PROCESS MODELING, ANALYSIS AND ENACTMENT WITH THE *FTG+PM* FORMALISM

Istvan David
istvan.david@umontreal.ca

geodes SEMINAR

July 8, 2022

What is a process?

precedes, hasEdgeTo…

- Basic structure: partially ordered set of activities (A, ≤)
  - Typically: one init (*meet*) and one finish (*join*) node → lattice

- Workflow?
  - Pretty much the same as the process
  - *Perhaps* more emphasis on the repeatable activities (“work”)
  - Business context: always processes, often directly associated with business goals for controlling purposes (PDCA, DMAIC, OODA, etc.) – *cf.* **BP**MN, **BP**EL, etc.

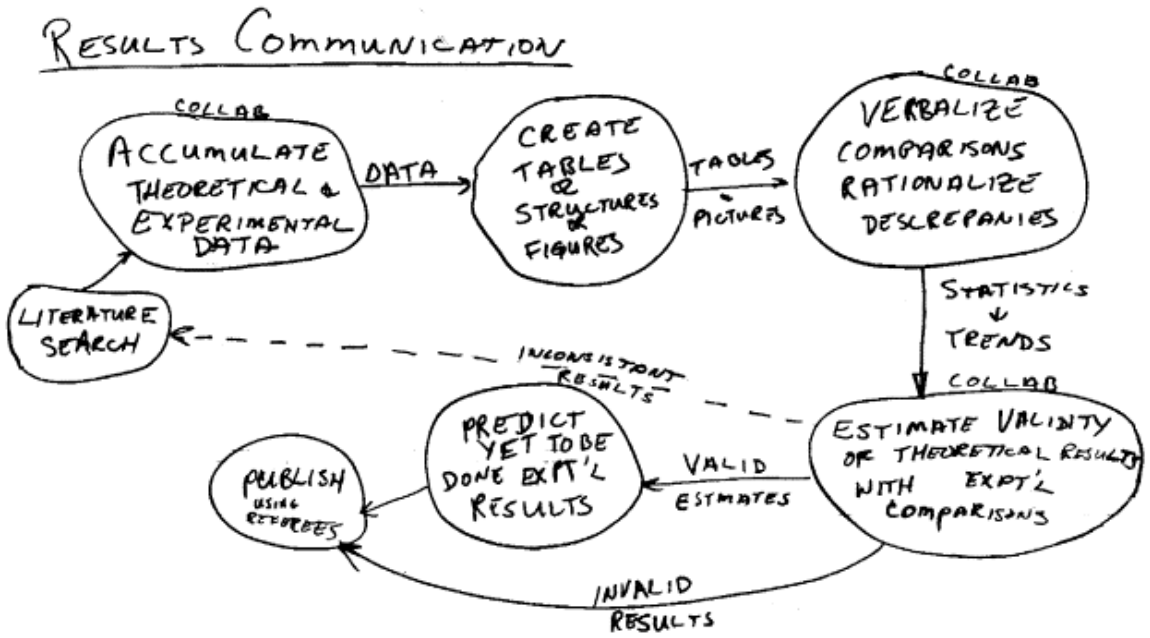
Process modeling

Why? We want to **be explicit** about the sequence of activities that help reaching a goal.

Process modeling

Why? We want to **be explicit** about the sequence of activities that help reaching a goal.

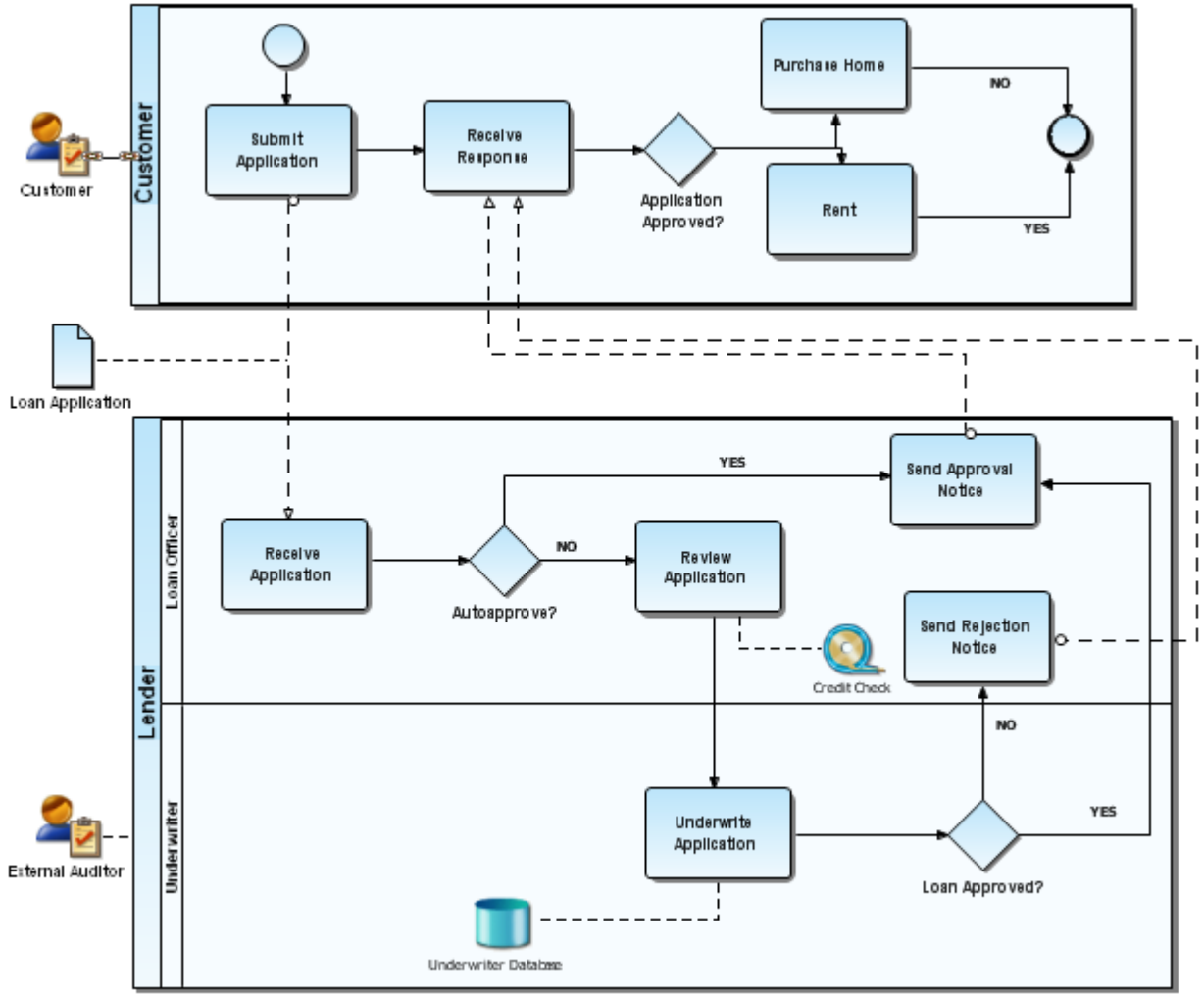
- Visualization, better understanding (visual inspection)



Process modeling

Why? We want to **be explicit** about the sequence of activities that help reaching a goal.

- Visualization, better understanding (visual inspection)
- Analysis, optimization



Process modeling

Why? We want to **be explicit** about the sequence of activities that help reaching a goal.

- Visualization, better understanding (visual inspection)
- Analysis, optimization
- Improving program design

```
12  module org.proxima.metamodels.generator
13
14  import org.eclipse.emf.mwe.utils.*
15  import org.eclipse.emf.mwe2.ecore.*
16
17  var projectName = "org.proxima.metamodels"
18  var processGenModelPath = "platform:/resource/${projectName}/model/metamodels.genmodel"
19  var enactmentGenModelPath = "platform:/resource/${projectName}/model/enactment.genmodel"
20  var codegenGenModelPath = "platform:/resource/${projectName}/model/codegen.genmodel"
21  var directory = "${projectName}/src"
22
23  Workflow {
24          bean = StandaloneSetup {
25                  scanClassPath = true
26                  platformUri = ".."
27                  registerGenModelFile = processGenModelPath
28                  registerGenModelFile = enactmentGenModelPath
29                  registerGenModelFile = codegenGenModelPath
30          }
31
32          component = DirectoryCleaner {
33                  directory = directory
34          }
35
36          component = org.eclipse.emf.mwe2.ecore.EcoreGenerator {
37          genModel = processGenModelPath
38          srcPath = "platform:/resource/${directory}"
39          }
40
41          component = org.eclipse.emf.mwe2.ecore.EcoreGenerator {
42          genModel = enactmentGenModelPath
43          srcPath = "platform:/resource/${directory}"
44          }
45
46          component = org.eclipse.emf.mwe2.ecore.EcoreGenerator {
47          genModel = codegenGenModelPath
48          srcPath = "platform:/resource/${directory}"
49          }
50  }
```

