MSDL Research

Pieter Mosterman
Sadaf Mustafiz, Levi Lucio
Bentley Oakes, Maris Jukss

Hans Vangheluwe
Bruno Barroca

Bart Meyers
Simon Van Mierlo, Yentl Van Tendeloo, Istvan David, Claudio Gomes

Joachim Denil
Ken Vanherpen

McGill School of Computer Science

neCIS
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
http://dsm-tp.org
Research Topics

**Analysis, Validation, Verification, Testing and Accreditation**
- Analysis and Verification of Model Transformations, Debugging, Instrumentation, Tracing, etc.

**Language Engineering**
- Domain-Specific Languages, Model Transformation, Design-Space Exploration (web-based) Visual and Textual Modelling Environments, etc.

**Simulation**
- Co-Simulation, Discrete-event, DEVS, continuous time, a-causal (e.g., Modelica), physics-based (e.g., Bond Graph), etc.

**Deployment & Resource-optimized Execution**
- Platforms (e.g. AUTOSAR, CAN, etc.), Deployment-Space Exploration, Virtualization, Models@run-time, Efficient execution of model transformations, etc.

**Model Management and Process**
- FTG+PM, Safety (ISO 26262, Railway, etc.), Agile Modelling, Consistency management, Experimental frames, etc.
09:00: Coffee at Hans' office (M.G.116)

09:30: welcome and high-level overview (Hans Vangheluwe)

09:40 - 10:00: Keynote: Riding the Line Between the Formal and Non-Formal in Modeling (Rick Salay)

10:00 - 11:10: Session on Modeling Language Engineering
10:00: Modelverse (Yentl Van Tendeloo)
10:10: Semantic Languages for Developing Correct Language Translations (Bruno Barroca)
10:20: Modular Language Composition (Cláudio Gomes)
10:30: Verification of Domain-Specific Models with ProMoBox (Bart Meyers)
10:40: Dynamic Structure Modelling for Causal Block Diagrams (Yves Maris)

10:50: 20 min discussion

11:10: coffee

11:20 - 12:20: Session on Simulation Techniques
11:20: PythonPDEVS (Yentl Van Tendeloo)
11:30: SCCD: a Statecharts and Class Diagrams hybrid (Simon Van Mierlo)
11:40: Discontinuity Propagation in Hybrid System Simulation (Cláudio Gomes)
11:50: Co-simulation: Simulator Coupling Approaches (Cláudio Gomes)
12:00: Debugging (Simon Van Mierlo)

12:10: 20 min discussion

12:30: lunch (sandwiches)

13:30 - 14:40: session on processes and optimization
13:30: FTG+PM (Hans Vangheluwe)
13:40: Engineering Process Transformation to Manage (In)Consistency (István Dávid)
13:50: Tool and Contracts for the Co-Design of Cyber-Physical Systems (Ken Vanherpen)
14:00: Experimental Frames (Joachim Denil)
14:10: Agility in the MBSE Process (Joachim Denil)

14:20: 20 min discussion

14:40: coffee

15:00 - 15:30: session on deployment/resource optimized execution
15:00: Deployment for AUTOSAR (Joachim Denil)
15:10: Activity in PythonPDEVS (Yentl Van Tendeloo)

15:20: 10 min discussion

15:30 - 16:00: session on model transformation
15:30: Efficient and Usable Model Transformations (Maris Jukss - Skype)
15:40: Fully Verifying Graphical Contracts on Model Transformations (Bentley Oakes - Skype)

15:50: 10 min discussion

16:00: social event: beer tasting
Welcome to the homepage of the 8th International Workshop on Multi-Paradigm Modelling (MPM'14).

