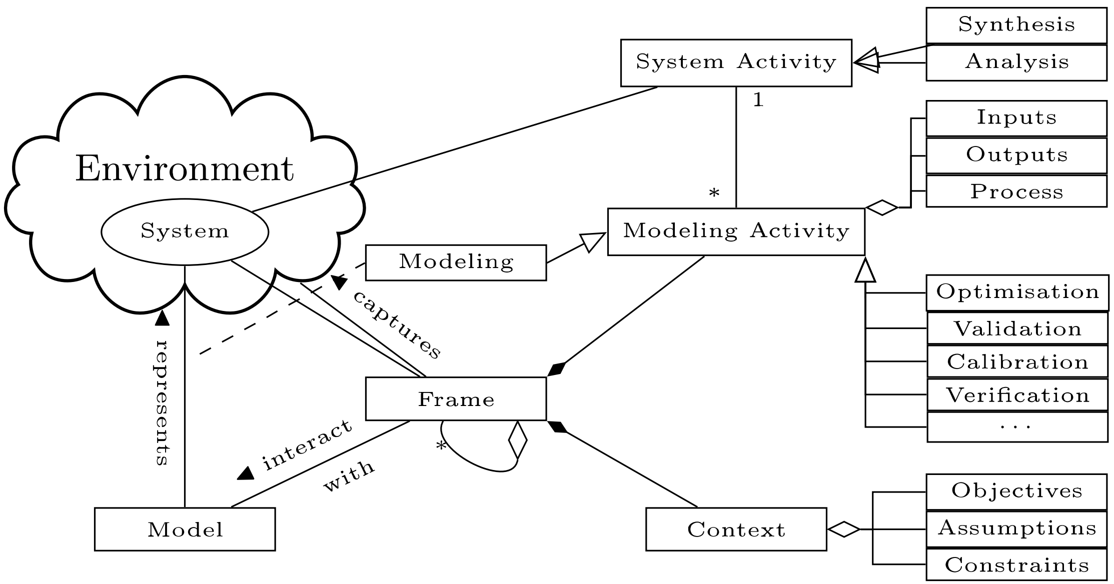
m	x	m	x	m	x	m	x	m	x
1	2.100	3	6.3749	5	10.4915	7	14.6081	9	19.0012
2	4.3166	4	8.4332	6	12.5489	8	16.7774		

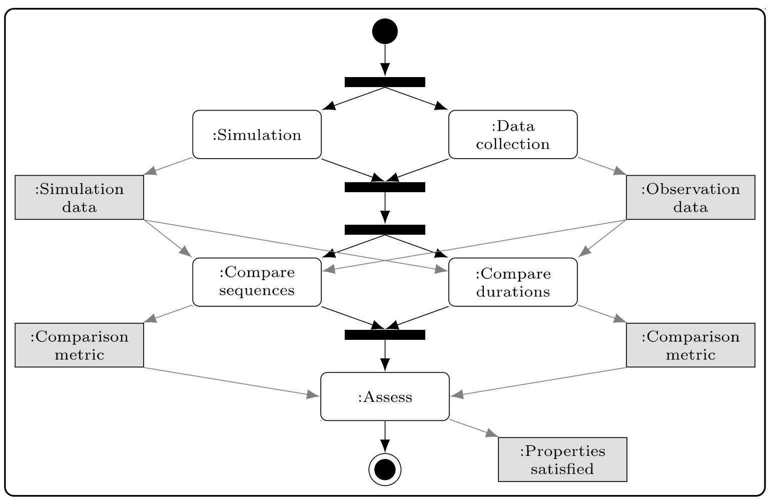


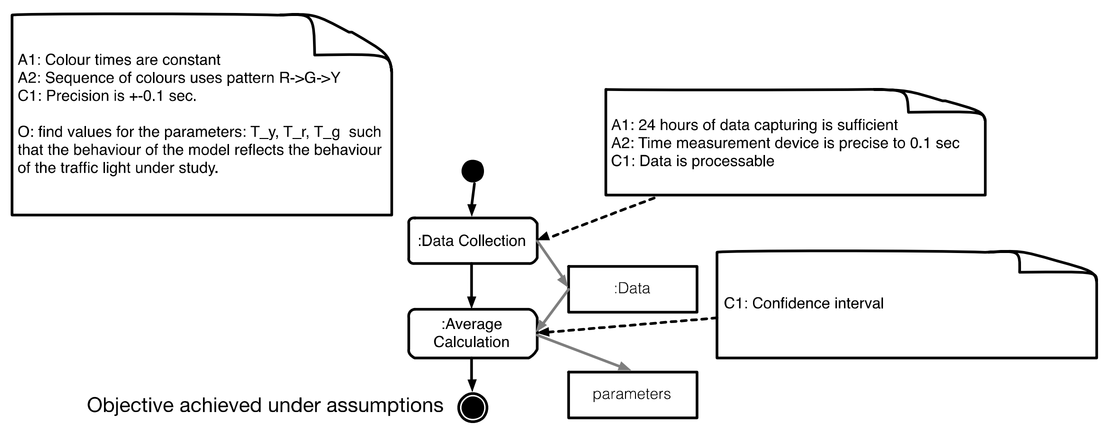


$r^2 = 0.9998413$ | $k = 4.759093$ | $\Omega_{in} = [1, 9]$ (force) | $\Omega_{out} = [2.101241, 18.91116]$

Not only for Model but for all parts of M&S cycle!







Future Directions

- Work with Alex, Rick and Stefan @ MPM4CPS Cost?
- Continue on the different M&S life-cycle elements
- Extended Case Study
  - Power window?
- Appropriate languages for modelling frames
- Extend formalization and frame relations
- Libraries with Frames (tool-building)

Lunch

Engineering Process Transformation

to Manage (In)consistency

Istvan David

MSDL Antwerp

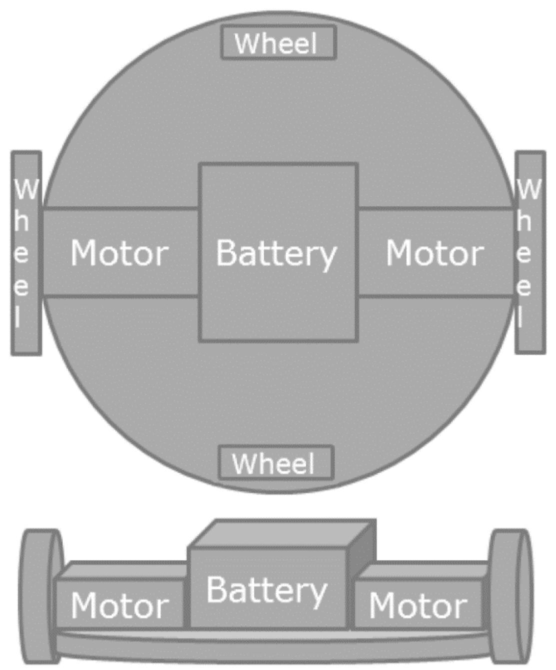
istvan.david@uantwerp.be

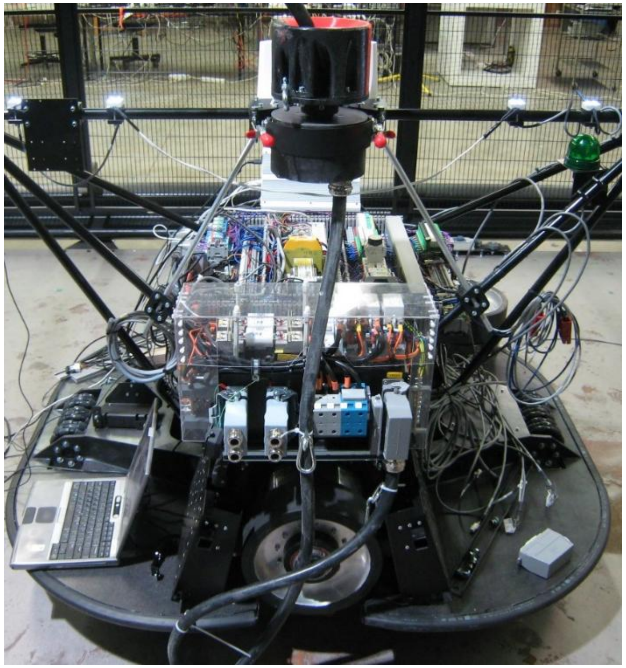
Summary

- Inconsistency management in engineering processes
- Inconsistencies → $$$
  - Late (or no) detection, numerous re-iterations…
- We provide:
  - A methodology, and
  - A tool for managing inconsistencies.

I. Dávid, J. Denil, K. Gadeyne, and H. Vangheluwe, "Engineering Process Transformation to Manage (In)consistency,"
in Proceedings of the 1st International Workshop on Collaborative Modelling in MDE (COMMitMDE 2016), pp. 7–16, http://ceur-ws.org/Vol-1717/, 2016.

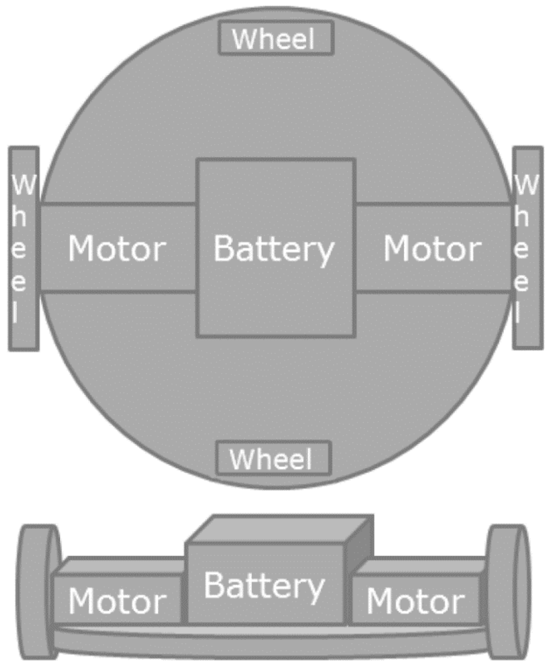
Engineering complex systems is hard!

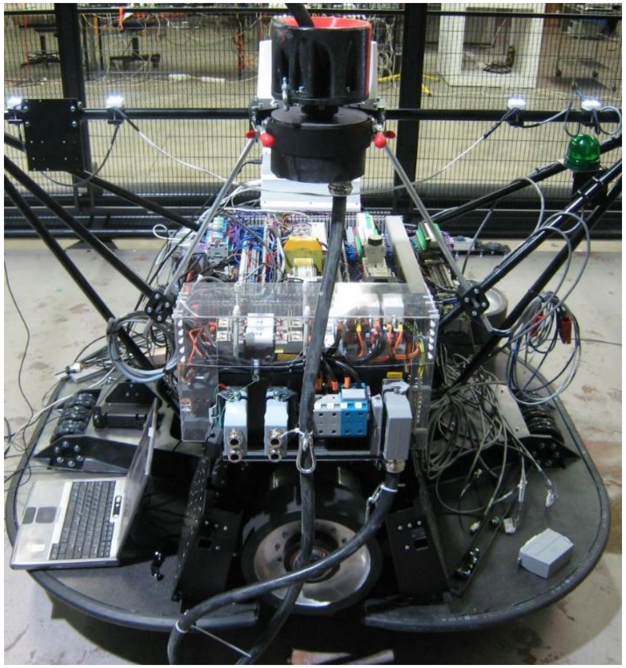




Engineering complex systems is hard!

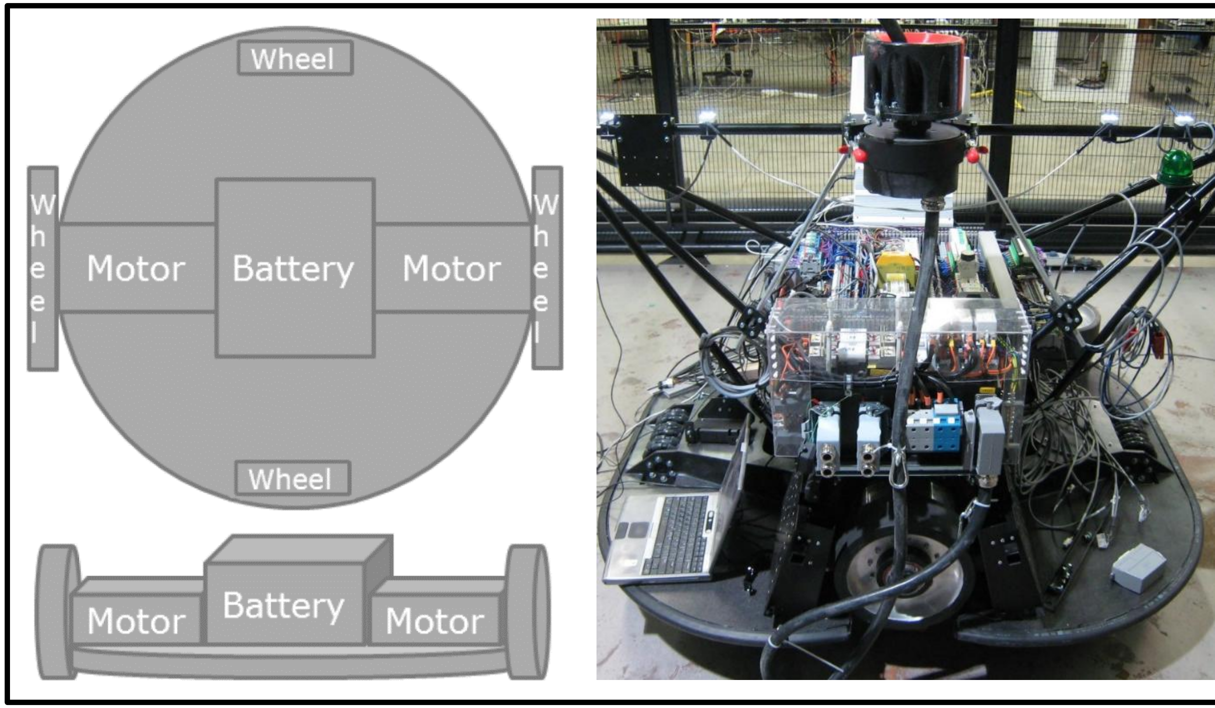
- Modeling
- Increased complexity
- Disparate domains
- Inconsistencie
s





Engineering complex systems is hard!

- Modeling
- Increased complexity
- Disparate domains
- Inconsistencies



CORRECTNESS **EFFICIENCY**

McGill School of Computer Science
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Managing inconsistencies

- *Rather than thinking about removing inconsistency we need to think about "managing consistency"* – Finkelstein
  - Tolerate, analyze, prevent…

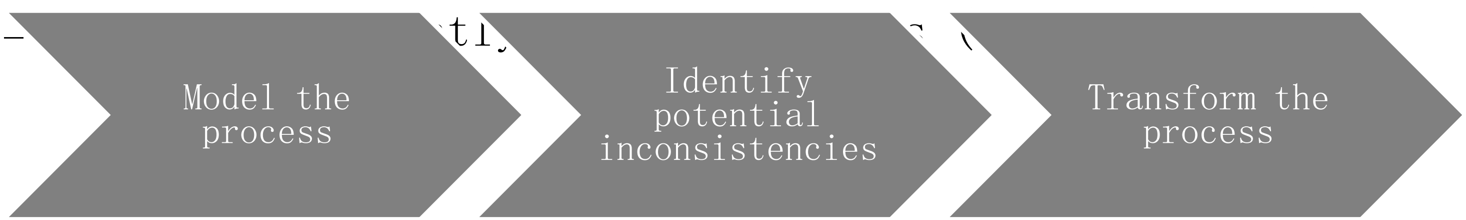
Managing inconsistencies

- *Rather than thinking about removing inconsistency we need to think about "managing consistency"* – Finkelstein
  - Tolerate, analyze, prevent…
- Processes!
  - Understand the lifecycle of models
  - …and their relation with (semantic) properties
  - ...and consequently: inconsistencies (origin, impact)

McGill School of Computer Science
Flanders Make Manufacturing Innovation Network
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Managing inconsistencies

- *Rather than thinking about removing inconsistency we need to think about "managing consistency"* – Finkelstein
  - Tolerate, analyze, prevent…
- Processes!
  - Understand the lifecycle of models
  - …and their relation with (semantic) properties

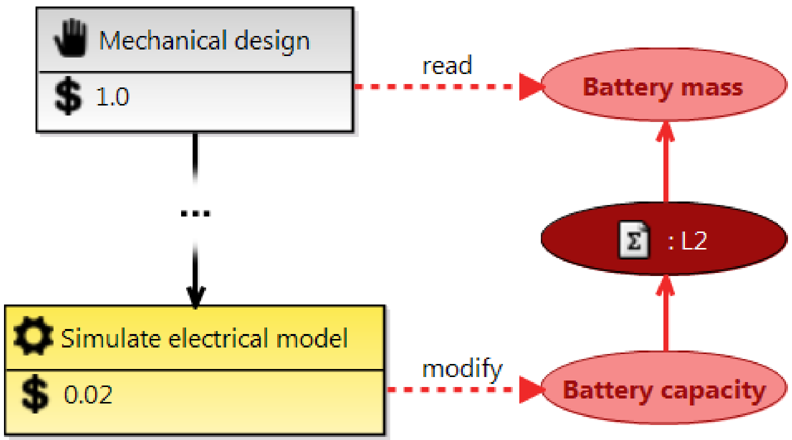


Managing inconsistencies

- *Rather than thinking about removing inconsistency we need to think about "managing consistency"* – Finkelstein
  - Tolerate, analyze, prevent…
- Processes!
  - Understand the lifecycle of models
  - …and their relation with (semantic) properties

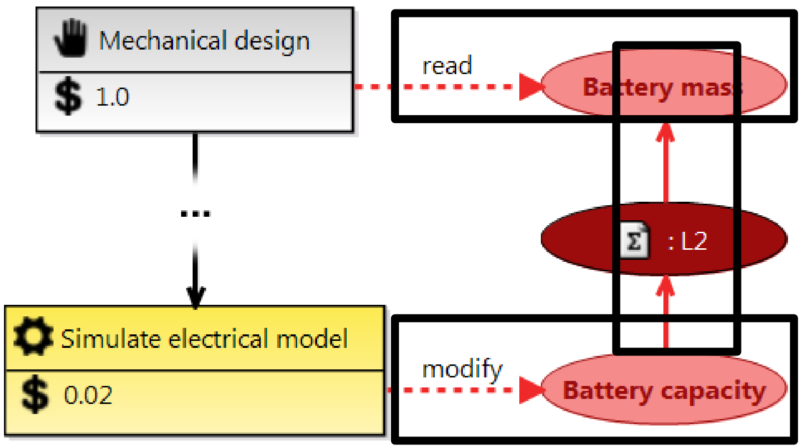
Process modeling and transformation

- Appropriate process modeling formalism?
  - Extended FTG+PM



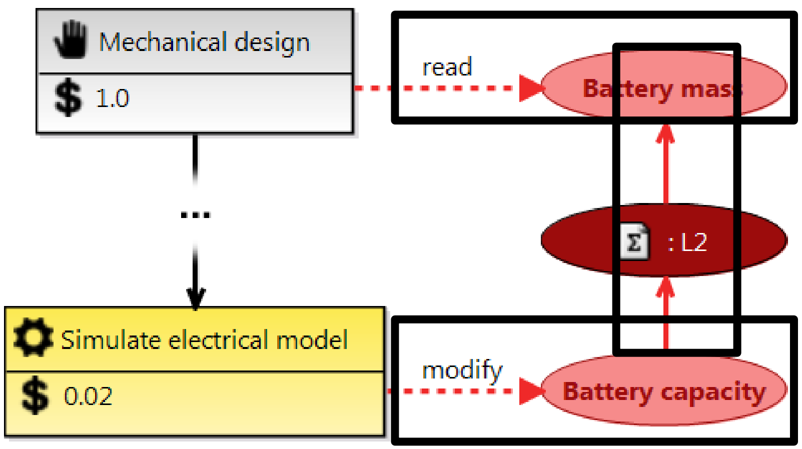
Process modeling and transformation

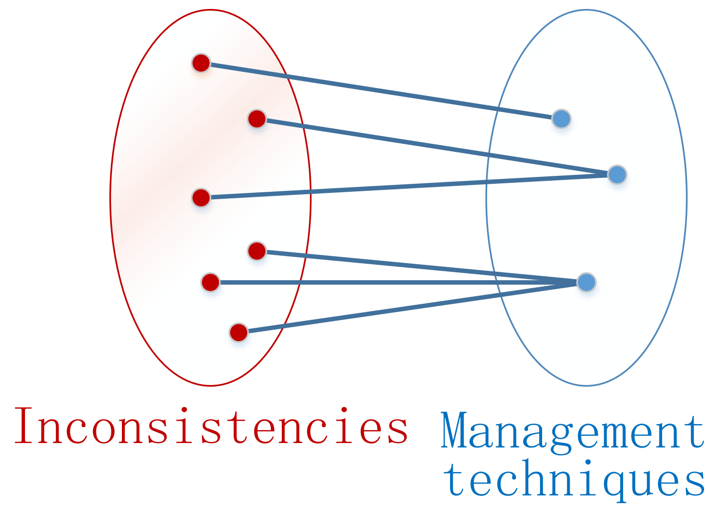
- Appropriate process modeling formalism?
  - Extended FTG+PM



Process modeling and transformation

- Appropriate process modeling formalism?
  - Extended FTG+PM

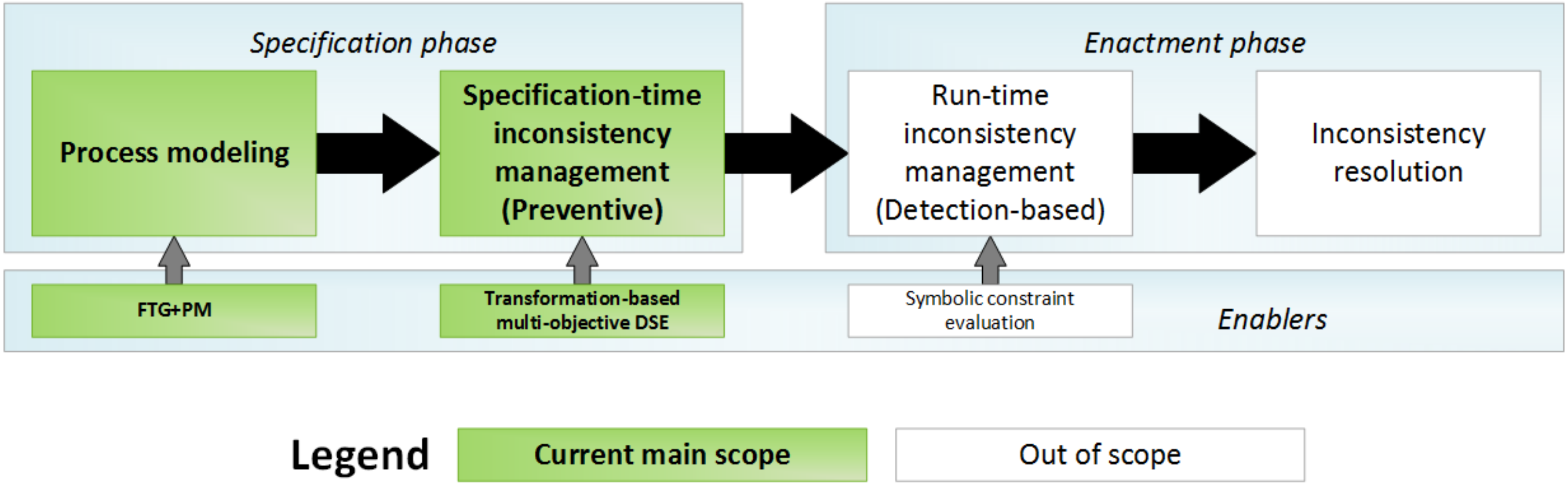




- It's an optimization problem
  - Matching ICs with ICMs while keeping transit costs at minimum
  - Challenge: impact of ICM techniques on the process

Roadmap

- Methodology+tooling
- Future work
  - Cost/performance modeling
  - Resolution techniques to be revisited
- Fits into a larger framework (see the other



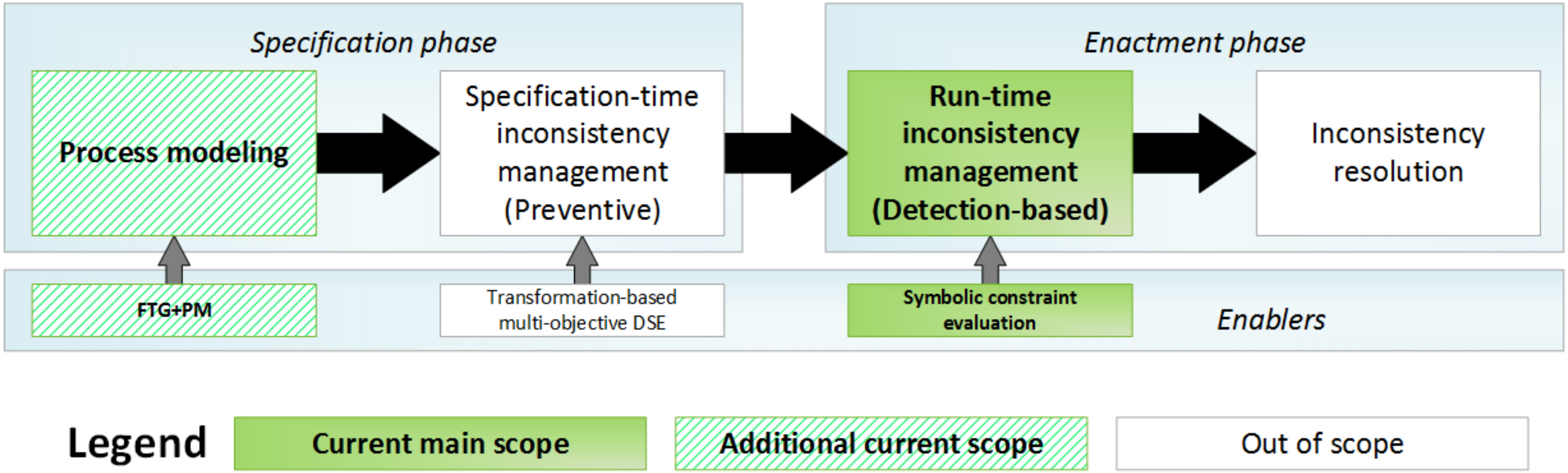
Modeling and enactment support for early detection of inconsistencies in engineering processes

Istvan David

MSDL Antwerp

istvan.david@uantwerp.be

Summary

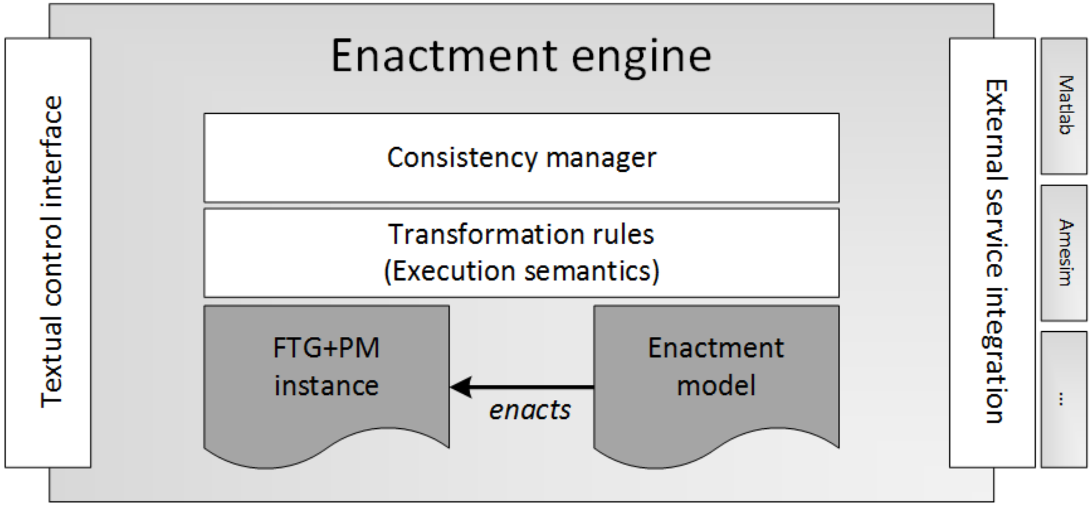


- Early inconsistency detection
- We provide:
  - A methodology for formalizing inconsistencies, and
  - An enactment engine for running the managed process.

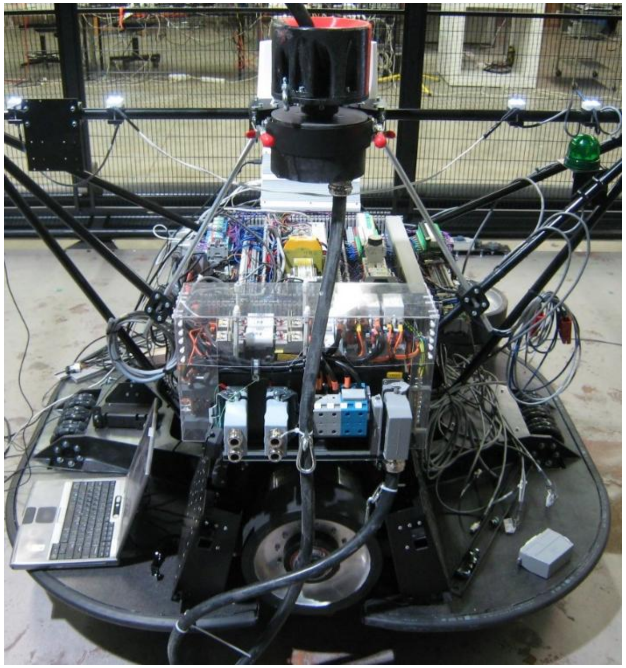
I. Dávid, B. Meyers, K. Vanherpen, Y. Van Tendeloo, K. Berx, and H. Vangheluwe, "Modeling and enactment support for early detection of inconsistencies in engineering processes,"
Submitted, under review, 2nd International Workshop on Collaborative Modelling in MDE (COMMitMDE 2017)

Process enactment

– Process modeling is a must, but it's not enough
– Process enactment is required to ensure consistency



Example

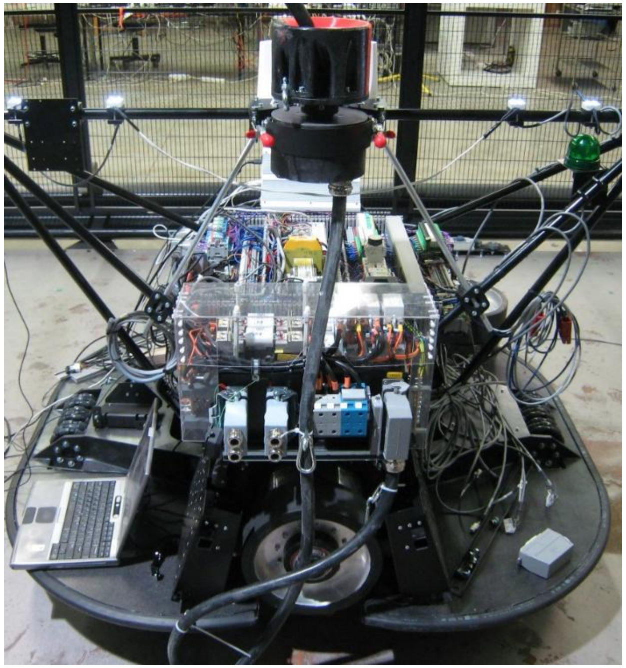


- $m_T = m_P + m_M + m_B$

- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

- mass > 0 [kg]

Example



- $m_T = m_P + m_M + m_B$

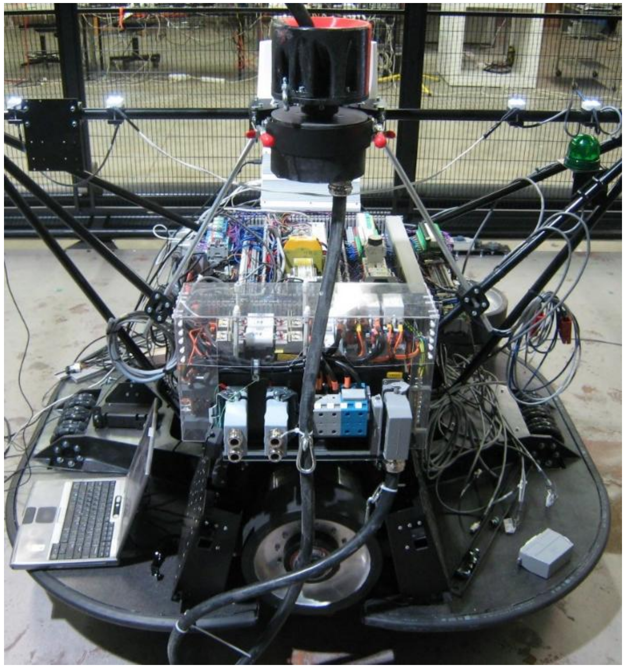
- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

- mass > 0 [kg]

Step 1

A platform is selected with a mass of 100kg. (m_P=100 [kg])

Example



– $m_T = m_P + m_M + m_B$

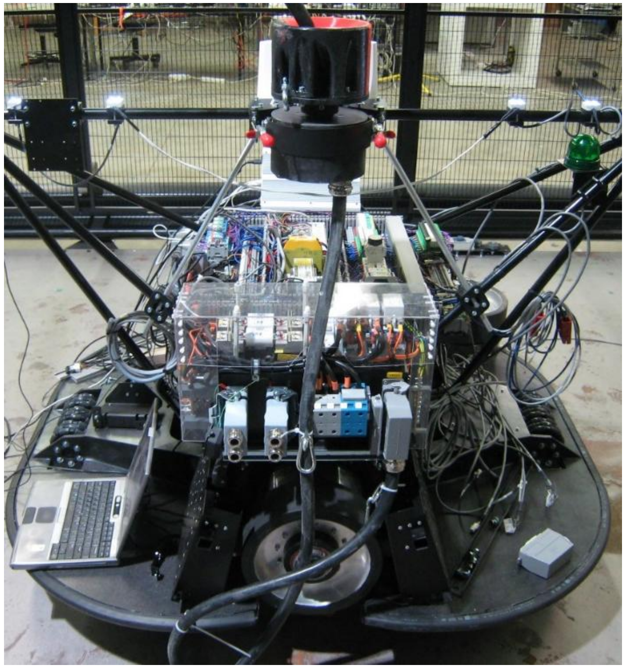
– $m_T \leq 150$ [kg],
$m_P \leq 100$ [kg],
$m_M \leq 50$ [kg],
$m_B \leq 10$ [kg]

