Multi-Paradigm Modelling and Language Engineering

Hans Vangheluwe

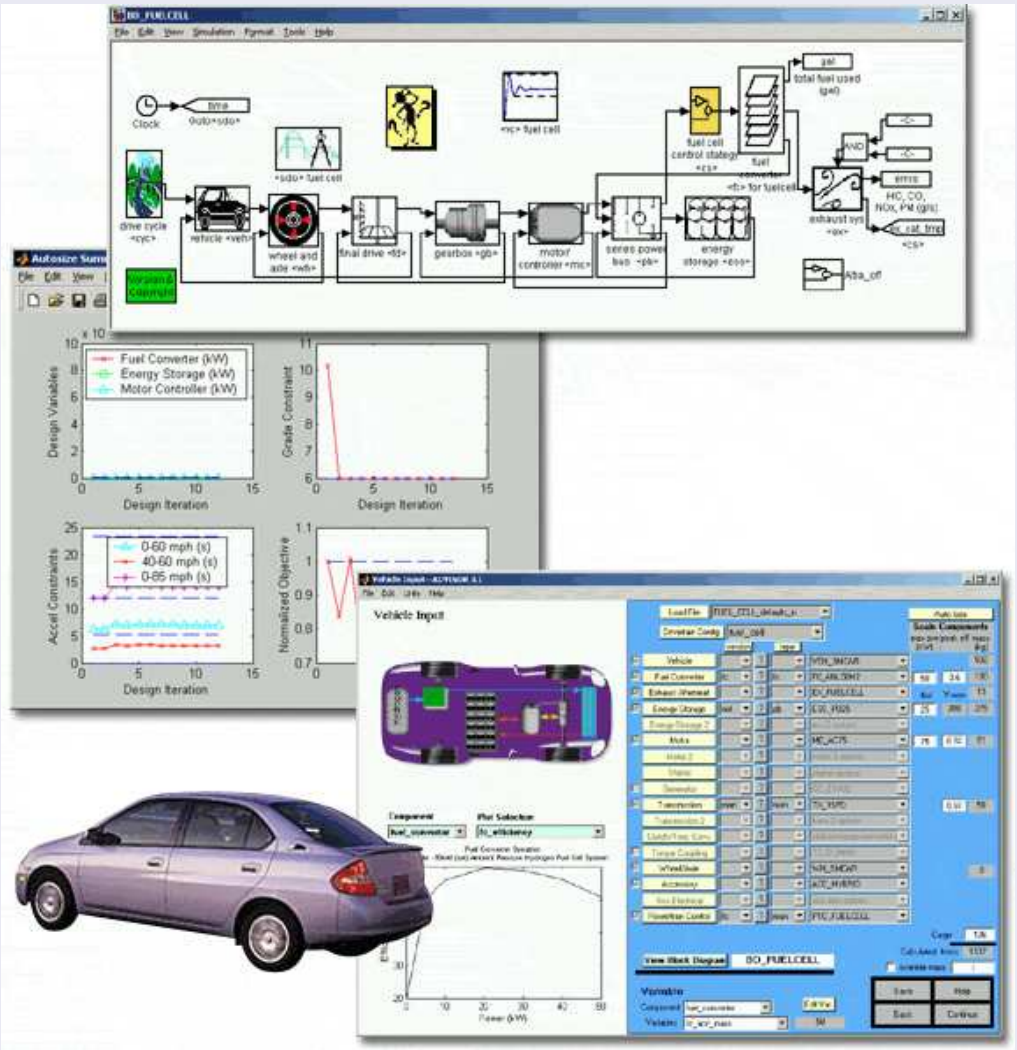
McGill

Modelling, Simulation and Design Lab (MSDL)
School of Computer Science, McGill University,
Montréal, Canada