Multi-Paradigm Modelling (MPM) is a research field focused on solving the challenge of combining, coupling, and integrating rigorous models of some reality at different levels of abstraction and views, using adequate modeling formalisms and semantic domains, with the goal to simulate (for optimization) or realize systems that may be physical, software, or a combination of both. The key challenges are finding adequate Model Abstractions, Multi-formalism modeling, Model Transformation and the application of MPM techniques and tools to Complex Systems. MPM theories/methods/technologies have been successfully applied in the field of software architectures, control system design, model integrated computing, and tool interoperability. The seventh Workshop on Multi-Paradigm Modelling (MPM) is aimed at furthering the state-of-the-art as well as to define future directions of this emerging research area by bringing together world experts in the field for an intense one-day workshop.

Organizers:
- Daniel Balasubramanian, Vanderbilt University, USA
- Christophe Jacquet, Supélec Systems Sciences, France
- Pieter Mosterman, Real-time and Modeling & Simulation Technologies, The MathWorks, USA and School of Computer Science, McGill University, Canada
- Sahar Kokaly, NECIS, Canada
- Tamás Mészáros, Budapest University of Technology and Economics, Hungary

2016 Bellairs CAMPaM workshop

Welcome to the homepage of the thirteenth Bellairs CAMPaM workshop.

The workshop aims to further the state-of-the-art in Computer Automated Multi-Paradigm Modelling (CAMPaM) as well as to define future directions of this emerging research area by bringing together world experts in the field for an intense one-week workshop.

The workshop will be held Friday 29 April (arrival) - Friday 6 May (departure) 2016 at McGill University's Bellairs campus. The actual workshop starts on Saturday morning and continues for 5 full days until Wednesday evening. Although it is possible to depart on Thursday, most participants leave on Friday to do some sightseeing on Thursday (in particular, to visit Crane Beach).

The workshop takes the Daugavpils seminar format -- bring a critical mass of top researchers together in a relatively remote location and soon new ideas will flow -- one step further: the Bellairs facilities are relatively primitive (and cheap) and there are no distractions such as typically found in hotels.

Organizers:
- Hans Vangheluwe, Department of Mathematics and Computer Science, University of Antwerp, Belgium, and School of Computer Science, McGill University, Canada.
- Pieter Mosterman, Real-time and Modeling & Simulation Technologies, The MathWorks, USA and School of Computer Science, McGill University, Canada.
The DSM-TP International Summer School provides an opportunity for learning and discussion about Domain Specific Modeling.

The School takes place from the 22nd till the 26th of August 2016 at the Université de Genève in Switzerland. This year the Summer School is organized by the Software Modeling and Verification Group (Genève, Switzerland) (SMV) in close collaboration with the Modelling, Simulation and Design Lab (Antwerp, Belgium and Montreal, Canada) and the Departamento de Informática (Portugal), who organized the previous editions of this Summer School.

Over the last decades, the complexity of systems we study and design (such as Cyber-Physical Systems) has grown exponentially. To manage this complexity, industry and academia now explicitly model different aspects of the structure and behaviour of systems, at the most appropriate level(s) of abstraction, using the most appropriate modelling formalism(s).

Dedicated modelling formalisms, also known as Domain Specific Languages, are used increasingly to maximally constrain the
Welcome to the COST Action IC1404 Multi-Paradigm Modelling for Cyber-Physical Systems

01 ABOUT
   Mission - Organization - Become a member!

02 ICT COST Action IC1404
   Introduction - Memorandum of Understanding

03 DOCUMENTS
   Newsletters - Dissemination Materials -
   - Calls (STSMs, Schools, etc.) -
   - Internal Reports and Minutes -
   - Administrative Information

04 NEWS AND UPCOMING EVENTS

05 USEFUL LINKS
   Contacts - Related Projects -
   - COST Administration (E-COST) -
   - Partner COST Actions - Related Events

06 INTERNAL SERVICES
   Management Committee (MC) -
   - Administrators (Zope management)
Papyrus Industrial Consortium Research/Academia

* Research
  + Promotion of research projects
  + Better access to research funding
  + Research collaborations
  + Better access to industrial problems
  + Possibility to interact with the industry on the development of relevant solutions
  + Facilitate tech transfer
  + Facilitate the recruitment of new students
    o Students are motivated by industrial interactions/collaborations