Process modeling formalisms

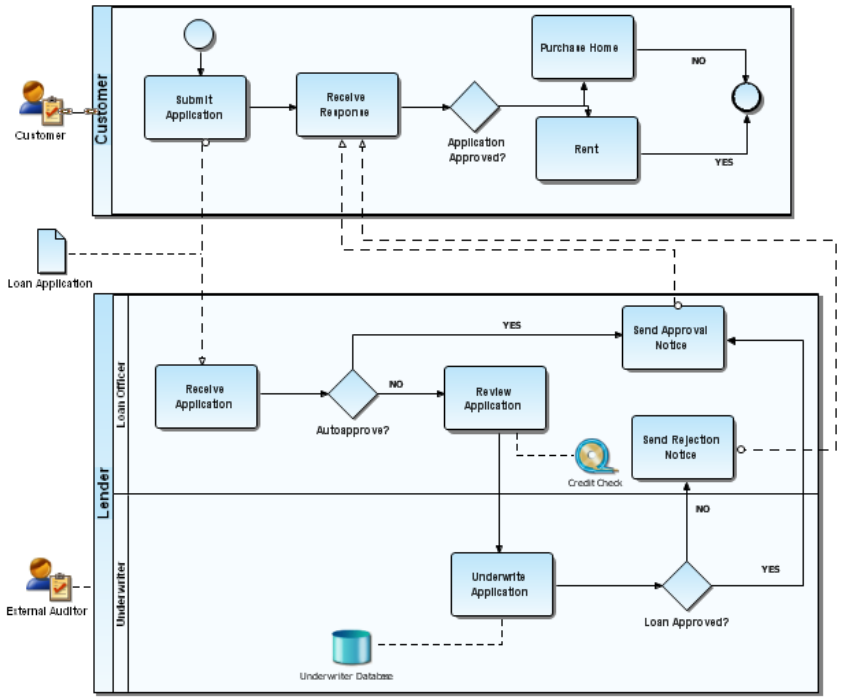
Formalism

=

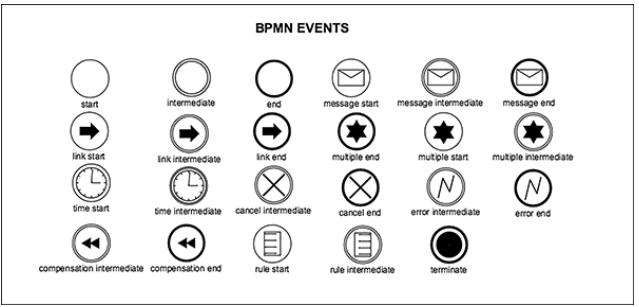
Language + Semantics

=

Concrete syntax + Abstract syntax + Semantics

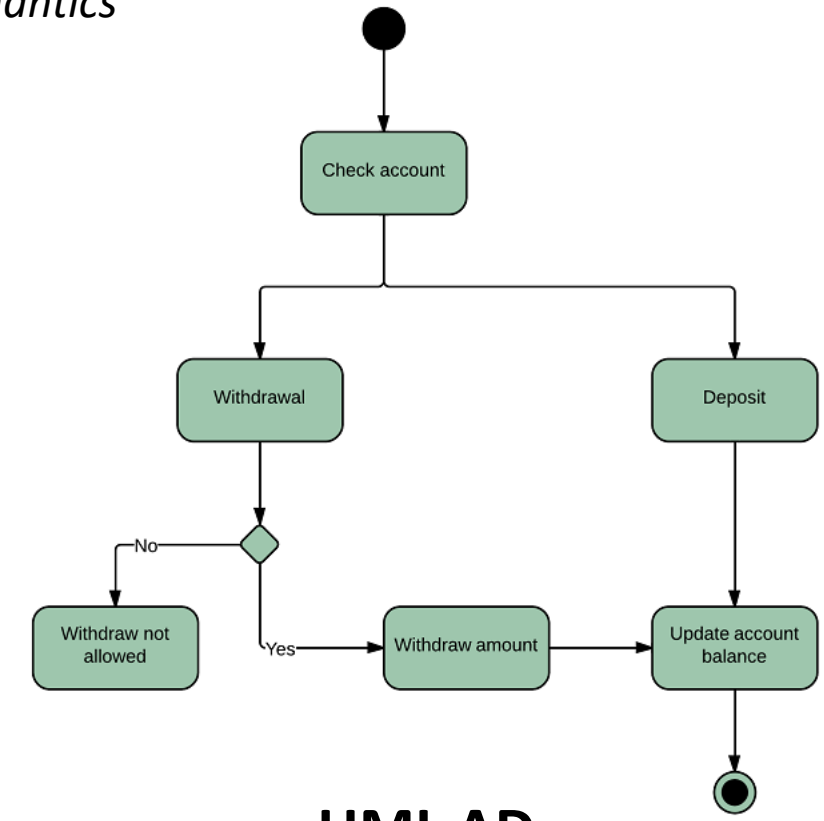


BPMN



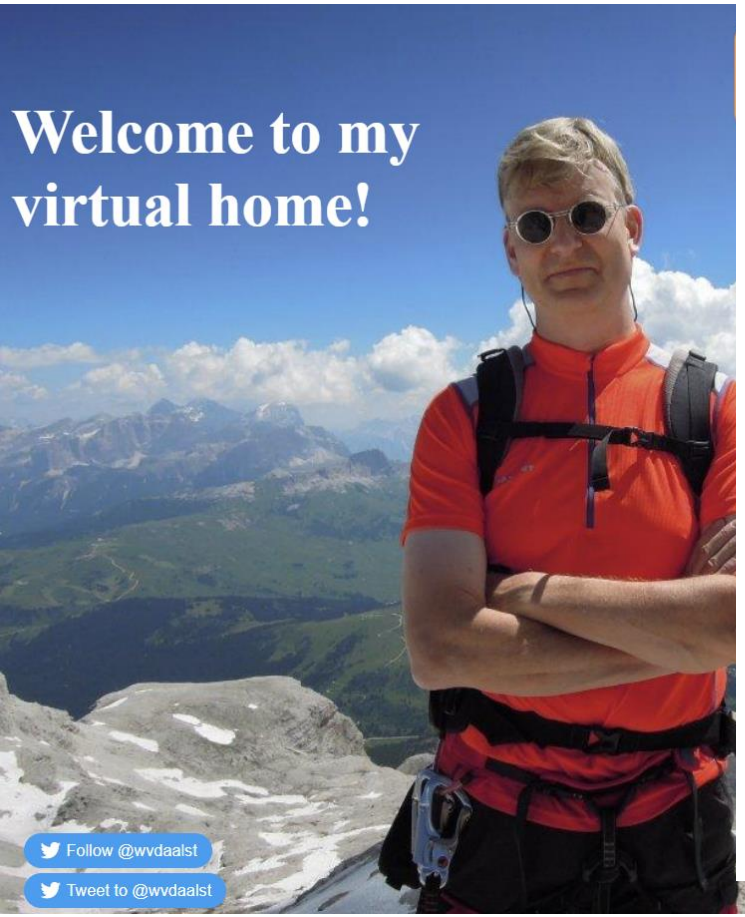

```
module org.proxima.metamodels.generator

import org.eclipse.emf.mwe.utils.*
import org.eclipse.emf.mwe2.ecore.*

var projectName = "org.proxima.metamodels"
var processGenModelPath = "platform:/resource/${projectName}/model/metamodels.genmodel"
var enactmentGenModelPath = "platform:/resource/${projectName}/model/enactment.genmodel"
var codegenGenModelPath = "platform:/resource/${projectName}/model/codegen.genmodel"
var directory = "${projectName}/src"

Workflow {
        bean = StandaloneSetup {
                scanClassPath = true
                platformUri = ".."
                registerGenModelFile = processGenModelPath
                registerGenModelFile = enactmentGenModelPath
                registerGenModelFile = codegenGenModelPath
        }

        component = DirectoryCleaner {
                directory = directory
        }

        component = org.eclipse.emf.mwe2.ecore.EcoreGenerator {
        genModel = processGenModelPath
        srcPath = "platform:/resource/${directory}"
        }

        component = org.eclipse.emf.mwe2.ecore.EcoreGenerator {
        genModel = enactmentGenModelPath
        srcPath = "platform:/resource/${directory}"
        }

        component = org.eclipse.emf.mwe2.ecore.EcoreGenerator {
        genModel = codegenGenModelPath
        srcPath = "platform:/resource/${directory}"
        }
}
```

Eclipse MWE



UML AD

van der Aalst Workflow patterns



WORKFLOW PATTERNS

HOME | PATTERNS | EVALUATIONS | VENDORS | ABOUT | LINKS | YAWL | IMPACT | DOCUMENTATION | CONTACTS

- Control
- Resource
- Data
- Exception Handling
- Presentation
- Log Imperfection

Welcome to the Workflow Patterns home page

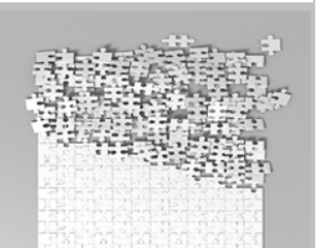
The Workflow Patterns initiative is a joint effort of Eindhoven University of Technology (led by Professor Wil van der Aalst) and Queensland University of Technology (led by Professor Arthur ter Hofstede) which started in 1999. The aim of this initiative is to provide a conceptual basis for process technology. In particular, the research provides a thorough examination of the various perspectives (control flow, data, resource, and exception handling) that need to be supported by a workflow language or a business process modelling language. The results can be used for examining the suitability of a particular process language or workflow system for a particular project, assessing relative strengths and weaknesses of various approaches to process specification, implementing certain business requirements in a particular process-aware information system, and as a basis for language and tool development.

On this web site you will find detailed descriptions of patterns for the various perspectives relevant for process-aware information systems: control-flow, data, resource, exception handling and event log imperfections. In addition you will find detailed evaluations of various process languages, (proposed) standards for web service compositions, and workflow systems in terms of this patterns.

We encourage interactions with interested parties about this research and its applications. For example, vendors can provide self-assessments of evaluations of their products (see the Vendors Corner). Also, we appreciate any feedback in relation to our evaluations (e.g. errors or inaccuracies).

IMPORTANT ANNOUNCEMENT: EVENT LOG IMPERFECTION PATTERN COLLECTION!

The quality of event data used in a process mining analysis impacts heavily on the accuracy and reliability of analysis results. The newly added Log Imperfection pattern collection includes 11 event log imperfection patterns, distilled from our experiences in preparing event logs, that neatly capture some commonly encountered quality issues associated with event logs. These patterns form the basis for a systematic approach to identifying and repairing event log quality issues which benefits both the process of preparing an event log and the quality of the resulting event log.



Wil van der Aalst

Université de Montréal

van der Aalst Workflow patterns

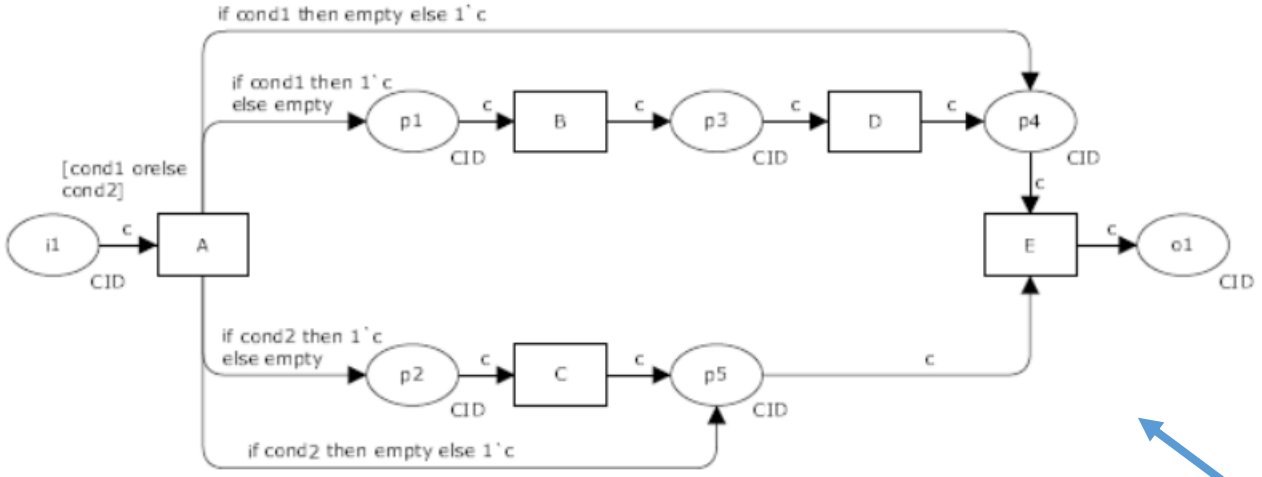


Figure 7: Structured synchronizing merge pattern

BPMN	1.0	+	Supported through t
XPDL	2.0	+	Supported by the Ol
UML ADs	2.0	-	Not supported. The

Advanced Branching and Synchronization Patterns

Here we present a series of patterns which characterise more complex branchi patterns are often not directly supported or even able to be represented in many *Merge, Multi-Merge and Discriminator.*

In this revision, the *Multi-Choice* and *Multi-Merge* have been retained in their prev been recognized that there are a number of distinct alternatives to the manner in *Synchronizing Merge* (WCP7), the *Acyclic Synchronizing Merge* (WCP37) and the *Ge*

In a similar vein, the original *Discriminator* pattern is divided into six (6) disti (WCP29), the *Structured Partial Join* (WCP30), the *Blocking Partial Join* (WCP31) an a more flexible AND-join useful in concurrent processes.

Of these patterns, the original descriptions for the *Synchronizing Merge* and the *Di*

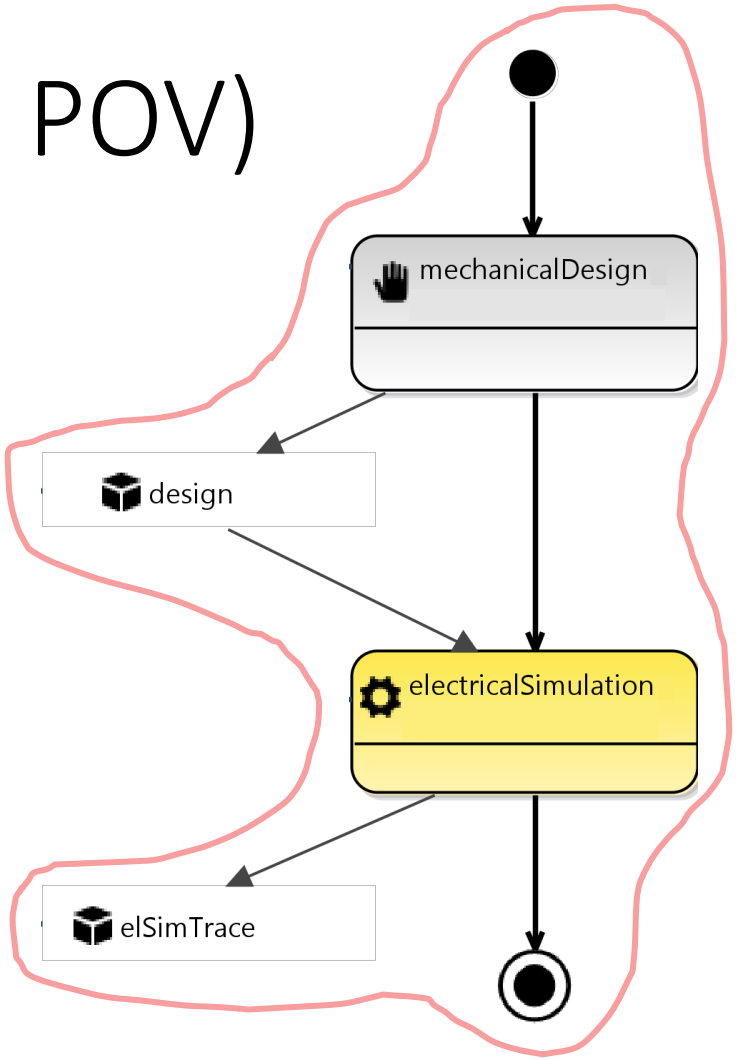
6. Multi-Choice
7. Structured Synchronizing Merge
8. Multi-Merge
9. Structured Discriminator
28. Blocking Discriminator
29. Cancelling Discriminator
30. Structured Partial Join
31. Blocking Partial Join
32. Cancelling Partial Join
33. Generalised AND-Join
37. Local Synchronizing Merge
38. General Synchronizing Merge
41. Thread Merge
42. Thread Split