– mass > 0 [kg]

Step 1
A platform is selected with a mass of 100kg. (m_P=100 [kg])

Step 2
A motor is selected with a mass of 50kg. (m_M=50 [kg])

Example



– $m_T = m_P + m_M + m_B$

– $m_T \leq 150$ [kg],

$m_P \leq 100$ [kg],

$m_M \leq 50$ [kg],

$m_B \leq 10$ [kg]

– mass > 0 [kg]

Step 1

A platform is selected with a mass of 100kg. (m_P=100 [kg])

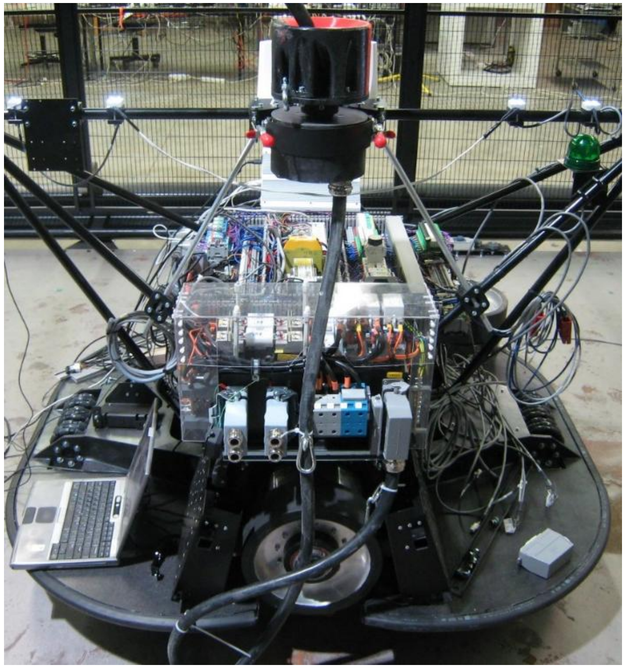
Step 2

A motor is selected with a mass of 50kg. (m_M=50 [kg])

Step 3

A battery is selected with a mass of 10 kg. (m_B= 10 [kg])

Example



- $m_T = m_P + m_M + m_B$

- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

- mass > 0 [kg]

Step 1

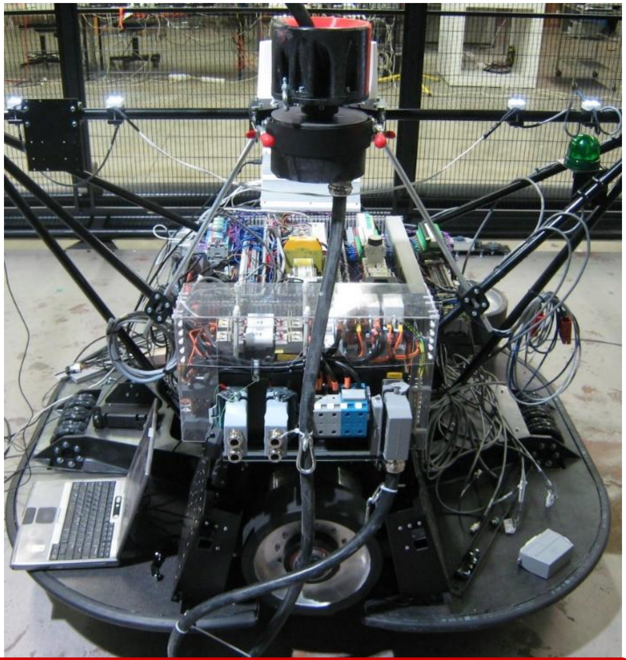
A platform is selected with a mass of 100kg. (m_P=100 [kg])

Step 2

A motor is selected with a mass of 50kg. (m_M=50 [kg])

Step 3

A battery is selected with a mass of 10 kg. (m_B= 10 [kg])



Example

- $m_T = m_P + m_M + m_B$

- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

- mass > 0 [kg]

Step 1

A platform is selected with a mass of 100kg. (m_P=100 [kg])

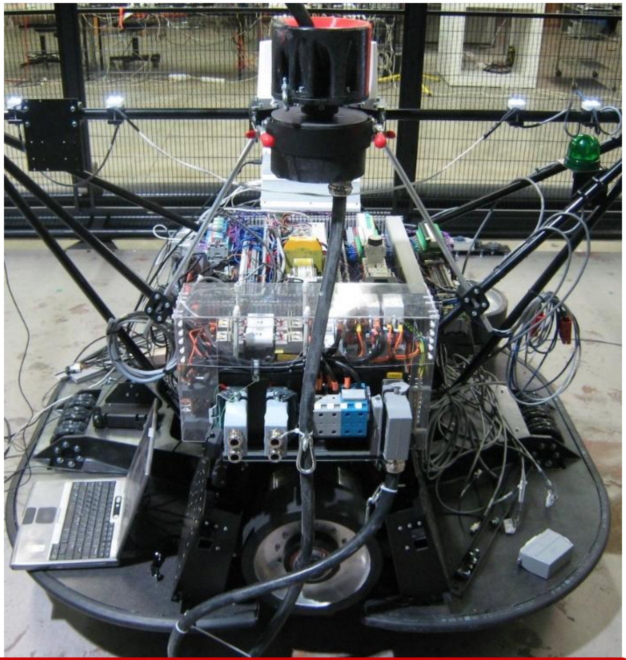
Step 2

A motor is selected with a mass of 50kg. (m_M=50 [kg])

Step 3

A battery is selected with a mass of 10 kg. (m_B= 10 [kg])

Example



- $m_T = m_P + m_M + m_B$

- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

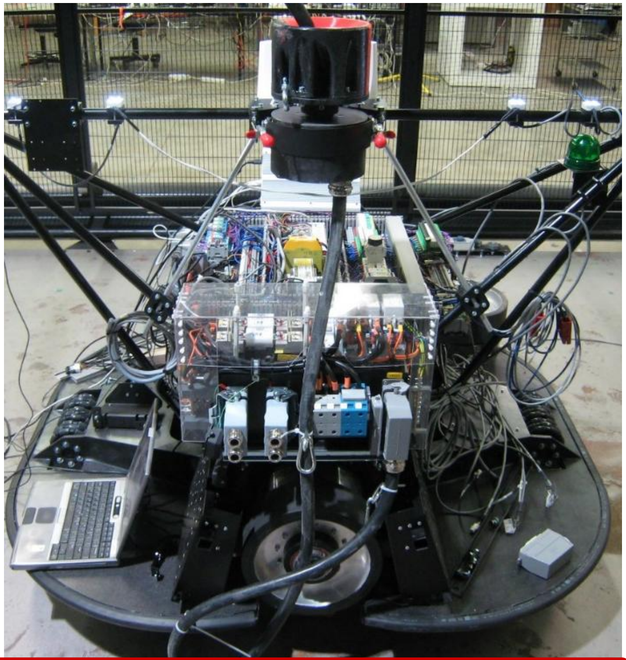
- mass > 0 [kg]

Step 1
A platform is selected with a mass of 100kg. (m_P=100 [kg])

Step 2
A motor is selected with a mass of 50kg. (m_M=50 [kg])

Step 3
A battery is selected with a mass of 10 kg. (m_B= 10 [kg])

Example



- $m_T = m_P + m_M + m_B$

Attribute

- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

Step 1
A platform is selected with a mass of 100kg. (m_P=100 [kg])

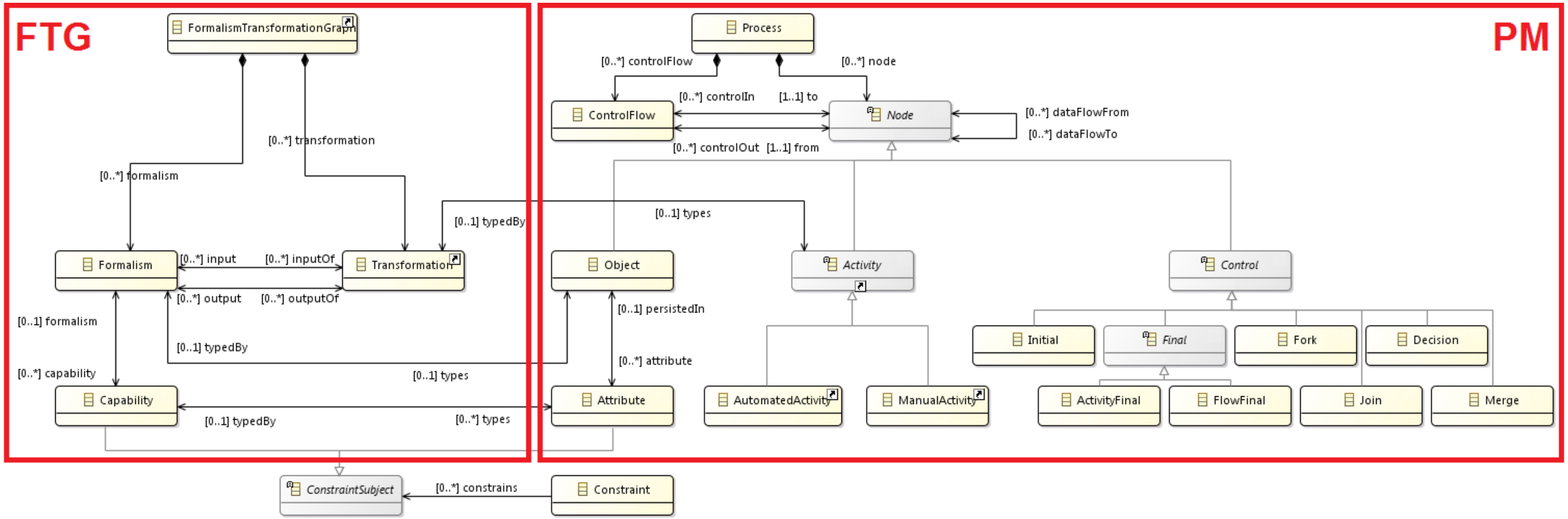
Step 2
A motor is selected with a mass of 50kg. (m_M=50 [kg])

Step 3
A battery is selected with a mass of 10 kg. (m_B= 10 [kg])

- mass > 0 [kg]

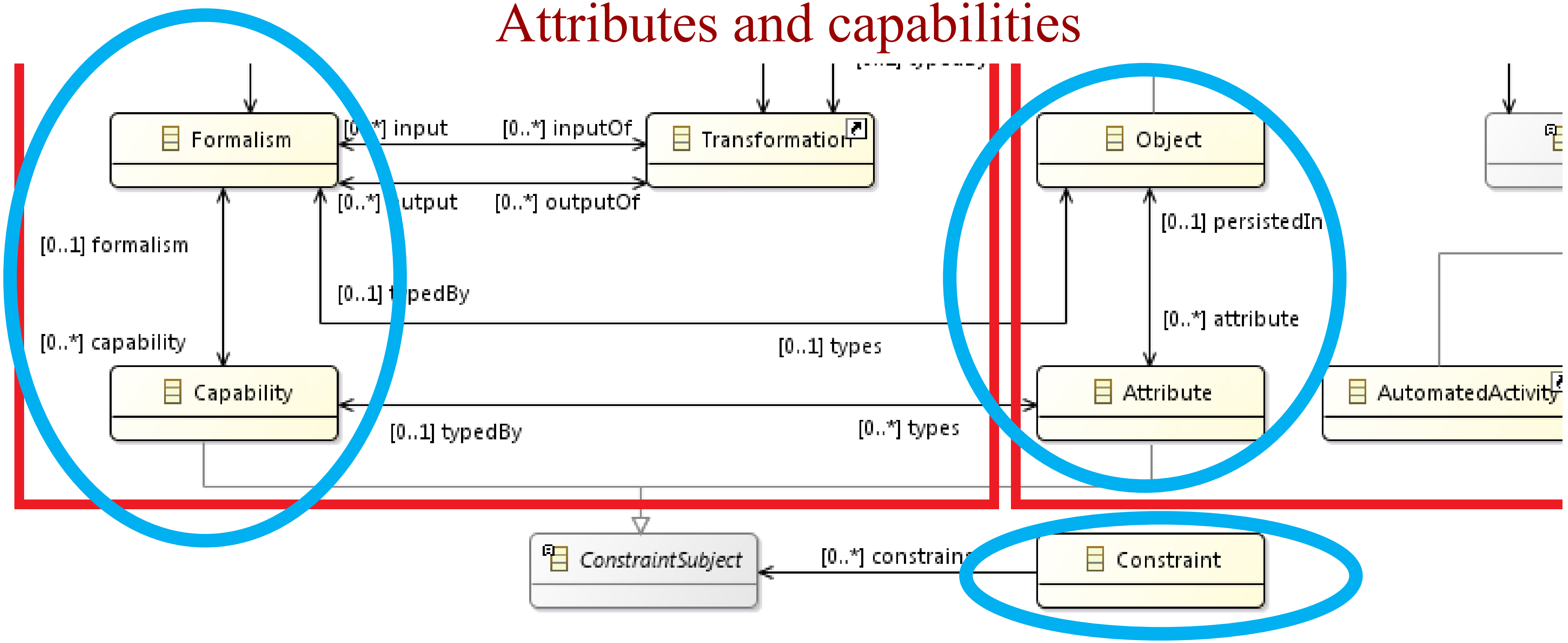
Capability

Attributes and capabilities

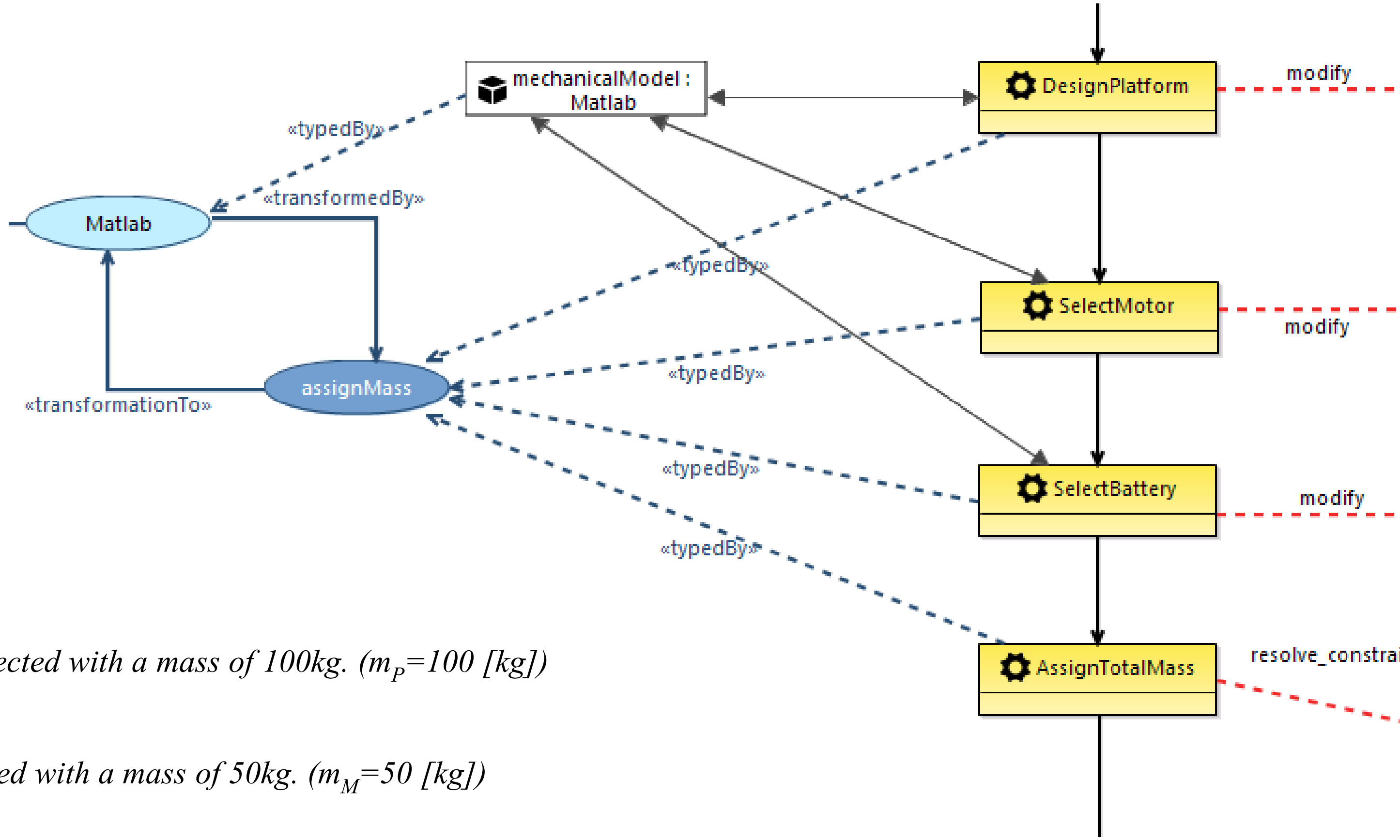


School of Computer Science

Attributes and capabilities



Modeling the process



Step 1

A platform is selected with a mass of 100kg. (m_P=100 [kg])

Step 2

A motor is selected with a mass of 50kg. (m_M=50 [kg])

Step 3

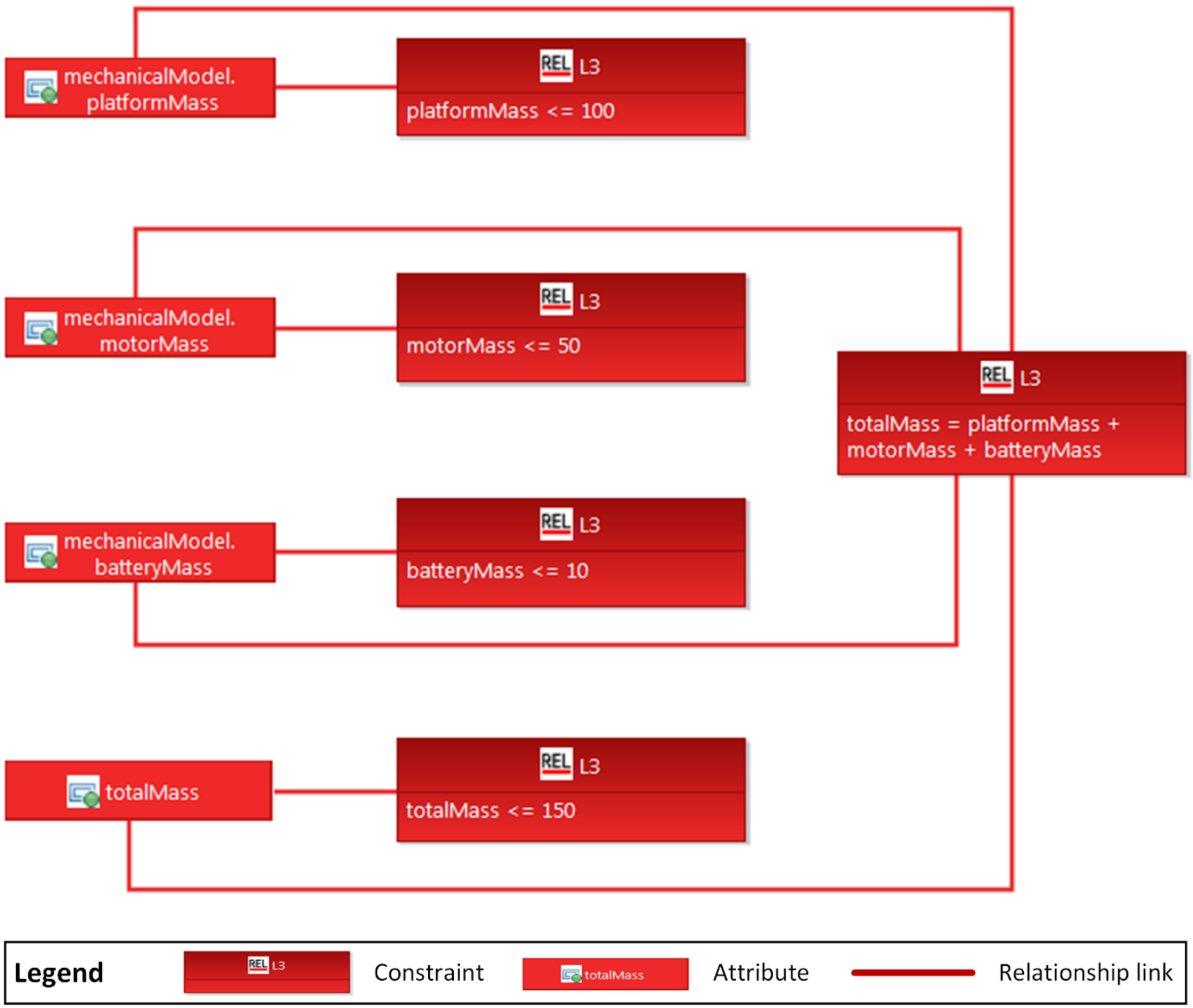
A battery is selected with a mass of 10 kg. (m_B= 10 [kg])

Modeling the process

- $m_T = m_P + m_M + m_B$

- $m_T \le 150$ [kg],
  $m_P \le 100$ [kg],
  $m_M \le 50$ [kg],
  $m_B \le 10$ [kg]

- mass > 0 [kg]

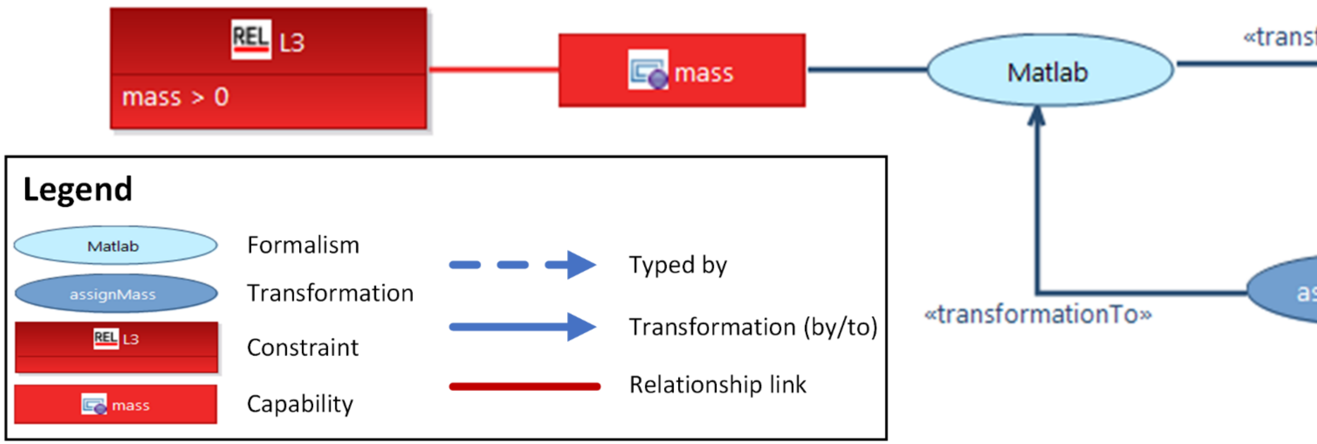


Modeling the process

- $m_T = m_P + m_M + m_B$

- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

- mass > 0 [kg]

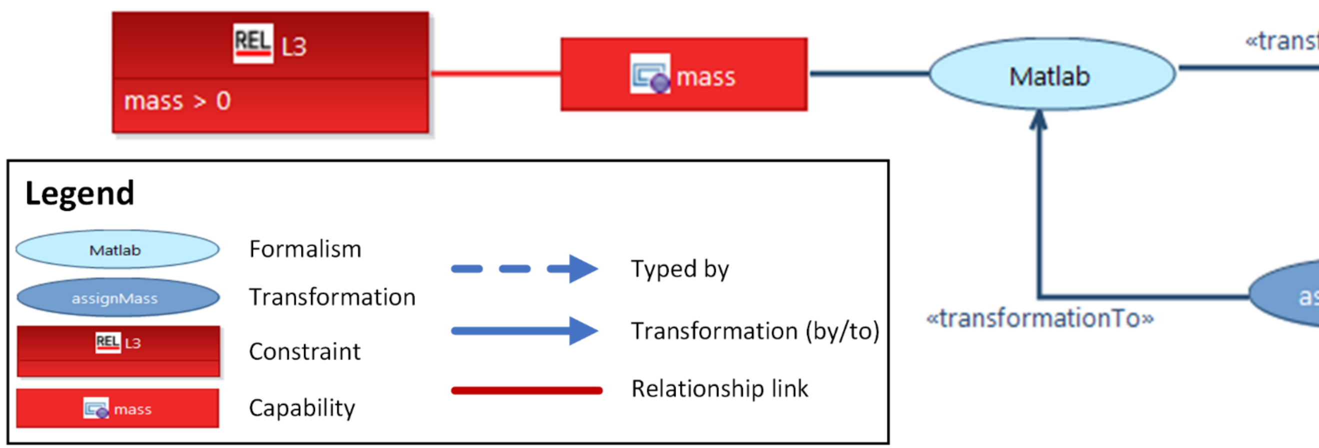


Modeling the process

– $m_T = m_P + m_M + m_B$

– $m_T \leq 150$ [kg],
$m_P \leq 100$ [kg],
$m_M \leq 50$ [kg],
$m_B \leq 10$ [kg]

– mass > 0 [kg]



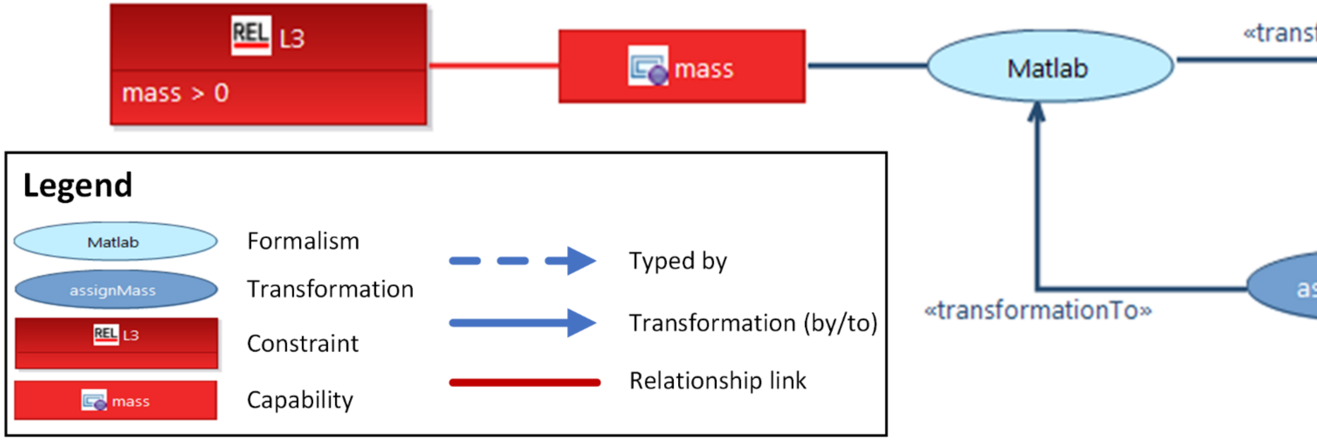
Evaluation of capability constraints
Any constraint applied on a capability imposes a constraint on every attribute typed by that capability.

Modeling the process

- $m_T = m_P + m_M + m_B$

- $m_T \leq 150$ [kg],
  $m_P \leq 100$ [kg],
  $m_M \leq 50$ [kg],
  $m_B \leq 10$ [kg]

- mass > 0 [kg]



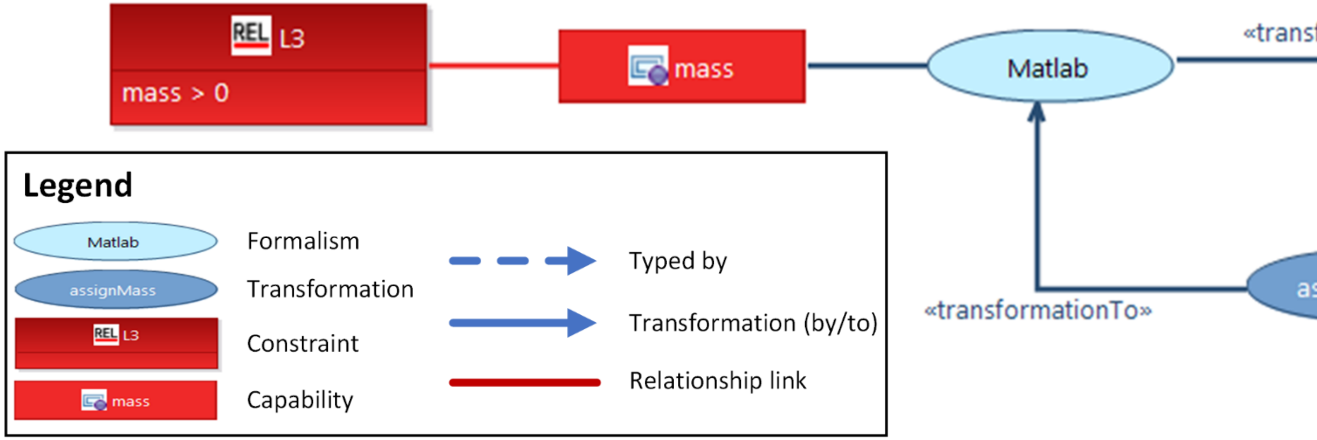
Evaluation of capability constraints
Any constraint applied on a capability imposes a constraint on every attribute typed by that capability.

$$0\text{ [kg]} < m_T \leq 150\text{ [kg]},$$
$$0\text{ [kg]} < m_P \leq 100\text{ [kg]},$$
$$0\text{ [kg]} < m_M \leq 50\text{ [kg]},$$
$$0\text{ [kg]} < m_B \leq 10\text{ [kg]}$$

Modeling the process

– $m_T = m_P + m_M + m_B$

– $m_T \leq 150$ [kg],
$m_P \leq 100$ [kg],
$m_M \leq 50$ [kg],
$m_B \leq 10$ [kg]

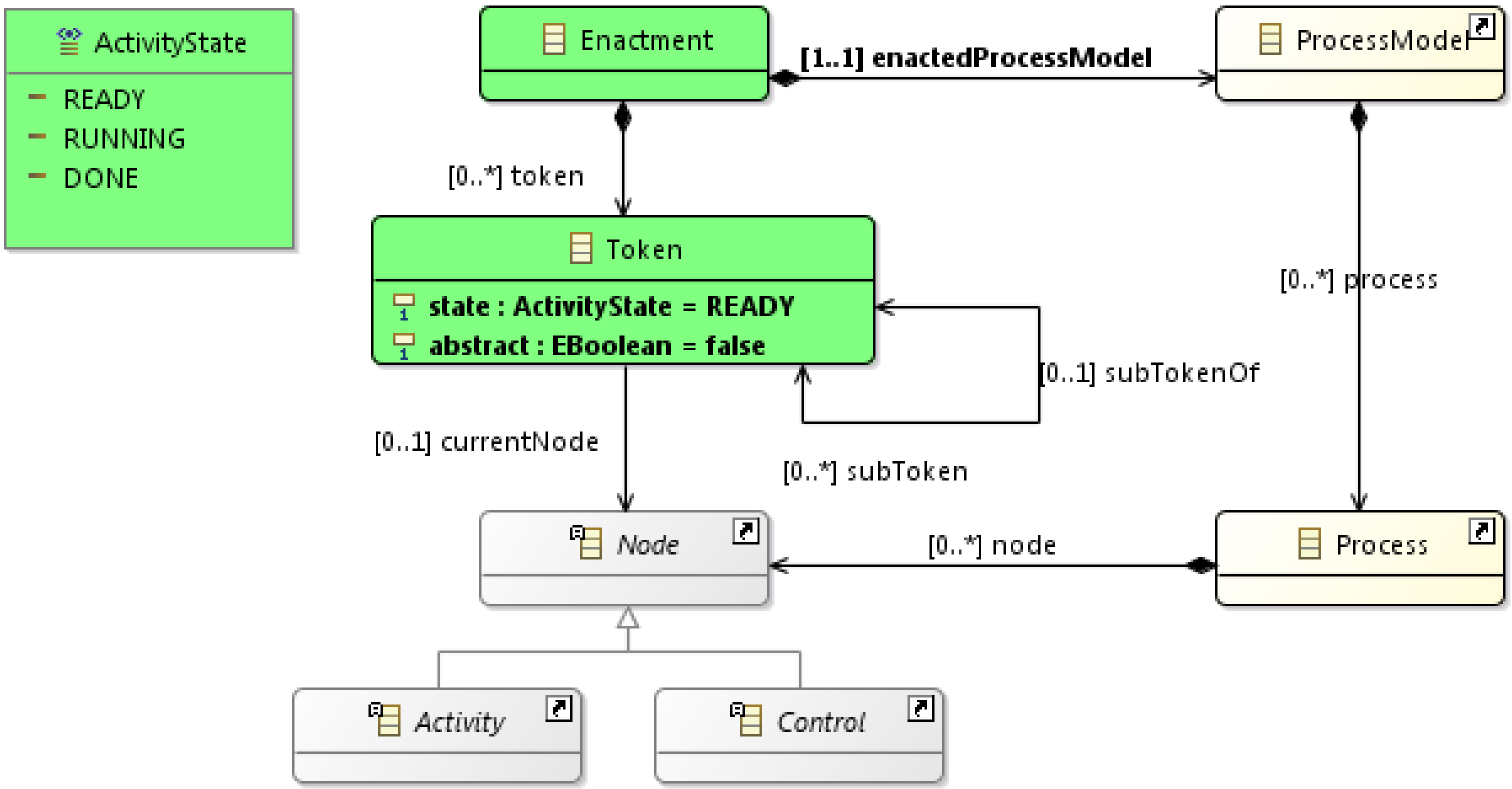


– mass > 0 [kg]

Evaluation of capability constraints
Any constraint applied on a capability imposes a constraint on every attribute typed by that capability.