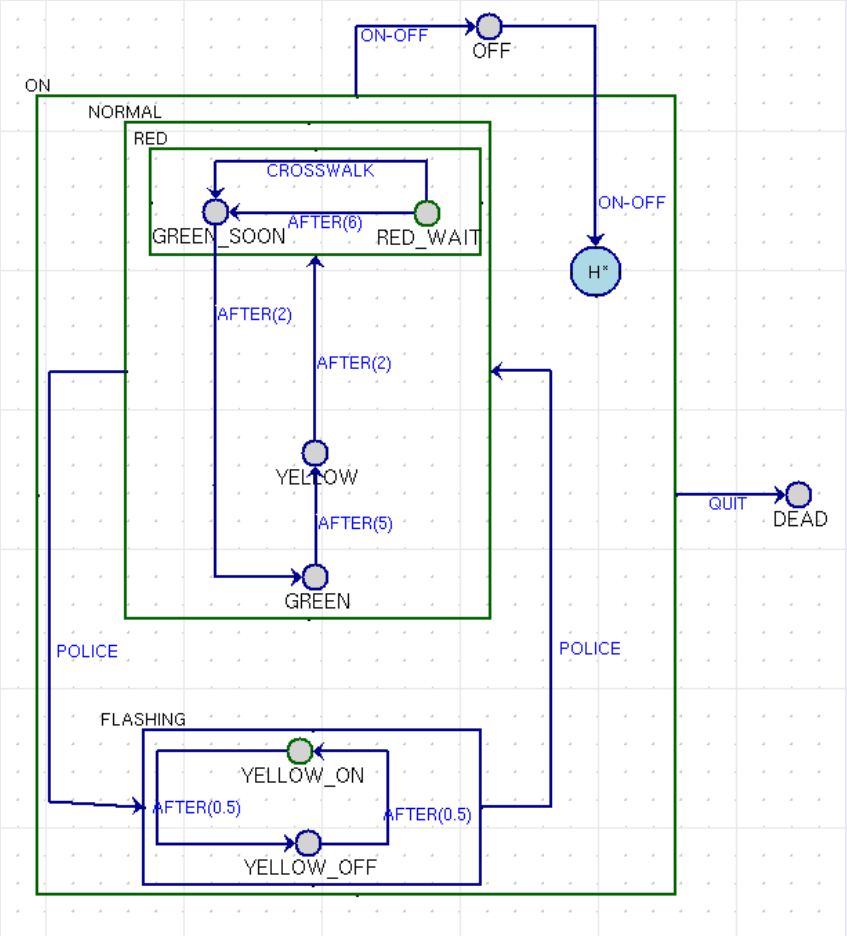
CAMPaM Workshop
7 April 2009

Bellairs Research Institute, Barbados

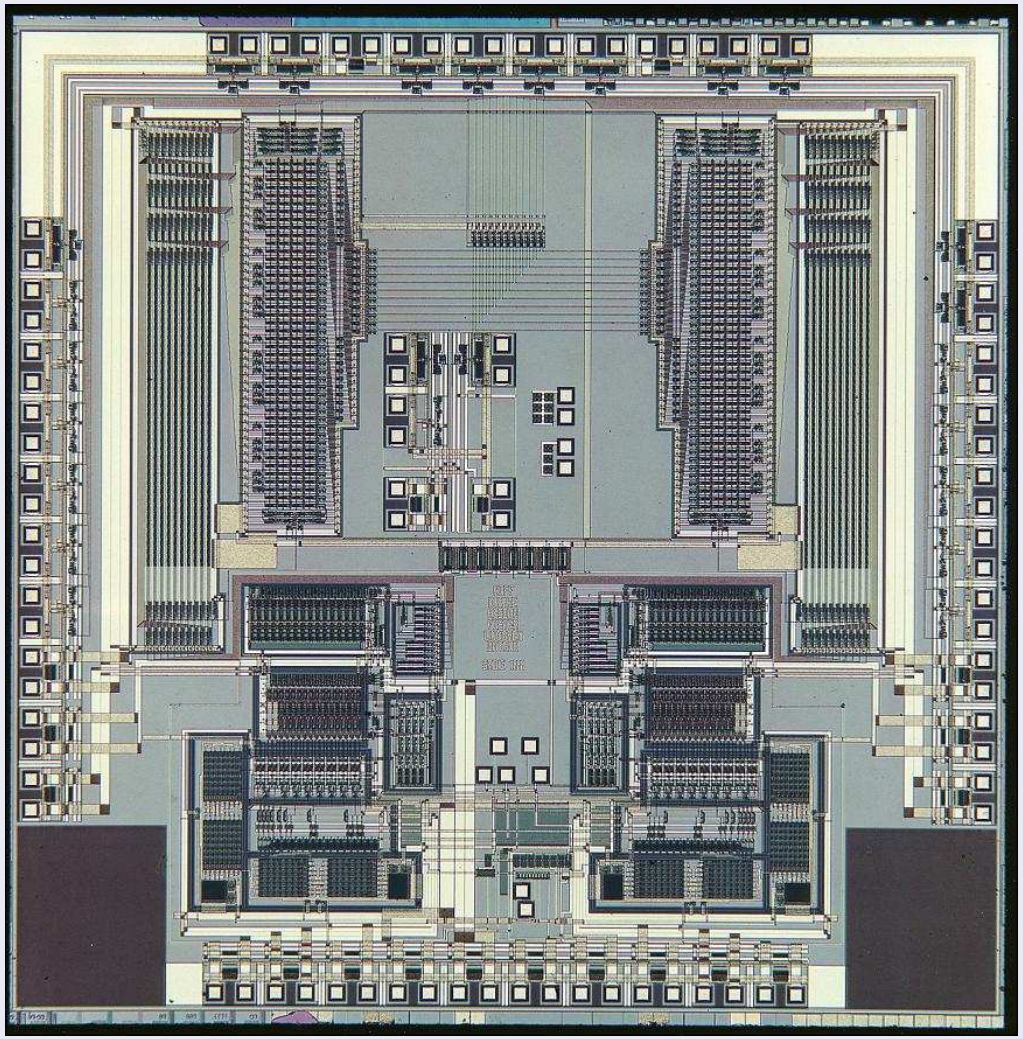
Dealing with Complexity



Different Abstraction Levels (need morphism)