Université de Montréal

Engineering processes

Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…

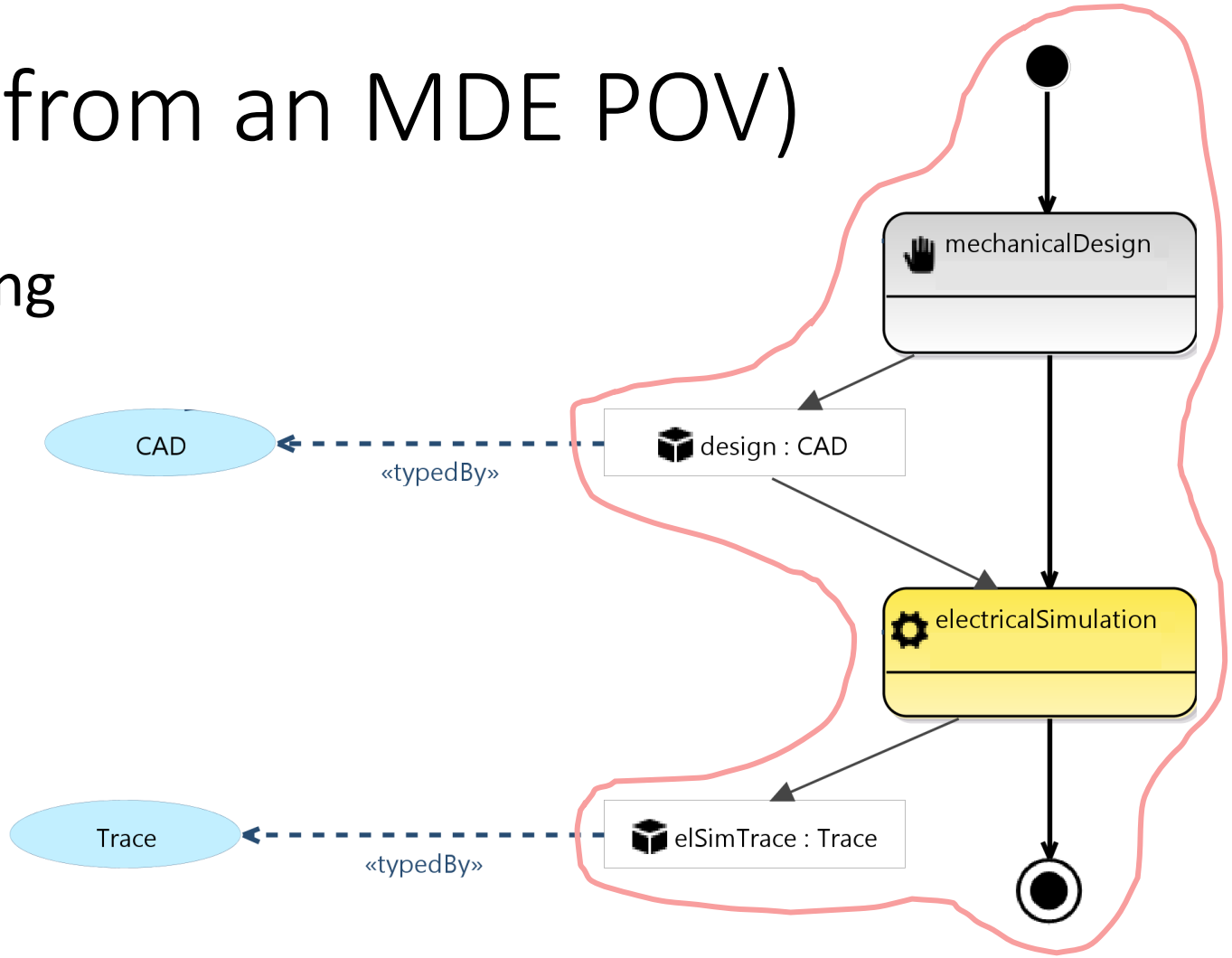


Process Model

Syntax: UML AD

Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…

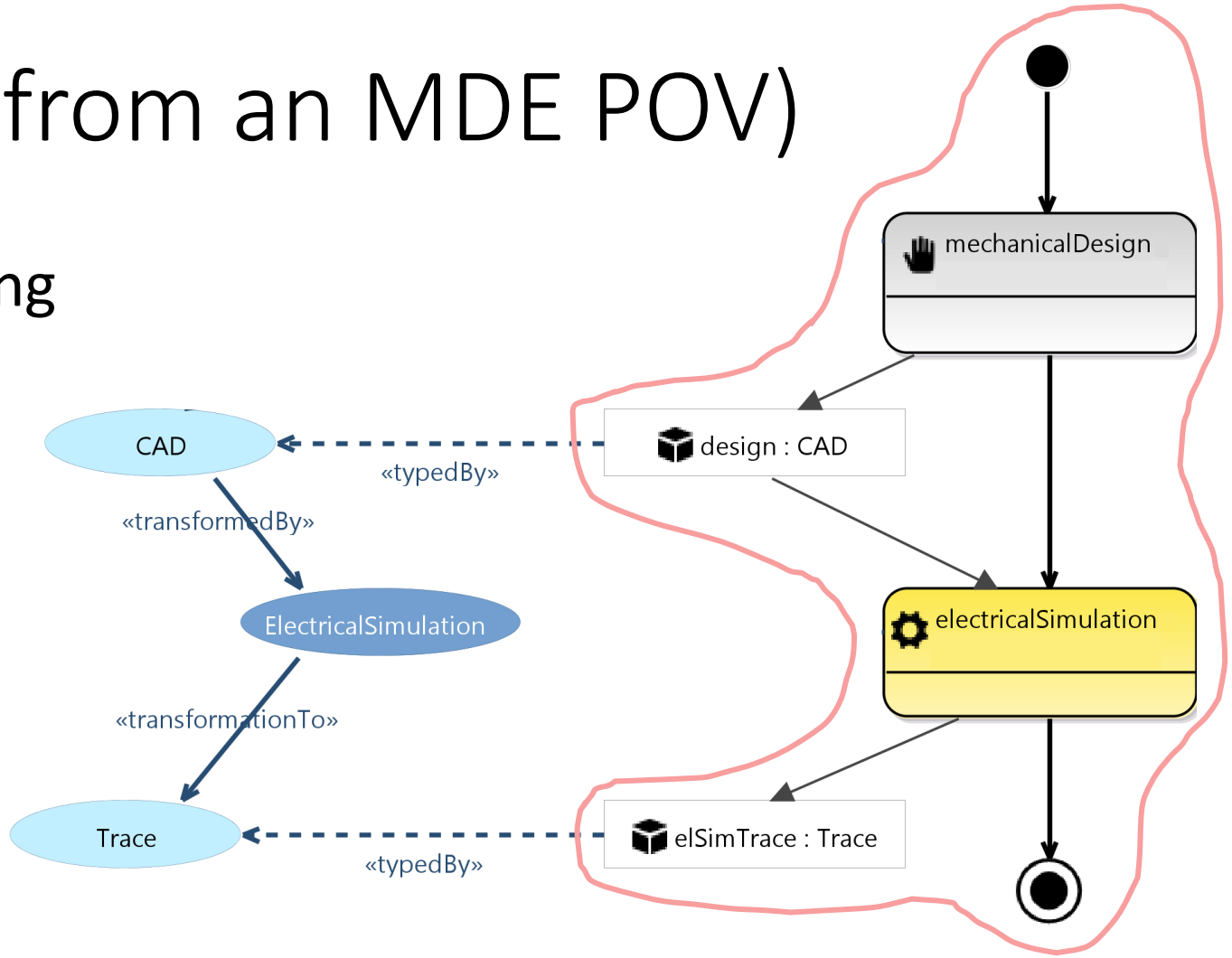


Process Model

Syntax: UML AD

Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…

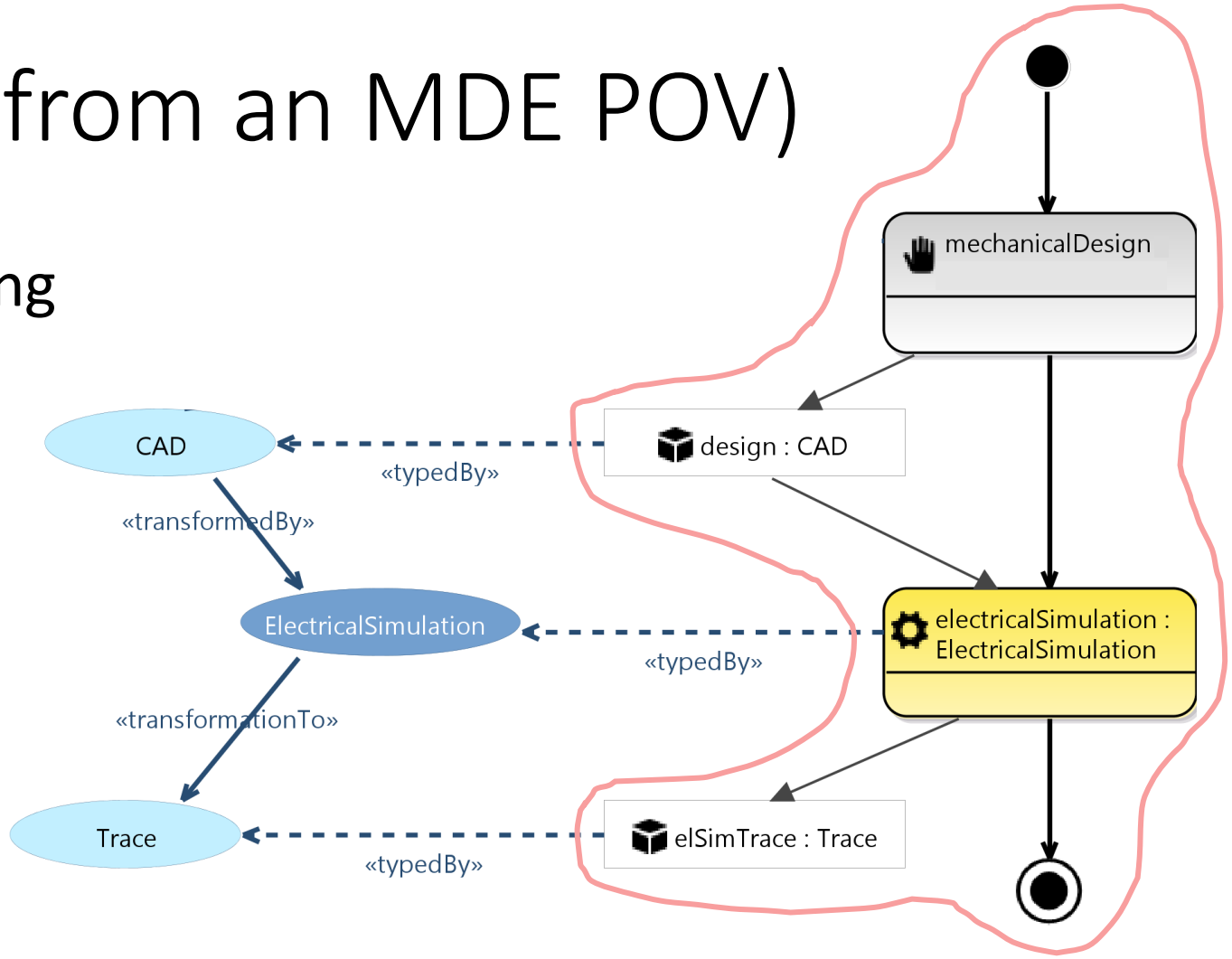


Process Model

Syntax: UML AD

Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…

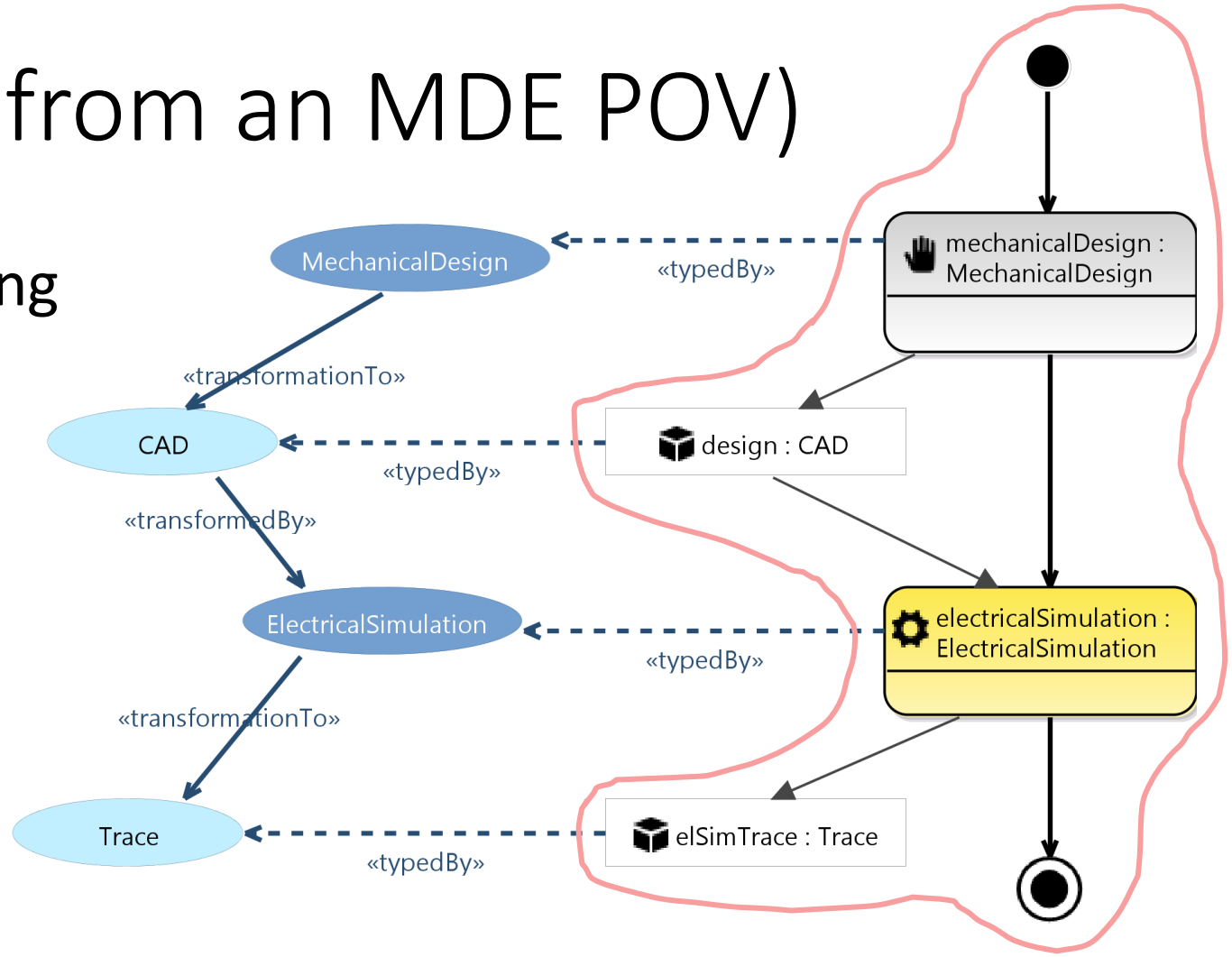


Process Model

Syntax: UML AD

Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…

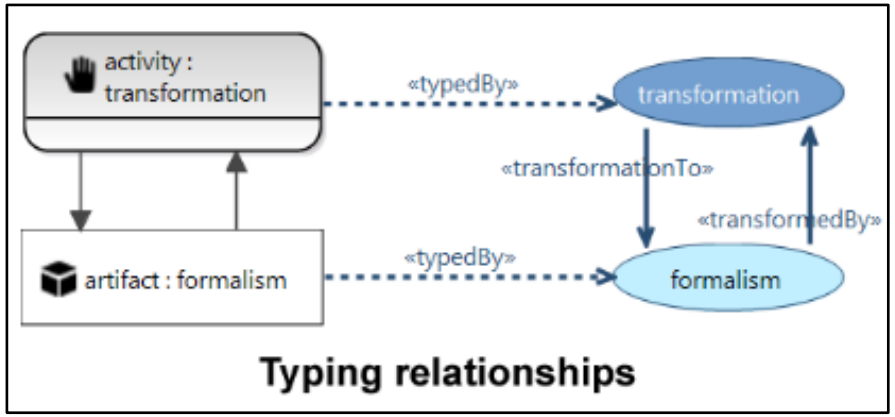


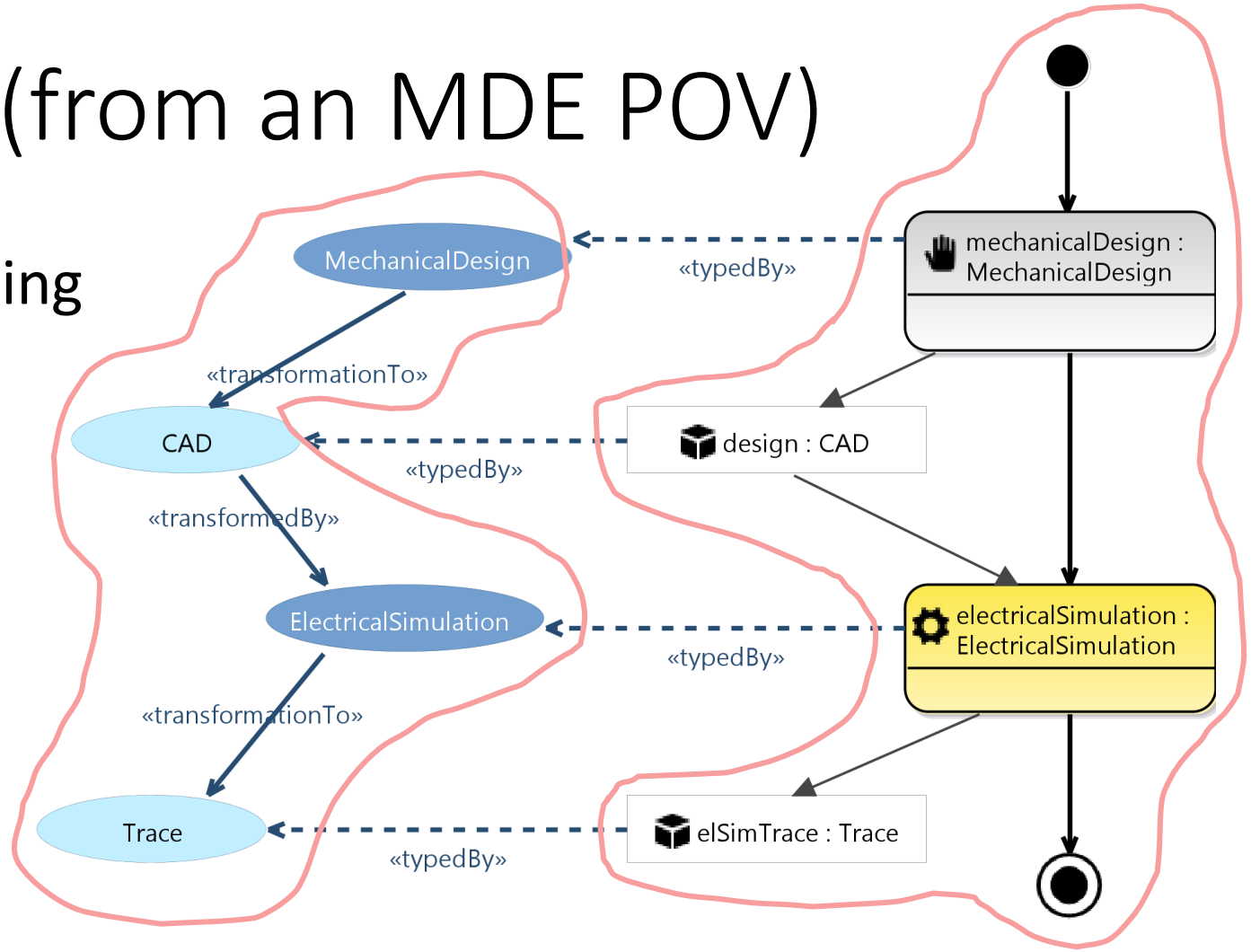
Process Model

Syntax: UML AD

Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…





Formalism Transformation Graph

Process Model

FTG+PM

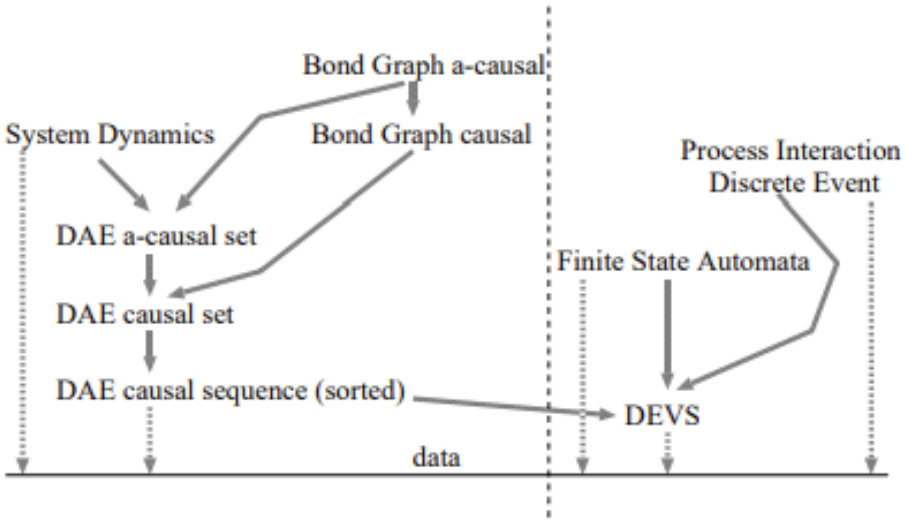
Syntax: UML AD

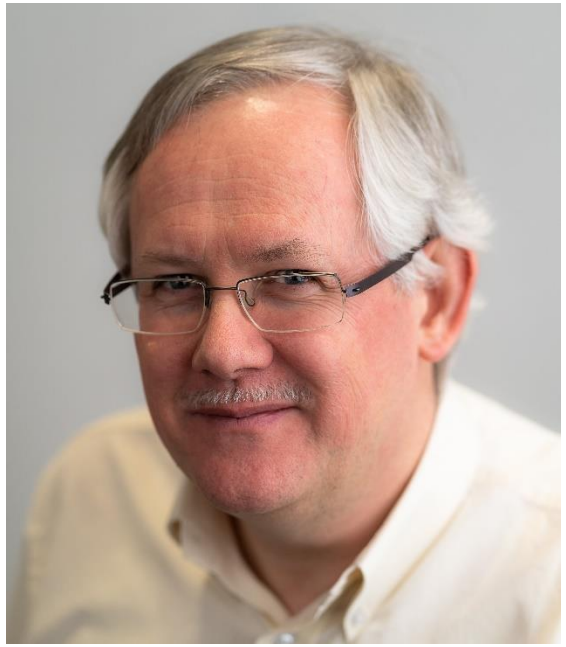
Lúcio, L., Mustafiz, S., Denil, J., Meyers, B., & Vangheluwe, H. (2012). The formalism transformation graph as a guide to model driven engineering. *School Comput. Sci., McGill Univ., Tech. Rep. SOCS-TR2012*, *1*.