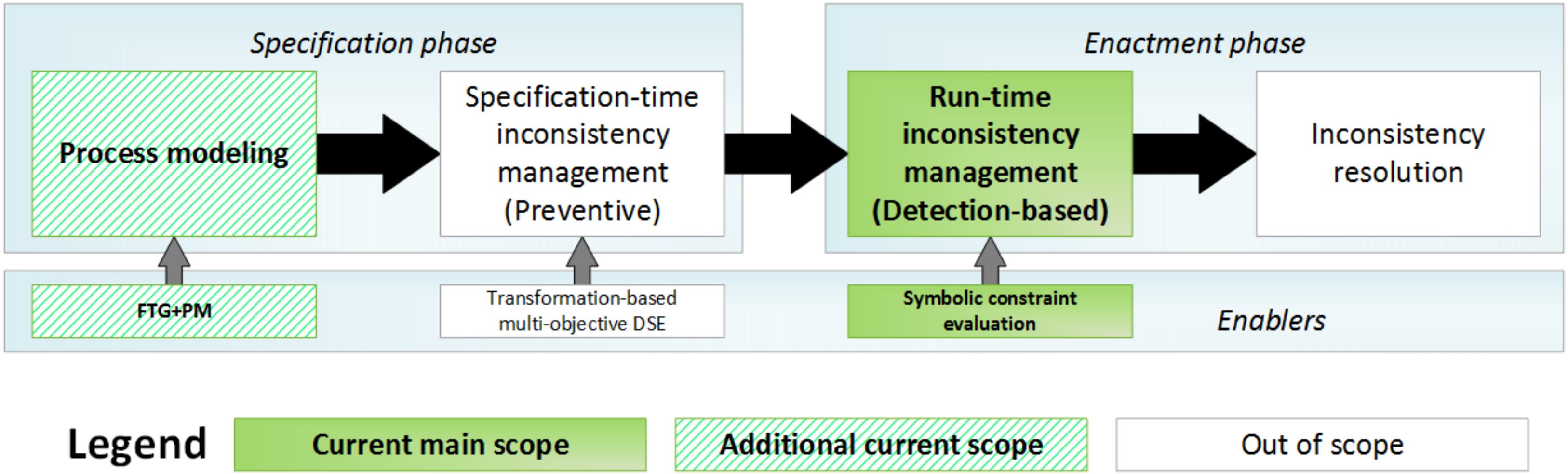
$0 \text{ [kg]} < m_T \leq 150 \text{ [kg]},$
$\mathbf{0 \text{ [kg]} < m_P \leq 100 \text{ [kg]},}$
$\mathbf{0 \text{ [kg]} < m_M \leq 50 \text{ [kg]},}$
$0 \text{ [kg]} < m_B \leq 10 \text{ [kg]}$

Process enactment



Roadmap

- Methodology and tooling provided
  - Tooling: fully modeled execution
  - Interfacing with Matlab/Simulink and AMESim
- Future work
  - Combine with specification-time inconsistency management



Enabling Contract-based Design in Engineering Processes

Istvan David

MSDL Antwerp

istvan.david@uantwerp.be

McGill School of Computer Science

NECSIS

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

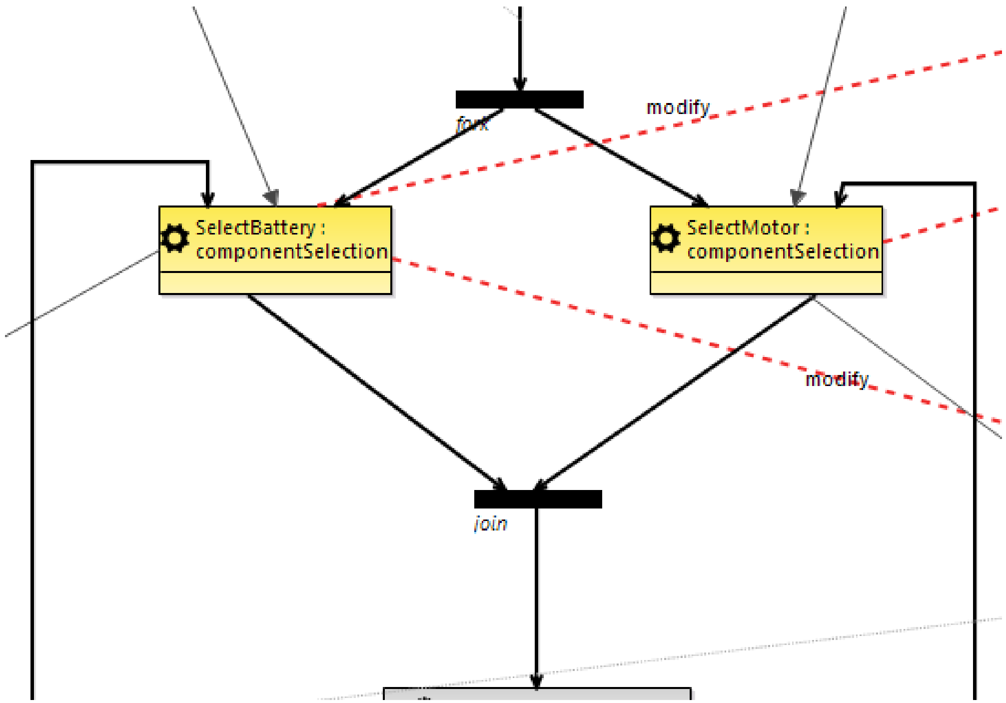
Ansymo Antwerp Systems & Software Modelling University of Antwerp

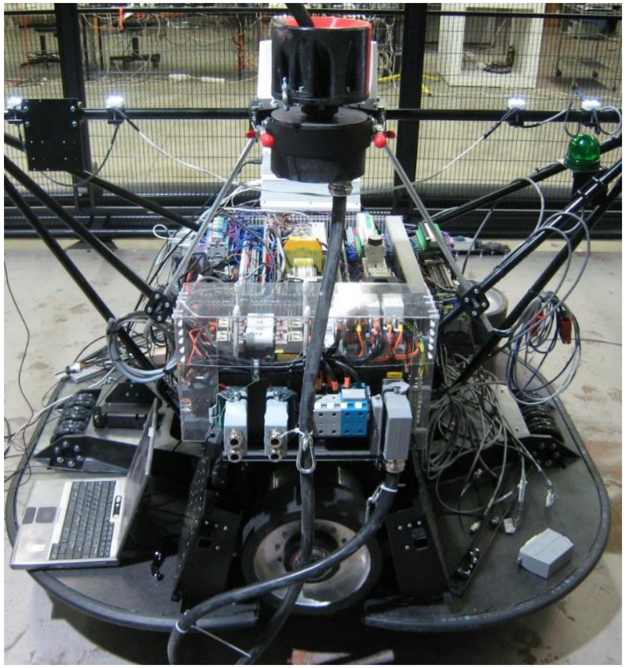
Summary

- Ensuring consistency in parallel branches of the enacted process
  - Preventive technique
- We provide:
  - A methodology for ensuring consistency by contracts
  - Tooling for
    - modeling and enacting the process, and
    - specifying contracts, and
    - use contracts as a preventive technique for inconsisteny mgmt.

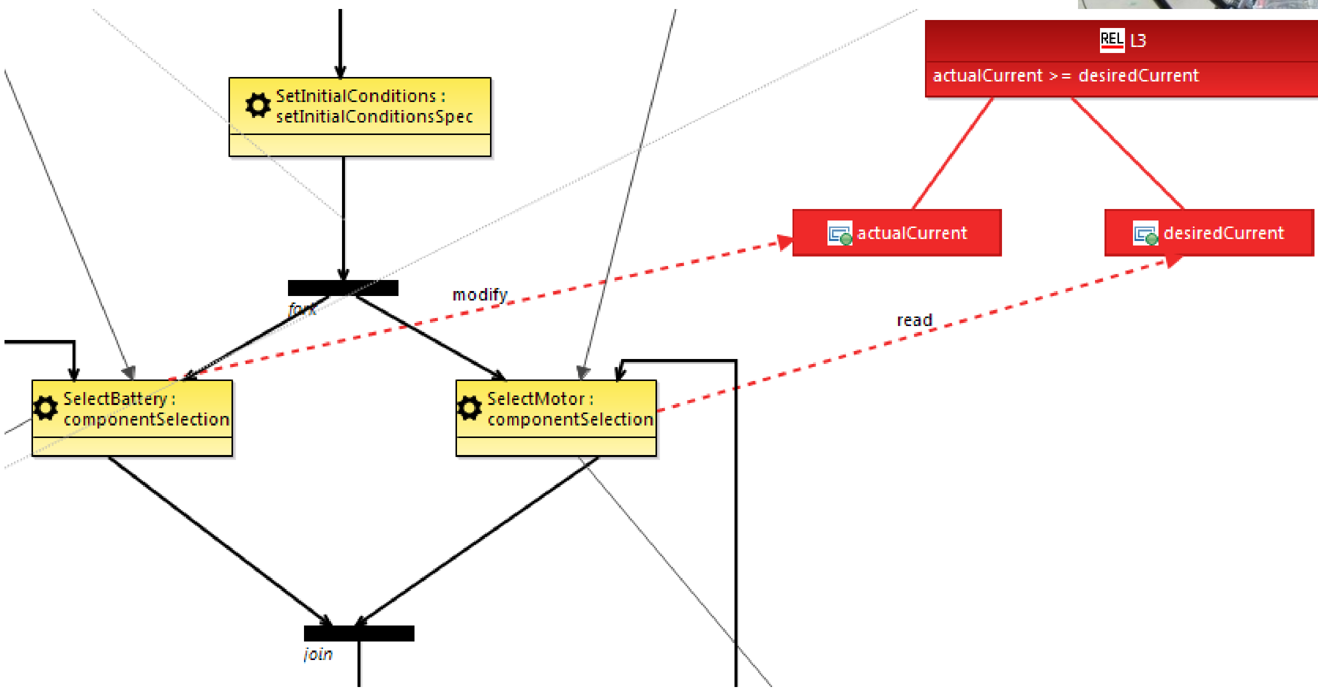
Example

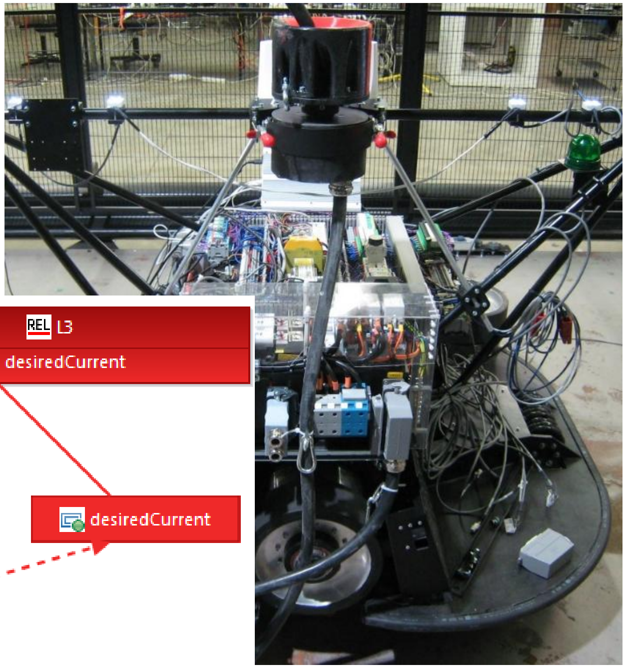
- The motor and the battery are selected in parallel





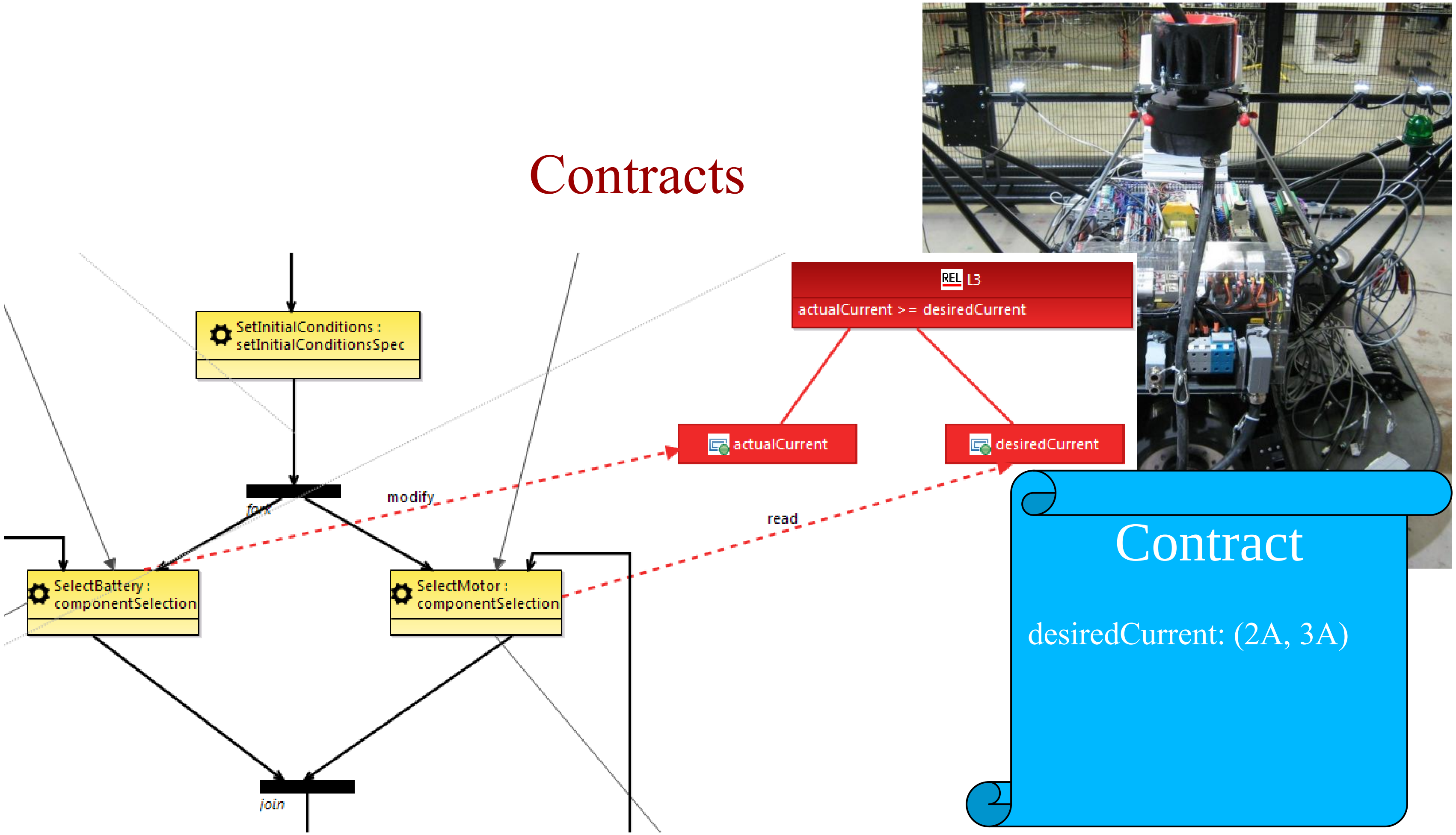
Example





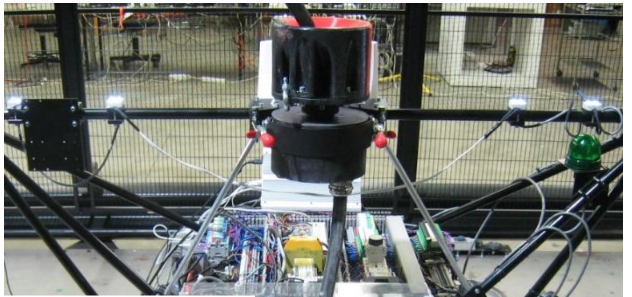
- Driving the motor *assumes* a minimum current from the battery
- The battery *guarantees* a minimum current for the motor

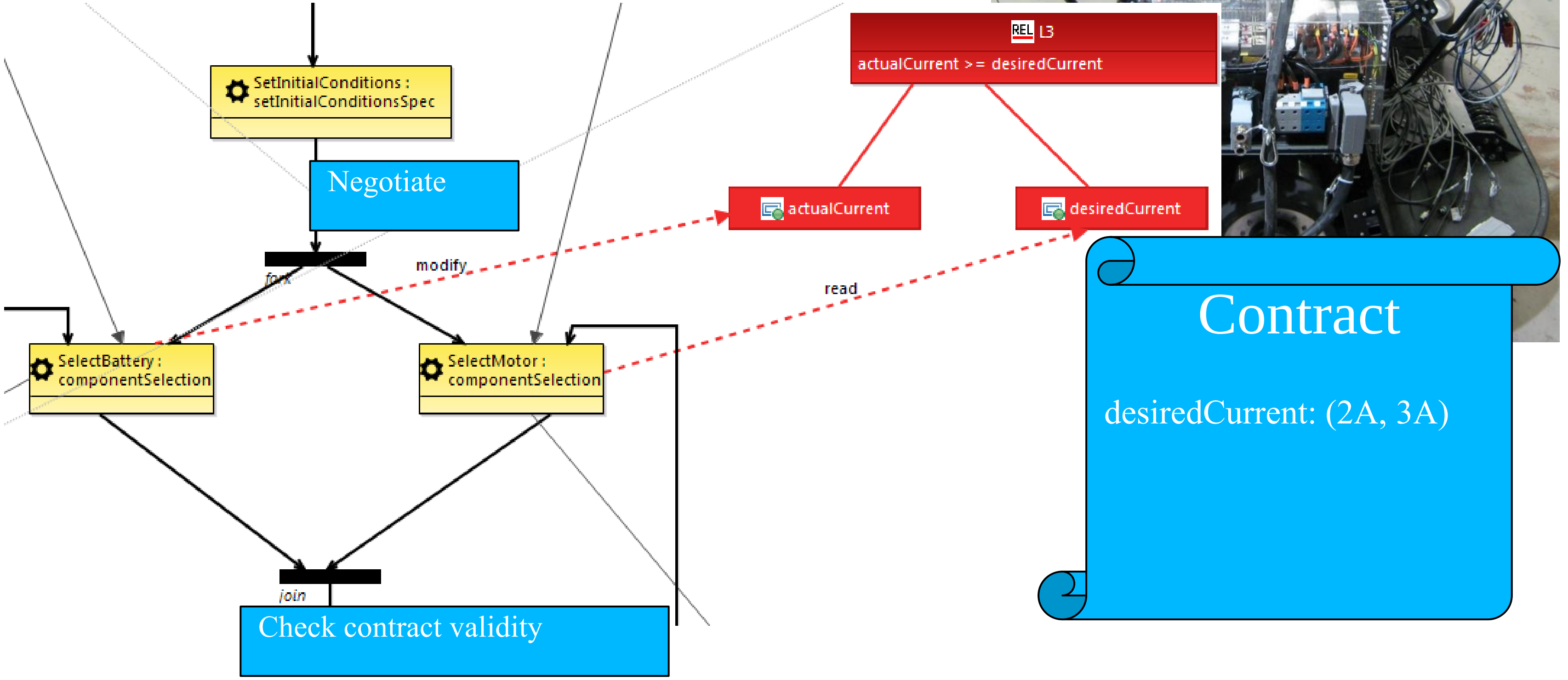
Contracts



- Driving the motor *demands* a minimum current from the battery
- The battery *guarantees* a minimum current for the motor

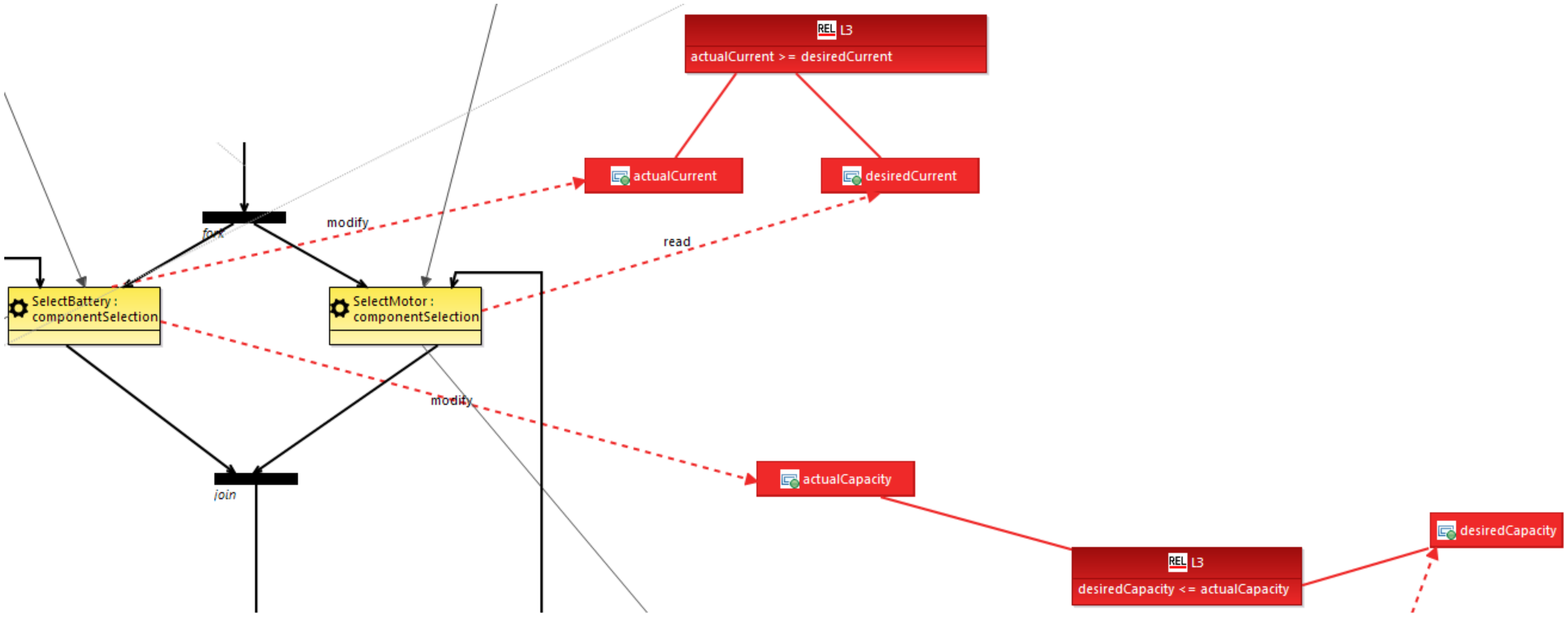
Contracts



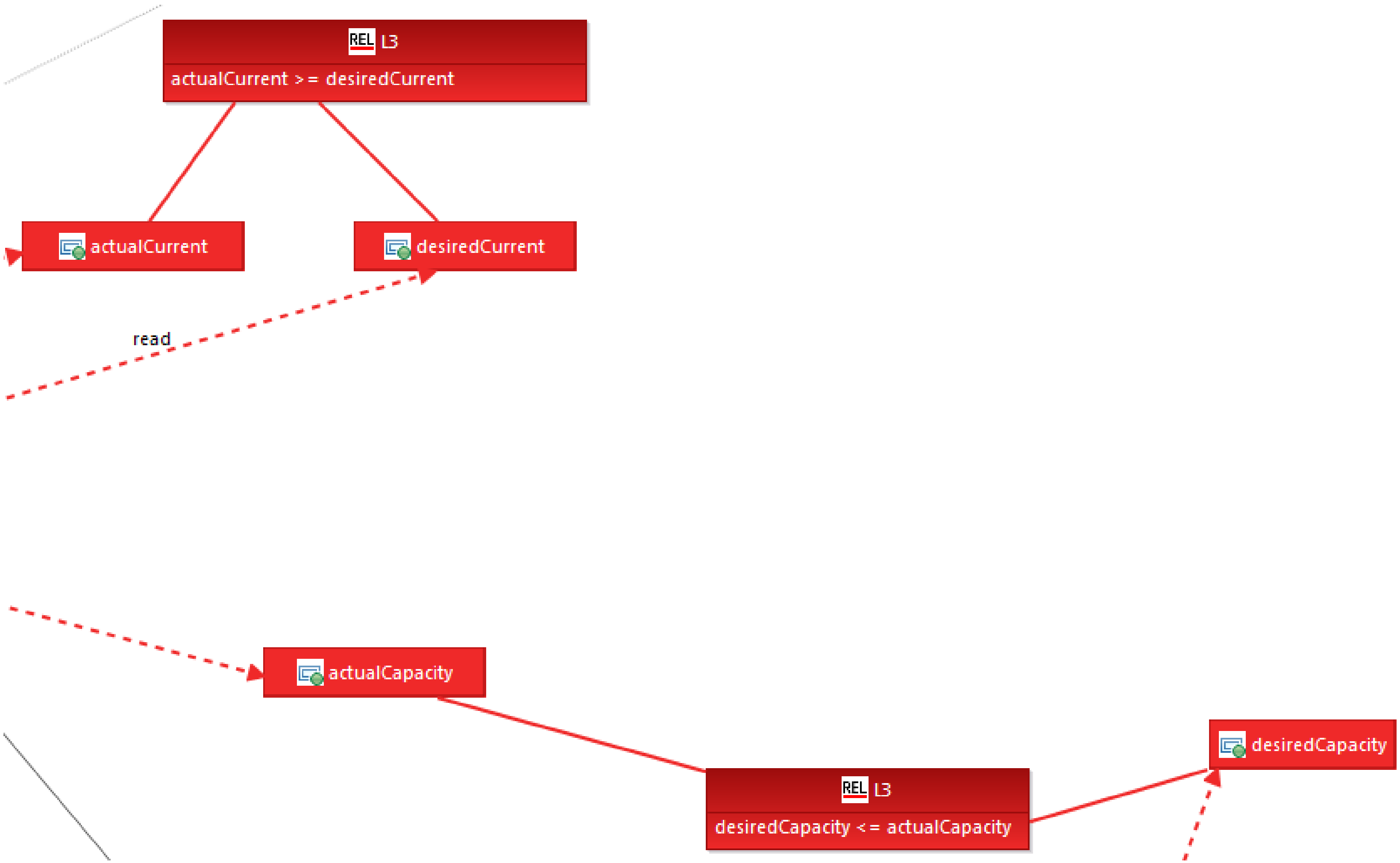


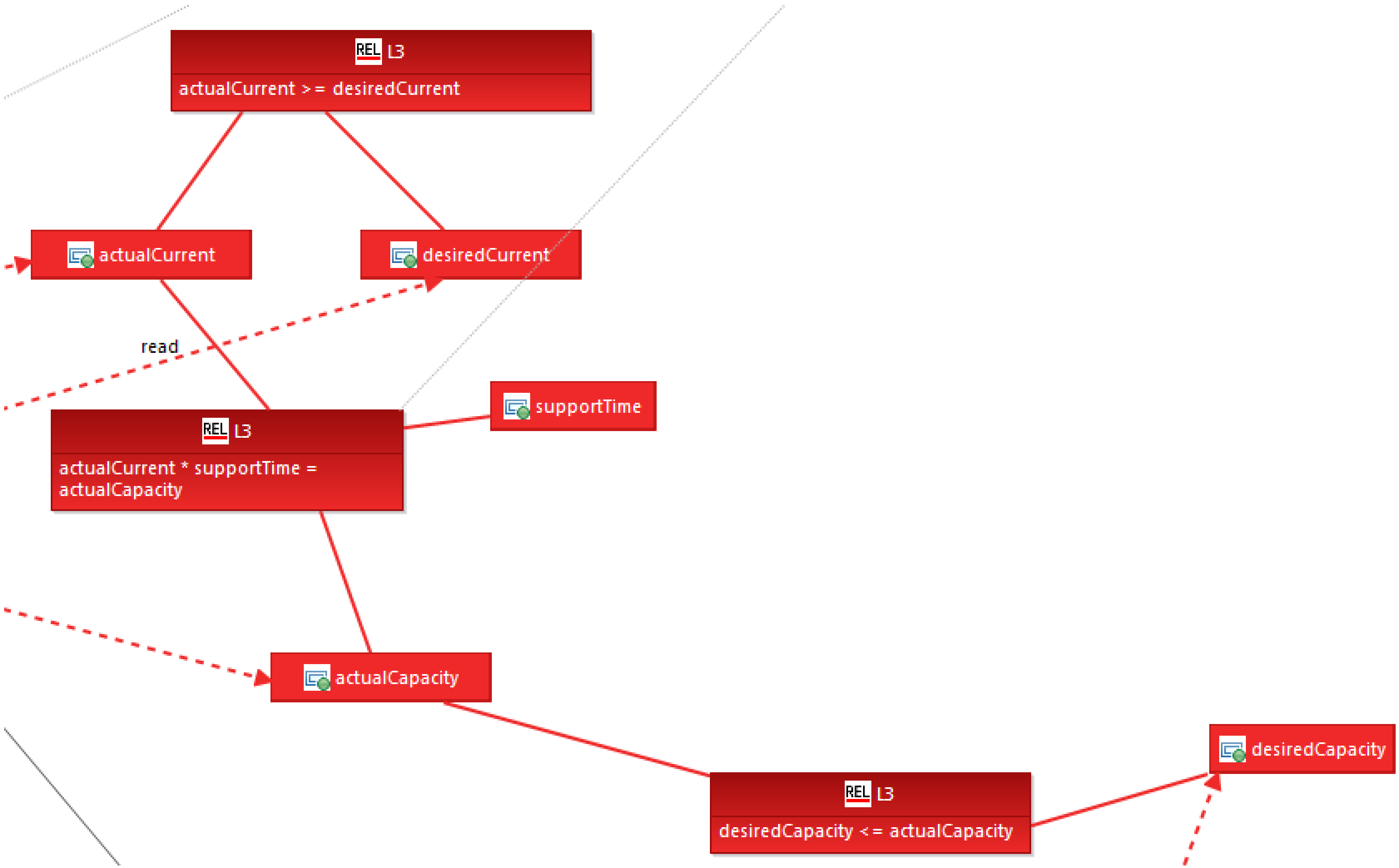
- Driving the motor *demands* a minimum current from the battery
- The battery *guarantees* a minimum current for the motor