* Teaching
  + Sharing of teaching material
    o It is very time consuming to develop quality teaching material
    o Establish a critical mass that will ensure better quality and stability
  + Collaborate between universities and with the industry on course projects
  + Consortium provides “rover” for class projects
some Projects/Funding/Collaborators
(academic collaborators not listed)

http://www.cost.eu/COST_Actions/ict/Actions/IC1404

http://www.mpm4cps.eu

http://www.necsis.ca/

http://www.modelwriter.eu/

http://www.mbse4mechatronics.org/
MODELLING LANGUAGE ENGINEERING
Modelverse

Yentl Van Tendeloo
Universiteit Antwerpen
yentl.vantendeloo@uantwerpen.be
Modelverse: Motivation
Modelverse: Explicit Type/Instance Relation
Modelverse: Multi-Conformance

Semantic Languages for Developing Correct Language Translations

Bruno Barroca
McGill University
bbarroca@cs.mcgill.ca
DSL + MDD

- DSLs restrict possibility of making errors
- Model-Driven Development (MDD)
- Rapid Prototyping
- Properties guaranteed by construction
- Software and Systems Certification

- Can the Push-Button?
DSL Semantics in the Push button

- How to define semantics for a DSL?

Model $\vdash$ DSL

$[[\text{Model}]]^\text{DSL} \triangle [[\ t \ (\text{Model}) \ ]]^\text{Ex.p}$
Other ways for Specifying Semantics?

- Read the paper: “Modular Design of Hybrid Languages by Explicit Modeling of Semantic Adaptation”
- And: “Towards Modular Language Design using Language Fragments: The Hybrid Systems Case Study”
- Use of **State Charts** to define the simulator/interpreter of a language in a **modular** way: FSM, CBD, FSM+CBD
DSL Semantics outside the Box?

- How to be sure that we got what we really wanted?

\[
[[ ]]^{DSL} \Delta \{\text{axiom}\}
\]

\[
\text{axiom} \Delta \text{pre} \rightarrow \text{pos}
\]
Already done..

- Read the paper: “Semantic Languages for Developing Correct Language Translations”
- A tool was developed in EMF/Prolog
  - [https://github.com/githubbrunob/DSLTransGI](https://github.com/githubbrunob/DSLTransGI)
    /tree/master/SOSBuilder
    /tree/master/dsltransAnalysis
SOS Language for the rescue
Ability to detect errors on the translation...
Next steps on DSLTrans+SOS

- Current notion of minimality
  - Generate model sources based on the assumption that each transformation rule was: (0) not executed, (1) executed once.
  - We assume that ‘Executed once’ is no different from ‘Executed many times’

- Explore/develop notion of minimality
  - Restrict match/apply pair generation to conforming to the source metamodel
  - Restrict match/apply pair that:
    - produce a non-empty transition system on SOS
    - explore/triggers all axioms defined in source SOS
  - Automatically Generate transformation rules, and transformation rules auto-fix!
Next steps on SOS

- Analyze finiteness of a SOS semantics

or

- Ensure finiteness by construction?
Modular Language Composition

Cláudio Gomes
Universiteit Antwerpen
claudio.goncalvesgomes@uantwerpen.be
Heterogeneity in Languages

Control Logic (FSAs)

Motor and window mechanics (ODEs)
De-constructing an Hybrid Language
Language Specification Fragments (LSF)
Constructing an Hybrid Language
The Big Picture
Thank you!