Application #0: Multi-paradigm modeling (MPM)

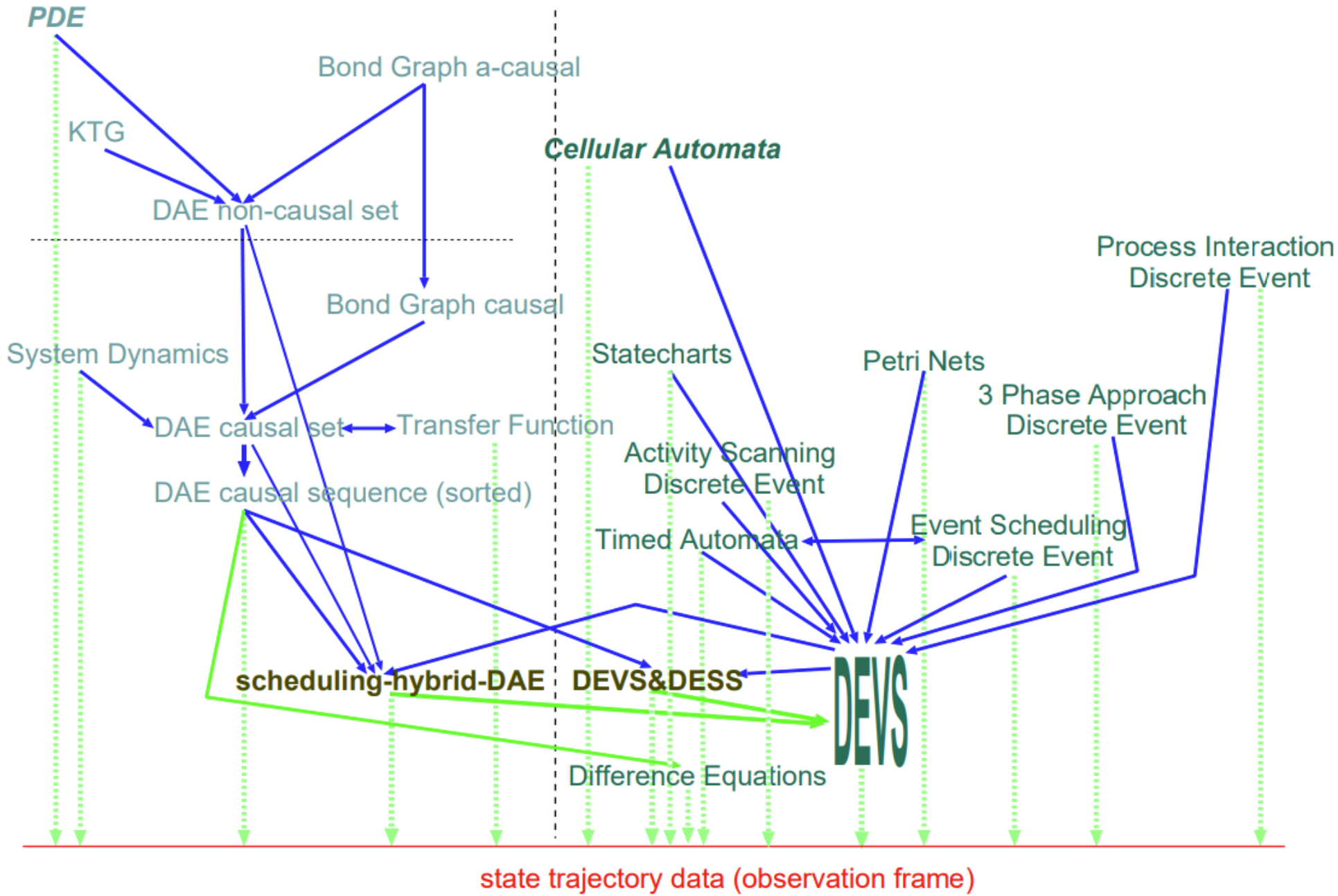
> *Transformation between formalism has different uses.*
> *Firstly, certain questions about a system can only be answered in certain formalisms, necessitating **model transformation** to the appropriate formalism.*
> *Secondly, to elicit the meaning of a coupled model consisting of sub-models in different formalisms, transformation of these sub-models to a **common formalism** is appropriate.*

Formalism Transformation Lattice





Vangheluwe, H., and Vansteenkiste, G.. "A multi-paradigm modeling and simulation methodology: Formalisms and languages." In *Simulation in Industry, Proceedings 8th European Simulation Symposium*, pp. 168-172. 1996.



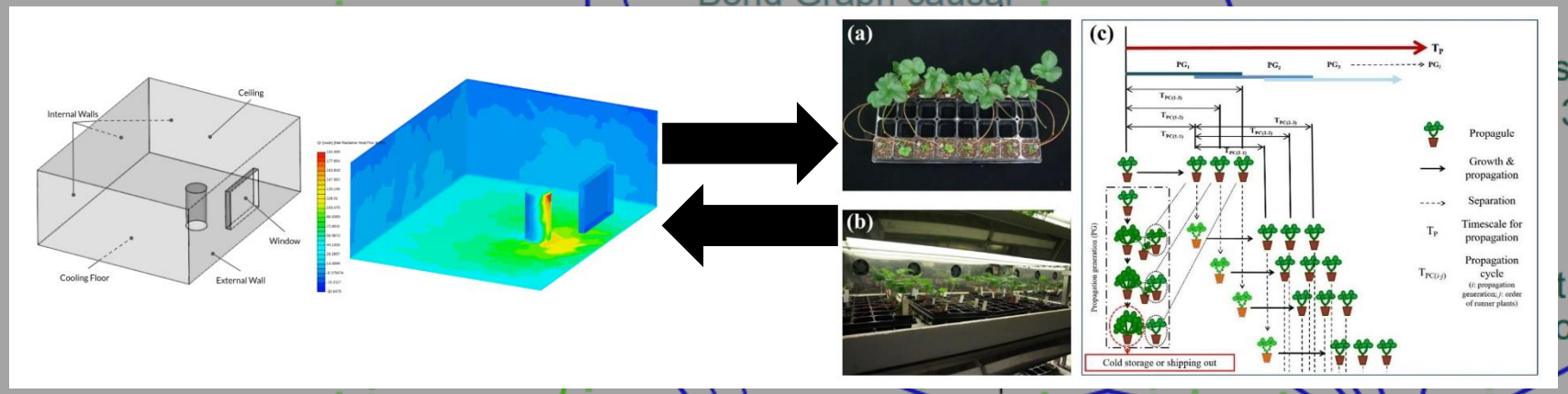
de Lara, J., and Vangheluwe, H. (2002, October). Computer aided multi-paradigm modelling to process Petri-nets and Statecharts. In *International Conference on Graph Transformation* (pp. 239-253). Springer, Berlin, Heidelberg.

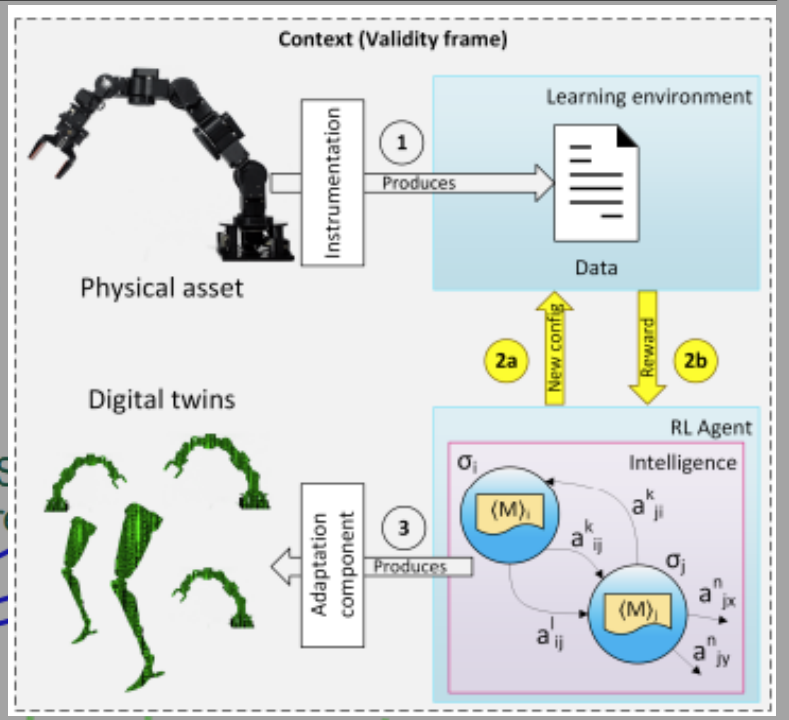
DEVS as a Common Denominator for Multi-formalism Hybrid Systems Modelling

Hans L.M. Vangheluwe
School of Computer Science, McGill University
McConnell Engineering Bldg. room 328
3480 University Street
Montreal, Quebec, Canada H3A 2A7
hv@cs.mcgill.ca

DEVS MODEL CONSTRUCTION AS A REINFORCEMENT LEARNING PROBLEM

Istvan David
Eugene Syriani
Department of Computer Science and Operations Research (DIRO) – Université de Montréal, Canada
istvan.david@umontreal.ca, syriani@iro.umontreal.ca



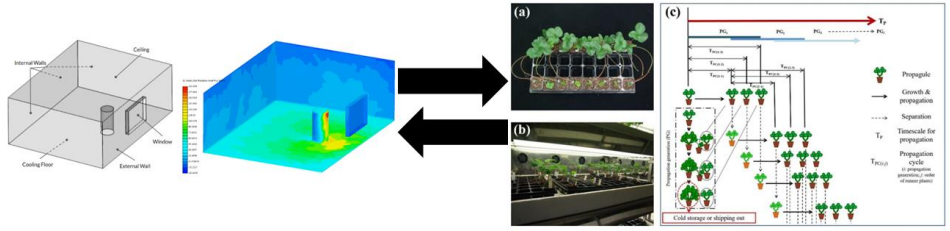


DEVS as a Semantic Domain for Programmed Graph Transformation

Eugene Syriani and Hans Vangheluwe

McGill University, School of Computer Science, Montréal, Canada,
{esyria,hv}@cs.mcgill.ca

luwe, H. (2002, October). Computer aided multi-paradigm modelling to process Petri-nets and Statecharts. *nce on Graph Transformation* (pp. 239-253). Springer, Berlin, Heidelberg.



PDE

KTG

Bond Graph a-causal

Cellular Automata

DAE non-causal set

Bond Graph causal

System Dynamics

Statecharts

Petri Nets

Process Interaction
Discrete Event

3 Phase Approach
Discrete Event

DAE causal set ↔ Transfer Function

DAE causal sequence (sorted)

Activity Scanning
Discrete Event

Timed Automata

Event Scheduling
Discrete Event

scheduling-hybrid-DAE

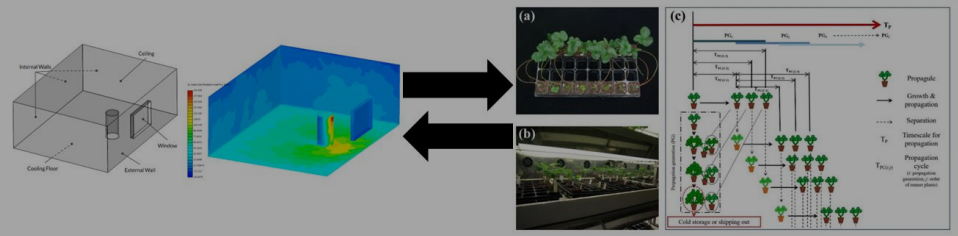
DEVS&DESS

DEVS

Difference Equations

state trajectory data (observation frame)

de Lara, J., and Vangheluwe, H. (2002, October). Computer aided multi-paradigm modelling to process Petri-nets and Statecharts. In *International Conference on Graph Transformation* (pp. 239-253). Springer, Berlin, Heidelberg.



PDE

Bond Graph a-causal

KTG

Cellular Automata

DAE non-causal set

Process Interaction

MPM:
Model everything explicitly,
at the most appropriate level(s) of abstraction,
using the most appropriate modelling formalism(s)
and processes.