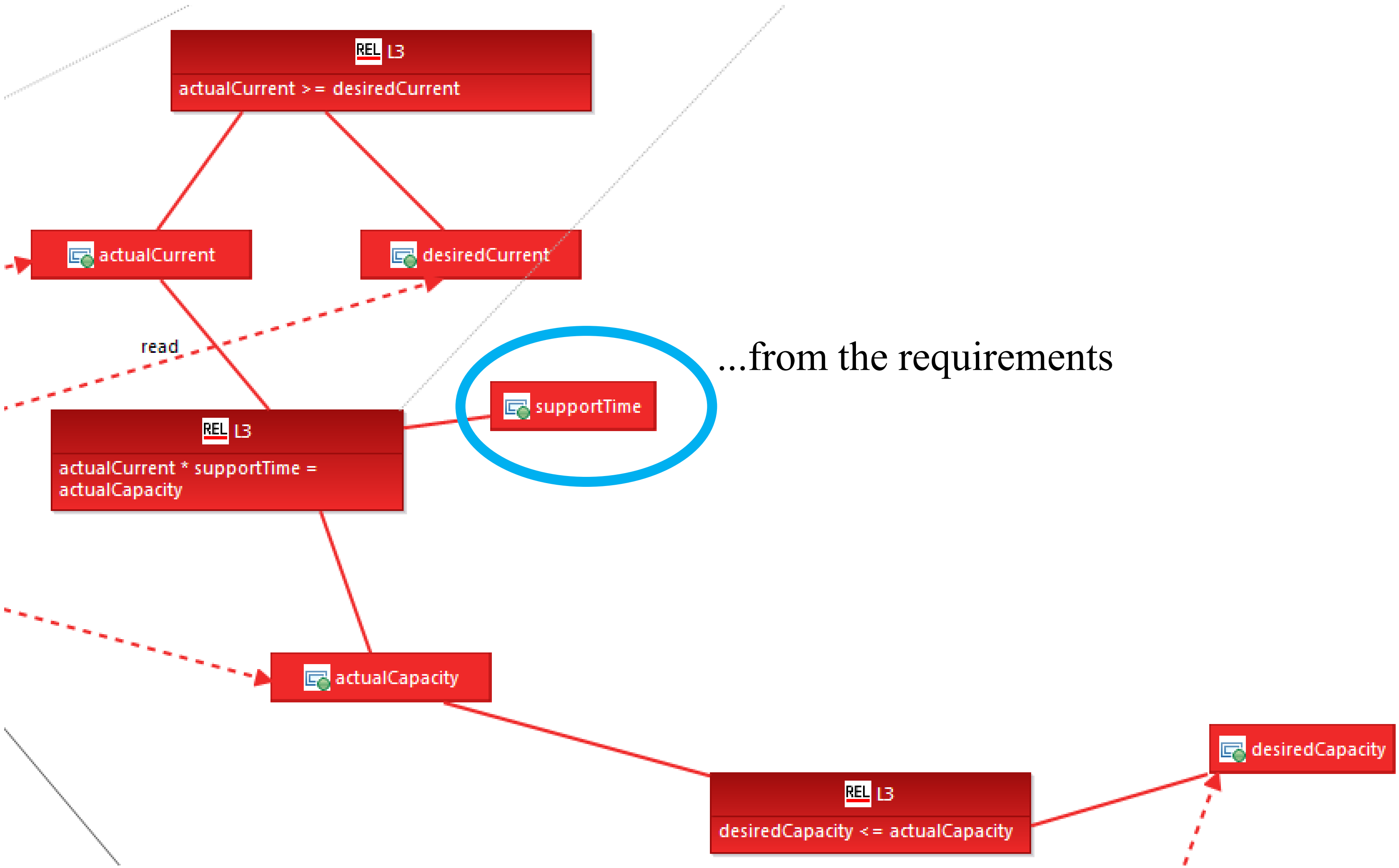
Leveraging attributes and constraints



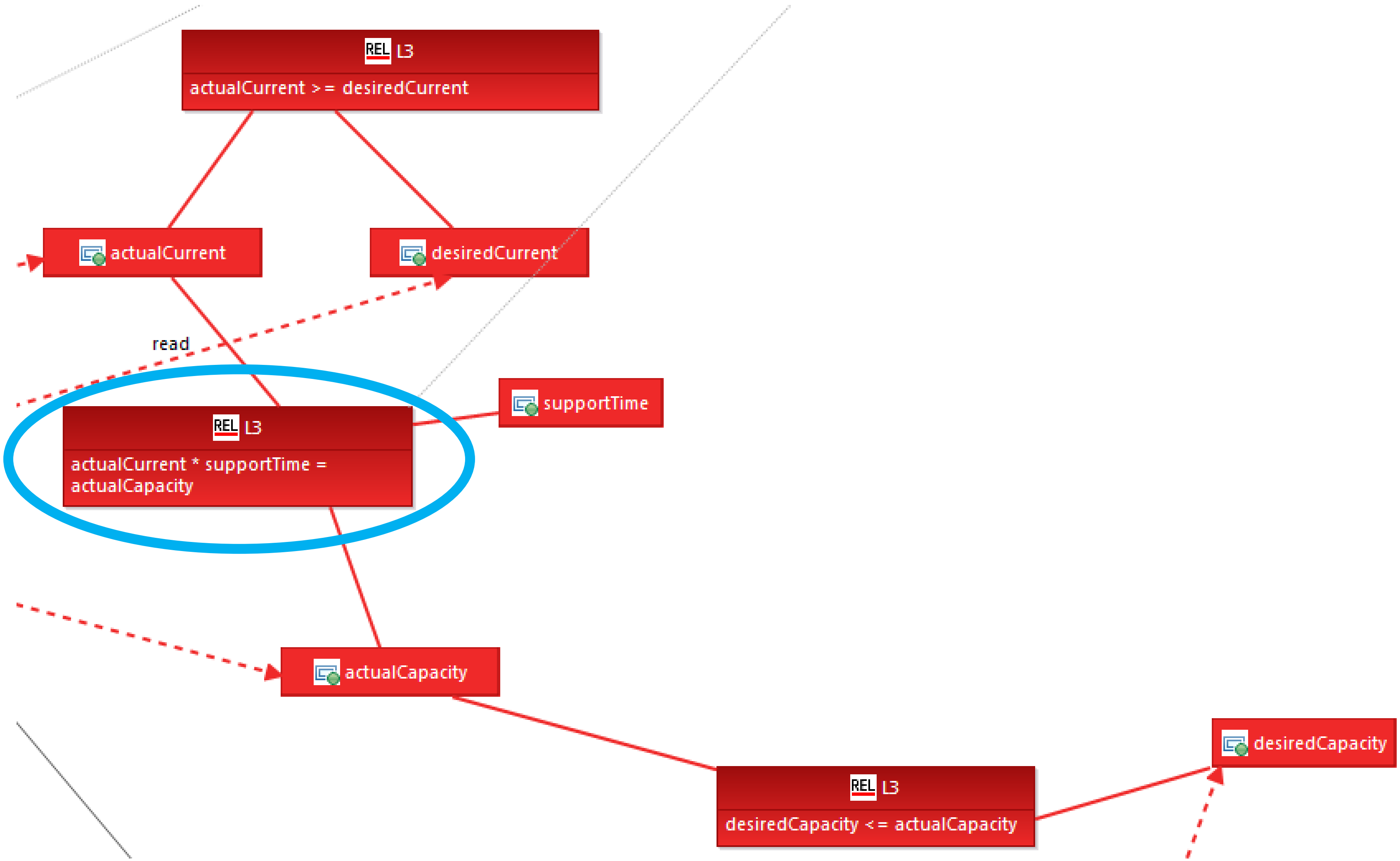
Leveraging attributes and constraints

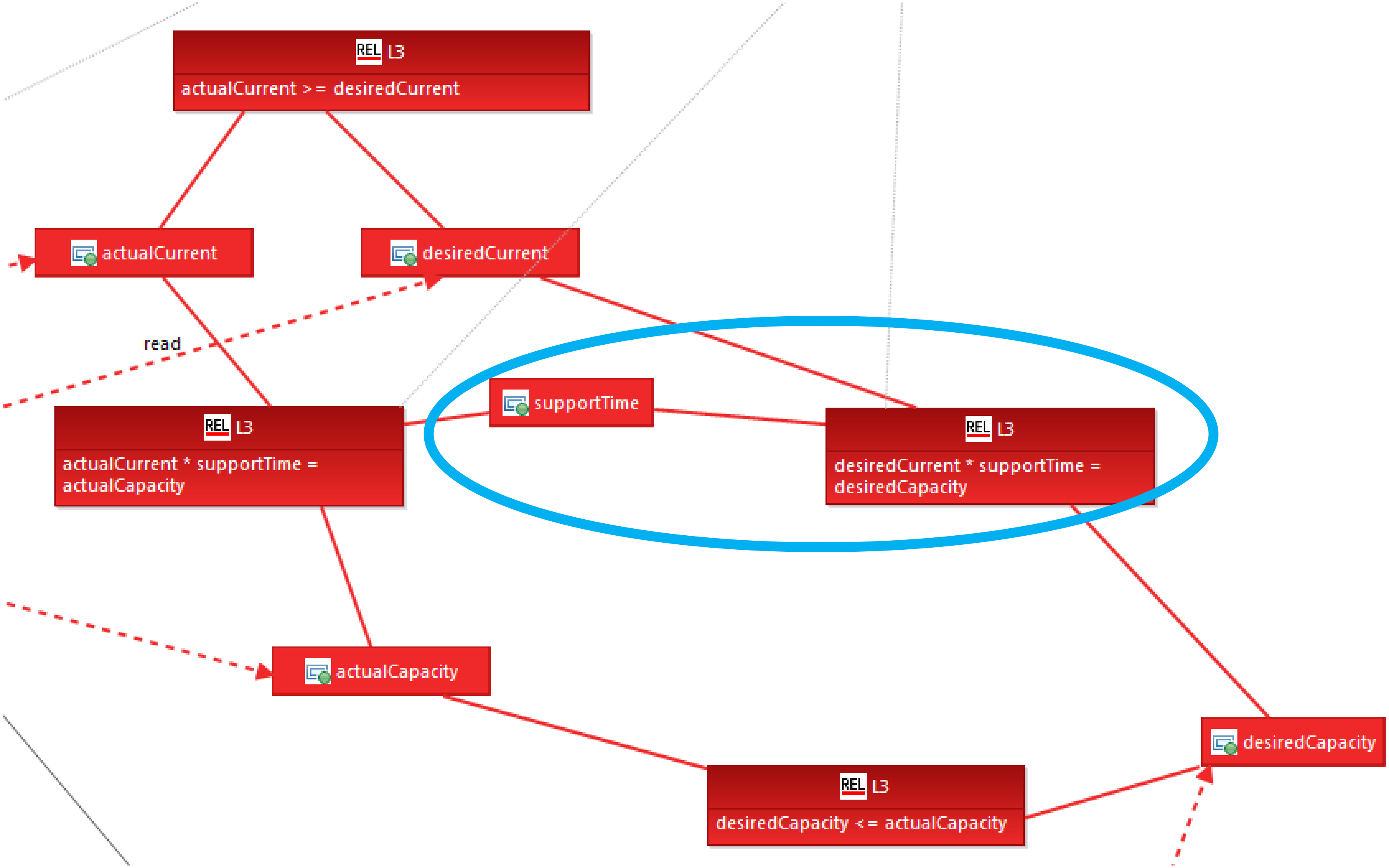


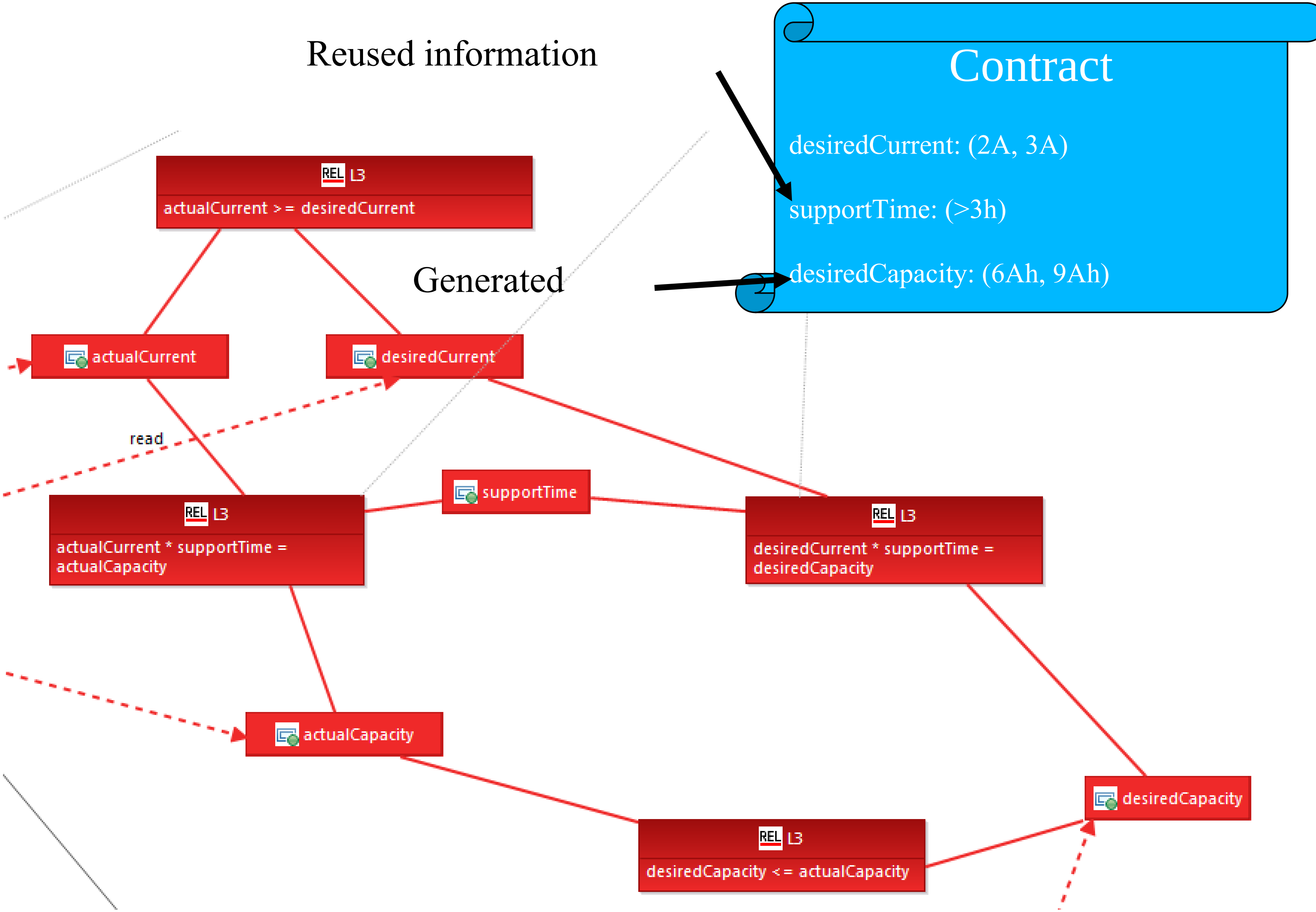

REL L3
actualCurrent >= desiredCurrent
actualCurrent
desiredCurrent
read
supportTime
REL L3
actualCurrent * supportTime = actualCapacity
actualCapacity
desiredCapacity
REL L3
desiredCapacity <= actualCapacity



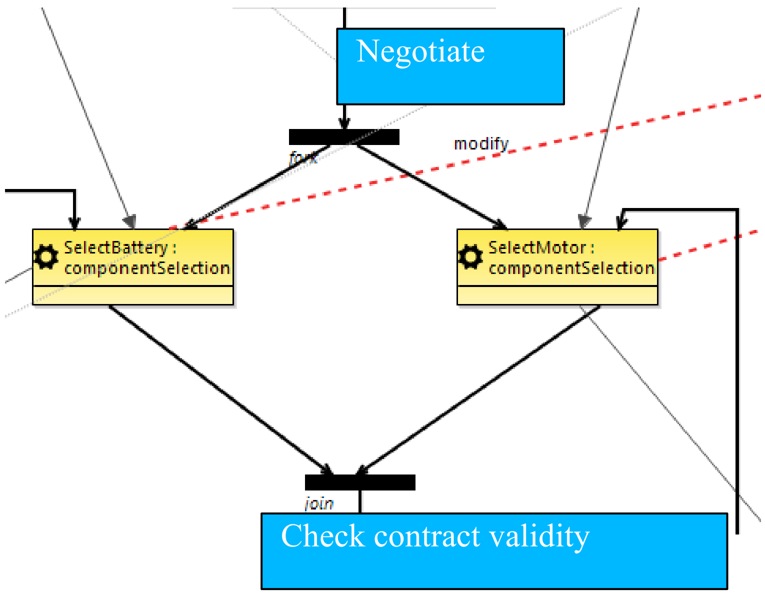
...from the requirements


REL L3
actualCurrent >= desiredCurrent
actualCurrent
desiredCurrent
read
supportTime
REL L3
actualCurrent * supportTime =
actualCapacity
actualCapacity
desiredCapacity
REL L3
desiredCapacity <= actualCapacity


REL L3
actualCurrent >= desiredCurrent
actualCurrent
desiredCurrent
read
REL L3
actualCurrent * supportTime = actualCapacity
supportTime
REL L3
desiredCurrent * supportTime = desiredCapacity
actualCapacity
desiredCapacity
REL L3
desiredCapacity <= actualCapacity

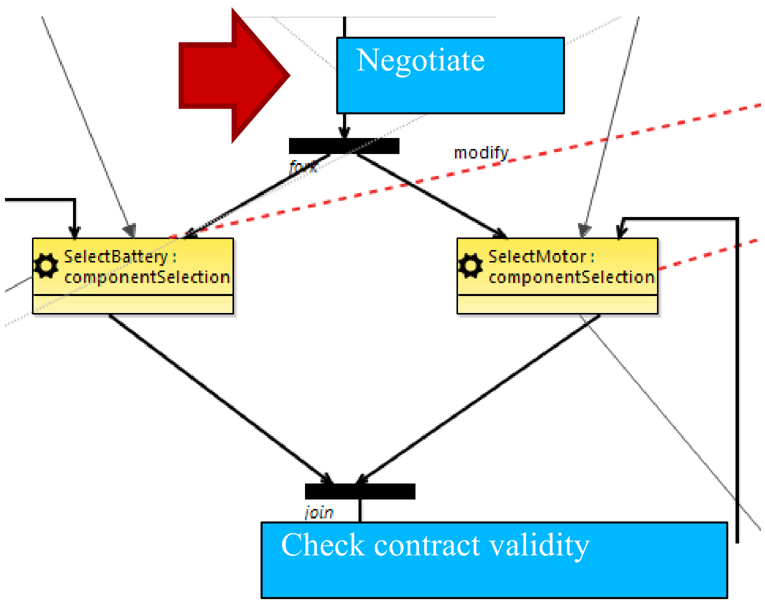

Reused information
Contract
desiredCurrent: (2A, 3A)
supportTime: (>3h)
desiredCapacity: (6Ah, 9Ah)
Generated
REL L3
actualCurrent >= desiredCurrent
actualCurrent
desiredCurrent
read
supportTime
REL L3
actualCurrent * supportTime = actualCapacity
REL L3
desiredCurrent * supportTime = desiredCapacity
actualCapacity
desiredCapacity
REL L3
desiredCapacity <= actualCapacity

Enactment



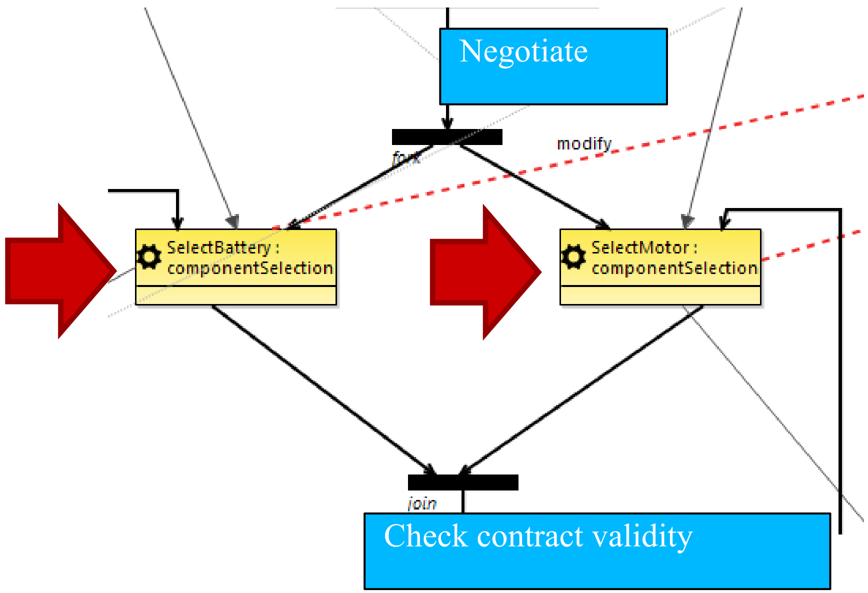
Negotiate
fork
modify
SelectBattery : componentSelection
SelectMotor : componentSelection
join
Check contract validity

Enactment



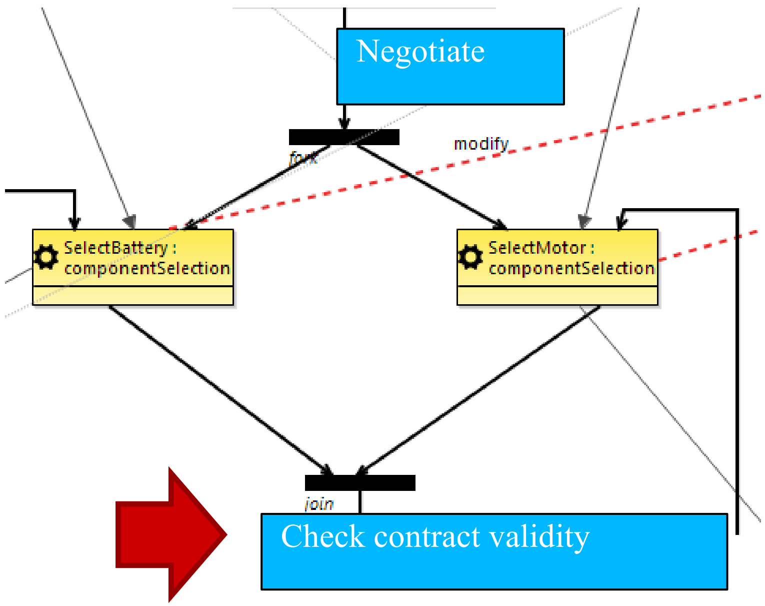
– Negotiate a contract based modify-read pairs of intents.

Enactment



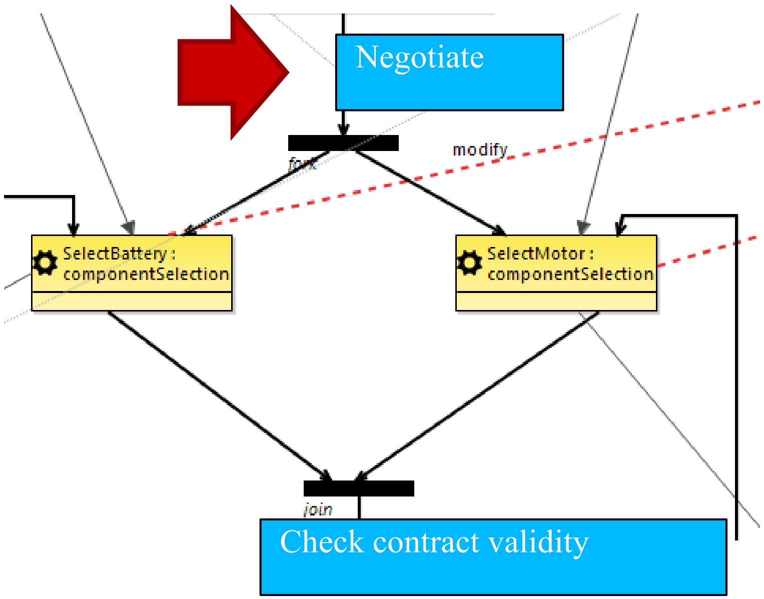
- Negotiate a contract based modify-read pairs of intents.
- Consistency between the parallel branches is managed by the contract. From the process engine's point of view, this is a „safe zone".

Enactment



- Negotiate a contract based modify-read pairs of intents.
- Consistency between the parallel branches is managed by the contract. From the process engine's point of view, this is a „safe zone".
- Upoin joining the branches, the contract is checked.

Alternative execution semantics



- Negotiate a contract
- Map its contents to new constraints of the attributes

Contributions

- From the process point of view:
  - CBCD as an inconsistency management technique
- From the CBCD point of view:
  - Less work during contract negotiation, as part of it can be inferred
  - If sufficient information is provided, the contract can be fully generated
- Integrated tooling
  - Process tool + CBCD tool

Roadmap

- Ongoing research, but the added value to the SOTA is obvious
- Tasks:
  - Work out an example ✔
  - Identify added value vs our previous work on
    - processes, and
    - contract-based design.
  - Provide tooling
- Target venue: ETAPS/FASE (submission in October)

McGill School of Computer Science

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Contract-Based Co-Design (CBCD)

Ing. Ken Vanherpen

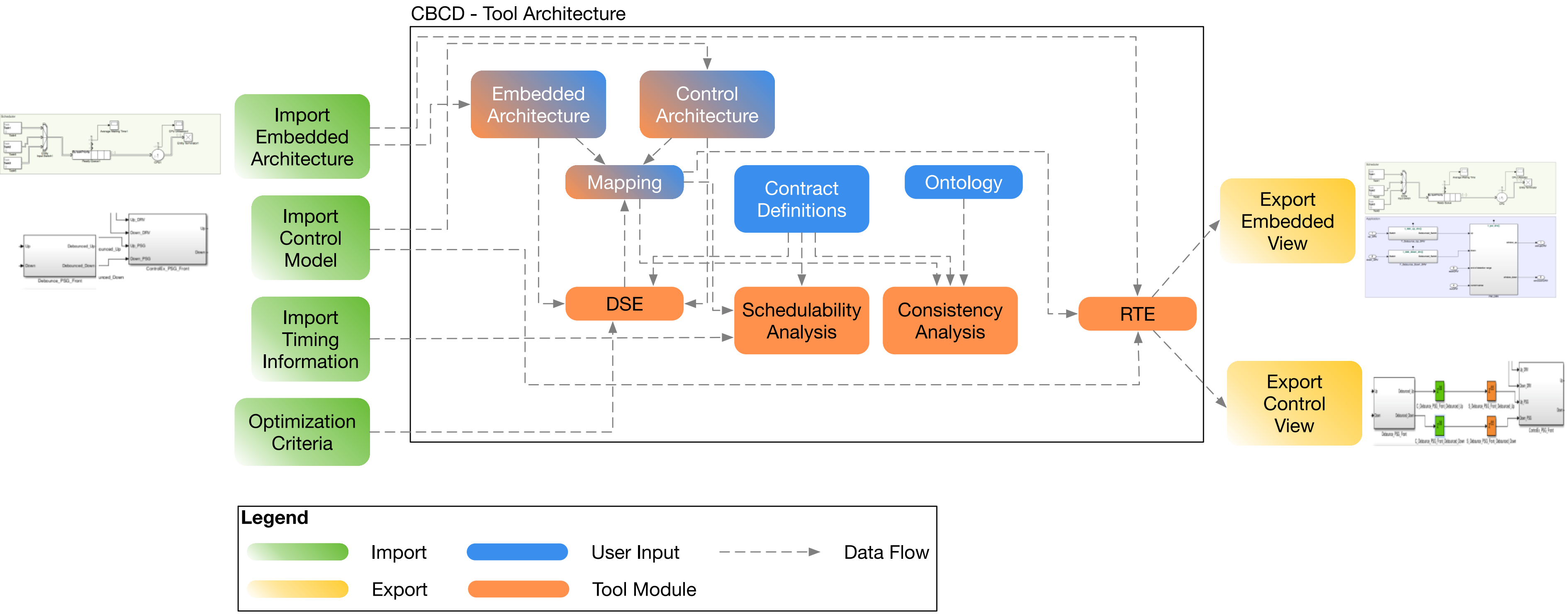
ken.vanherpen@uantwerpen.be

http://msdl.cs.mcgill.ca/people/ken/

School of Computer Science

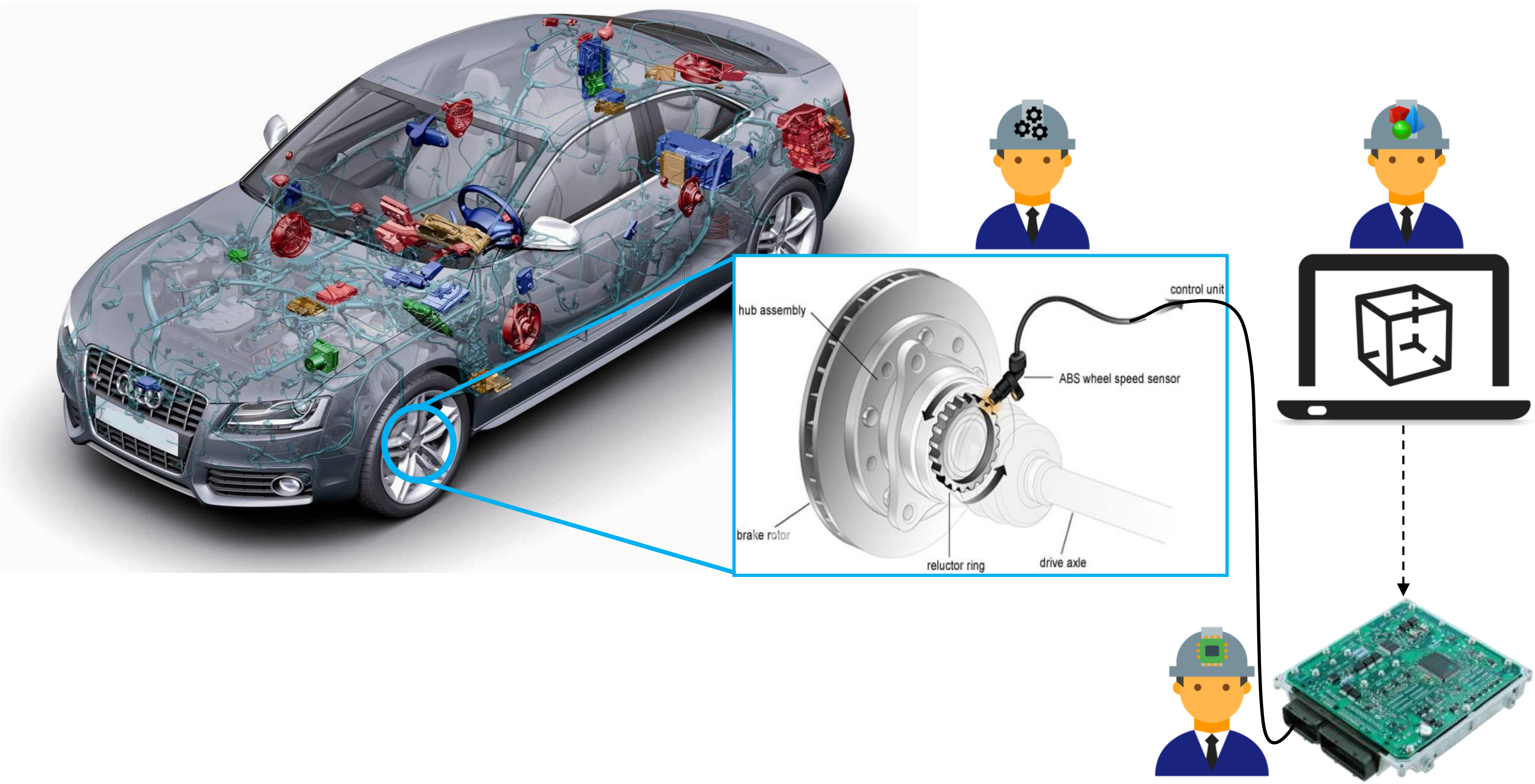
Ansymo
Antwerp Systems & Software Modelling
University of Antwerp