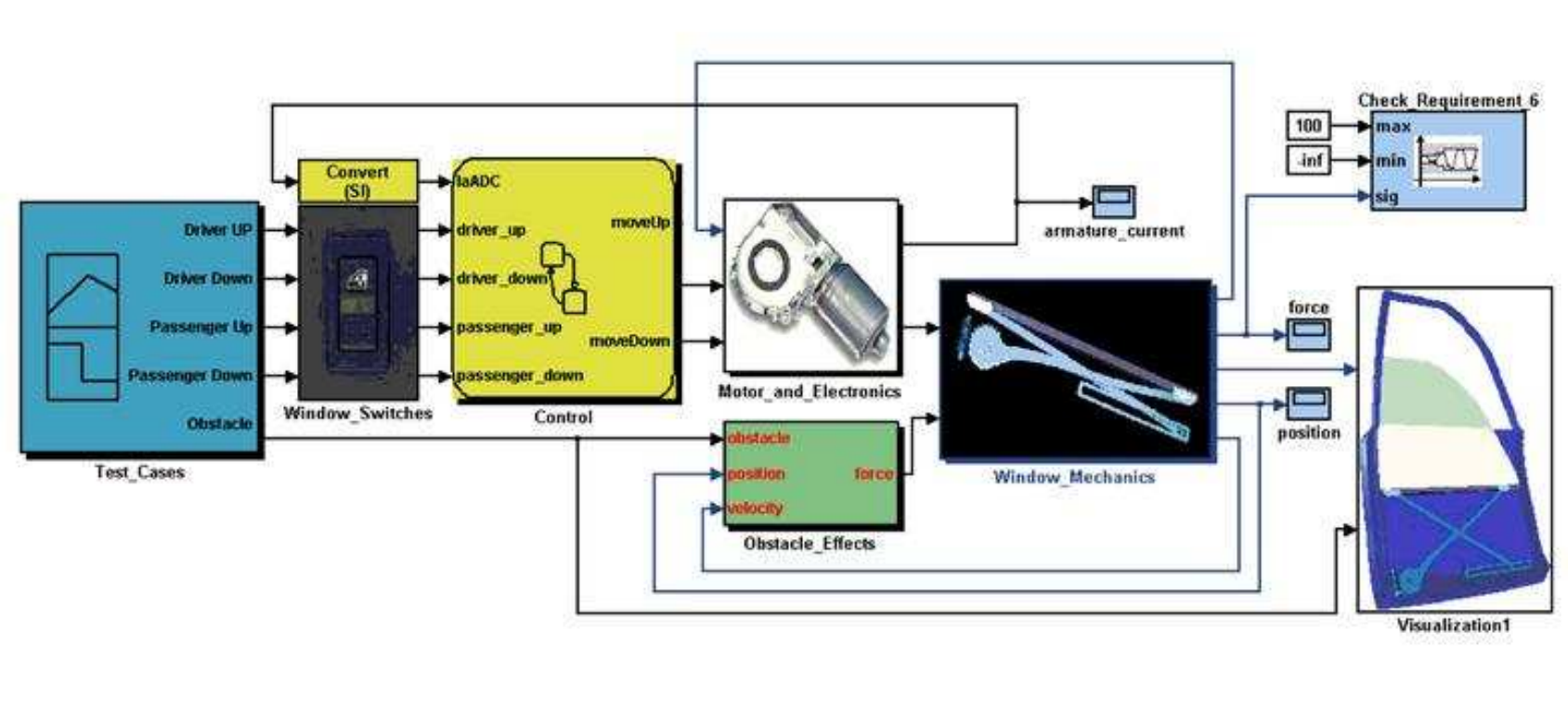


Large Number of Components



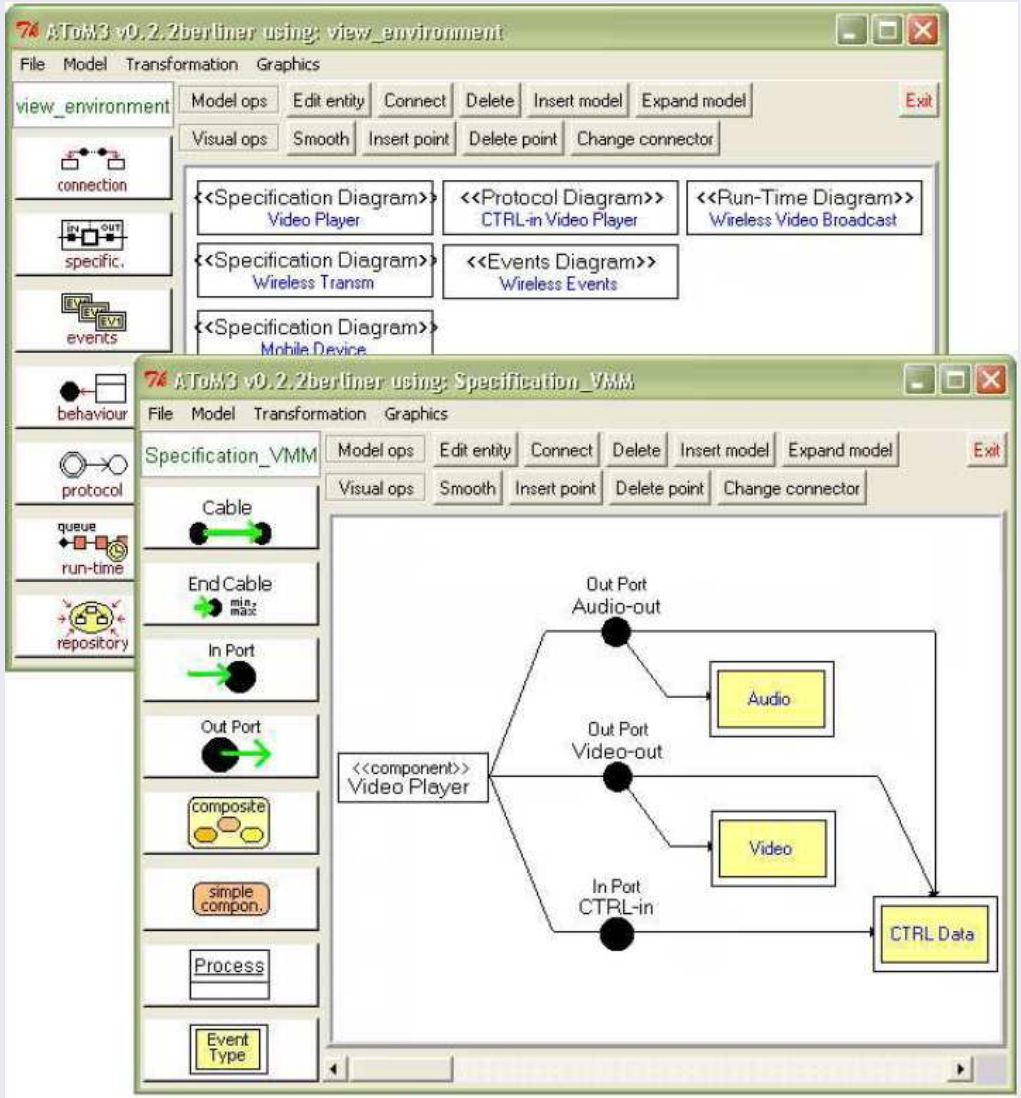
Multiple Formalisms: Power Window

Components in Different Formalisms

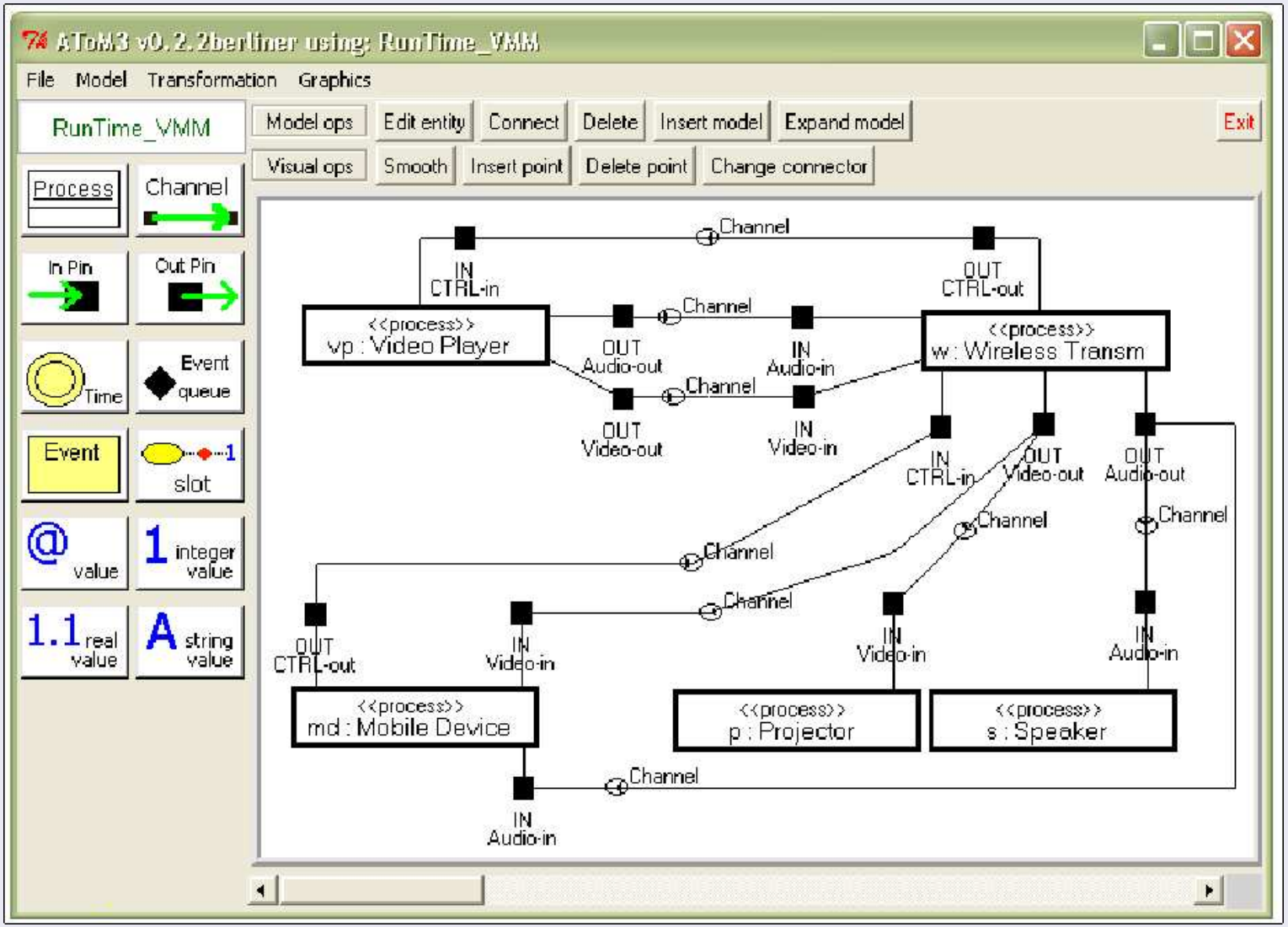


www.mathworks.com/products/demos/simulink/PowerWindow/html/PowerWindow1.html

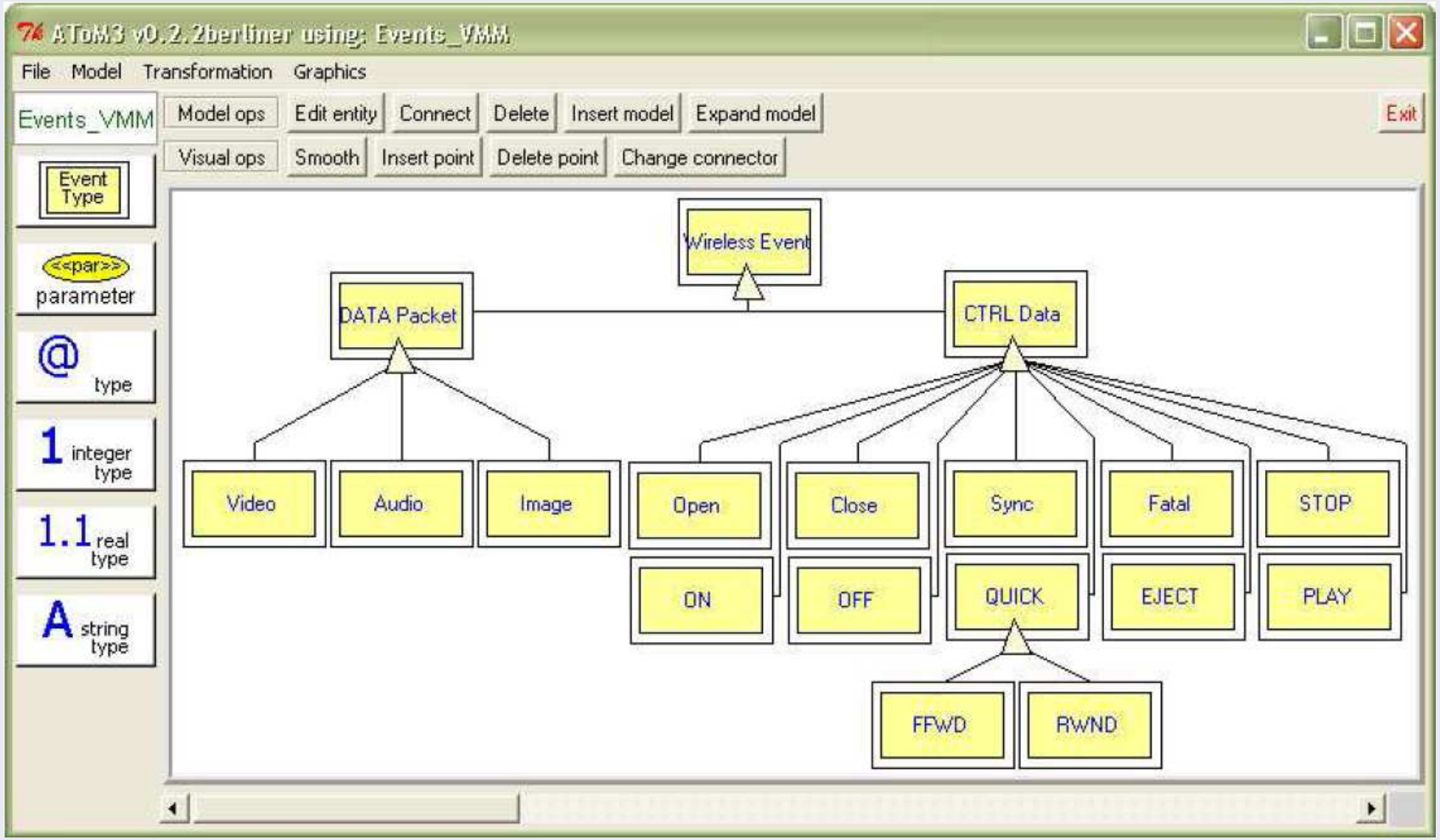
Multiple Views in Different Formalisms (need consistency!)