DEVS

Difference Equations

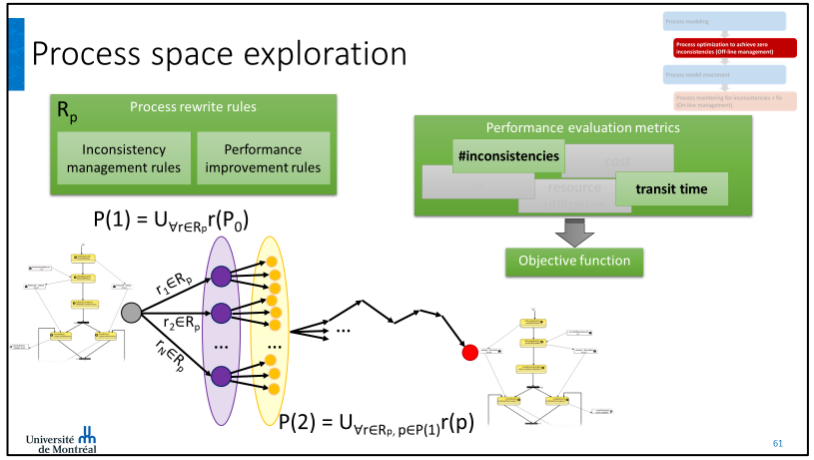
state trajectory data (observation frame)

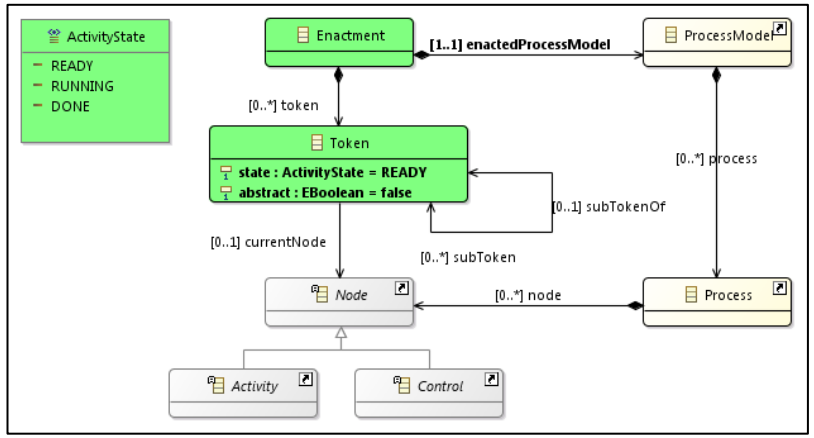
de Lara, J., and Vangheluwe, H. (2002, October). Computer aided multi-paradigm modelling to process Petri-nets and Statecharts. In *International Conference on Graph Transformation* (pp. 239-253). Springer, Berlin, Heidelberg.

Other applications

Model transformation chains

- Analysis, optimization, execution
- Reuse, traceability, certification, etc

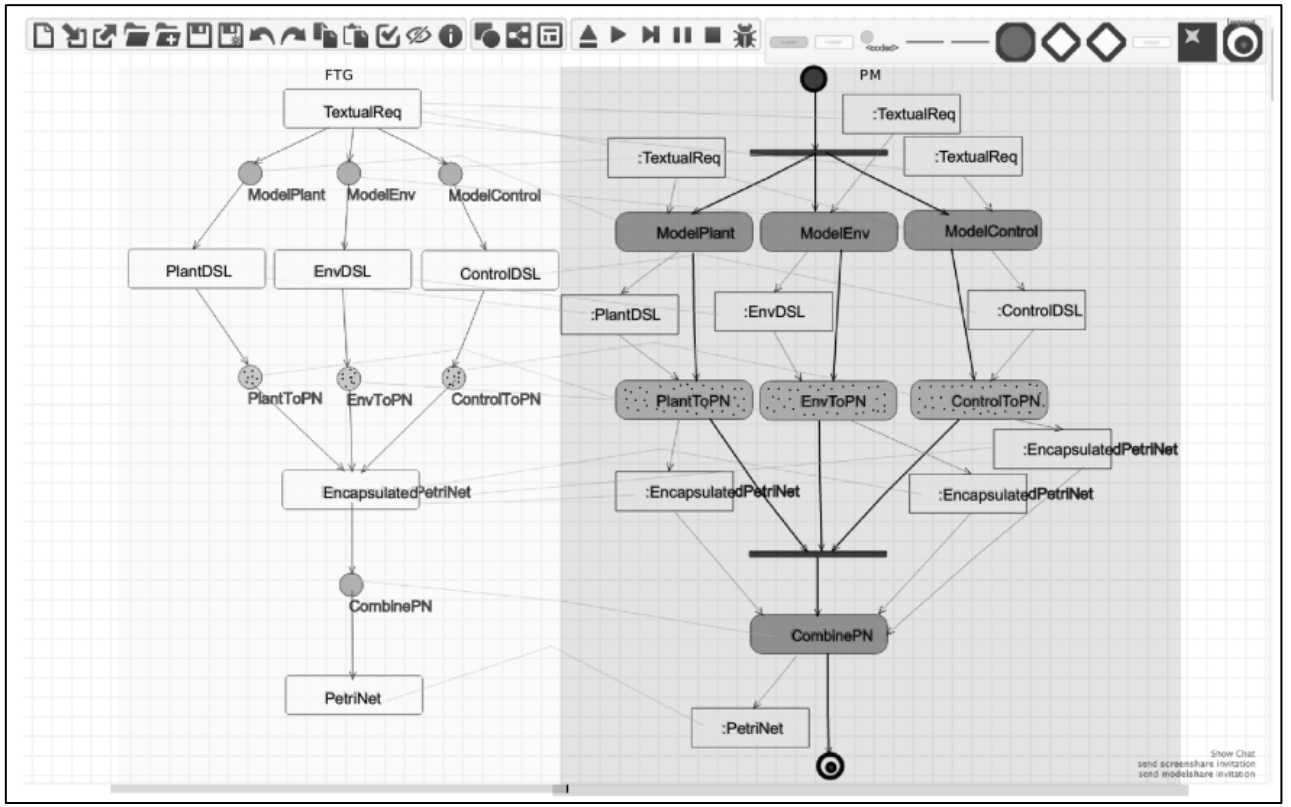




FTG+PM: An Integrated Framework for Investigating Model Transformation Chains

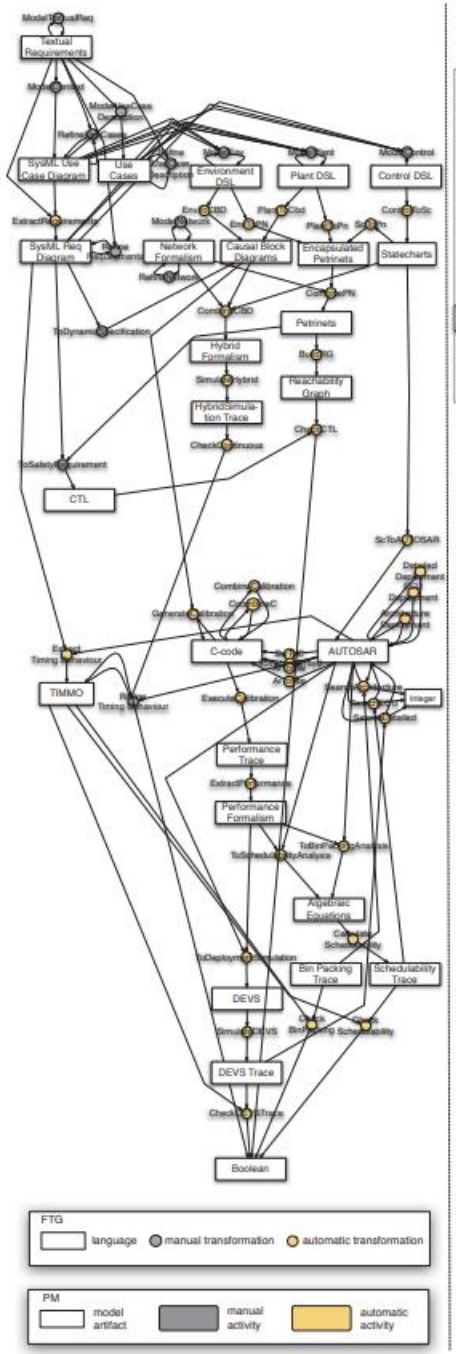
Levi Lúcio[1], Sadaf Mustafiz[1], Joachim Denil[2,1], Hans Vangheluwe[2,1], and Maris Jukss[1]

[1] School of Computer Science, McGill University, Canada
{levi,sadaf,joachim.denil,hv,mjukss}@cs.mcgill.ca
[2] University of Antwerp, Belgium
{joachim.denil,hans.vangheluwe}@ua.ac.be



The Power Window case study

- Goal: development of a Power Window software controller
  - Complex embedded system: time-critical, safety-critical, hard real-time
  - Characteristic elements: requirements, architecture, plant-environment-control, verification via PN, hybrid behavioral simulation, deployment (AUTOSAR SWCs), code gen for ECUs



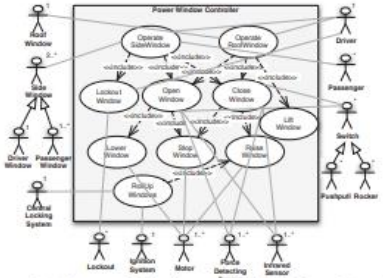


Figure 4: Power Window: SysML Use Case Diagra

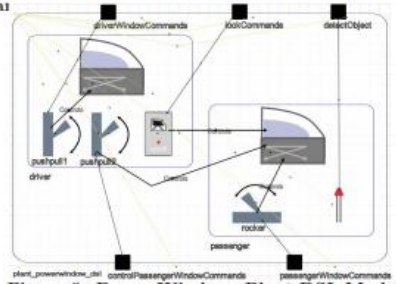


Figure 5: Power Window: Plant DSL Model

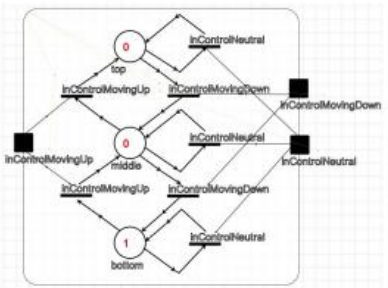


Figure 6: Power Window: Petri Net for Driver Window

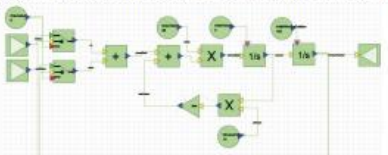


Figure 7: Power Window: Causal Block Diagram (CBD) for Plant Model

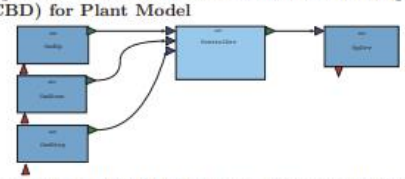


Figure 8: AUTOSAR Software Component Model

Université de Montréal

DEVS as a Semantic Domain for Programmed Graph Transformation

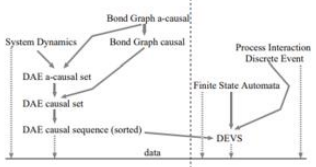
Eugene Syriani and Hans Vangheluwe

McGill University, School of Computer Science, Montréal, Canada,
{esyria,hv}@cs.mcgill.ca

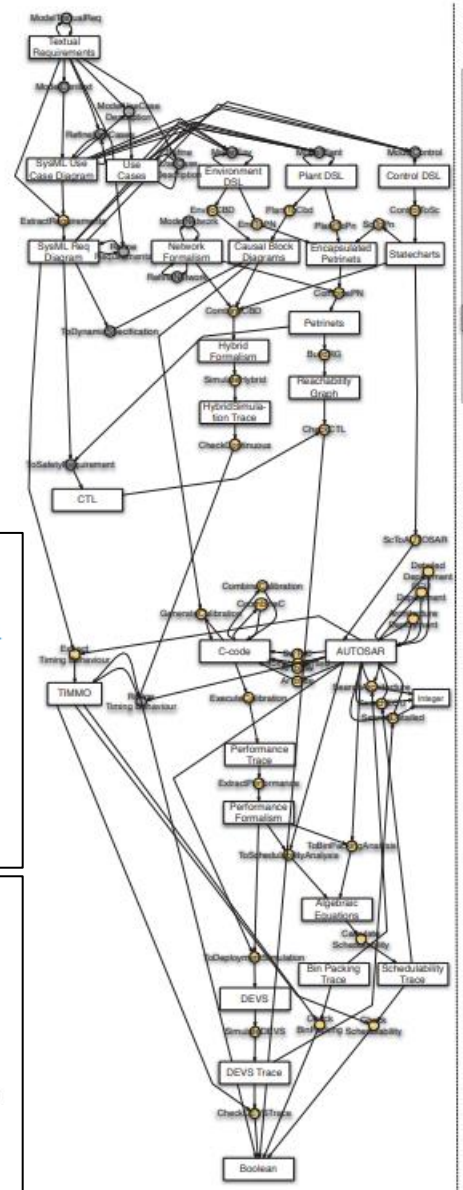
Application #0: Multi-paradigm modeling (MPM)