Summary

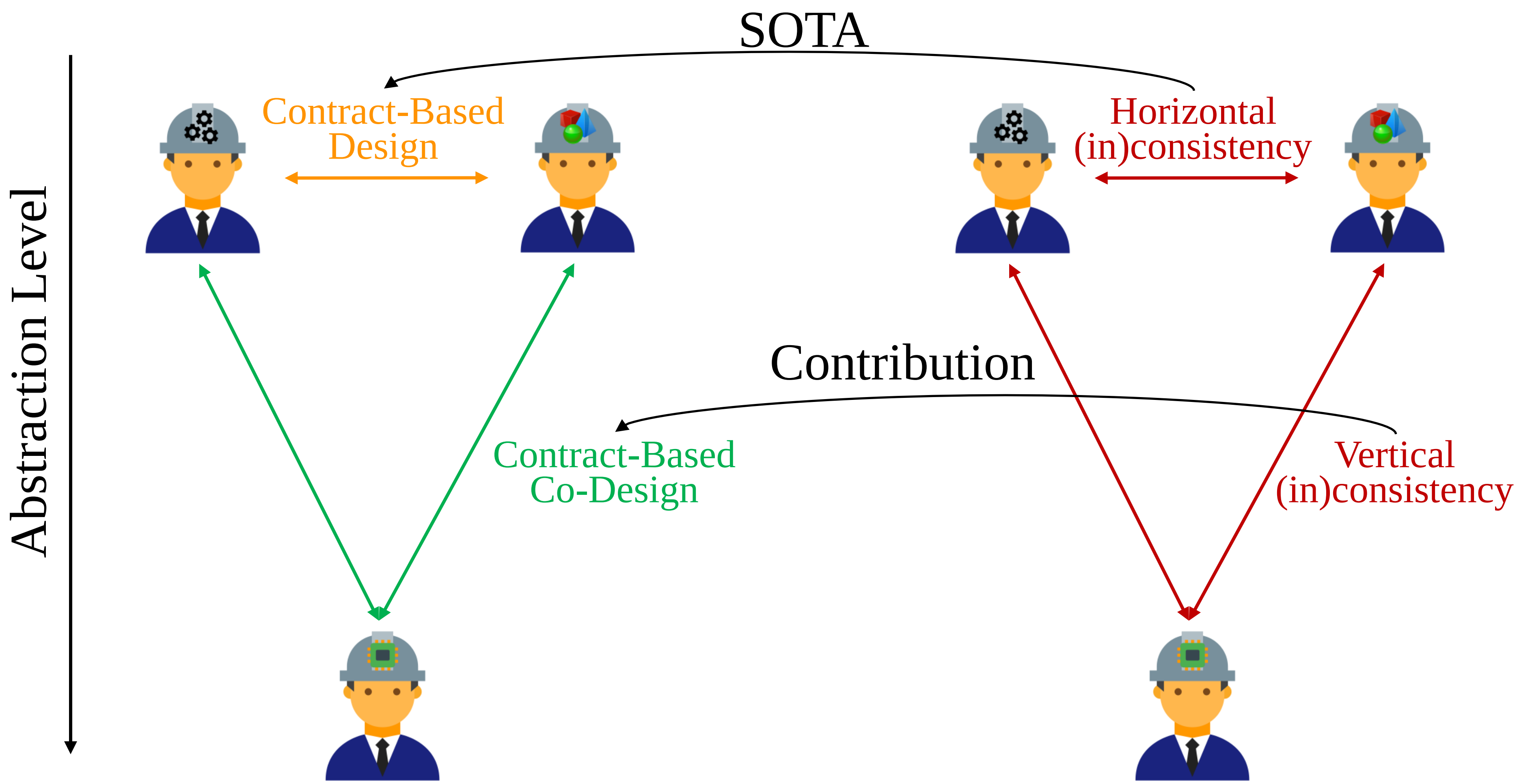


K. Vanherpen et al. Ontological Reasoning as an Enabler of Contract-Based Co-Design. CyPhy, 2016.

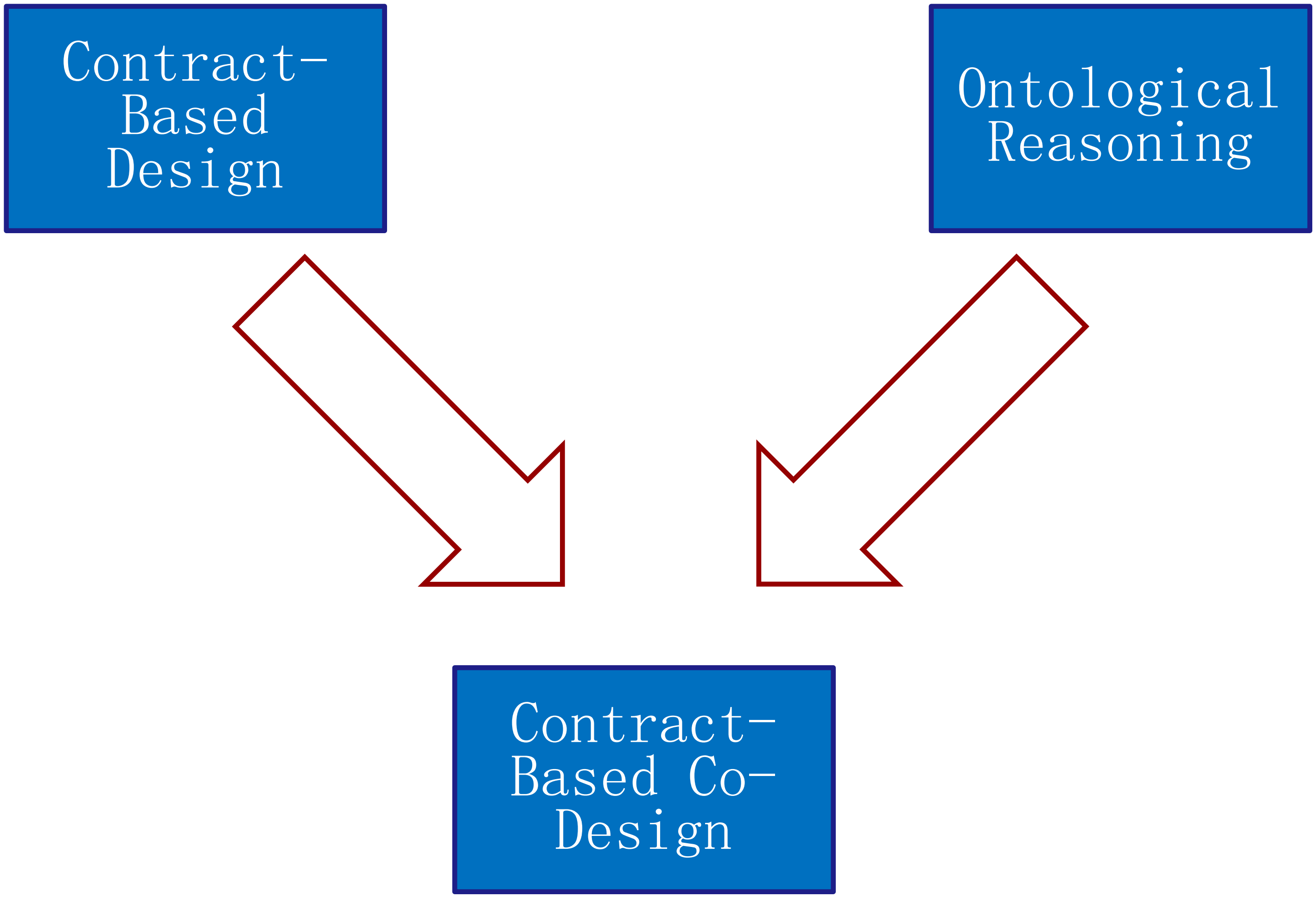
Problem Statement



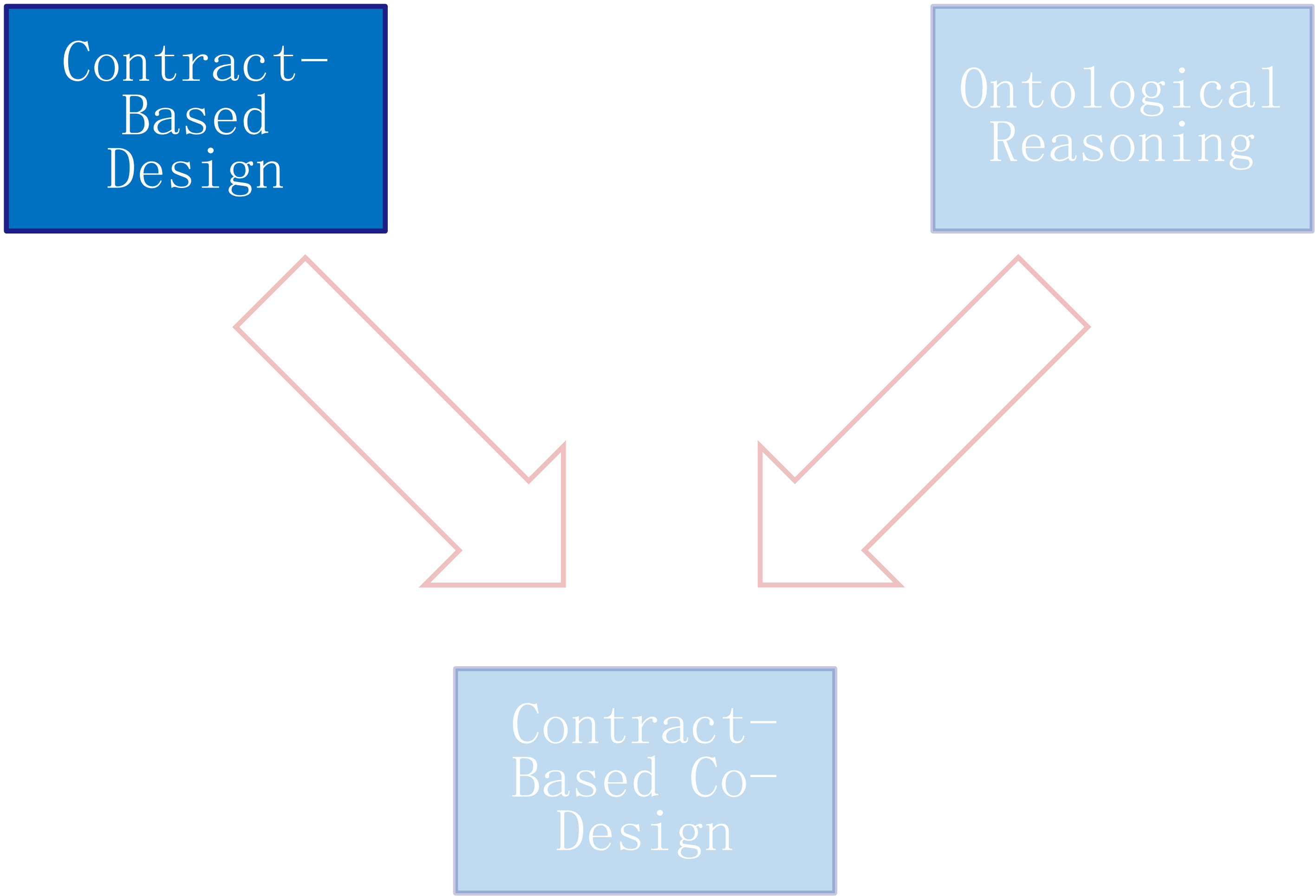
Contract-Based Co-Design (CBCD)



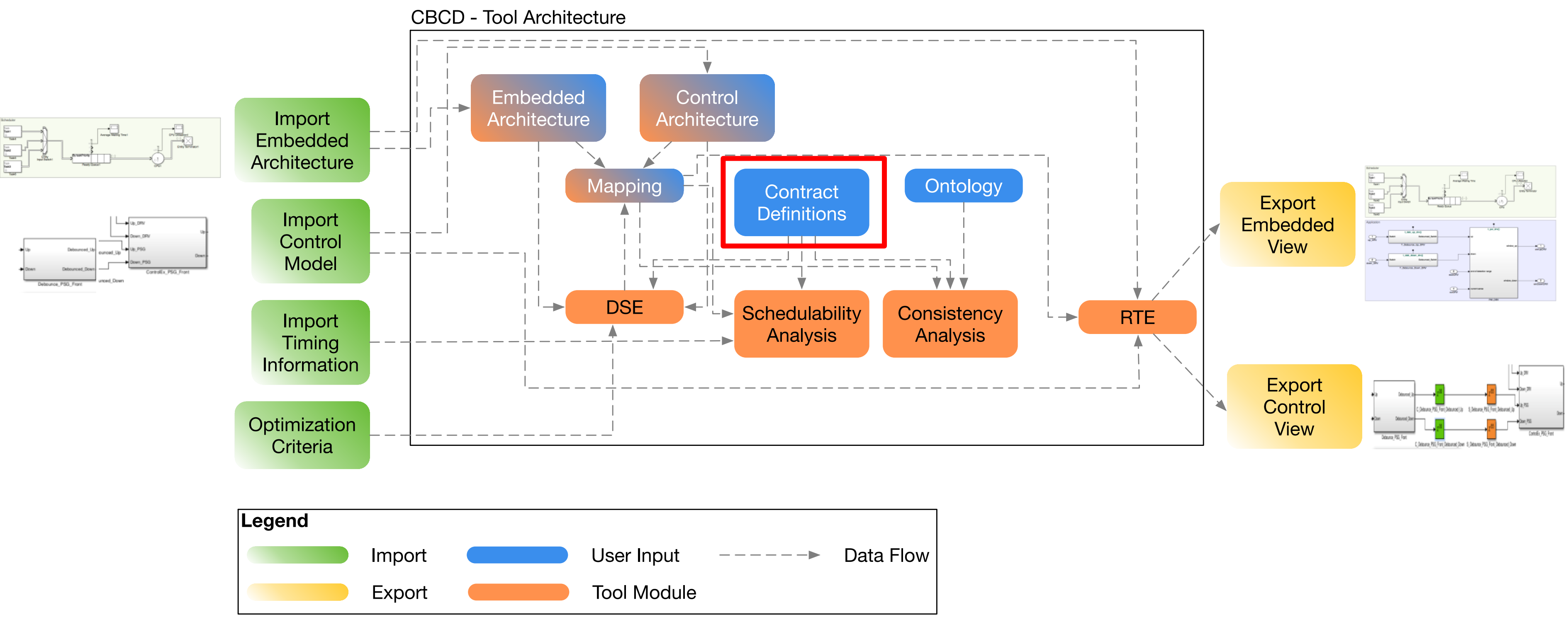
CBCD Theory


Contract-Based Design
Ontological Reasoning
Contract-Based Co-Design

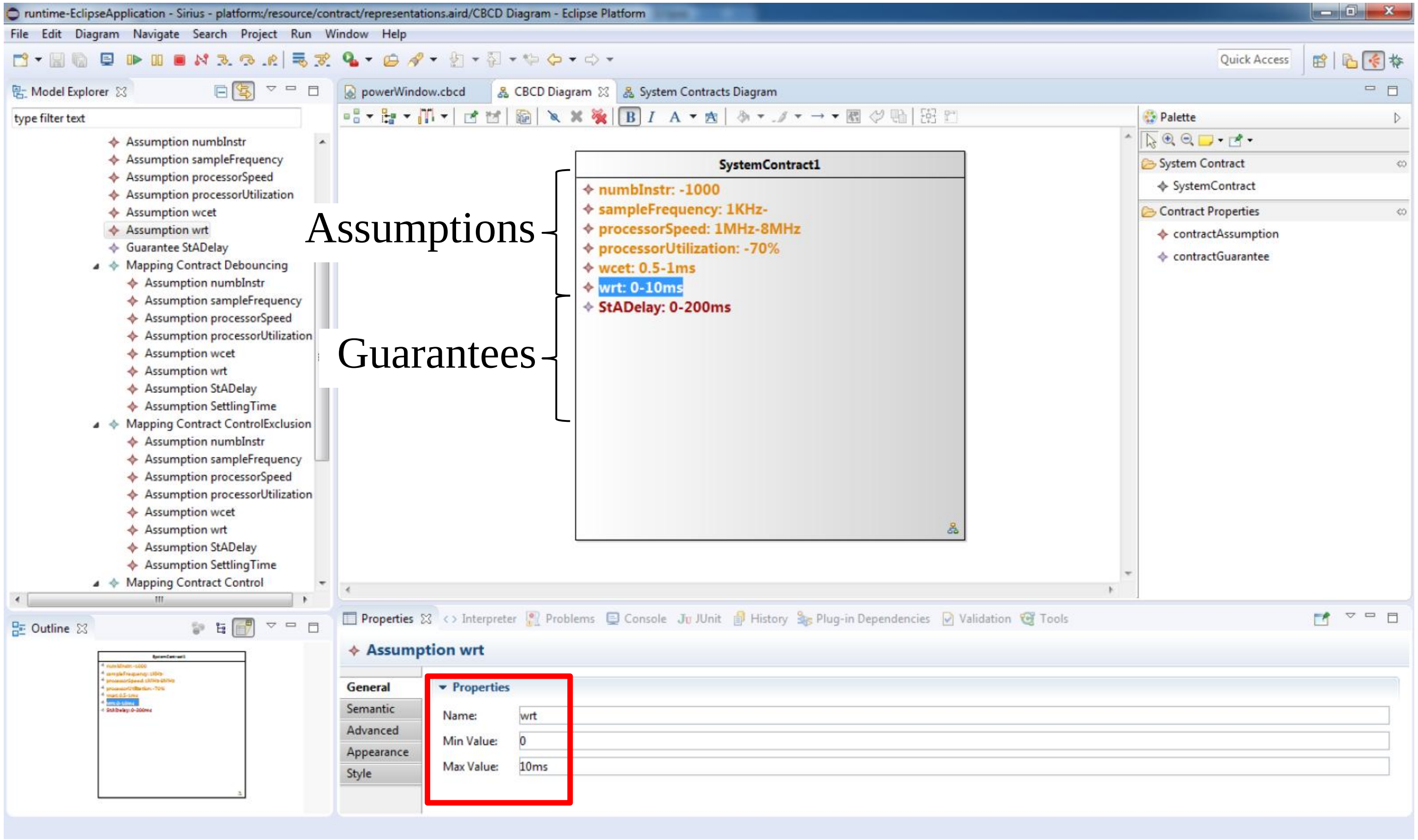
CBCD Theory


Contract-Based Design
Ontological Reasoning
Contract-Based Co-Design

CBCD Tool – Contract Definition



CBCD Tool – Contract Definition



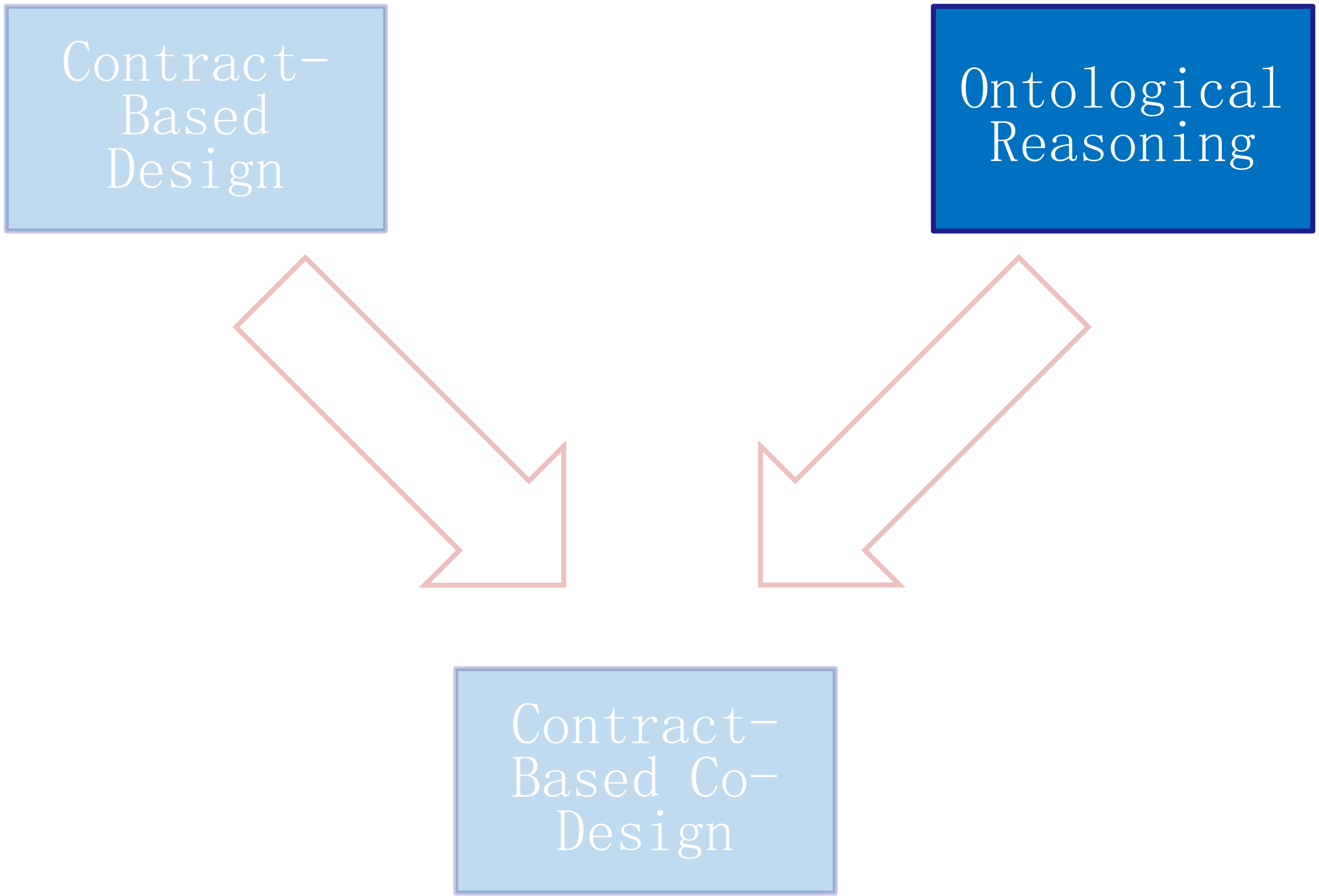
McGill School of Computer Science

neesis

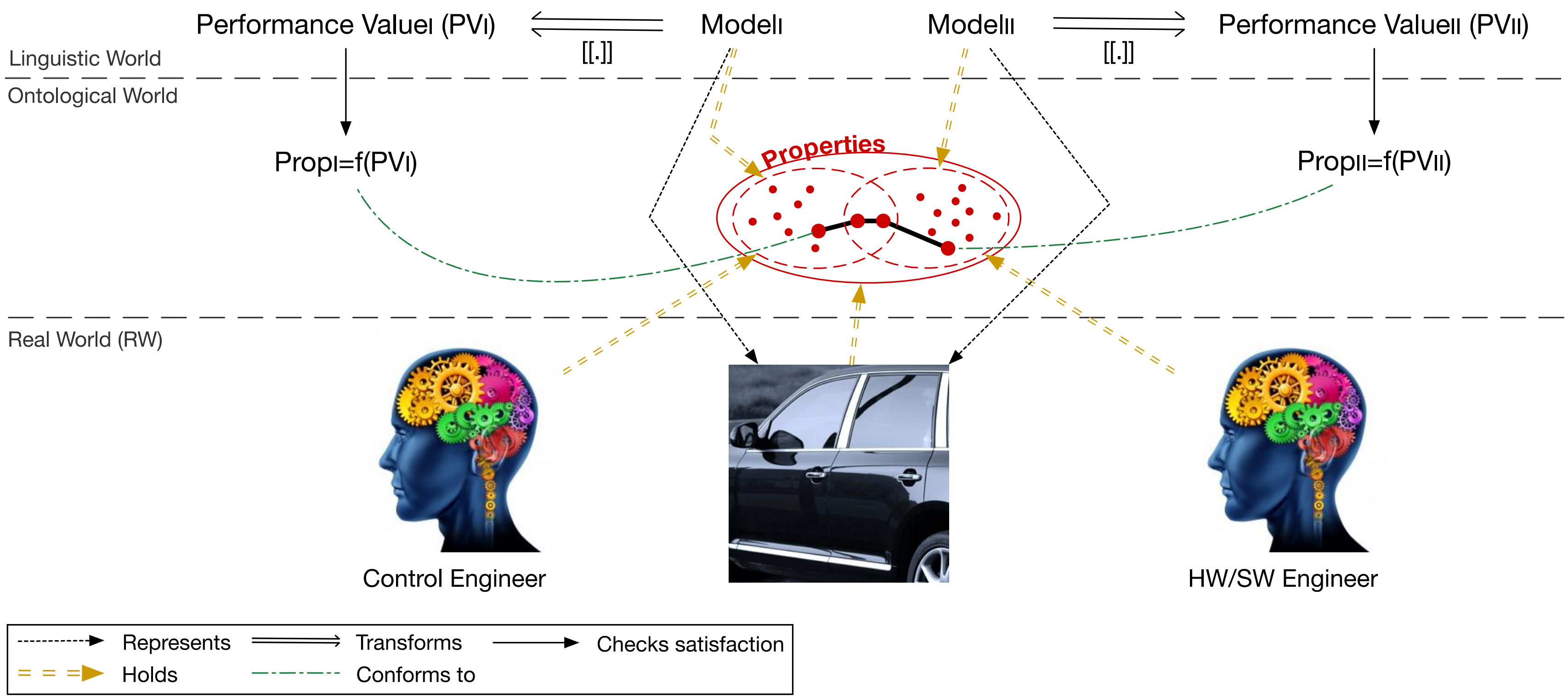
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

Ansymo Antwerp Systems & Software Modelling University of Antwerp

CBCD Theory – Ontological Reasoning

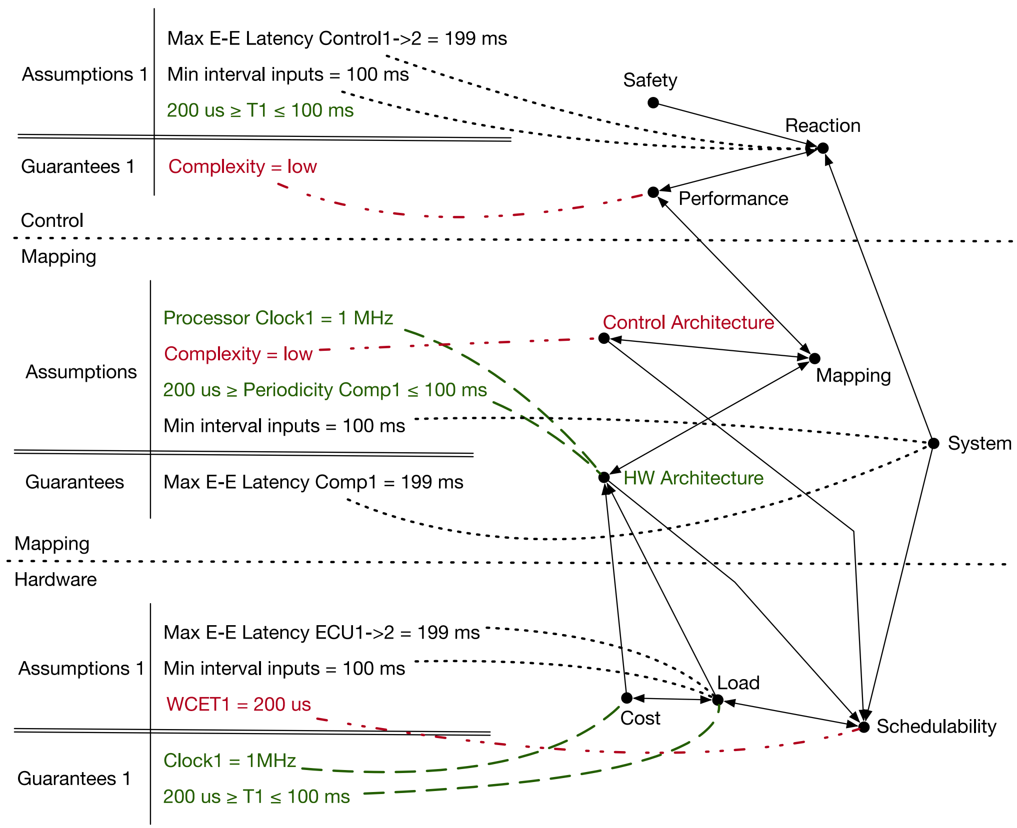


CBCD Theory – Ontological Reasoning



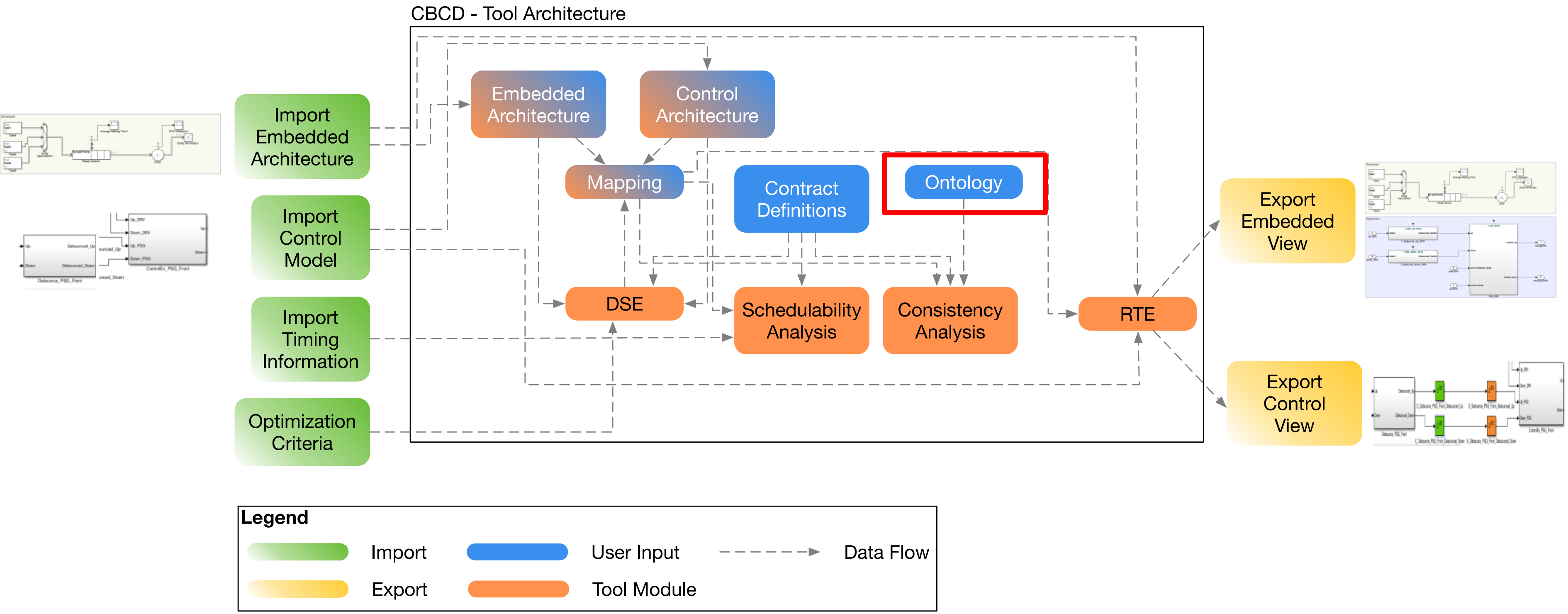
K. Vanherpen et al. Ontological Reasoning for Consistency in the Design of Cyber-Physical Systems. CPPS, 2016.

CBCD Theory – Ontological Reasoning



K. Vanherpen et al. Ontological Reasoning as an Enabler of Contract-Based Co-Design. CyPhy, 2016.

CBCD Tool – Ontology

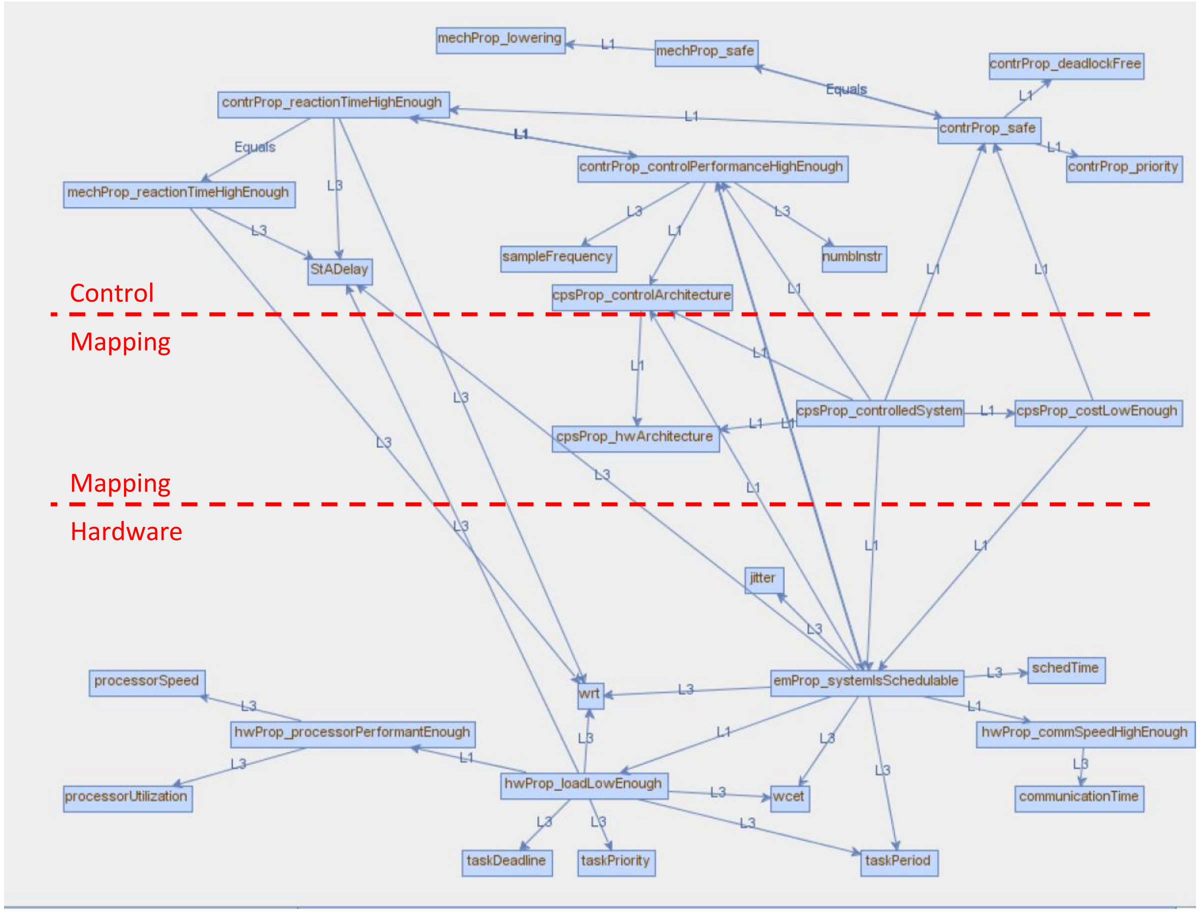


McGill School of Computer Science

Flanders Make Manufacturing Innovation Network

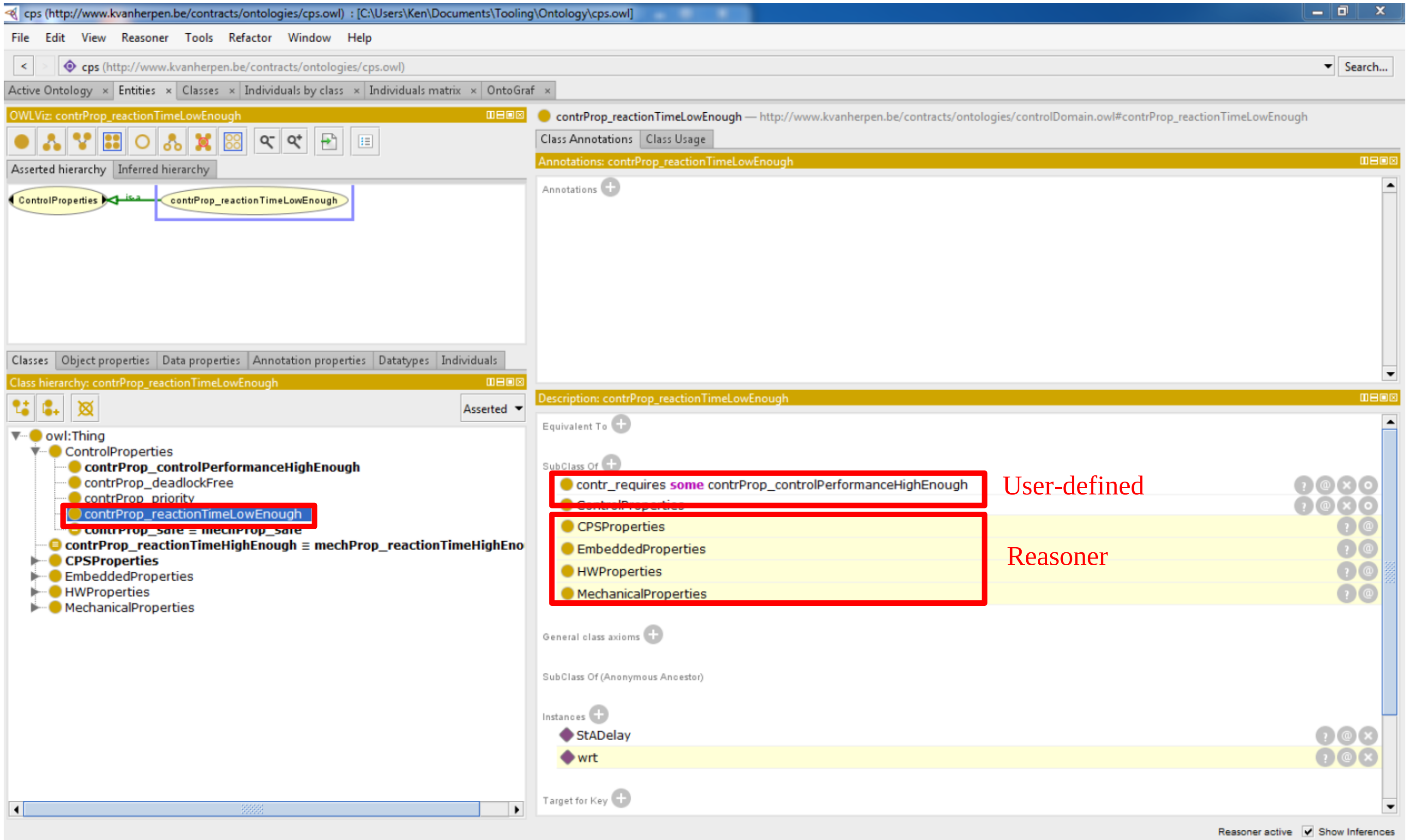
Ansymo Antwerp Systems & Software Modelling University of Antwerp