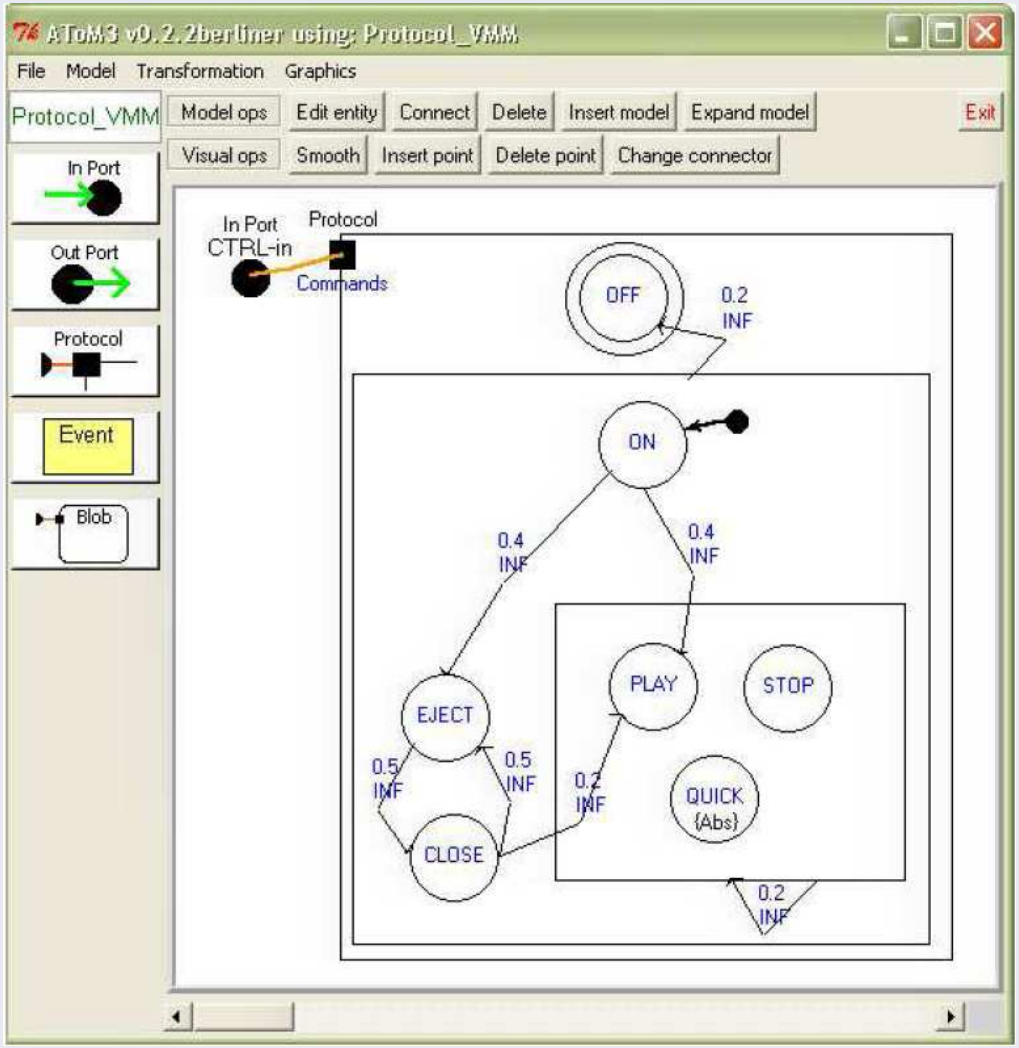
View: Structure Diagram



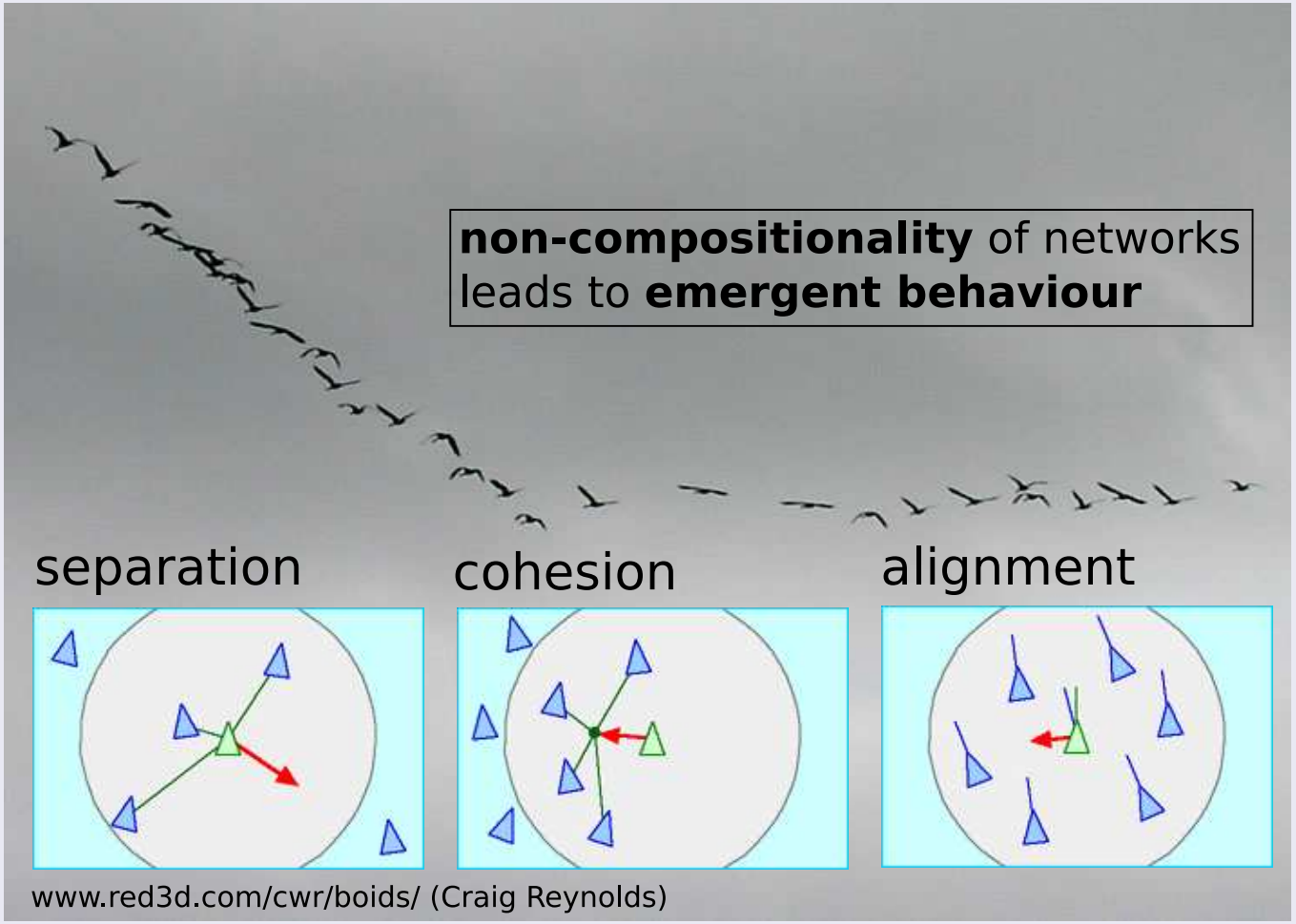
View: Events Diagram



View: Protocol Machine



Non-compositional/Emergent Behaviour



Multi-Paradigm Modelling
(minimize *accidental complexity*)

- at the most appropriate **level of abstraction**
- using the most appropriate **formalism(s)**
  Differential Algebraic Equations, Petri Nets, Bond Graphs, Statecharts, CSP, Queueing Networks, Lustre/Esterel, . . .
- with **transformations** as first-class models

Pieter J. Mosterman and Hans Vangheluwe.

Computer Automated Multi-Paradigm Modeling: An Introduction. Simulation 80(9):433–450, September 2004.