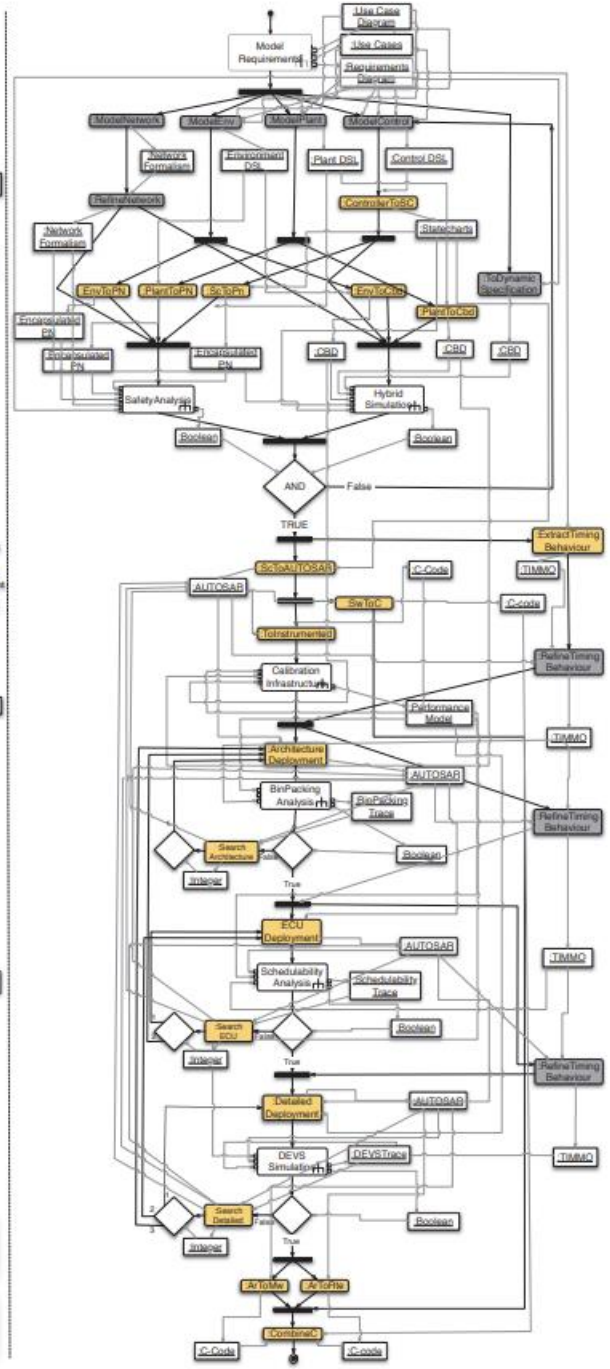
Transformation between formalism has different uses.
Firstly, certain questions about a system can only be answered in certain formalisms, necessitating ***model transformation*** *to the appropriate formalism.*
Secondly, to elicit the meaning of a coupled model consisting of sub-models in different formalisms, transformation of these sub-models to a ***common formalism*** *is appropriate.*

Formalism Transformation Lattice



Vangheluwe, H., and Vansteenkiste, G.. "A multi-paradigm modeling and simulation methodology: Formalisms and languages." In *Simulation in Industry, Proceedings 8th European Simulation Symposium*, pp. 168-172. 1996.





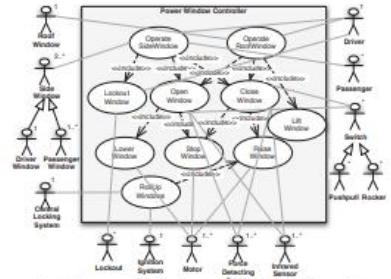


Figure 4: Power Window: SysML Use Case Diagra

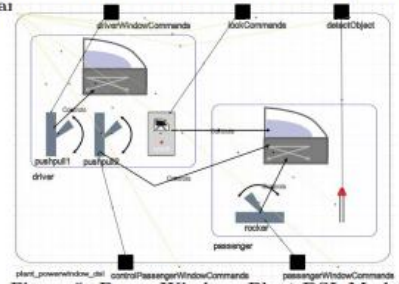


Figure 5: Power Window: Plant DSL Model

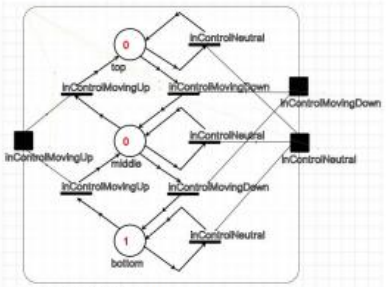


Figure 6: Power Window: Petri Net for Driver Window

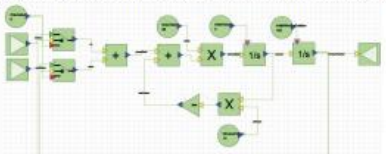


Figure 7: Power Window: Causal Block Diagram (CBD) for Plant Model

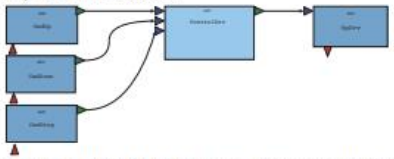
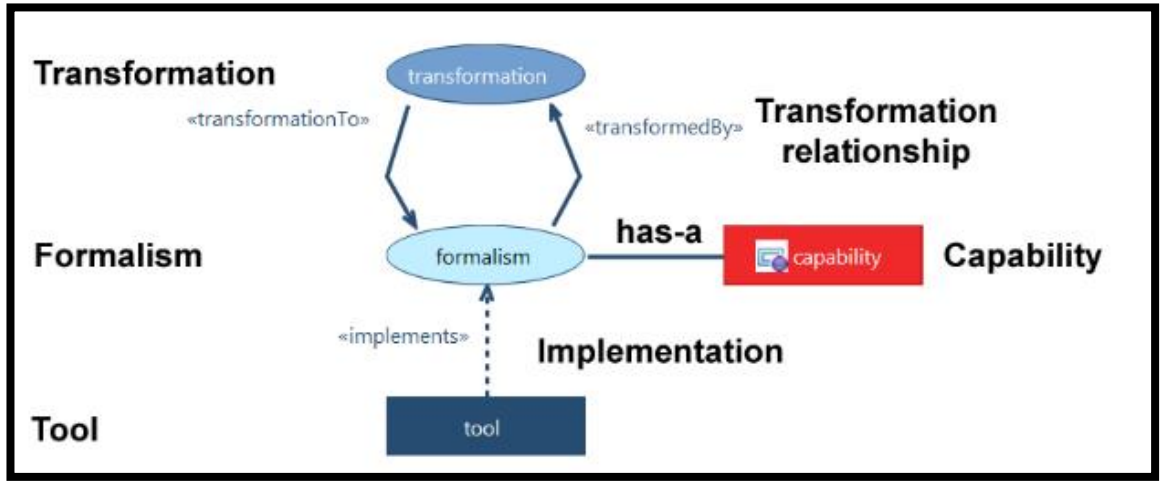
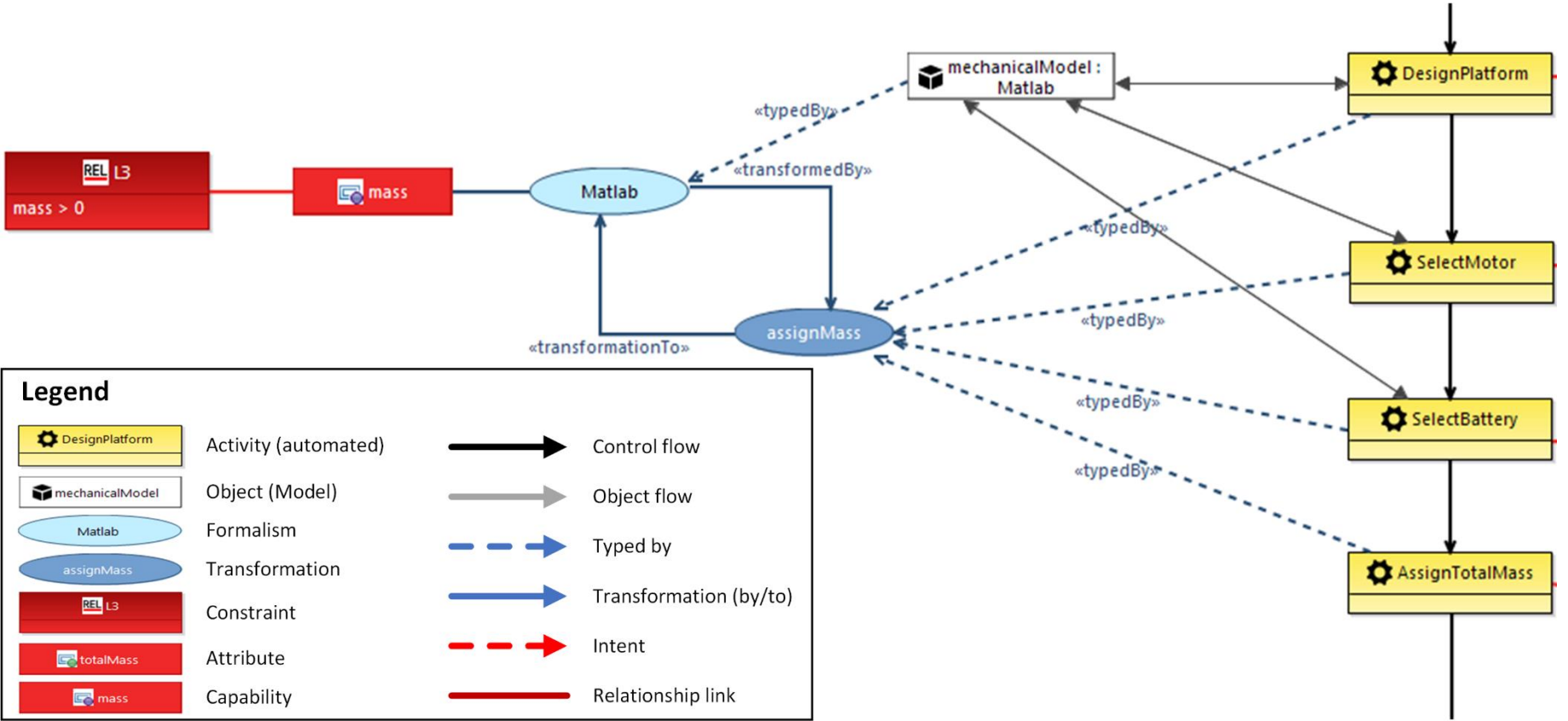


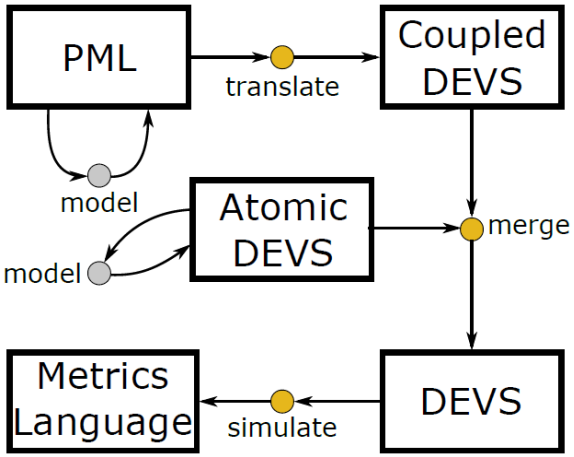
Figure 8: AUTOSAR Software Component Model