- Ugaz, Rafael. Weaving of domain-specific languages: A literature review. 2014.
- Ugaz, Rafael. Weaving of domain-specific languages: Enabling technology. 2014.
- Ugaz, Rafael. Combination of Domain-Specific Languages. 2015.
Verification of Domain-Specific Models with ProMoBox

Bart Meyers
Universiteit Antwerpen
bart.meyers@uantwerpen.be
Properties for DSMLs: State of the Art

\[
\square((\neg 1 \land 3) \lor (\neg 2 \land 5)) \rightarrow (\neg 0 \lor \neg (\neg 2 \land \neg 5)) \lor (\neg 1 \lor \neg (\neg 2 \land \neg 5)) \lor (\neg 2 \lor \neg (\neg 2 \land \neg 5)) \lor (\neg 3 \lor \neg (\neg 2 \land \neg 5)) \lor (\neg 4 \lor \neg (\neg 2 \land \neg 5)) \lor (\neg 5 \lor \neg (\neg 2 \land \neg 5)) \lor (\neg 6 \lor \neg (\neg 2 \land \neg 5)) \lor (\neg 7 \lor \neg (\neg 2 \land \neg 5))
\]

Design \models \text{Property}
Properties for DSMLs: Property DSML

Design ⊨ Property
Properties for DSMLs: Five Languages

Multi-Paradigm Modelling of DSMLs

[[.|.]] Property Design Input Runtime Trace

reachesFloor forall after eventually

LTL .pml Spin .txt .trail

DSM Formal Methods

39
Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM

Properties for DSMLs: Consistency

Annotations

Consistency by construction

Design

Input

Runtime

Trace

Formal Methods

reachesFloor
forall 
1 after 
2 eventually 
3 4

DSM
Properties for DSMLs: Testing
Properties for DSMLs: Testing (Approach)
Conclusion and future

ProMoBox
- Annotations
- DSML generation
- Generic semantics

Model Checking
[SYN] Property Template
+ [SEM] Generic Promela compiler

Testing
[SYN] Test Template
+ [SEM] Generated operational semantics

Generation of test cases from properties
Publications


Dynamic structure modelling for Causal Block Diagrams

Yves Maris
Universiteit Antwerpen / McGill University
yves.maris@student.uantwerpen.be
Problem

- Expressiveness limited by fixed structure
- Changing model during simulation
- Staying consistent with CBD constructs
Solution

- Addition of structure block
  - Instantiation of new components
  - Operations for removal
  - Reinitialisation
- Triggered by event
Examples
Case Study

- Balls in elevator
- Doors open when elevator reaches floor
- Balls can enter and leave elevator through door
Session 2

SIMULATION TECHNIQUES
PythonPDEVS

Yentl Van Tendeloo
Universiteit Antwerpen
yentl.vantendeloo@uantwerpen.be
PythonPDEVS: Positioning

PythonPDEVS: Performance

PythonPDEVS: Features

PythonPDEVS: Future Work

- City
  - House
    - Generator
    - Queue
  - Road
    - Queue
    - Processor
    - Queue
  - Traffic light
  - Road
    - Queue
    - Processor
    - Collector
  - Commerce
    - Queue
SCCD: a Statecharts and Class Diagrams Hybrid

Simon Van Mierlo
Universiteit Antwerpen
simon.vanmierlo@uantwerpen.be
Complex Timed Autonomous Reactive Systems

Behavior
- Timed
- Autonomous
- Interactive
- Hierarchical

Structure
- Dynamic
- Hierarchical

Design? Statecharts + Class Diagrams = SCCD(XML)
SCCD Compiler
SCCD in and for Unity, a Commercial Game Engine

Tank Wars

SCCD: The Future

- Conformance
  - Initialization/Destruction
- Exceptions
- Dynamic Loading of SCCD Models
- Interfaces/Contracts: Protocol Machine
- Subtyping
  - Events as Objects
  - Behavior
- Object Referencing
- (Domain-specific) languages built on top of SCCD
  - Hierarchical Interactions
  - Process Languages
- De-/re-constructing hybrid languages