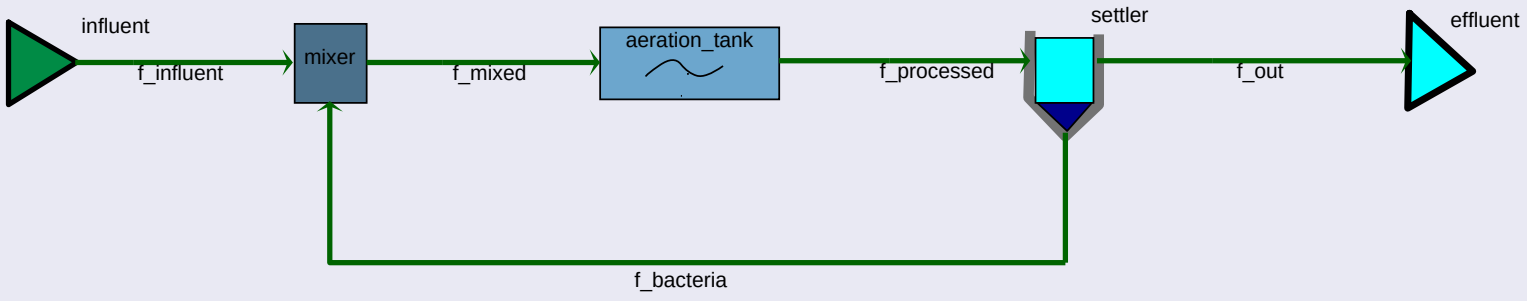
Special Issue: Grand Challenges for Modeling and Simulation.

Waste Water Treatment Plants (WWTPs)

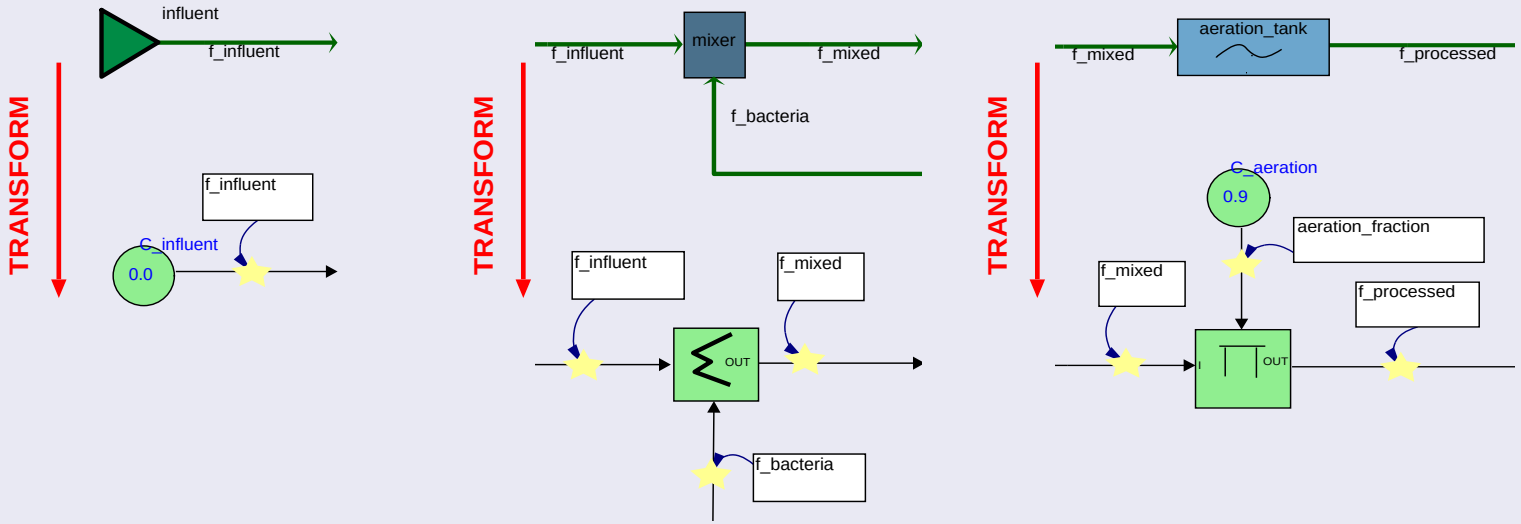
NATO's Sarajevo WWTP

www.nato.int/sfor/cimic/env-pro/waterpla.htm

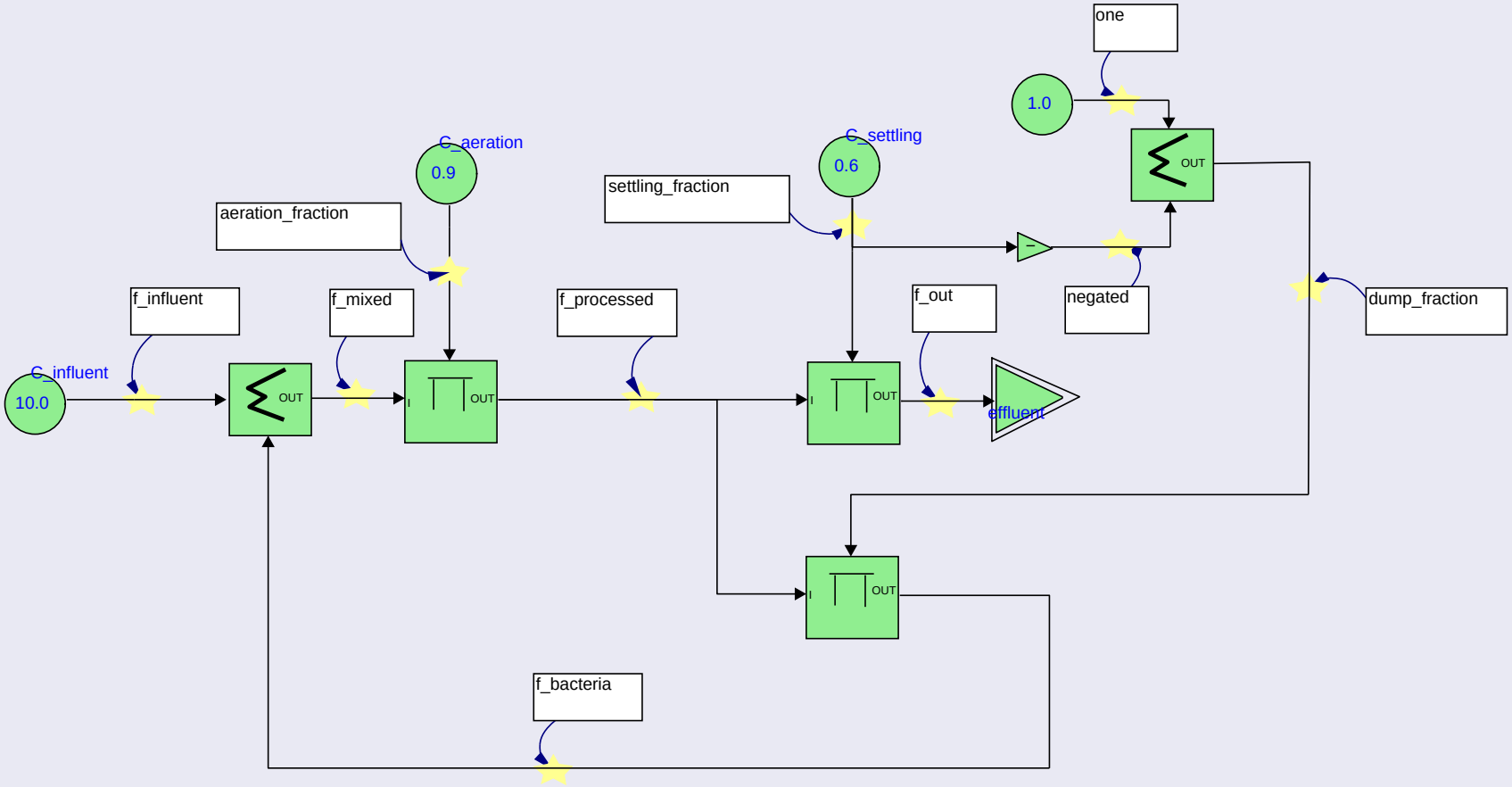
What does this WWTP model mean?