CBCD Tool – Ontology

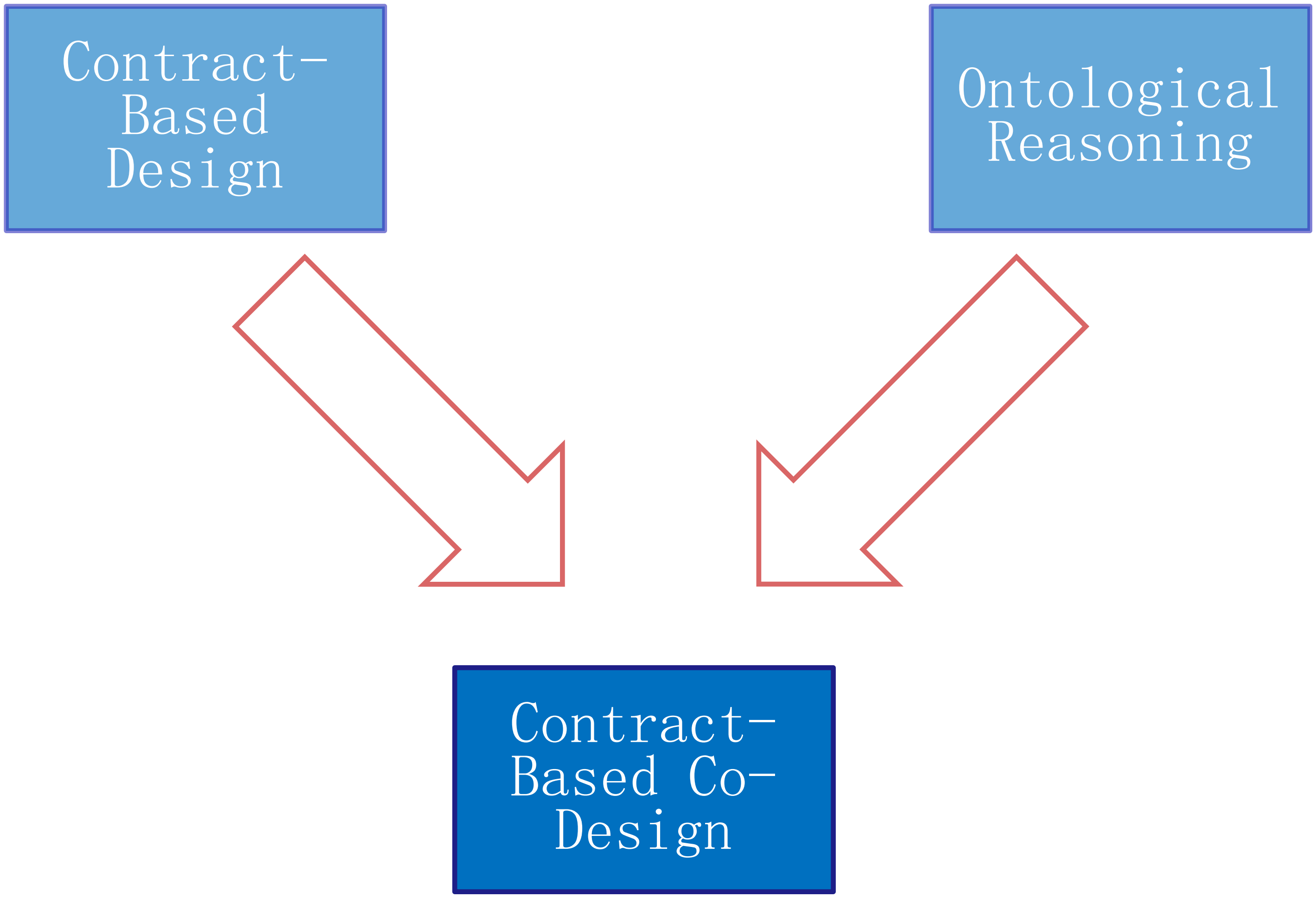


McGill School of Computer Science
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

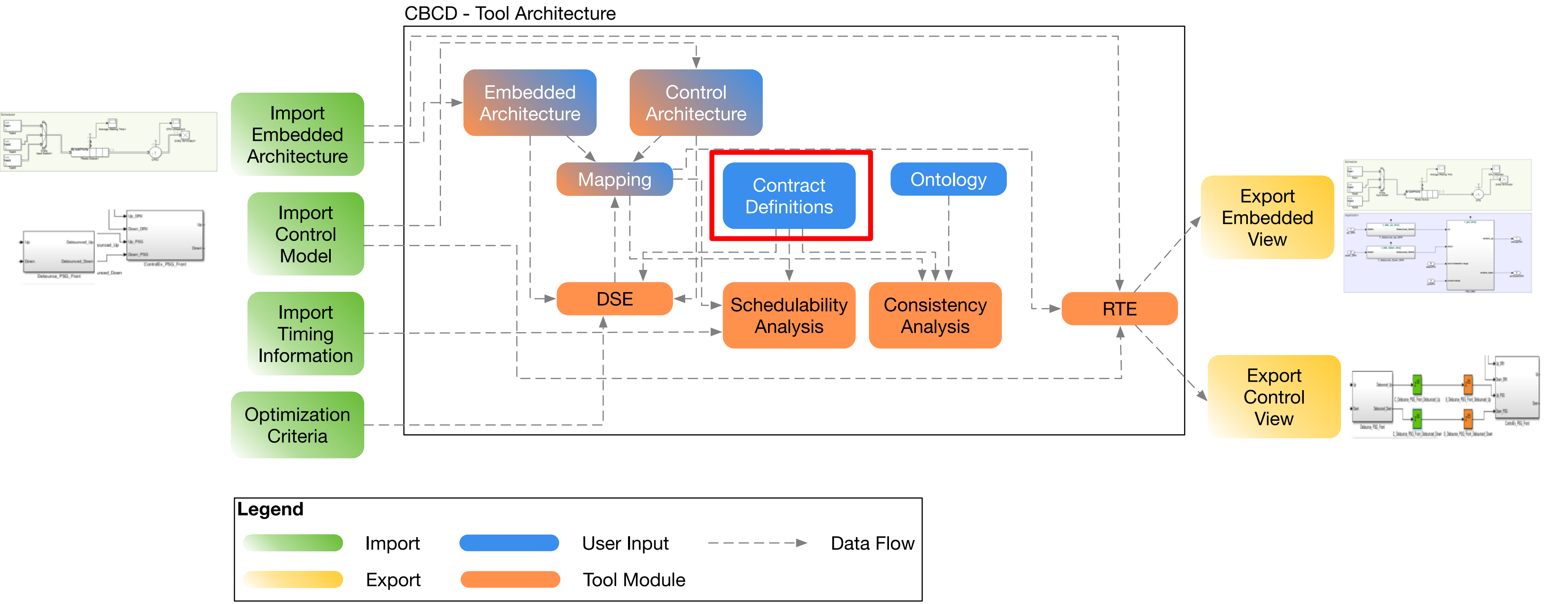
CBCD Tool – Ontology



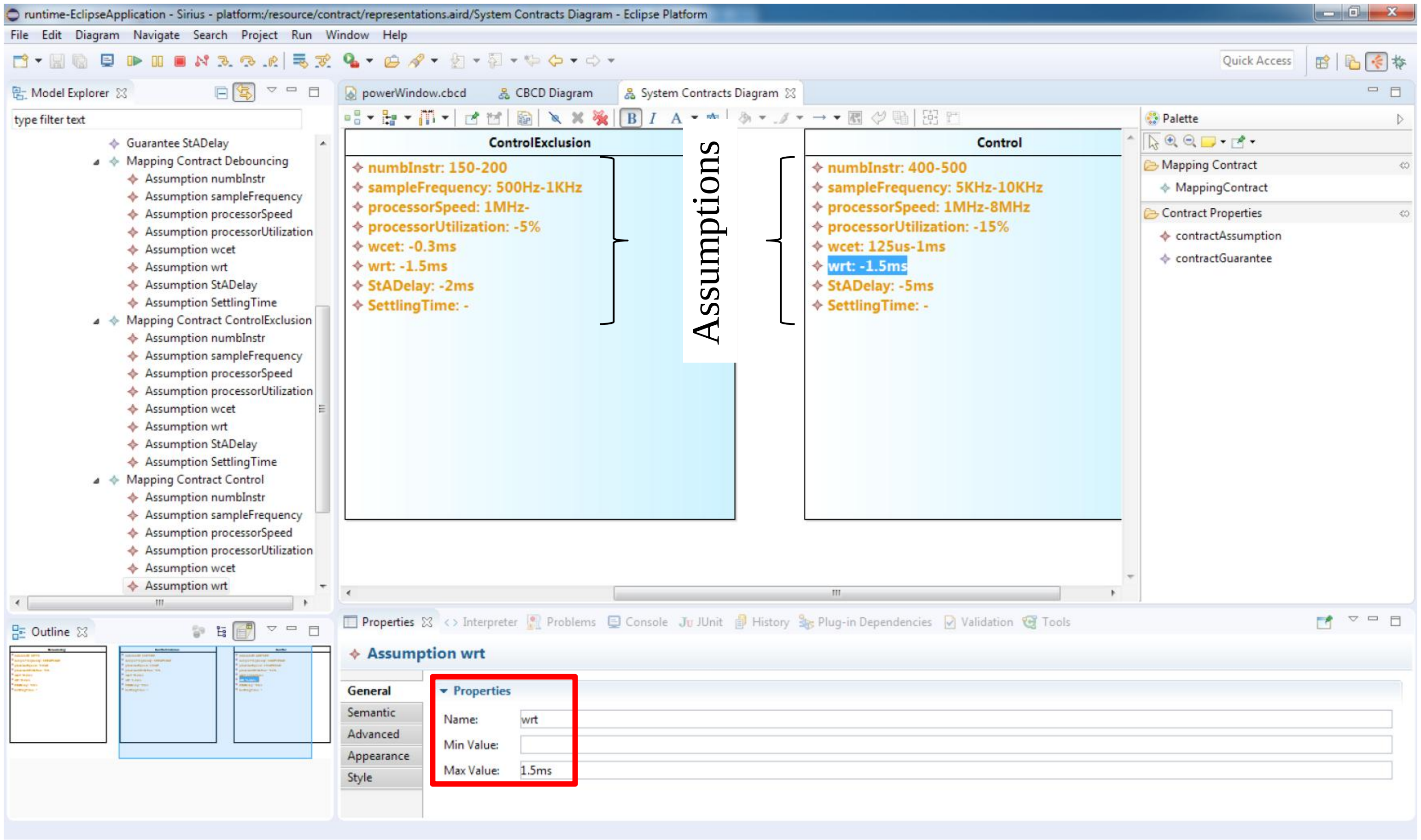
CBCD Theory


Contract-Based Design
Ontological Reasoning
Contract-Based Co-Design

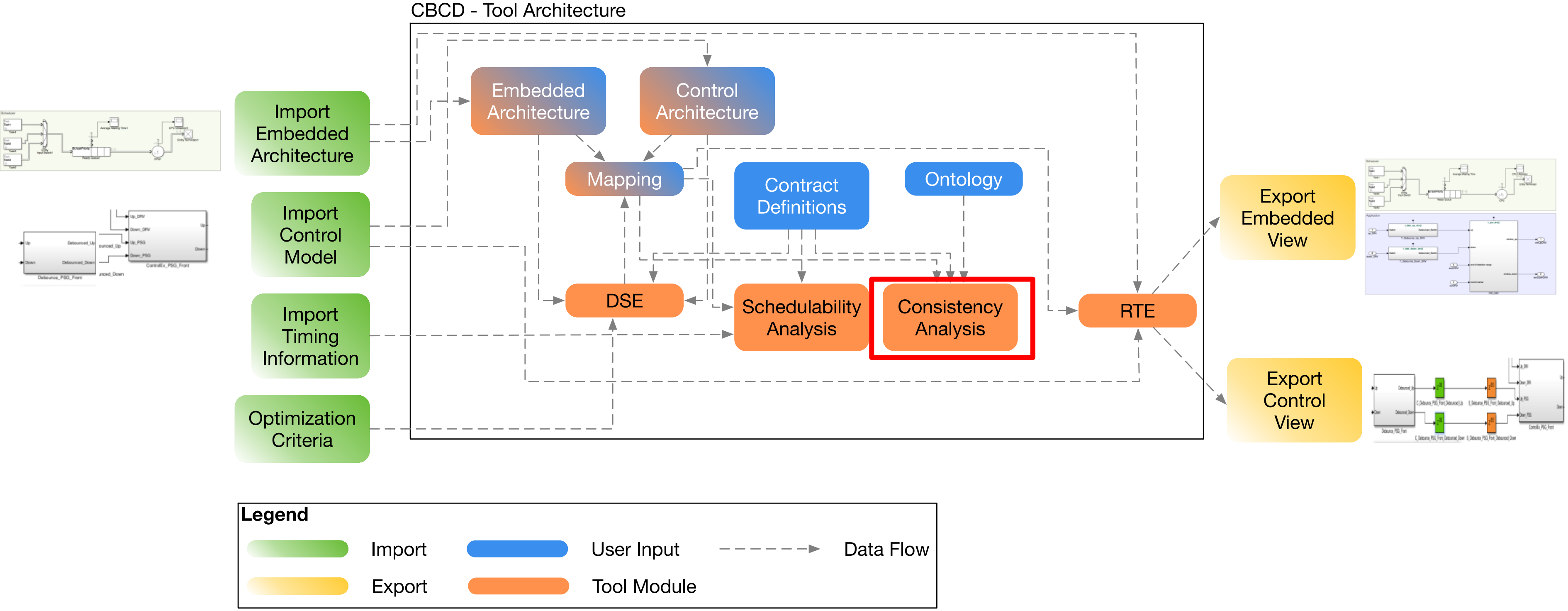
CBCD Tool – Contract Definition



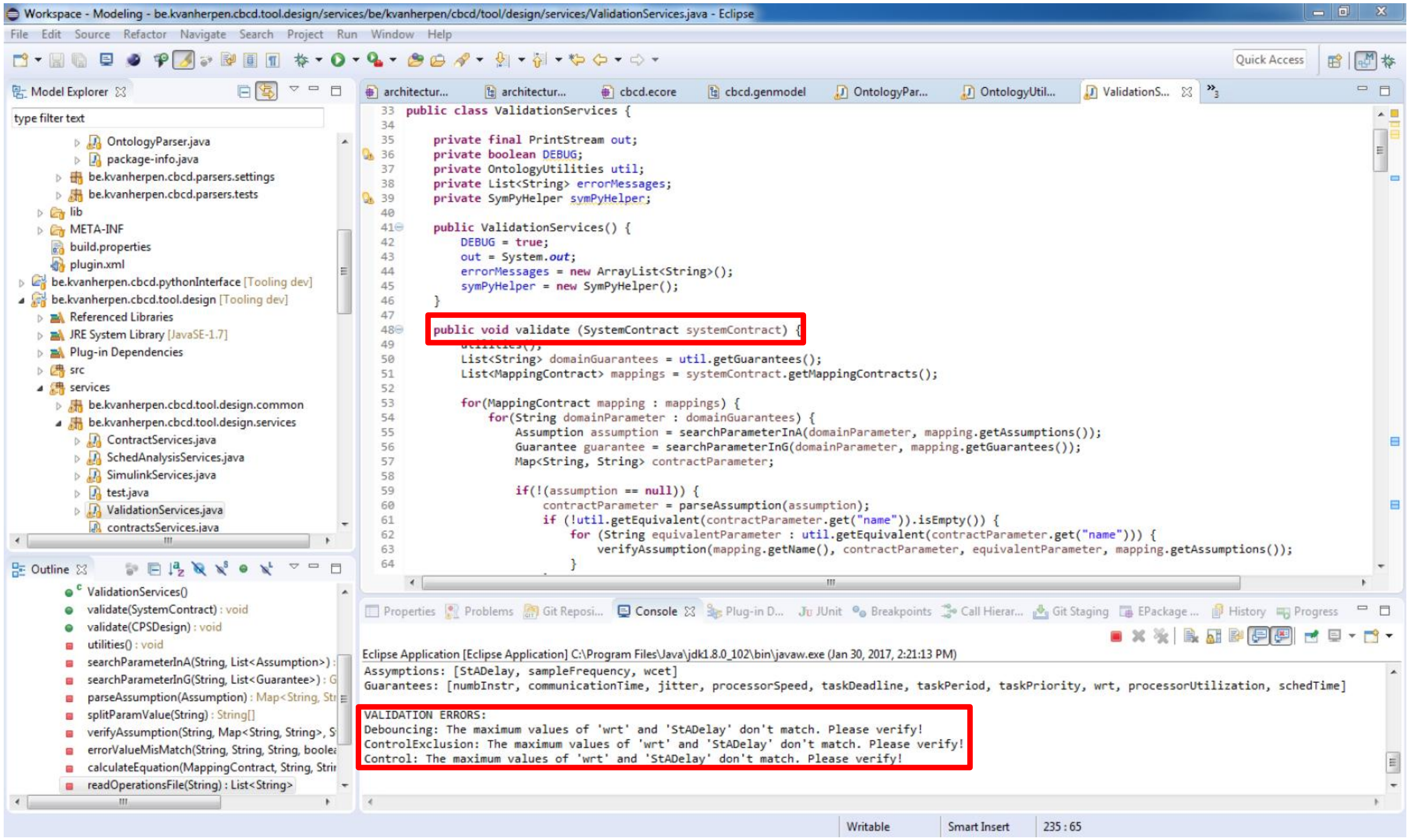
CBCD Tool – Mapping Contract



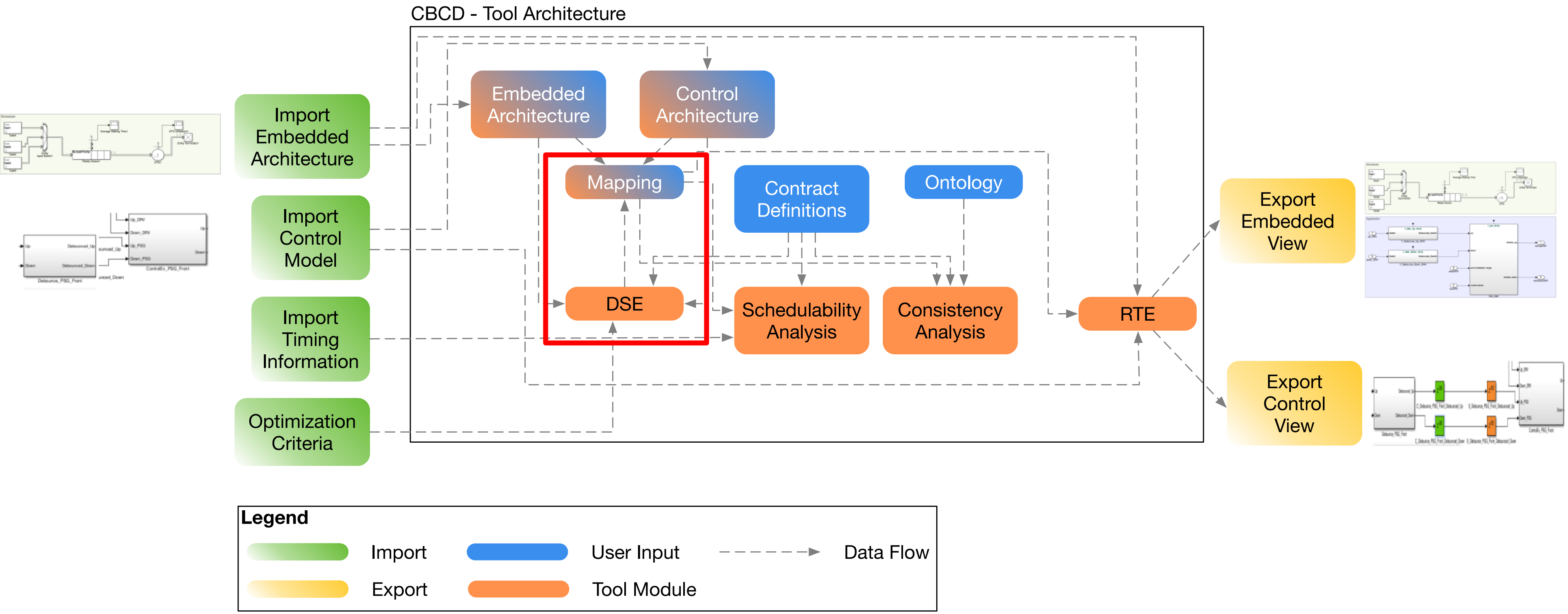
CBCD Tool – Contract Validation Analysis



CBCD Tool – Contract Validation Analysis



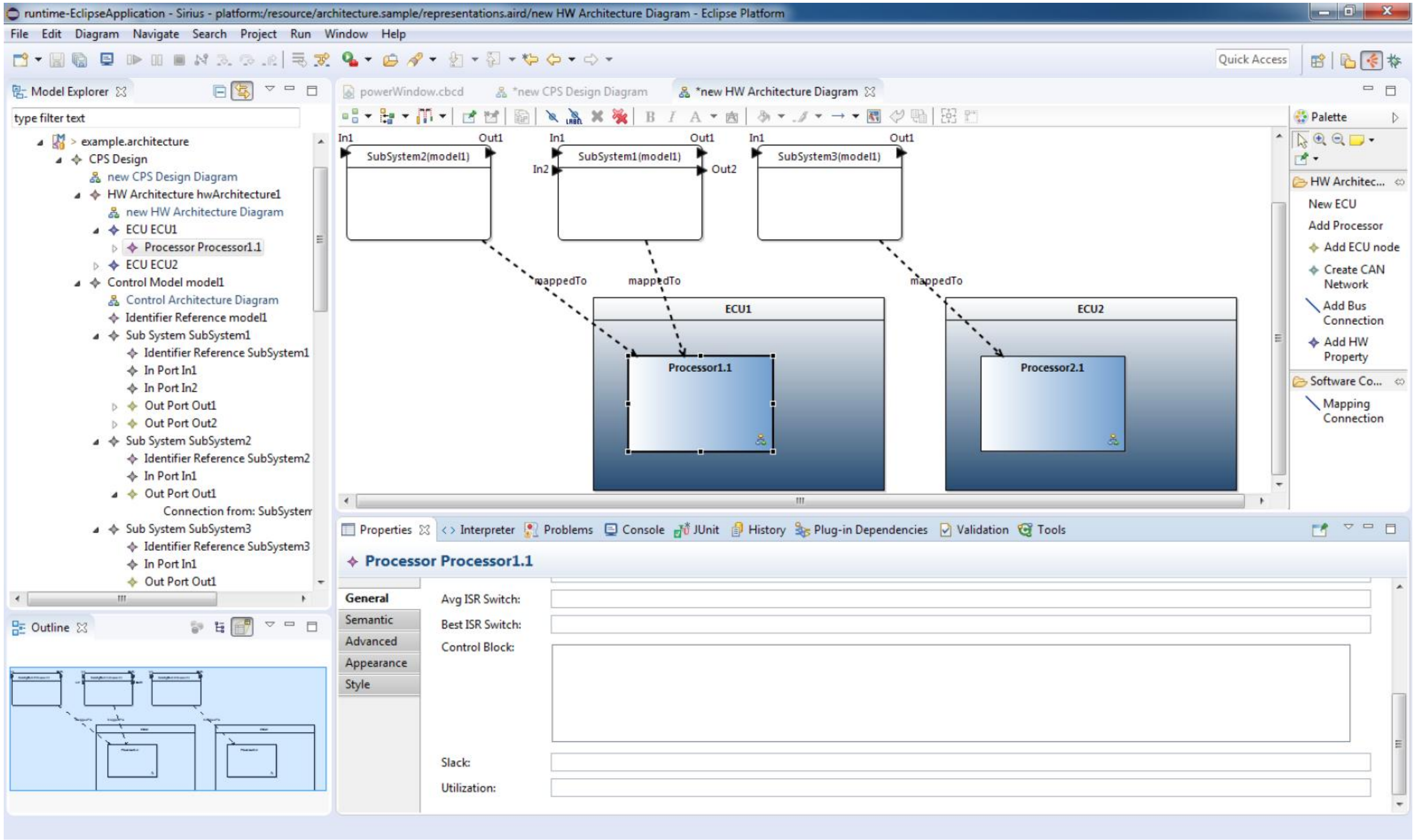
CBCD Tool – (DSE) Mapping



McGill School of Computer Science
nessis

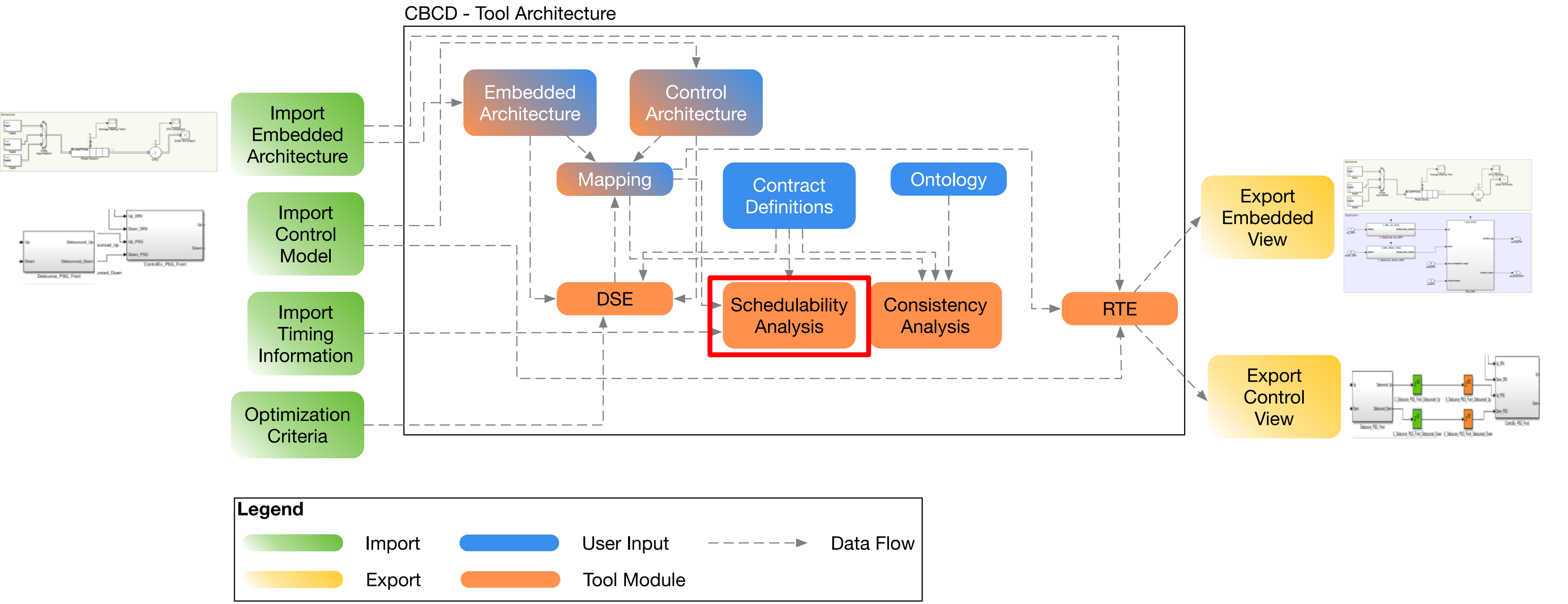
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

CBCD Tool – (DSE) Mapping

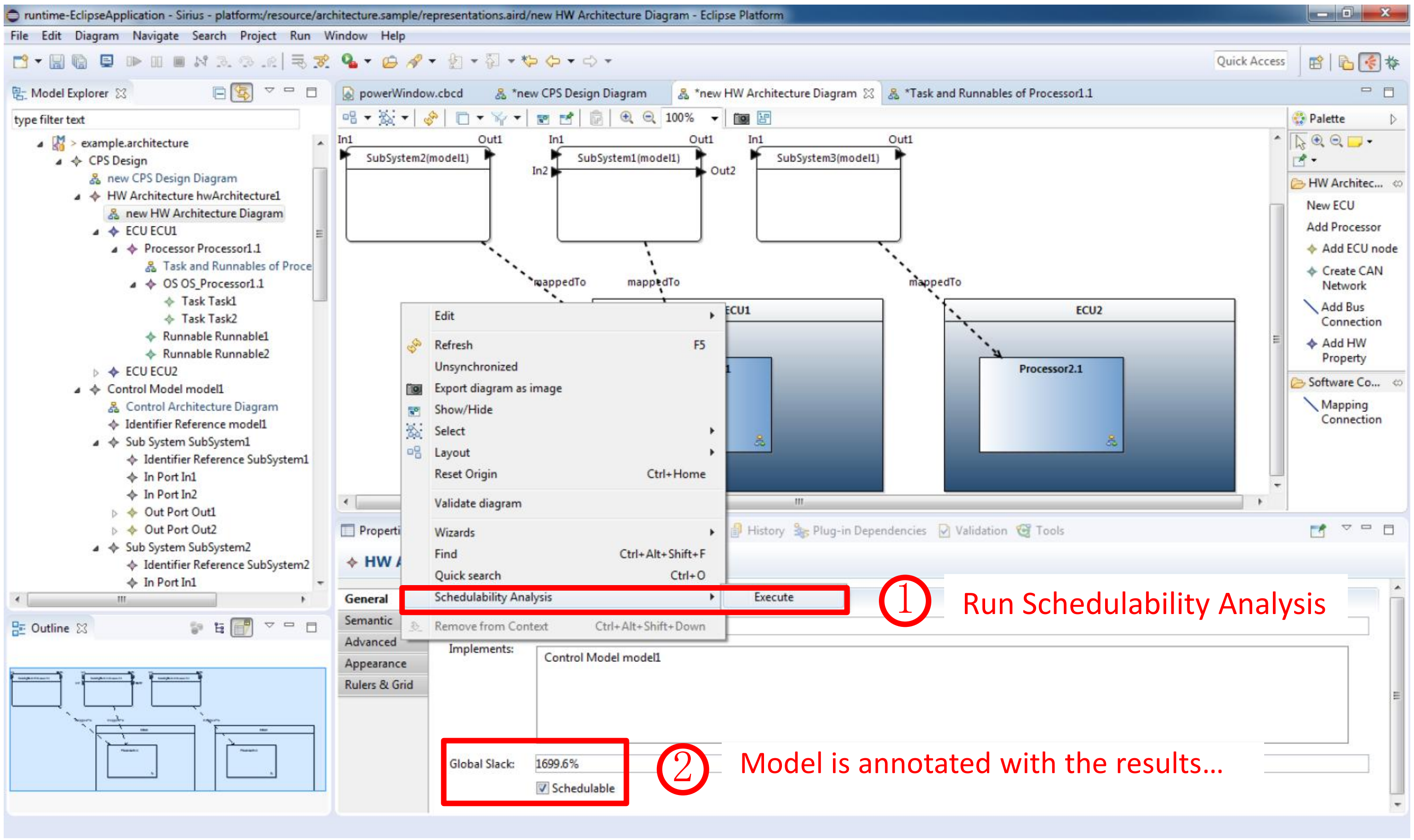


McGill School of Computer Science | necsis | FLANDERS MAKE MANUFACTURING INNOVATION NETWORK | Ansymo Antwerp Systems & Software Modelling University of Antwerp