Discontinuity Propagation in Hybrid System Simulation

Cláudio Gomes
Universiteit Antwerpen
claudio.goncalvesgomes@uantwerpen.be
Discontinuities
Impulses

\[ \int_{-\infty}^{t} \delta(v - t_0) dv = \begin{cases} 1 & \text{if } t_0 \leq t \\ 0 & \text{otherwise} \end{cases} \]

\[
\begin{align*}
&\lim_{t_2 \to 0} \left[ m \left( v(t_2) - v(t) \right) \right] = \int_{t}^{t_2} f(t) \, dt \\
&\lim_{t_2 \to 0} \left[ m \left( v(t_2) - v(t) \right) \right] = \int_{t}^{t_2} I(t-L_0) \, dt
\end{align*}
\]
Generalized Signal

\[ S(t) = \begin{cases} 
  t^2 & \text{if } t < 1 \\
  -(2-t)^2 & \text{if } t \geq 1
\end{cases} + a_{01} \delta(t - \tau_1) + a_{11} \delta'(t - \tau_2) \]

\[ S'(t) = \begin{cases} 
  2t & \text{if } t < 1 \\
  4 - 2t & \text{if } t \geq 1
\end{cases} - 2\delta(t - 1) + a_{01} \delta'(t - \tau_1) + a_{11} \delta^{(2)}(t - \tau_2) \]

\[ S^{(2)}(t) = \begin{cases} 
  2 & \text{if } t < 1 \\
  -2 & \text{if } t \geq 1
\end{cases} - 2\delta'(t - 1) + a_{01} \delta^{(2)}(t - \tau_1) + a_{11} \delta^{(3)}(t - \tau_2) \]

\[ S^{(3)}(t) = -4\delta(t - 1) - 2\delta^{(2)}(t - 1) + a_{01} \delta^{(3)}(t - \tau_1) + a_{11} \delta^{(4)}(t - \tau_2) \]

\[ S(t) = s(t) + \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} \delta^{(i)}(t - \tau_j) \]
Results - Bouncing Ball

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>MSE Hybrid CBDs</th>
<th>MSE Normal CBDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-2}$</td>
<td>$9.188382 \times 10^{-3}$</td>
<td>$2.012778 \times 10^{-2}$</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>$1.476189 \times 10^{-4}$</td>
<td>$2.93385 \times 10^{-4}$</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>$1.185938 \times 10^{-6}$</td>
<td>$2.474646 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MSE</th>
<th>Exec Time Hybrid CBDs</th>
<th>Exec Time Normal CBDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$9.128132 \times 10^{-3}$</td>
<td>2.887s (238%)</td>
<td>1.212s (100%)</td>
</tr>
</tbody>
</table>
Co-simulation: Simulator Coupling Approaches

Cláudio Gomes
Universiteit Antwerpen
claudio.goncalvesgomes@uantwerpen.be
The modern car

- Complexity
  - 40+ subsystems

- Competitive Market

- Concurrent Development
  - Late Integration Problems

- Distributed Development
  - Specialized suppliers
  - Late Integration (due to IP)
Simulators

\[ \text{Simulator} = \text{Solver} + \text{Model} \]

**Model**

\[
m_1 \ddot{x}_1 = -c_1 \cdot x_1 - d_1 \cdot \dot{x}_1 + F_c
\]

\[ x_1(0) = p_1; \quad \dot{x}_1(0) = s_1 \]

**Solver**

\[
\ddot{x}(t + h) := \ddot{x}(t) + f(\ddot{x}(t), u(t)) \cdot h
\]

\[ \dddot{x}(0) := x(0) \]

\[
S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i} \rangle
\]

\[ \delta_i : \mathbb{R} \times \mathbb{R} \times X_i \times U_i \rightarrow X_i \]

\[ \lambda_i : \mathbb{R} \times X_i \times U_i \rightarrow Y_i \text{ or } \mathbb{R} \times X_i \rightarrow Y_i \]

\[ x_i(0) \in X_i \]

\[ \phi_{U_i} : \mathbb{R} \times U_i \times \ldots \times U_i \rightarrow U_i \]
Co-simulation