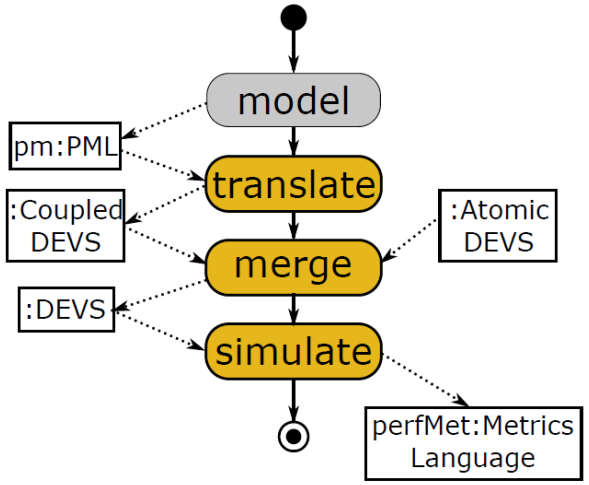
Tool orchestration



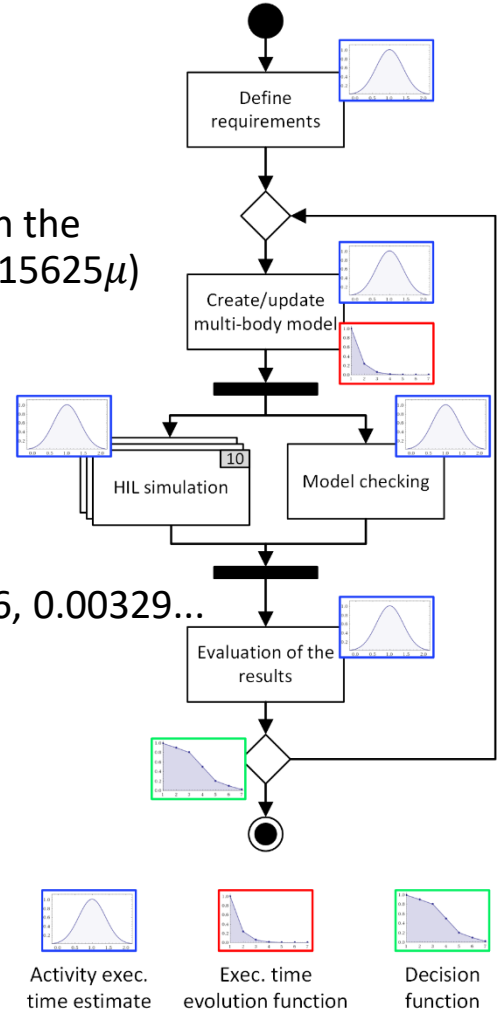


Performance evaluation of FTG+PM instances



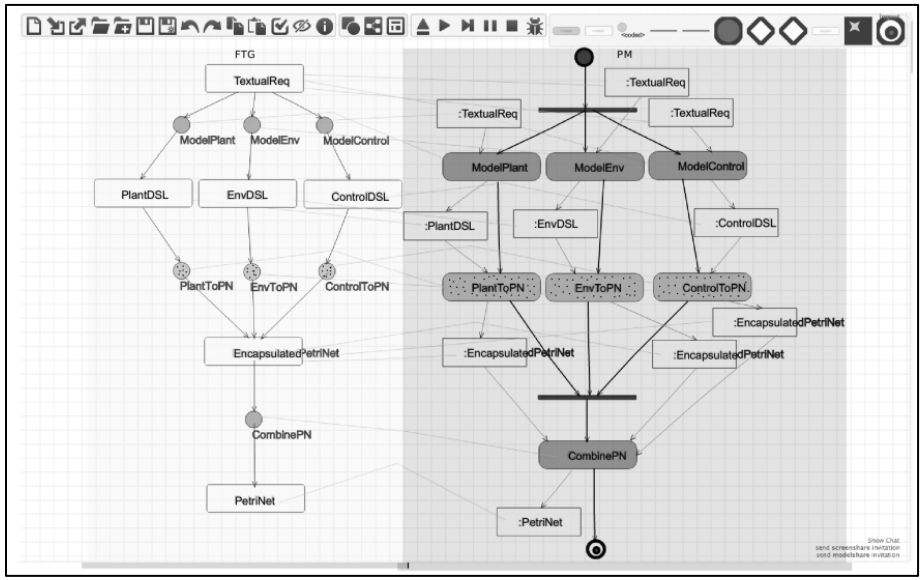


- Activity execution time
  - Rule
    - Gaussian distribution
    - 80% of the estimations within the 20% error range: t(a)=N(μ, 0.15625μ)
- Execution time evolution
  - Rule
    - $e^{-1/0.7i}$ (i: iteration)
  - Resulting values
    - 1.0, 0.2397, 0.05743, 0.01376, 0.00329...
- Decision function
  - Manually set
  - 0.99, 0.9, 0.8, 0.5, 0.2, 0.1

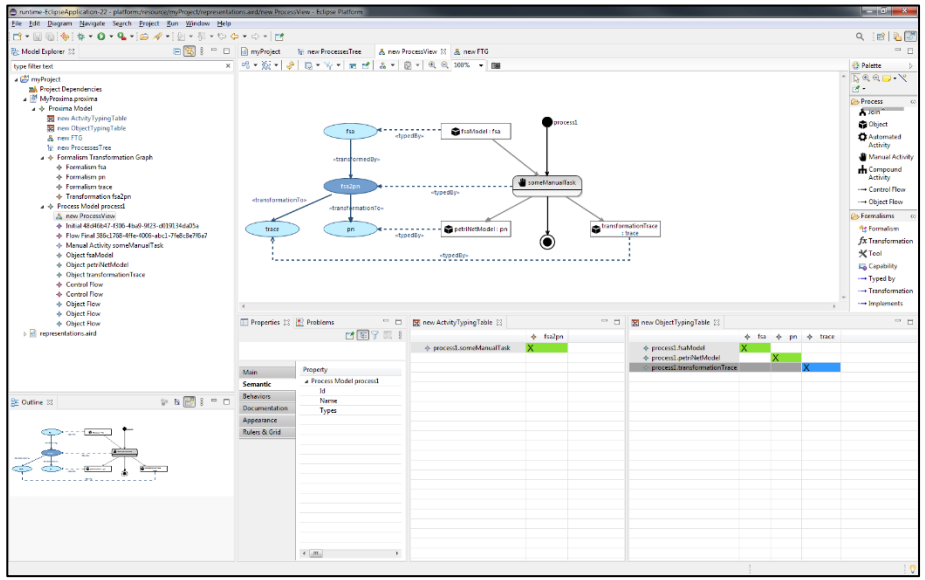


I. Dávid, Y. Van Tendeloo, H. Vangheluwe: „*Translating Engineering Workflow Models to DEVS for Performance Evaluation*”. In Proceedings of the 2018 Winter Simulation Conference, pp. 616-627. IEEE Press, 2018.

Tool support



AToMPM



PROxIMA

Modelverse specification

Yentl Van Tendeloo
Bruno Barroca
Simon Van Mierlo
Hans Vangheluwe

August 31, 2016

https://atompm.github.io/
http://proximatools.org/

Université de Montréal

Some limitations and open questions

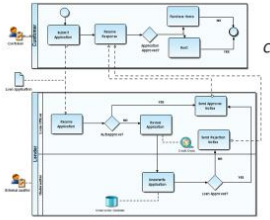
Potential topics to work on

- Inheritance/conformance in the FTG
- DEVS-based enactment
- Change management FTG
- Shortest path problems

- Process inference via the FTG
  - Process recommendations, process reuse
- FTG for machine learning
  - For the better understanding and generative construction of ML pipelines

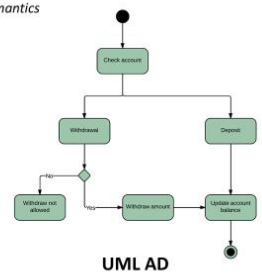
Process modeling formalisms

Formalism
=
Language + Semantics
=
Concrete syntax + Abstract syntax + Semantics



BPMN

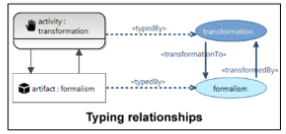
Eclipse MWE



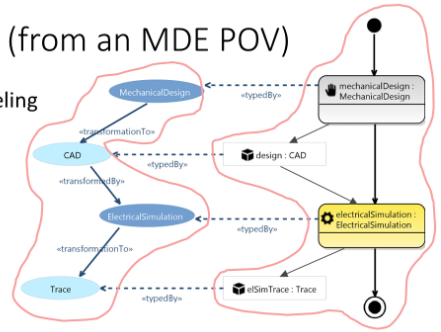
UML AD

Engineering processes (from an MDE POV)

- More emphasis on models, modeling formalisms, levels of abstraction, dependencies and traceability…



Typing relationships



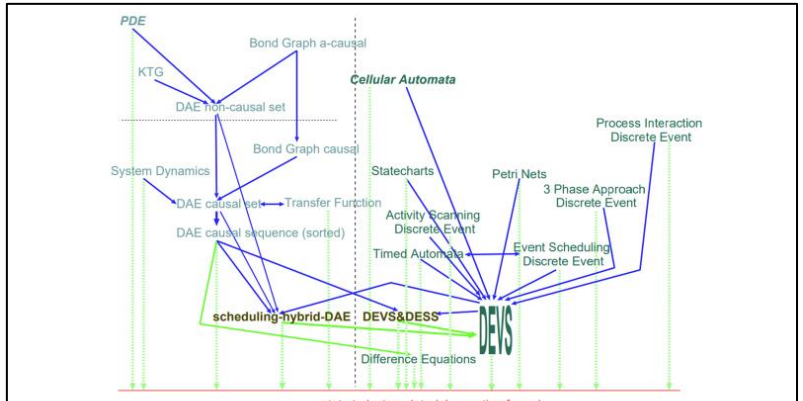
Formalism Transformation Graph

Process Model

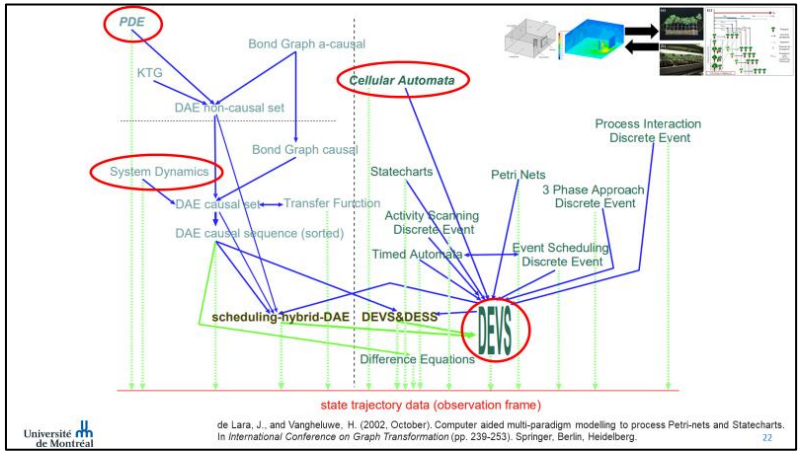
Syntax: UML AD

FTG+PM

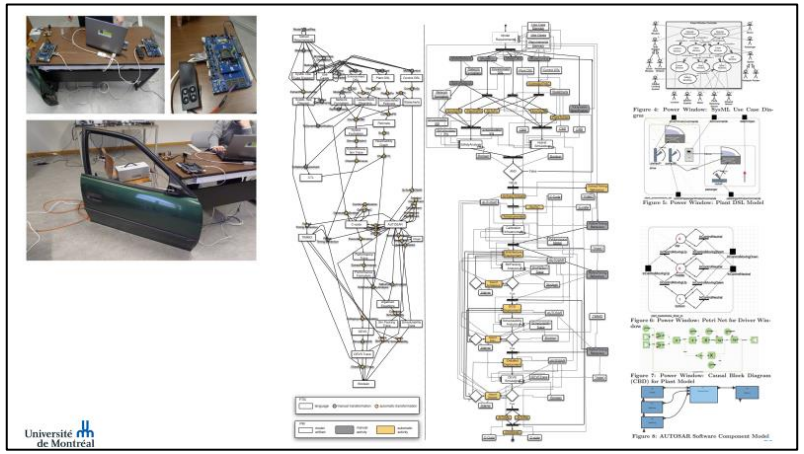
Lúcio, L., Mustafiz, S., Denil, J., Meyers, B., & Vangheluwe, H. (2012). The formalism transformation graph as a guide to model driven engineering. *School Comput. Sci., McGill Univ., Tech. Rep. SOCS-TR2012, 1.*



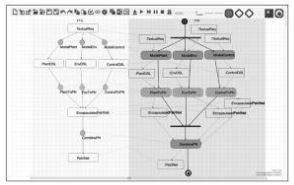
de Lara, J., and Vangheluwe, H. (2002, October). Computer aided multi-paradigm modelling to process Petri-nets and Statecharts. In *International Conference on Graph Transformation* (pp. 239-253). Springer, Berlin, Heidelberg.



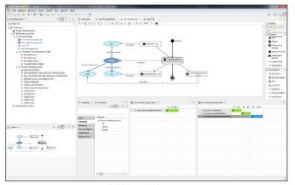
de Lara, J., and Vangheluwe, H. (2002, October). Computer aided multi-paradigm modelling to process Petri-nets and Statecharts. In *International Conference on Graph Transformation* (pp. 239-253). Springer, Berlin, Heidelberg.



Tool support



AToMPM



PROxIMA

https://atompm.github.io/
http://proximatools.org/