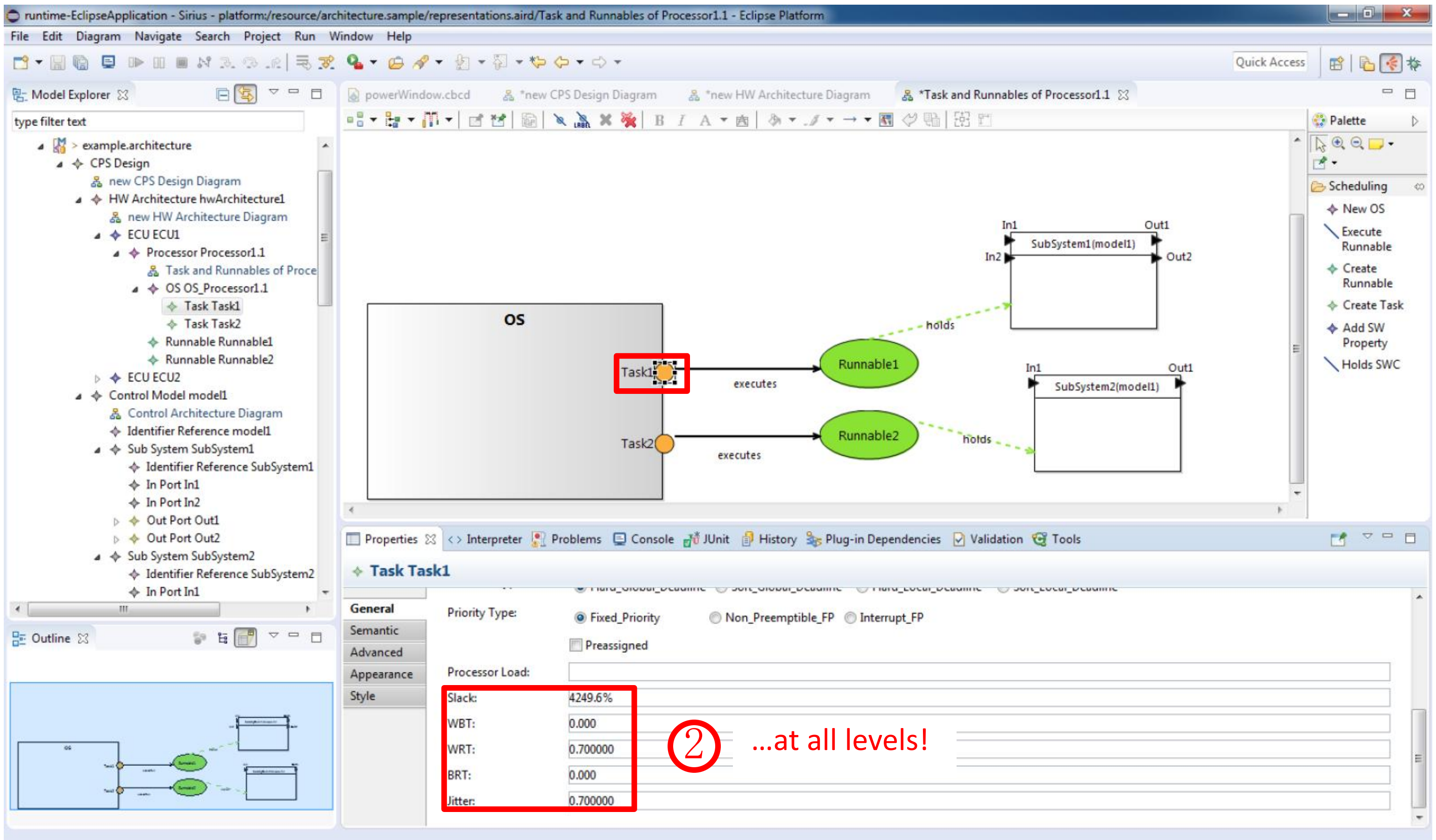
CBCD Tool – Schedulability Analysis



CBCD Tool – Schedulability Analysis



CBCD Tool – Schedulability Analysis

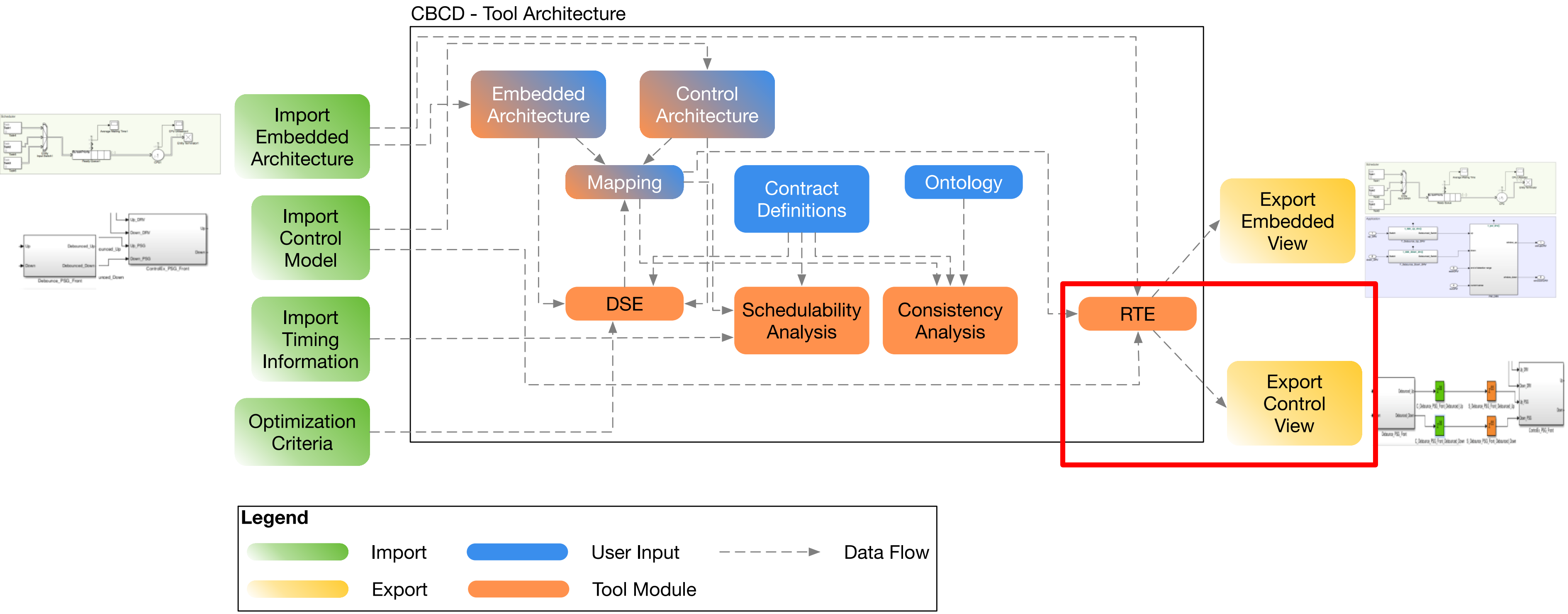


McGill School of Computer Science

Flanders Make — Manufacturing Innovation Network

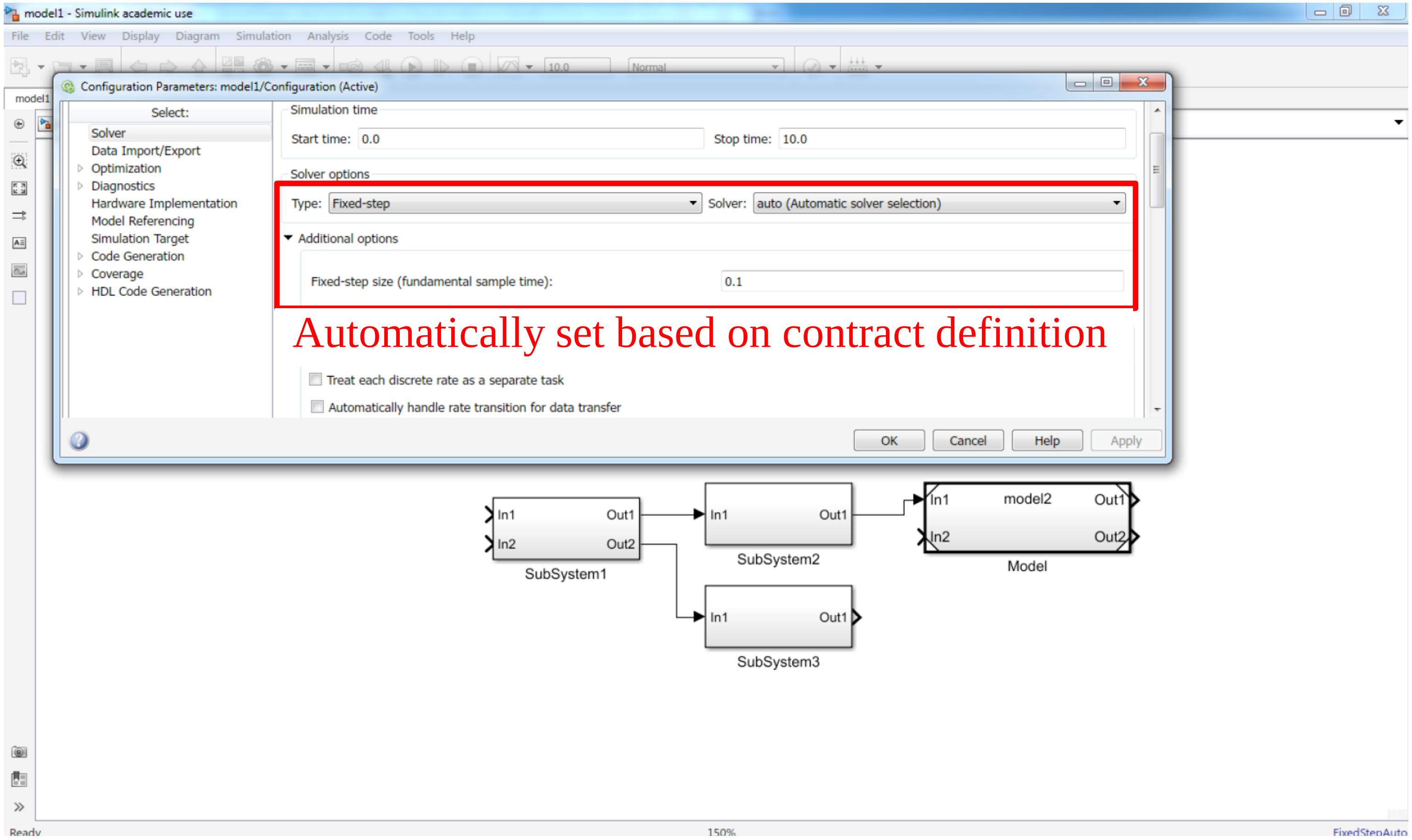
Ansymo — Antwerp Systems & Software Modelling — University of Antwerp

CBCD Tool – Export (Control) View



McGill School of Computer Science · nec$\!$sis · FLANDERS MAKE MANUFACTURING INNOVATION NETWORK · Ansymo Antwerp Systems & Software Modelling University of Antwerp

CBCD Tool – Export (Control) View



CBCD Tool – Export (Control) View

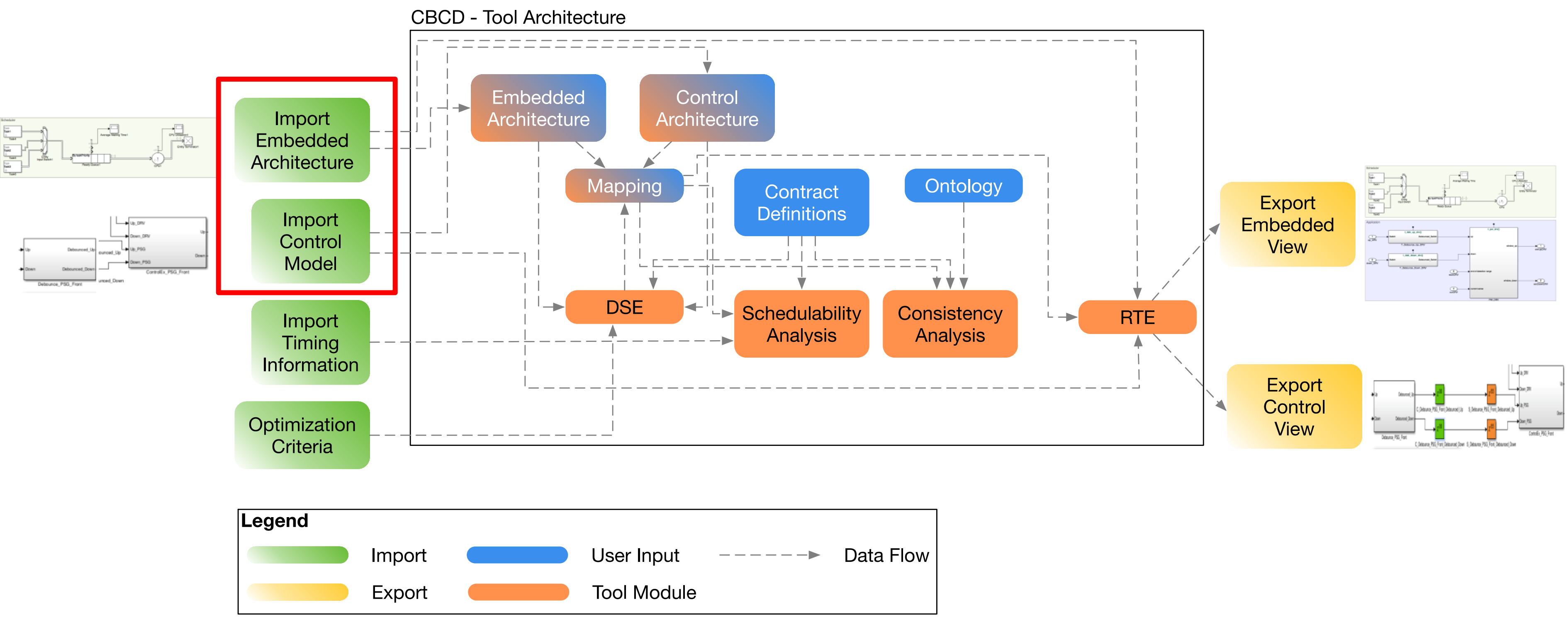
Annotating/updating a Simulink model with hardware properties:

K. Vanherpen, J. Denil, H. Vangheluwe, P. De Meulenaere, Model Transformations for Round-Trip Engineering in Control-Deployment Co-Design. Mod4Sim, 2015.

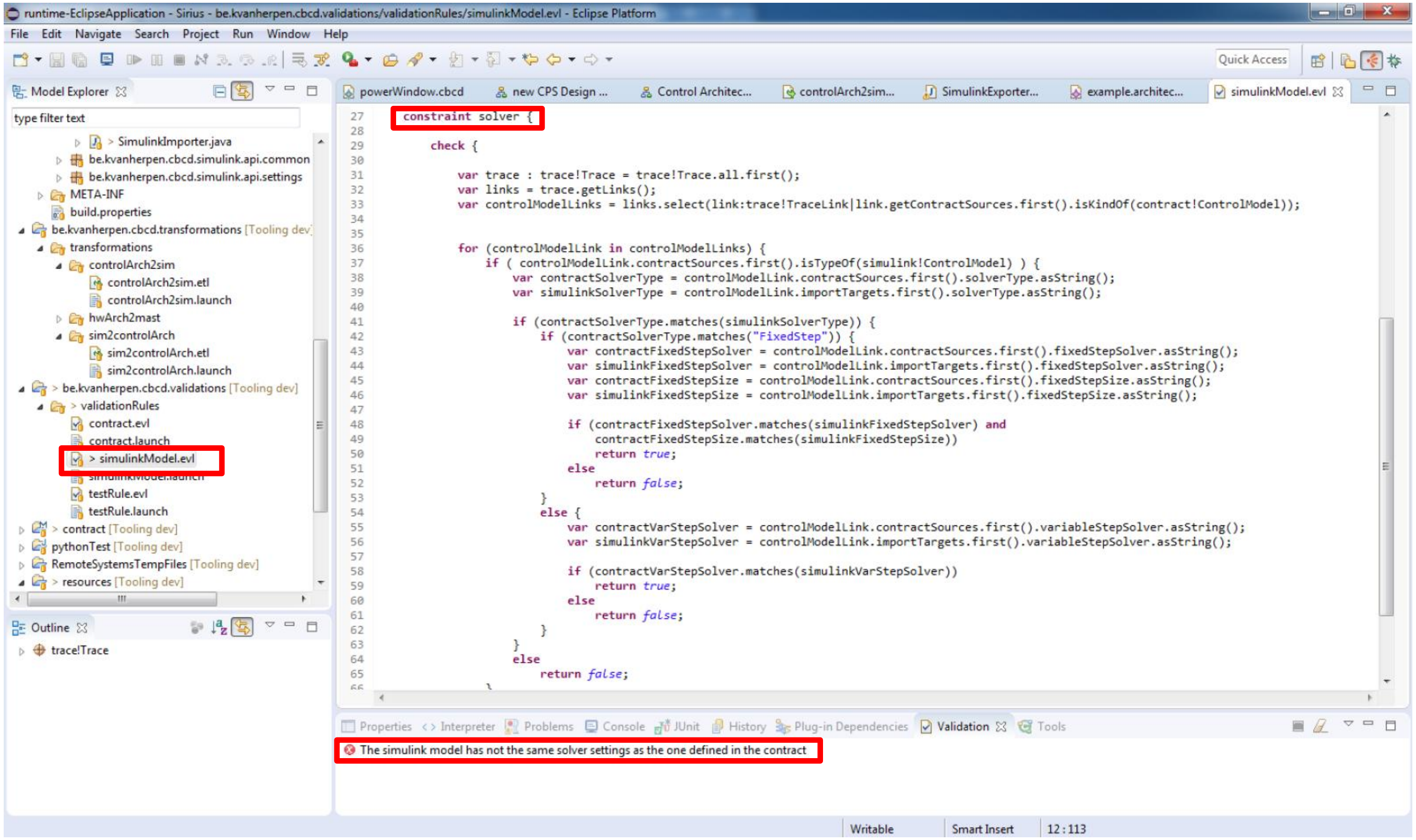
McGill School of Computer Science

Ansymo Antwerp Systems & Software Modelling University of Antwerp

CBCD Tool – Import (Control) View



CBCD Tool – Model Validation Analysis



Roadmap

- Support for horizontal contracts
- Enable composition and conjunction of contracts
- Inconsistency Management combined with Contract-Based Design
- RTE for embedded co-design view
- Sensitivity Analysis
- Contract Management
- Link with validity frames
- …

Variability for Controller Design

Bart Meyers

Universiteit Antwerpen

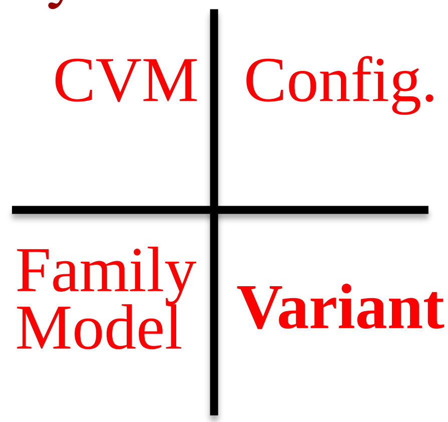
bart.meyers@uantwerpen.be

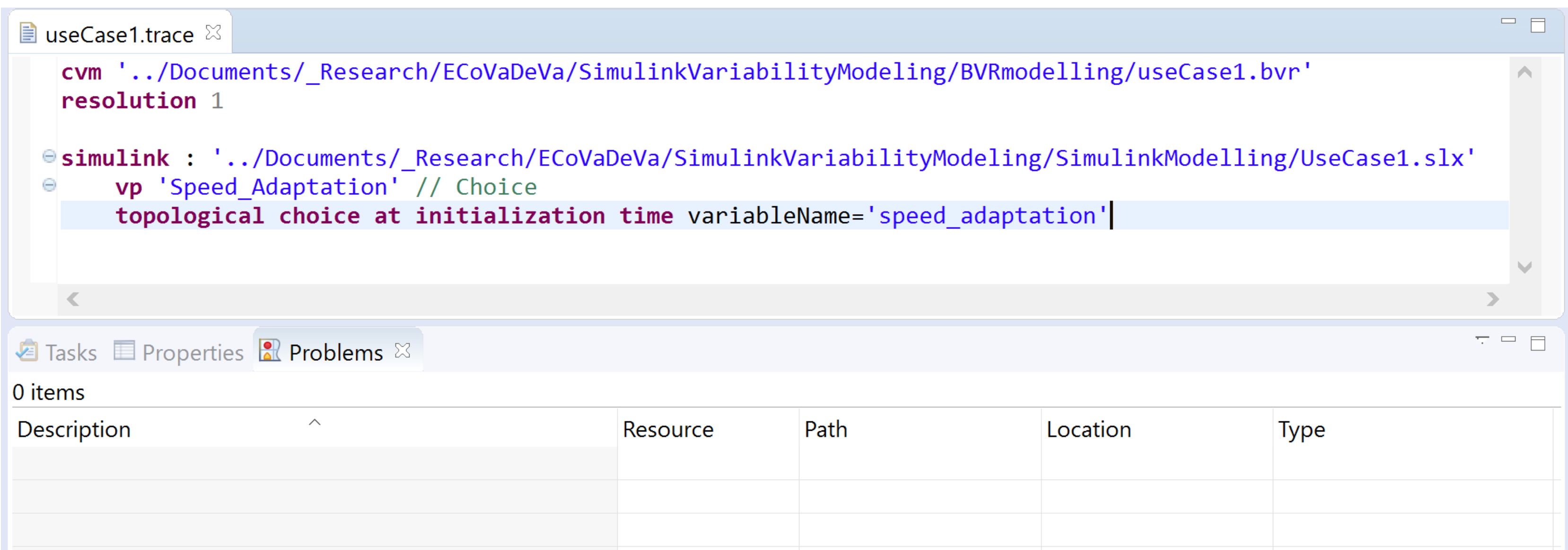
Summary

Variants in controller design

Ultimate goal:

- Generation of variants from:
  - Central variability model
  - Configuration
  - Family model
- Traceability tool to link all artefacts:



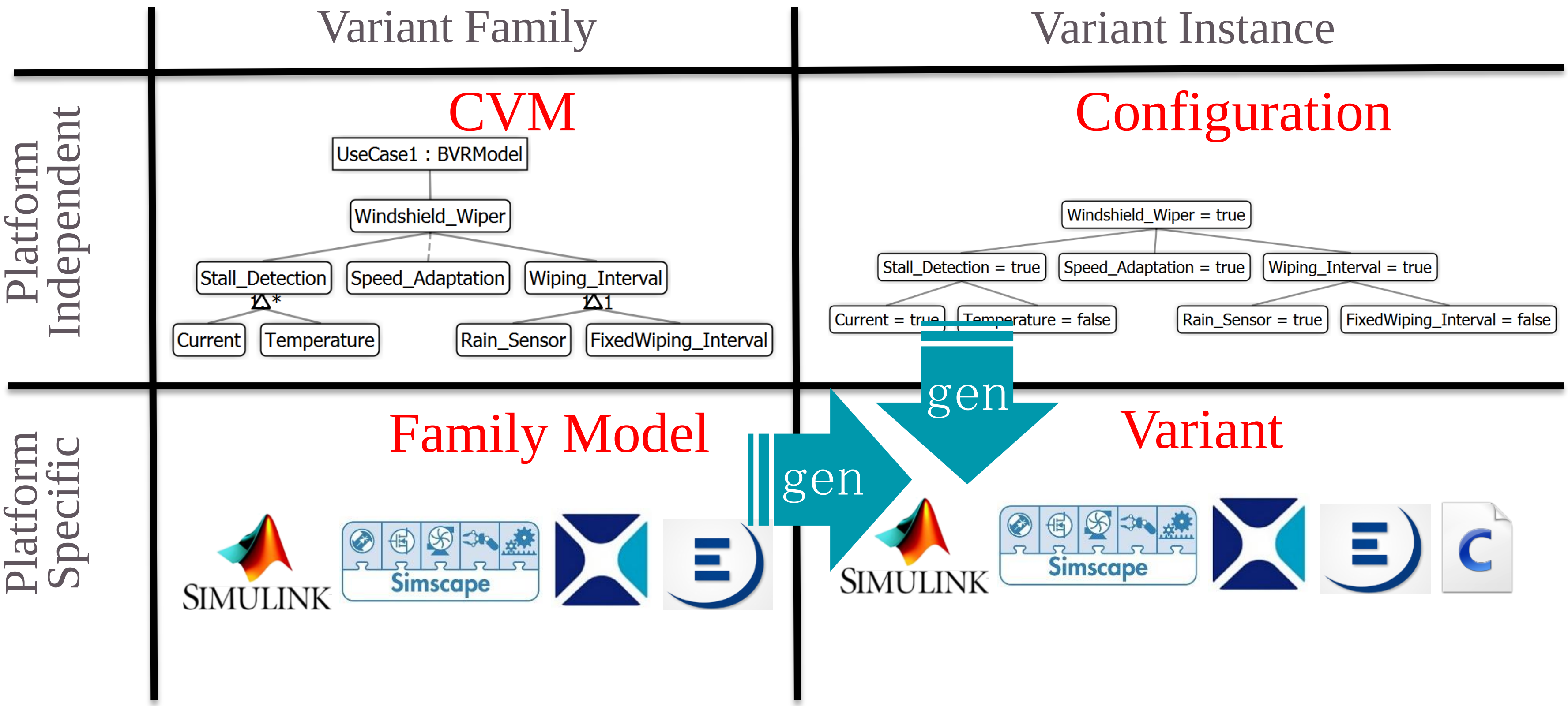


McGill School of Computer Science

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

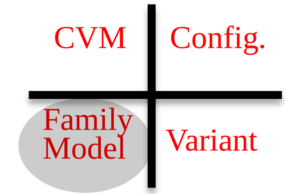
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Variability

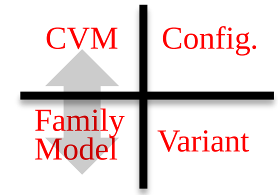


UA Tasks in ECoVaDeVa Project

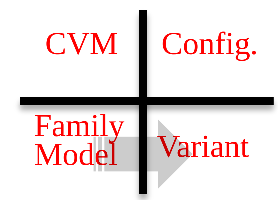
- Variability modeling in acausal models
  - Amesim/System Synthesis
  - Simscape
  - Modelica



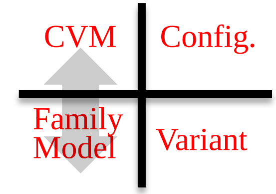
- Linking features to Variation Points in different tools
  - Necessary for variant generation
  - May generate links between configuration and variant
  - Correctness check



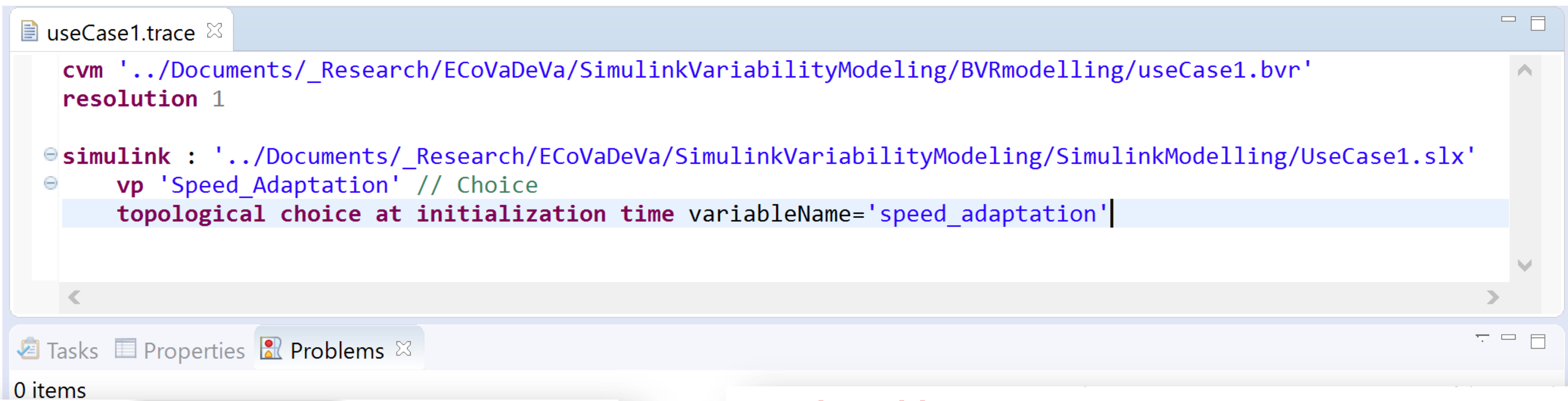
- Model transformation tool for Simulink and Amesim
  - Generate variant at model configuration time
  - Especially interesting for non-150% approaches



Traceability Tool

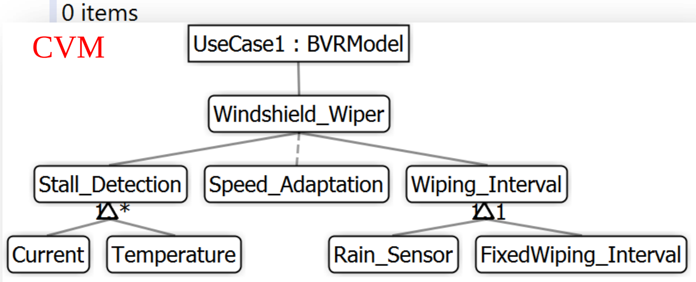


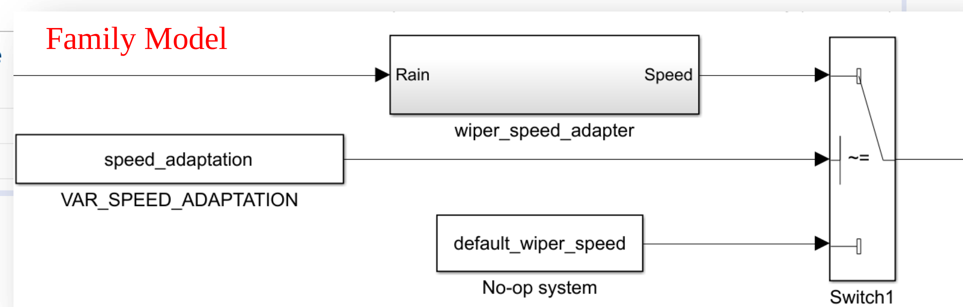
- Textual tool in Xtext
  - Accesses BVR model
  - Accesses Simulink model (to do: other types of models like SimScape, Amesim, EXAM, …)




```
cvm '../Documents/_Research/ECoVaDeVa/SimulinkVariabilityModeling/BVRmodelling/useCase1.bvr'
resolution 1

simulink : '../Documents/_Research/ECoVaDeVa/SimulinkVariabilityModeling/SimulinkModelling/UseCase1.slx'
    vp 'Speed_Adaptation' // Choice
    topological choice at initialization time variableName='speed_adaptation'
```



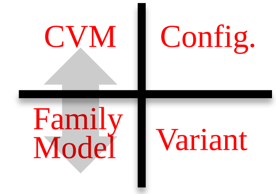


McGill School of Computer Science

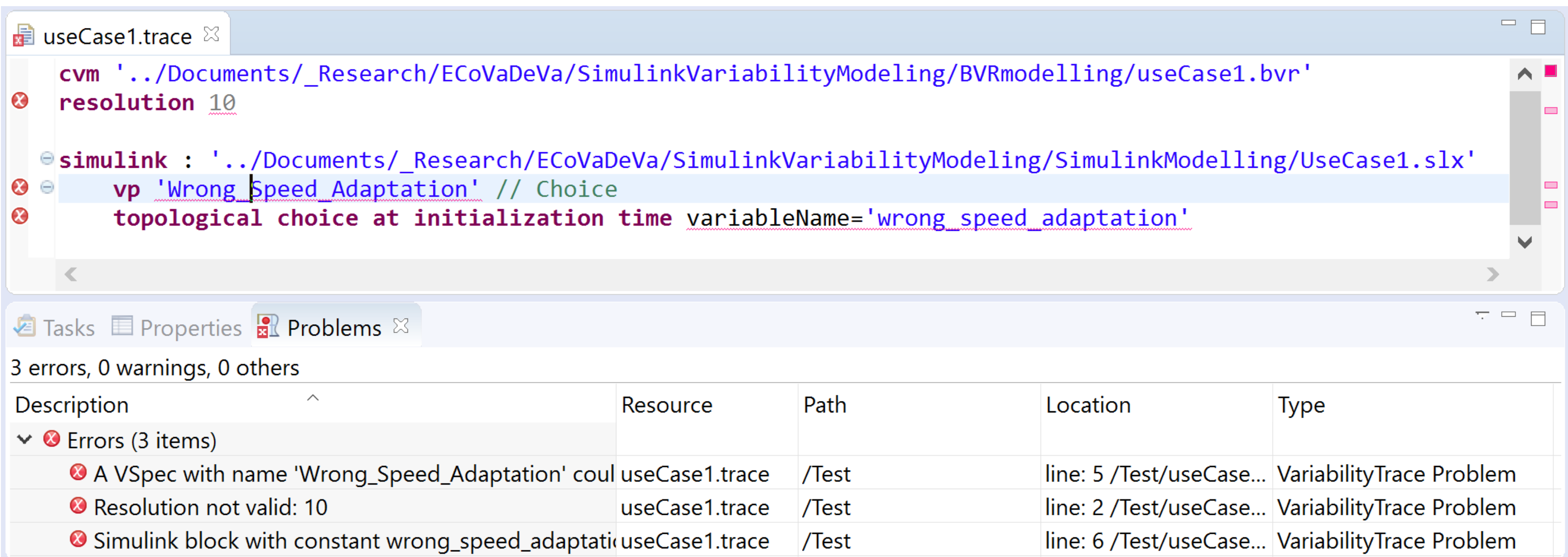
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

Ansymo Antwerp Systems & Software Modelling University of Antwerp

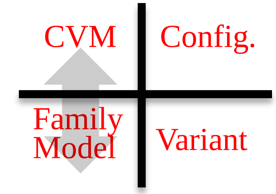
Traceability Tool



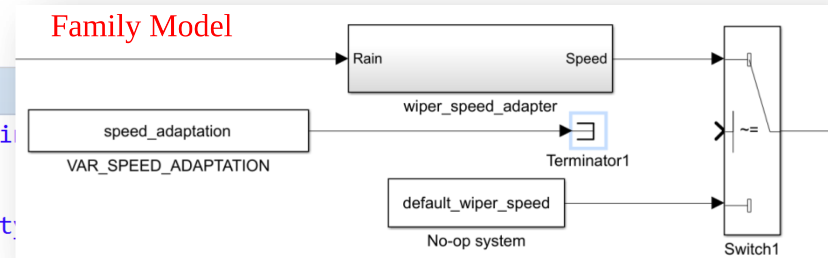
- Detection of errors
  - Paths, existence of elements, …



Traceability Tool



- Detection of errors/inconsistencies
  - Checking correctness of Simulink family model against CVM
    - E.g., there is a constant block named "speed_adaptation" but it's not connected to a switch



useCase1.trace

```
cvm '../Documents/_Research/ECoVaDeVa/SimulinkVariabilityModeli
resolution 1

simulink : '../Documents/_Research/ECoVaDeVa/SimulinkVariability
    vp 'Speed_Adaptation' // Choice
    topological choice at initialization time variableName='speed_adaptation'
```

Tasks | Properties | Problems

1 error, 0 warnings, 0 others

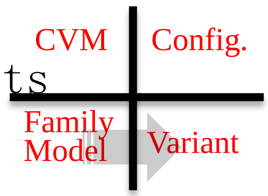
Description	Resource	Path	Location	Type
Errors (1 item)				
No Switch block connected to the given Constant bl	useCase1.trace	/Test	line: 6 /Test/useCase...	VariabilityTrace Problem