Co-sim. Scenario = Simulators + Coupling Conditions

Co-Simulator = Co-sim. Scenario + Orch. Algorithm

\[
CS = (S, L)
S = (S_1, \ldots, S_n)
L : Y_1 \times \ldots \times Y_n \times U_1 \times \ldots \times U_n \to \mathbb{R}^n
L(y_1, \ldots, y_n, u_1, \ldots, u_n) = 0
\]

1. \( u_1(t) = \lambda_1(x_1(t)) \)
2. \( u_2(t) = \lambda_2(x_2(t)) \)
3. \( x_1 = x_2 \)
4. \( t \to t + H \)
Orchestration Algorithm Concerns

- Heterogeneous Capabilities of Simulators
  - Accuracy
  - Algebraic Loops
  - Distribution
  - Modularity
Separation of Concerns via MDE

- **Objective**: Deal with Complex Orchestration Alg.
- **How**?
  - Transform Co-sim scenario to address each concern separately;
  - Reduce to a trivial form;
  - Add standard Orchestration Alg;
Example: Distribution Concern

- Across computers, small H incurs network communication cost.
- Large H leads to accuracy problem.
- Extrapolation made by simulators is inappropriate to the scenario.
- Complex orchestration mechanism required to deal with distribution correctly.
Debugging

Simon Van Mierlo
Universiteit Antwerpen
simon.vanmierlo@uantwerpen.be
- Discrete-Event: Statecharts, DEVS
- Continuous: Causal Block Diagrams
- Dynamic Structure: DSDEVS
- Rule-Based Model Transformation (see Maris' presentation)
- TODO: multi-formalism: co-simulation vs. semantic adaptation
- TODO: non-determinism, action language
Session 3

PROCESSES AND OPTIMISATION
FTG+PM

Hans Vangheluwe
Universiteit Antwerpen / McGill University
hv@cs.mcgill.ca
28 different modelling formalisms
50 transformations

FTG+PM (model mgmt. … consistency)
Engineering Process Transformation to Manage (In)Consistency

István Dávid
Universiteit Antwerpen
istvan.david@uantwerpen.be
Why inconsistencies?

- Complex engineered systems
  - Increased complexity, interplay between disparate domains
  - Multi-paradigm, multi-domain, collaborative modeling settings
  - Inconsistencies between models: due to semantic overlaps

- Inconsistencies → $$$
  - Late (or no) detection, numerous re-iterations…
What to do?

- 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)

Goal: manage potential inconsistencies

Weave in management patterns into the process
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
Process modeling and transformation

- Model the process
- Identify potential inconsistencies
- Transform the process

- Quantification of the optimality
  - Loops and decisions in the process
    → requires **stochastic** simulation
  - Multiple simulation strategies
    - Mapping to queueing networks
  - Custom strategies can be implemented and plugged in

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

– Formalism for modeling processes along with properties
– Optimization for consistency
  – …and eventually transit time of the process!
– Implementation
  – Process modeler (visual)
  – Characterization and management of inconsistencies via graph patterns and M2M transformations
Perspectives

– Enhancing the process model
  – Resources, ontological reasoning, enhanced cost model

– Tolerance
  – “Management” is more than just prevention
  – Temporal, parameter and design tolerance

– Link with tool chains and tool integration scenarios
  – OSLC

– Prototype
  – Process enactment, interfacing with engineering tools
Tool and Contracts for the Co-Design of Cyber-Physical Systems

Ken Vanherpen
Universiteit Antwerpen
ken.vanherpen@uantwerpen.be
Problem Statement
Current solution: Contract-Based Design (CBD)

**Contract for the control engineer**

- Assumptions
  - Sample time $\leq 0.8$ ms
  - Button signals are boolean
  - $\text{Res}_{\text{FORCE}} = 0.012$ V/N
  - $\text{Res}_{\text{MOTOR}} = 0.047$ V/RPM

- Guarantees
  - Safety $\leq 0.2$ mm
  - Reaction time $\leq 1$ ms
  - $\text{Comp}_{\text{CONTROL}} \leq 0.05$ ms