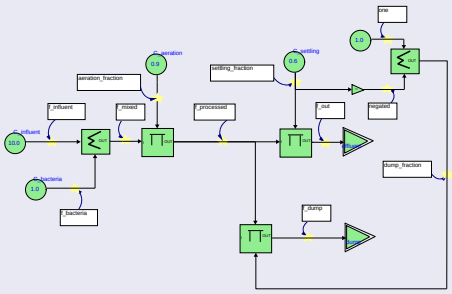
Transformation from WWTP to . . .



...its meaning (steady-state abstraction): Causal Block Diagram (CBD)



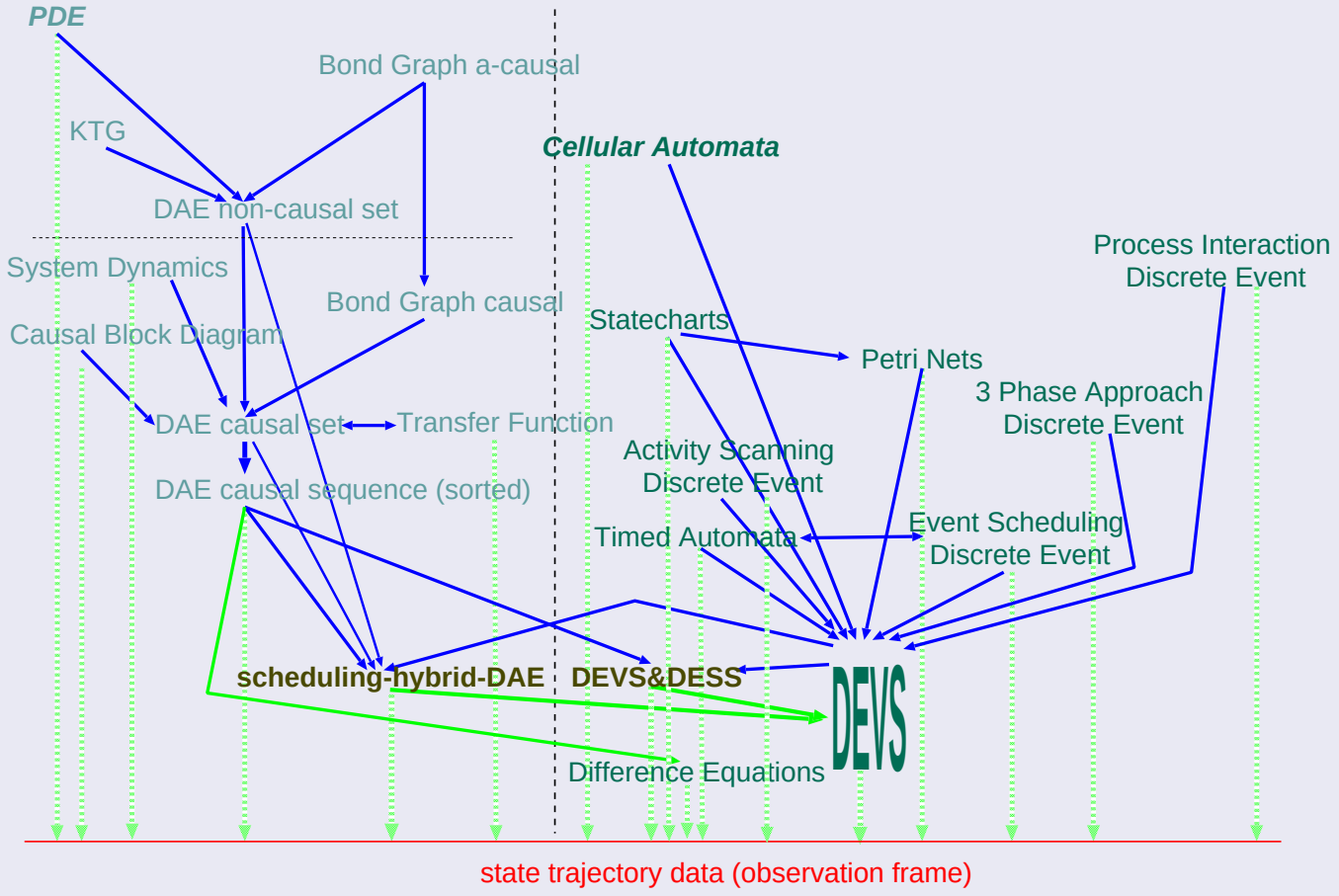
Meaning of the CBD

$$
[\![\text{  }]\!] = \begin{cases}
f\_influent &= C\_influent \\
f\_bacteria &= C\_bacteria \\
f\_mixed &= f\_influent + f\_bacteria \\
aeration\_fraction &= C\_aeration \\
f\_processed &= aeration\_fraction * f\_mixed \\
settling\_fraction &= C\_settling \\
negated &= -settling\_fraction \\
one &= 1 \\
dump\_fraction &= one + negated \\
f\_dump &= f\_processed * dump\_fraction \\
f\_out &= settling\_fraction * f\_processed
\end{cases}
$$

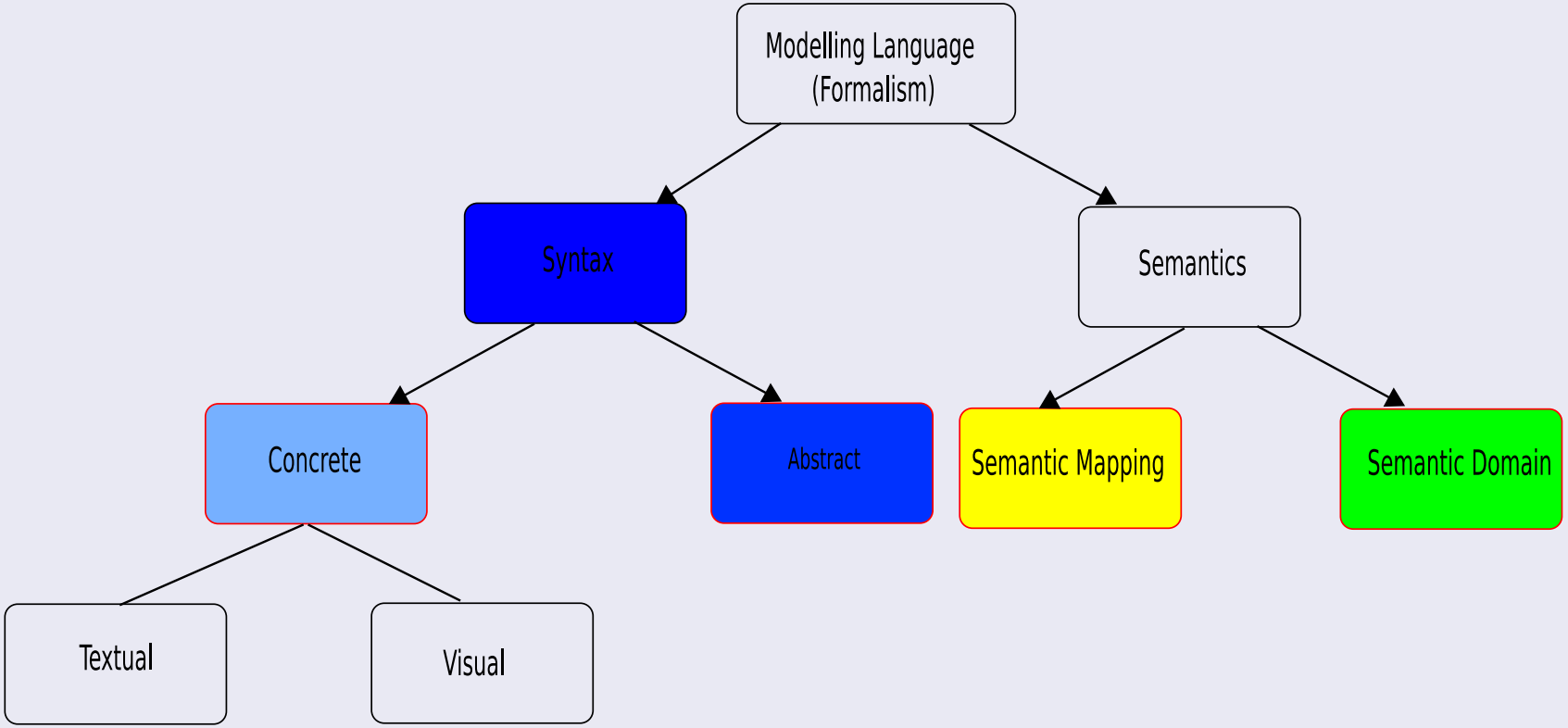
Other uses of transformation

- operational semantics (as opposed to denotational)
- requirements $\rightarrow$ design $\rightarrow$ application
- requirements $\rightarrow$ tests
- optimization
- ...
- multi-formalism modelling