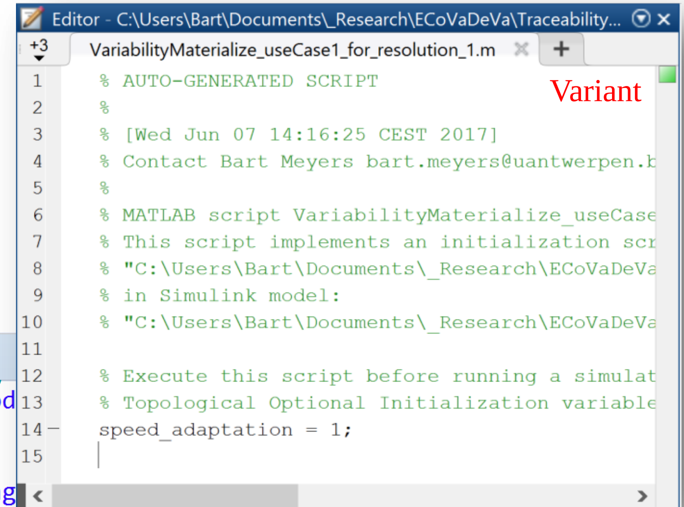
McGill School of Computer Science

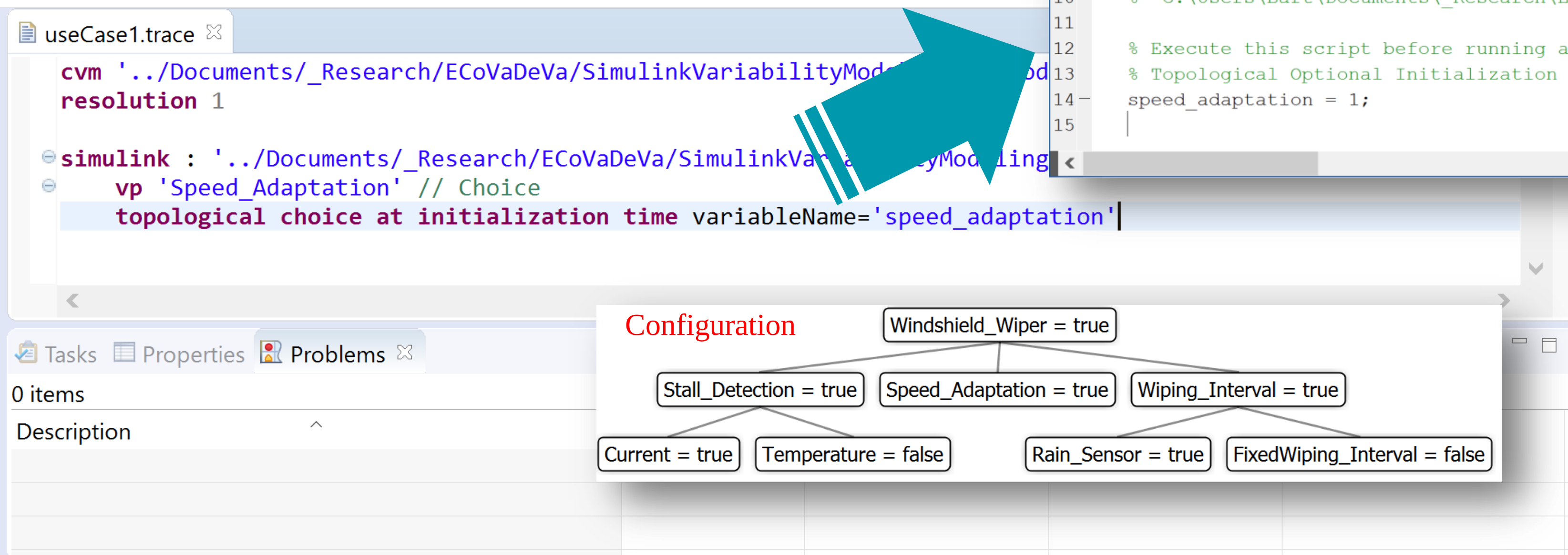
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

Ansymo Antwerp Systems & Software Modelling University of Antwerp

– Generation of variants







Roadmap

- Implement more variability techniques in traceability tool
- Look into variability modelling for tools for acausal modelling
  - I suspect this will be a major challenge

Agile Model-Based Systems Engineering

Joachim Denil

Joachim.Denil@uantwerpen.be

McGill School of Computer Science

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

Ansymo Antwerp Systems & Software Modelling University of Antwerp

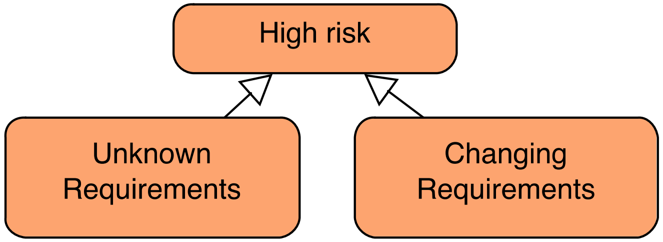
Summary

- Companies want to increase responsiveness to change in requirements
- Agile principles helped Software Engineers with same problem!
- Application to Systems Engineering is more difficult
- Modelling techniques and supporting tools could help in enabling Agile MBSE

McGill School of Computer Science neesis

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK Ansymo Antwerp Systems & Software Modelling University of Antwerp

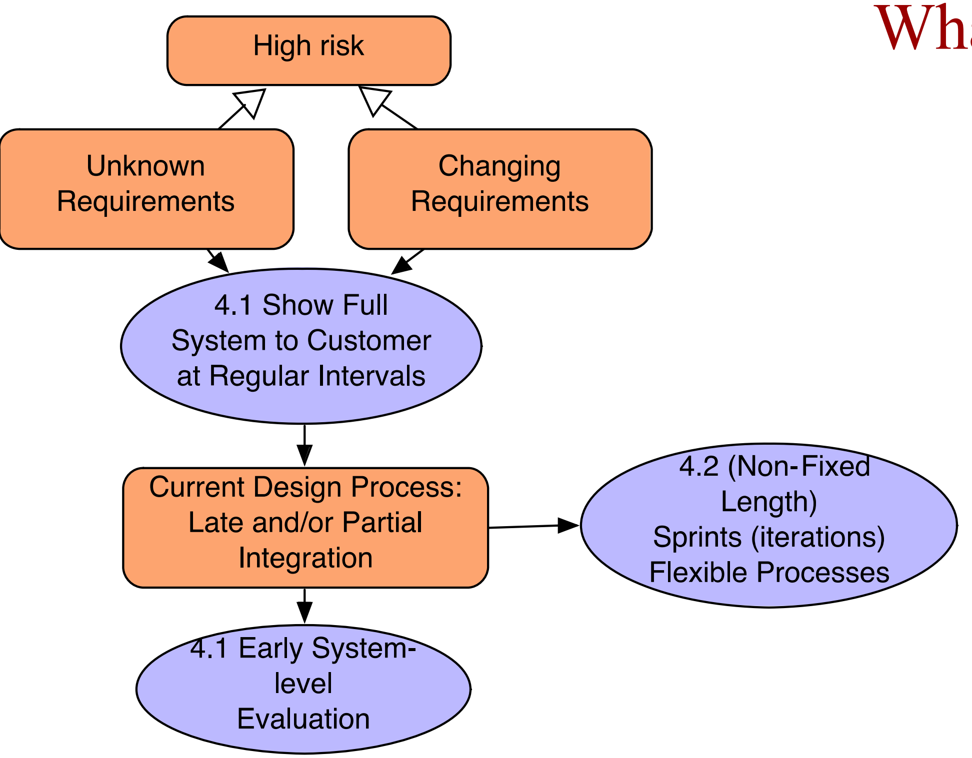
Why?

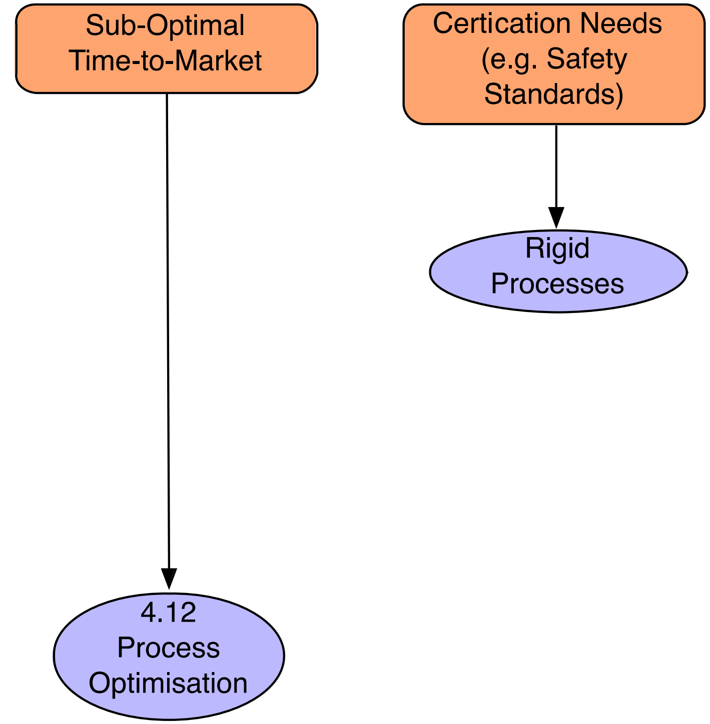


Sub-Optimal Time-to-Market

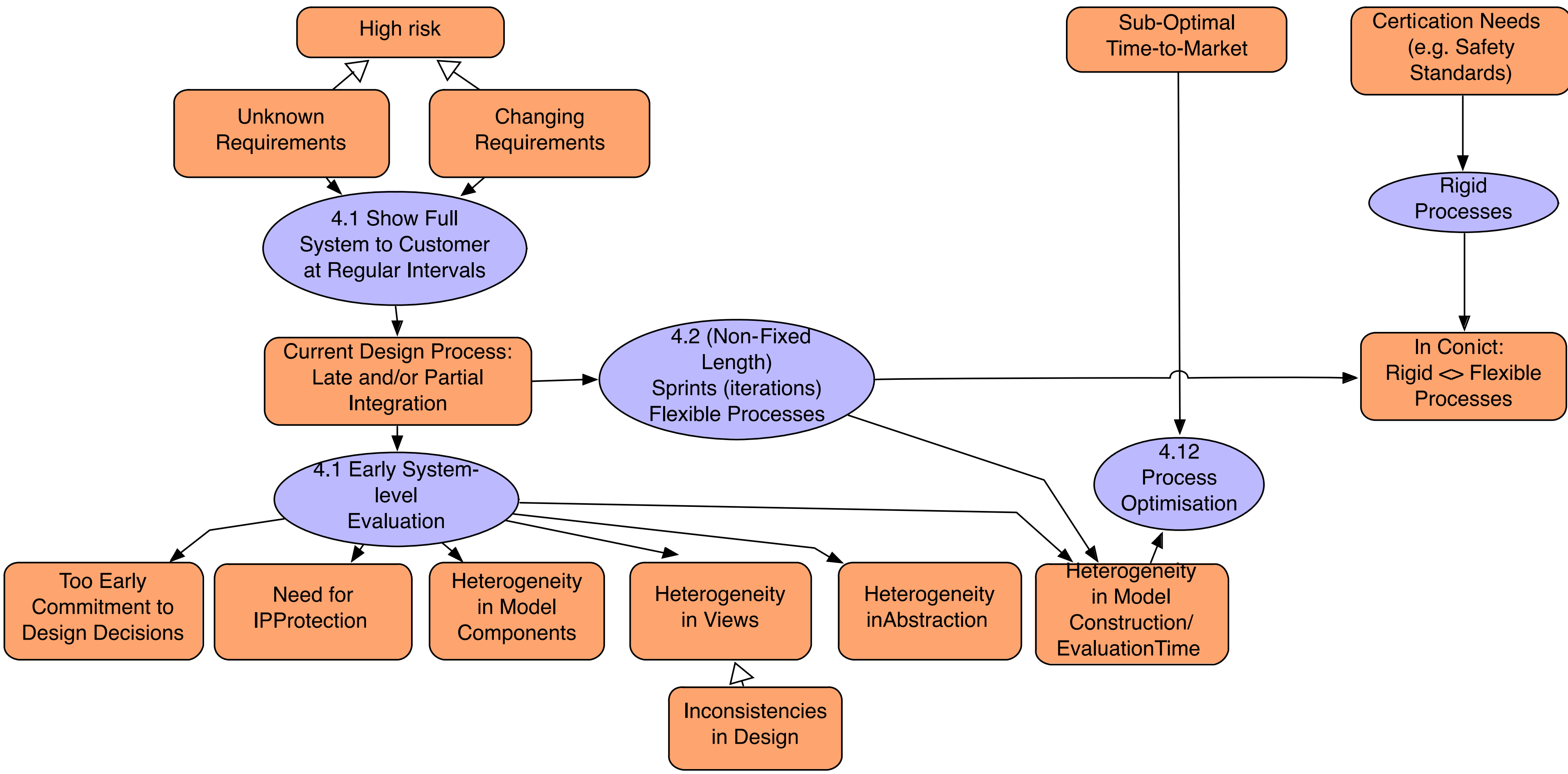
Certication Needs (e.g. Safety Standards)

What?


High risk
Unknown Requirements
Changing Requirements
4.1 Show Full System to Customer at Regular Intervals
Current Design Process: Late and/or Partial Integration
4.2 (Non-Fixed Length) Sprints (iterations) Flexible Processes
4.1 Early System-level Evaluation

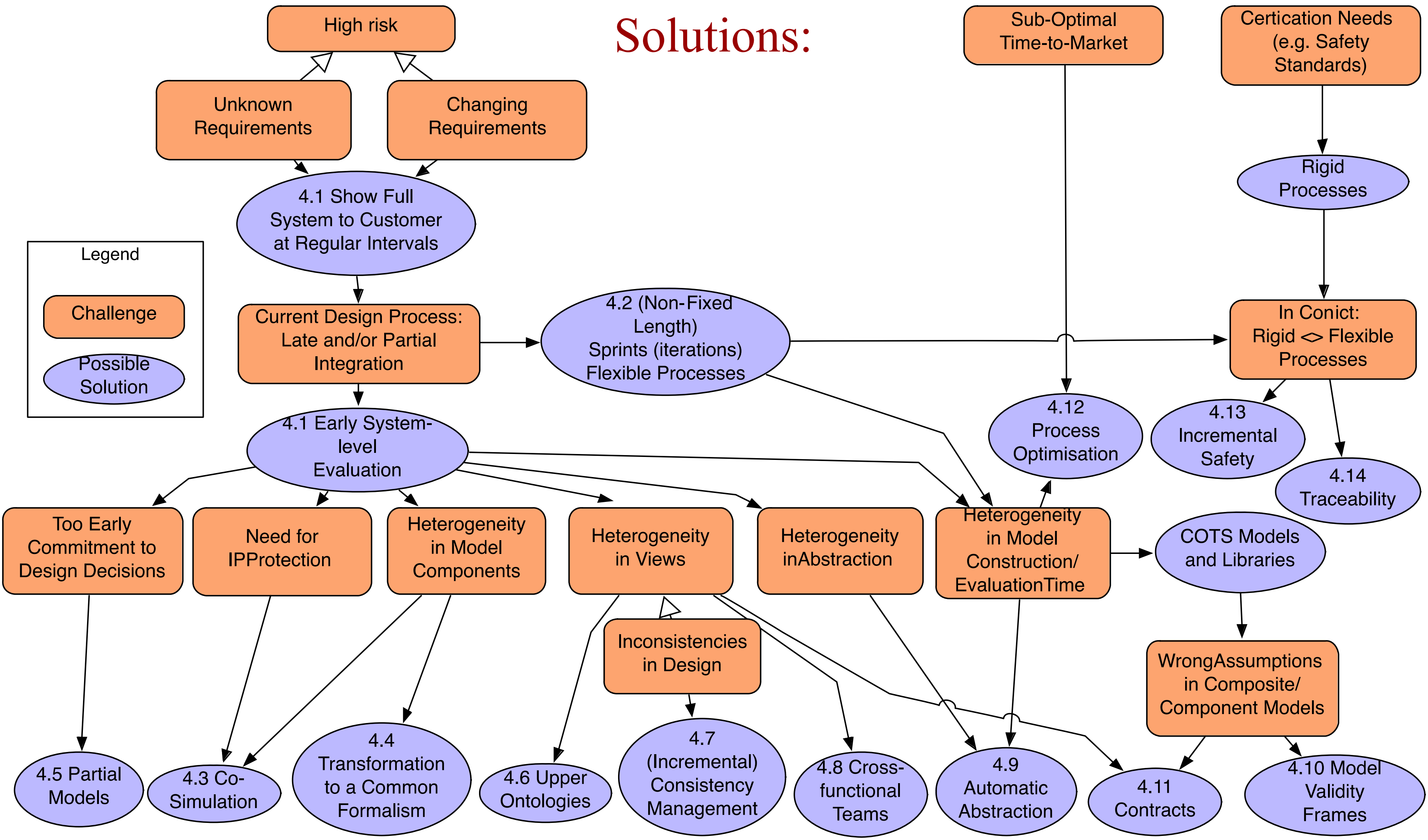

Sub-Optimal Time-to-Market
4.12 Process Optimisation
Certication Needs (e.g. Safety Standards)
Rigid Processes

But… for complex systems: e.g. CPS



McGill School of Computer Science
necsis
FLANDERS MAKE MANUFACTURING INNOVATION NETWORK
Ansymo Antwerp Systems & Software Modelling University of Antwerp

Solutions:


High risk
Unknown Requirements
Changing Requirements
4.1 Show Full System to Customer at Regular Intervals
Legend
Challenge
Possible Solution
Current Design Process: Late and/or Partial Integration
4.2 (Non-Fixed Length) Sprints (iterations) Flexible Processes
Sub-Optimal Time-to-Market
Certication Needs (e.g. Safety Standards)
Rigid Processes
In Conict: Rigid <> Flexible Processes
4.1 Early System-level Evaluation
4.12 Process Optimisation
4.13 Incremental Safety
4.14 Traceability
Too Early Commitment to Design Decisions
Need for IPProtection
Heterogeneity in Model Components
Heterogeneity in Views
Heterogeneity inAbstraction
Heterogeneity in Model Construction/ EvaluationTime
COTS Models and Libraries
Inconsistencies in Design
WrongAssumptions in Composite/ Component Models
4.5 Partial Models
4.3 Co-Simulation
4.4 Transformation to a Common Formalism
4.6 Upper Ontologies
4.7 (Incremental) Consistency Management
4.8 Cross-functional Teams
4.9 Automatic Abstraction
4.11 Contracts
4.10 Model Validity Frames

Future Directions

- From **Research Plan** to #research proposals
- FWO SBO Proposal
  - External Partners needed
  - Company support and Valorization needed
  - Select minimal set of Topics to enable Agile MBSE in company setting

Coffee

Research Plan and Projects under Submission and Accepted

(INES, ASET, EMPHYSIS)

Joachim

CoSys - MSDL

Joachim's Research Plan

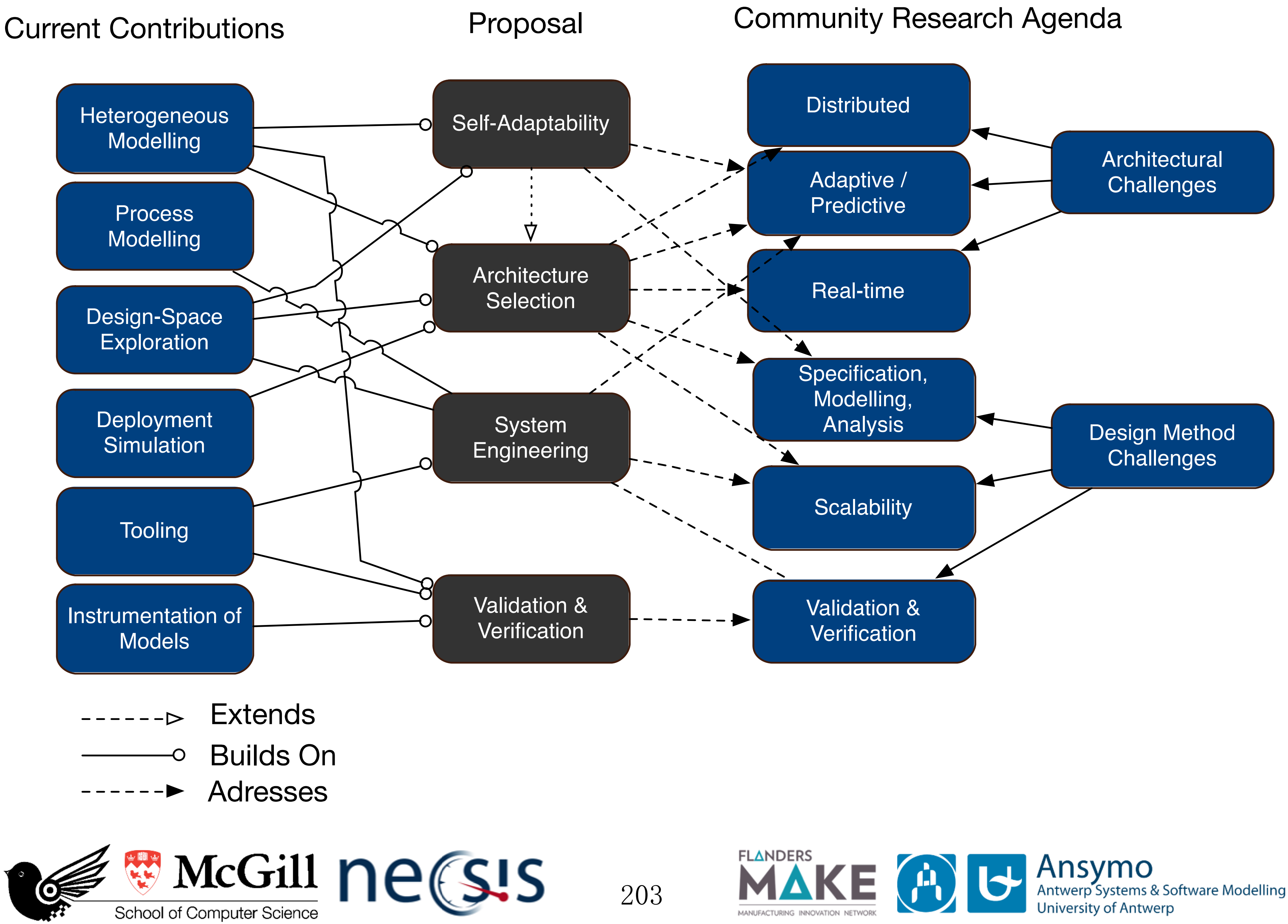
Joachim.Denil@uantwerpen.be

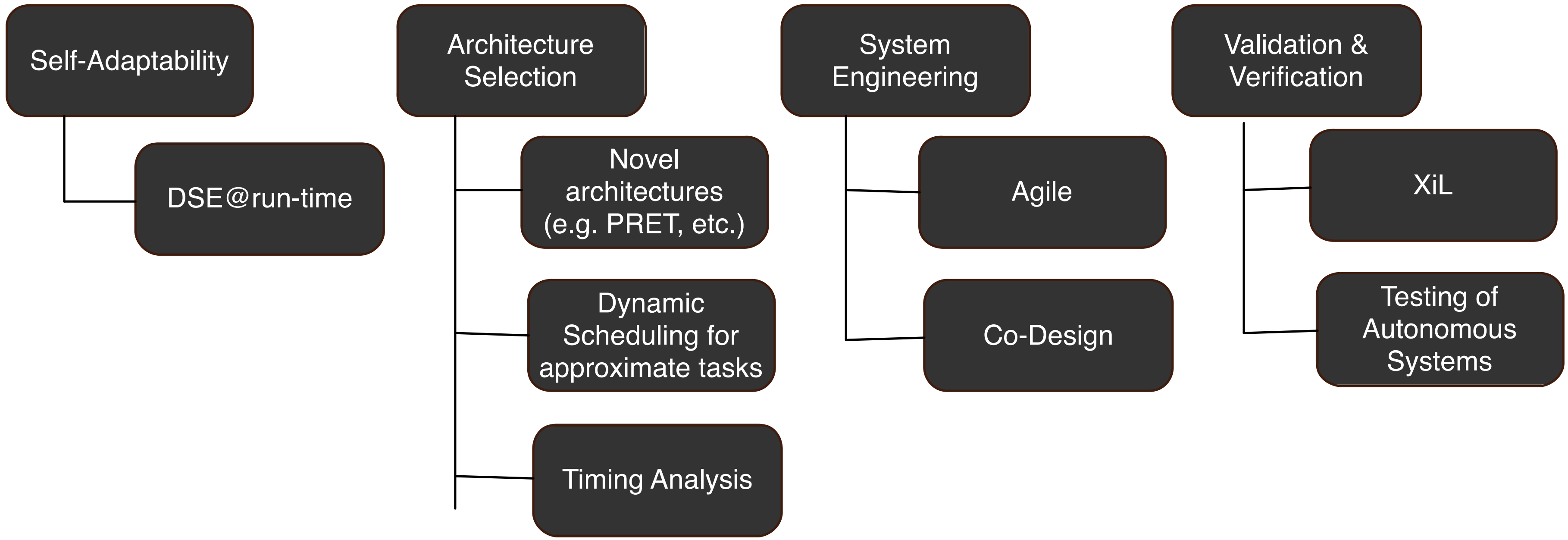
McGill School of Computer Science NECSIS

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK Ansymo Antwerp Systems & Software Modelling University of Antwerp


SMSC
LAN91C110
A9920-A767
8H12763


A380
AIRBUS


Current Contributions
Proposal
Community Research Agenda
Heterogeneous Modelling
Process Modelling
Design-Space Exploration
Deployment Simulation
Tooling
Instrumentation of Models
Self-Adaptability
Architecture Selection
System Engineering
Validation & Verification
Distributed
Adaptive / Predictive
Real-time
Specification, Modelling, Analysis
Scalability
Validation & Verification
Architectural Challenges
Design Method Challenges
Extends
Builds On
Adresses
McGill
School of Computer Science
FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK
Ansymo
Antwerp Systems & Software Modelling
University of Antwerp


Self-Adaptability
DSE@run-time
Architecture Selection
Novel architectures (e.g. PRET, etc.)
Dynamic Scheduling for approximate tasks
Timing Analysis
System Engineering
Agile
Co-Design
Validation & Verification
XiL
Testing of Autonomous Systems

Validation, Verification, Testing and Accreditation

Analysis and Verification of Model Transformations, Debugging, Instrumentation, Tracing, etc.

Language Engineering

Domain-Specific Languages, Model Transformation, (web-based) Visual and Textual Modelling Environments, etc.

Simulation

Co-Simulation, Discrete-event, DEVS, continuous time, acausal, Modelica, etc.

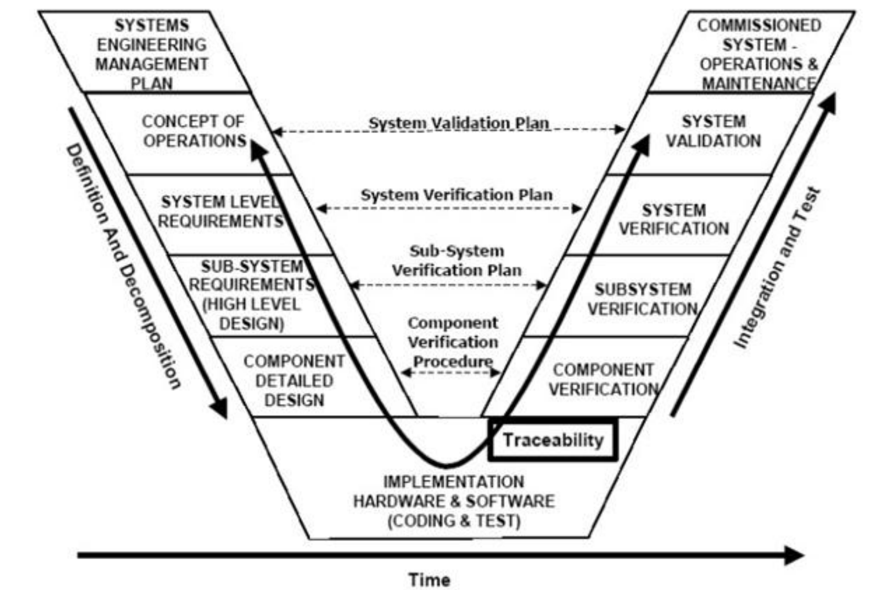
Deployment & Resource-optimized Execution

Platforms (e.g. AUTOSAR, CAN, etc.), Design-Space Exploration, Virtualization, Models@run-time, Efficient execution of model transformations, etc.

Model Management & Process

FTG+PM, Safety (ISO 26262, Railway, etc,), Agile Modelling, Consistency management, Experimental frames, etc.

Approved: INES: Eureka Project (O&O)



Siemens PLM Software

SIEMENS

BOEING

Work on:

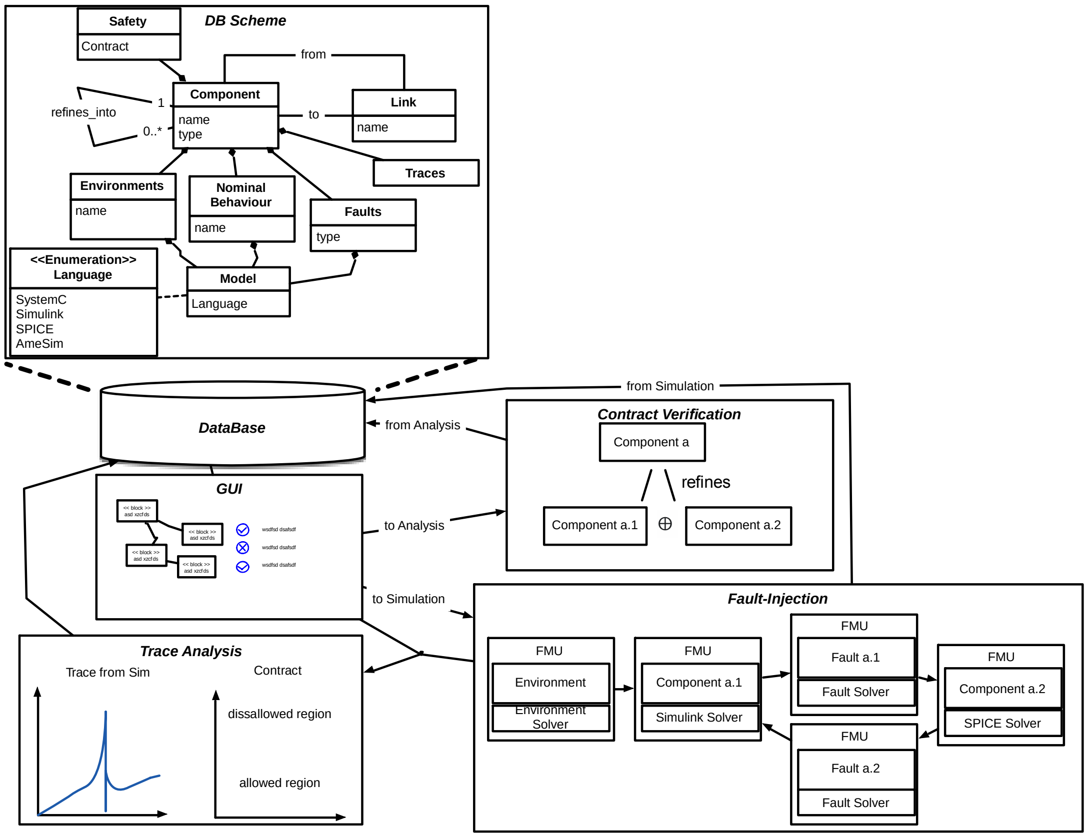
- Fault-injection
- Deployment Simulation
- Co-Simulation (MiL and HiL)
- Etc.

18 PM Pre-doc + 6 PM Post-doc

McGill School of Computer Science

Ansymo Antwerp Systems & Software Modelling University of Antwerp

In Submission: aSET (FM ICON)



Siemens PLM Software

SIEMENS

HSPRO

Leclanché

TENNECO

DANA

12 PM Pre-doc; 12 PM Post-doc

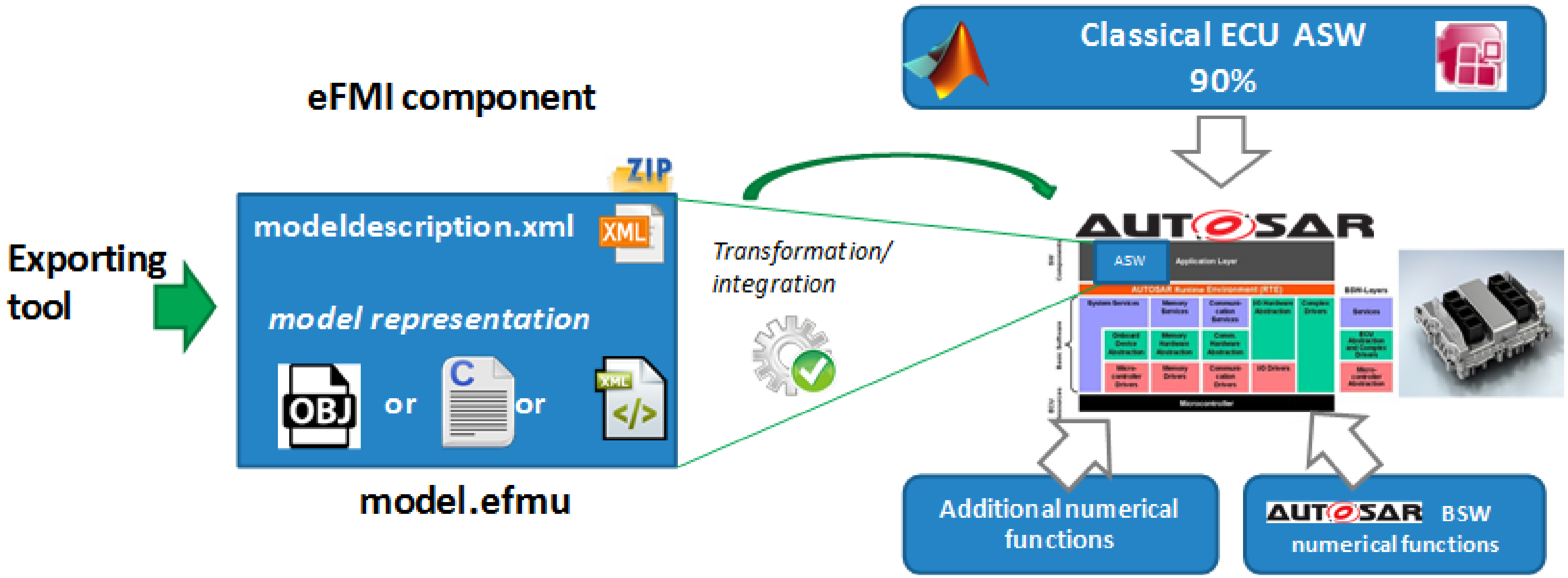
McGill School of Computer Science

FLANDERS MAKE MANUFACTURING INNOVATION NETWORK

Ansymo Antwerp Systems & Software Modelling University of Antwerp

In Submission: Emphysis (EU ITEA3)

BOSCH ETAS SIMULATION X Powered by ITI PIKETEC software engineering
Siemens PLM Software SIEMENS DANA cea DLR Honeywell



18 PM Pre-doc, 6 PM Post-doc

NEXOR Research Plan

Fons

Discussion on research threads, road maps, priorities,

why/what/how for customers

Conclusion*

Hans

* If we got this far…