**Contract for the embedded engineer**

- Assumptions
  - WCET$_{\text{CONTROL}} \leq 0.05$ ms

- Guarantees
  - $T_{\text{CONTROL}} \leq 0.8$ ms
  - Load$_{\text{ECU}} \leq 69\%$
  - $\text{Res}_{\text{FORCE}} = 0.012$ V/N
  - $\text{Res}_{\text{MOTOR}} = 0.047$ V/RPM

**Negotiation**
Ontological Reasoning to enable CBD – Example


Round-Trip Engineering (RTE) Method

Experimental Frames

Joachim Denil
Universiteit Antwerpen
joachim.denil@uantwerpen.be
Industrial Size Example...

\[ F = -kX \]
Experimental Frames

More than one function!
Complex Environments?
How to implement all of this?

A. Experiment Model

- Repeatable scientific experiments
- Workflow-like language!
B. Validity Frame

- Calibration:
  - We have real world data.
  - Parameter calibration of our model

- Validity
  - Under which assumptions is our model valid?

⇒ EF is not sufficient:
  - Spring-like behaviour only possible in combination with Mass
  - Dependency on solver!
  - Initial conditions and Parameters?
  - etc./.
% Create result file:

dir = datenr(datetime('now'));
mkdir(dir);
htmlfilename = sprintf('%s/calibrationresult.html', dir);
theHtmlfile = fopen(htmlfilename, 'w+');
fprintf(theHtmlfile, '<!DOCTYPE html>
');
fprintf(theHtmlfile, '<html>
');
fprintf(theHtmlfile, '<body>
');

% Turn off warnings
warning off;

% Read the measured data:
M = csvread('spring_measure.csv');
M = csvread('non_lin_spring.csv');

% Load the spring calibration frame:
open_system('spring_calibration');
force_original = M(:,1);
measured_position_original = M(:,2);

% Find linear regions:
for window = mindatapoints:size(force_original)
    % Do a linear regression on the data:
    force = force_original(l:l+window);
    measured_position = measured_position_original(l:l+window);
    lin = measured_position/force;
    %lin = force(2)/measured_position(2);

    % Decide on K-value
    k = lin;

    % Configure the solver:
    set_param('spring_calibration', 'StopTime', '3')

    % Now do a simulation
    sv = [];
    for i = 1:size(force)
        m = force(i);
    SimOut = sim('spring_calibration');
    sv = [sv max(position.Data)];
    end

% Calculate an error
% Here it is linear so r^2 is a good value.
% In case of non-linear, we could generate
custom error functions and decide on K-value...
\( r^2 = 9.998413e-01 \)
\( k = 4.759093e+00 \)
\( \Omega_{\text{in\_force}} = [10, 90] \)
\( \Omega_{\text{out\_displacement}} = [2.101241e+00, 1.891116e+01] \)
What is next?

– Property Frame:
  – Design-by-Contract:
    – What model can I use?
    – Substitutability!
    – However: what about emergent properties from composition?
    – Do we need a notion of Function before going to behaviour?
      – Spring can act as a mass as well!
Agility in the MBSE Process

Joachim Denil
Universiteit Antwerpen
joachim.denil@uantwerpen.be
AGILE MANIFESTO

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

But…

Twelve Principles of Agile Software

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

Deliver working software frequently, at a couple of weeks to a few months’ intervals, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

http://agilemanifesto.org

But…

We need to take a step back…

mmm… Executable models? Demonstrations?

Frequent… A change to my CAD model takes 3 weeks!
Agility in MBSE...

- What is agility about?
  - Changing requirements
  - Rapid customer/system feedback
  - Holistic instead of silos
  - Etc.