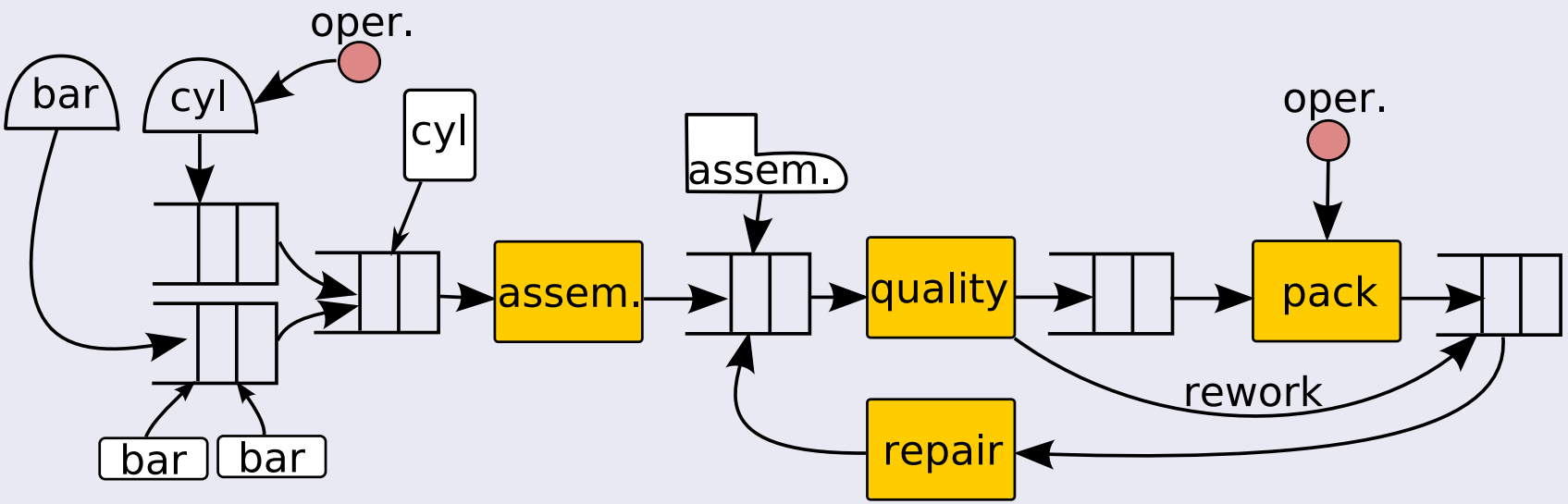
Multi-Formalism Modelling: Formalism Transformation Graph