- We need to take CPS/SIS/mechatronic context into account:
  - Cost of change
  - Safety!
  - Etc.
Solutions…

– Front-loading:
  – Make design decisions as early as possible in the process
  – Explore multiple solutions at the same time

– Early integration:
  – Use correct co-simulation to integrate as early as possible

– Explicit reasoning over processes
  – Short iteration cycles (with property support)
Session 4

DEPLOYMENT AND RESOURCE OPTIMISED EXECUTION
Deployment for AUTOSAR

Joachim Denil
Universiteit Antwerpen
joachim.denil@uantwerpen.be
Deployment Simulation

Design-Space Exploration

Variation:
- Type of ECU
- Number of ECUs
- Tasks, Priorities, ...

Constraints:
- Real-time Behavior

Optimality:
- Cost
- Extedibility

Gain (db)

1m 1u 1m 1u 5m 10m 10m ...

Multi-Abstraction in DSE

Explore at the model level!

Even More: What is the most appropriate technique?
For example: Rule-Based, Constraint-based, MILP, Genetic Algorithms, etc.

* Ken Vanherpen, Joachim Denil, Paul De Meulenaere, Hans Vangheluwe: Design-Space Exploration in MDE: An Initial Pattern Catalogue. CMSEBA@MoDELS 2014: 42-51
What is next?

- Which constraints? (see consistency work)
- Same Solver type vs. different solvers?
- Incrementality?
- Add (domain) information (sensitivity, etc.)
Activity in PythonPDEVS

Yentl Van Tendeloo
Universiteit Antwerpen
yentl.vantendeloo@uantwerpen.be
Activity: Motivation

Activity: Data Gathering

Activity: Results

Citylayout model for different migrators (5 nodes)

- No migration
- Activity tracking
- Custom activity tracking
- Custom activity Prediction

Time (s)

Microseconds artificial load
Session 5

MODEL TRANSFORMATION
Efficient and Usable Model Transformations

Maris Jukss
McGill University
maris.jukss@mail.mcgill.ca
Efficiency and Usability Issues

- Pattern matching is the most expensive operation
  - Based on subgraph isomorphism problem
- Debugging MT support lags behind code debugging
- Industrial application of MT may be hindered
Static Scope [1]

- Static scope is created by transformation engineer
- Unified hierarchical scope formalism
- Scoped transformation rules, reduced search space

Dynamic Scope [1]

- Discover scopes automatically (first match, optimistic)
- MT is observed, matches predicted (machine learning)

“Deep” Debugging of MTs
Fully Verifying Graphical Contracts on Model Transformations

Bentley James Oakes
McGill University
bentley.oakes@mail.mcgill.ca
Problem Statement

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

Given a model transformation... Does this contract hold on all input and output models?

Contract

Pre-condition

PhysicalNode  
Scheduler  
Service

Post-condition

CompositionType  
PPortPrototype

Does this contract hold on all input and output models?
- Symbolic execution - how rules interact and overlap
- Represent all possible transformation executions

- Contracts proved over all transformation executions
- “If pre-condition elements exist in input model, then post-condition elements exist in output model”


Many collaborations with Queens University (Canada) 
Selim, Lúcio, Cordy, Dingel, Oakes. “Specification and Verification of Graph-Based 
Model Transformation Properties” ICGT 2014. 
  – Verification of industrial transformation 
Selim, Cordy, Dingel, Lúcio, Oakes. “Finding and Fixing Bugs in Model 

Collaboration with Cláudio Gomes 
(University of Antwerp) et al. on SyVOLT 
Eclipse plug-in to build transformation and 
perform verification

Lúcio, Oakes, Gomes, Selim, Dingel, Cordy, 
Vangheluwe. “SyVOLT: Full Model 
Transformation Verification Using Contracts” 
MODELS 2015.
Also collaborating with Claudio on the verification of mbeddr, which is a set of languages designed to aid the development of embedded software.
Collaboration with Javier Troya (Universidad de Sevilla) and Manuel Wimmer (TU Wien)

Translate ATL transformations using a higher-order transformation into our language DSLTrans for contract proving

Multiple transformations translated, including industrial ATL transformation


SoSyM journal version in preperation