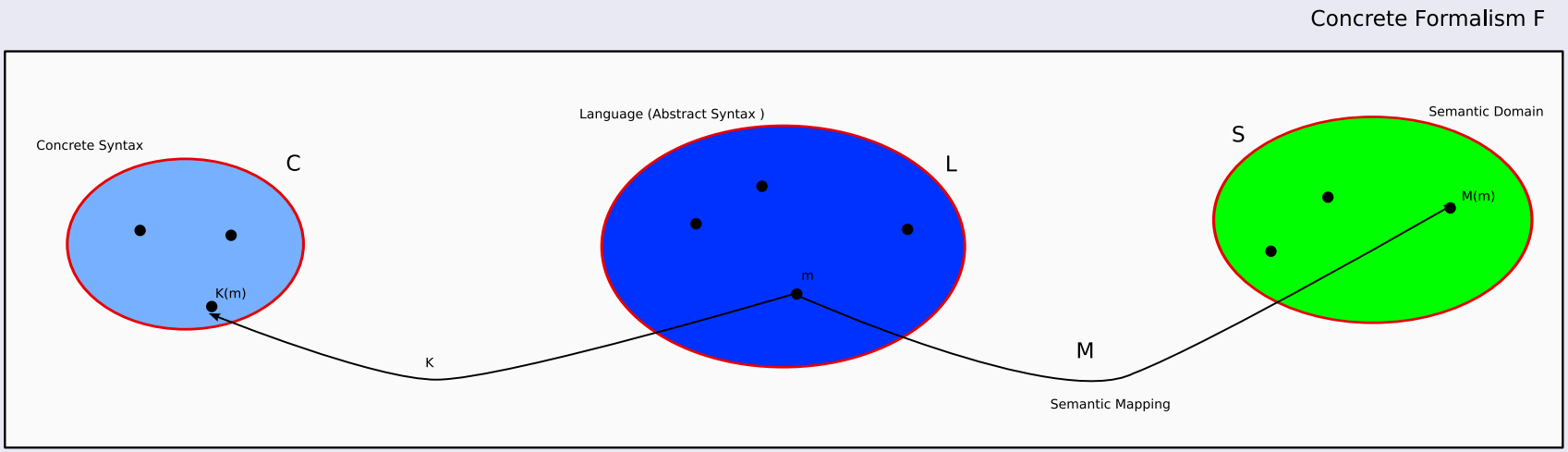
Dissecting a Modelling Language



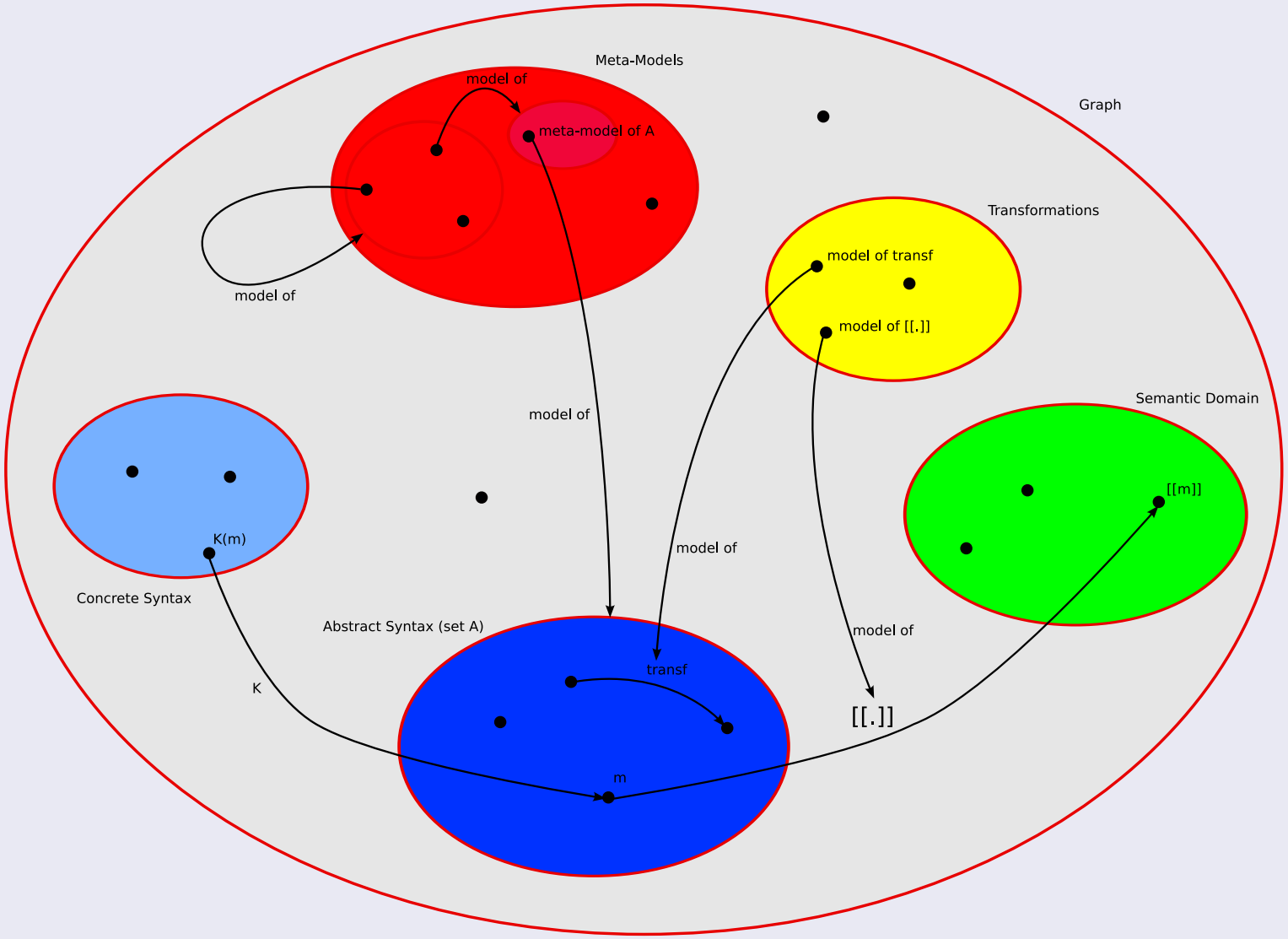
A Production System Model



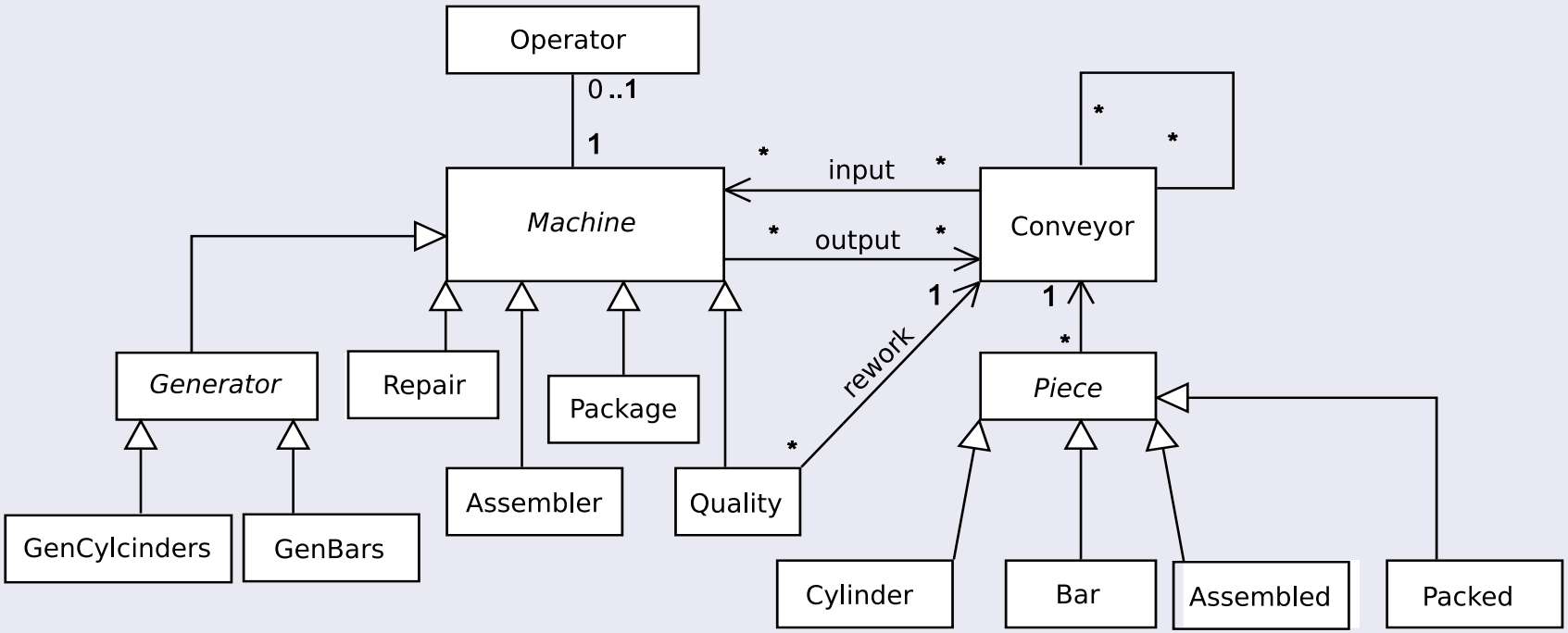
Modelling Languages as Sets of Models



Meta-Modelling

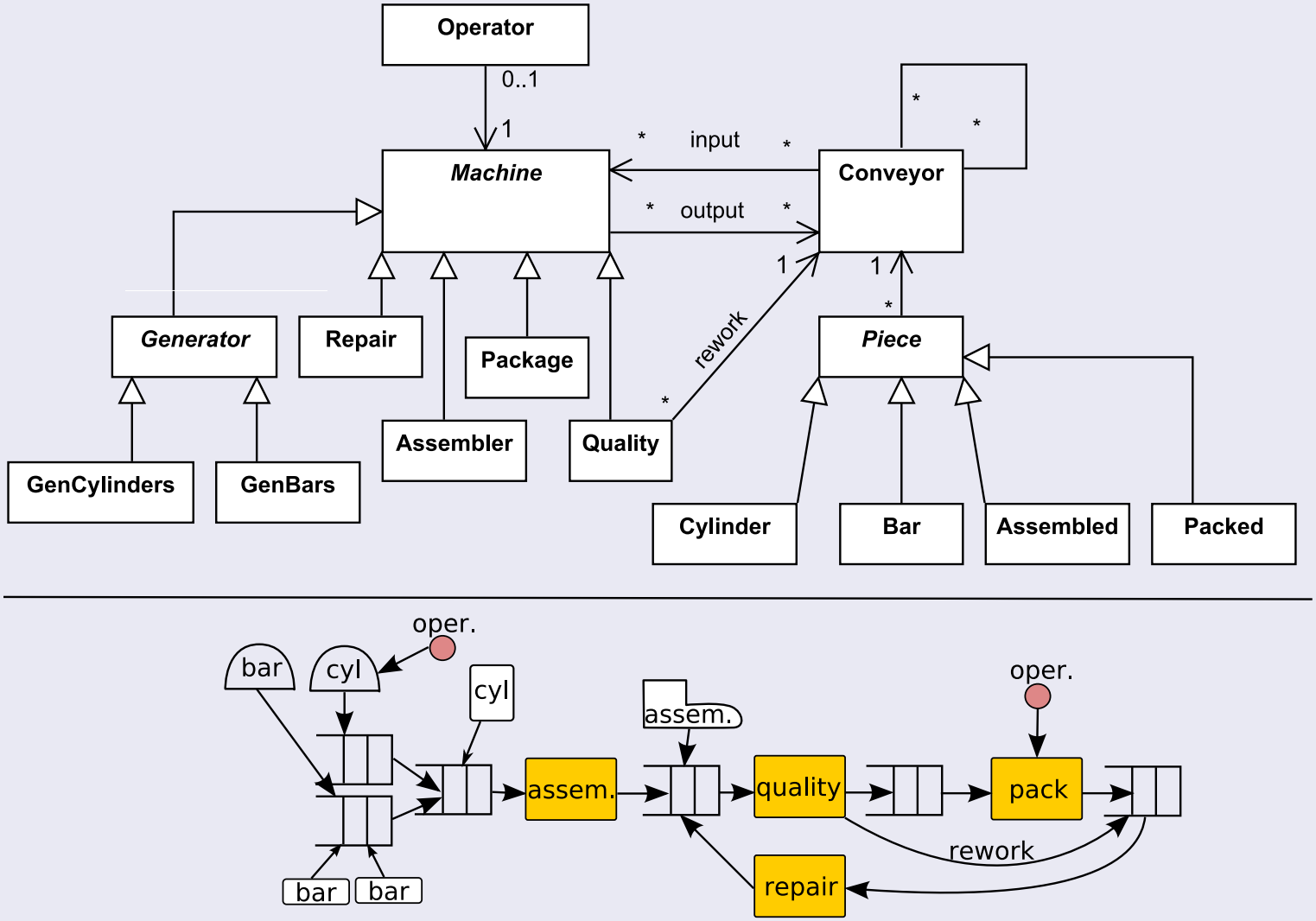


Modelling Abstract Syntax: Meta-Model



not shown: local and global contraints

Modelling Concrete Syntax (and UI Behaviour)

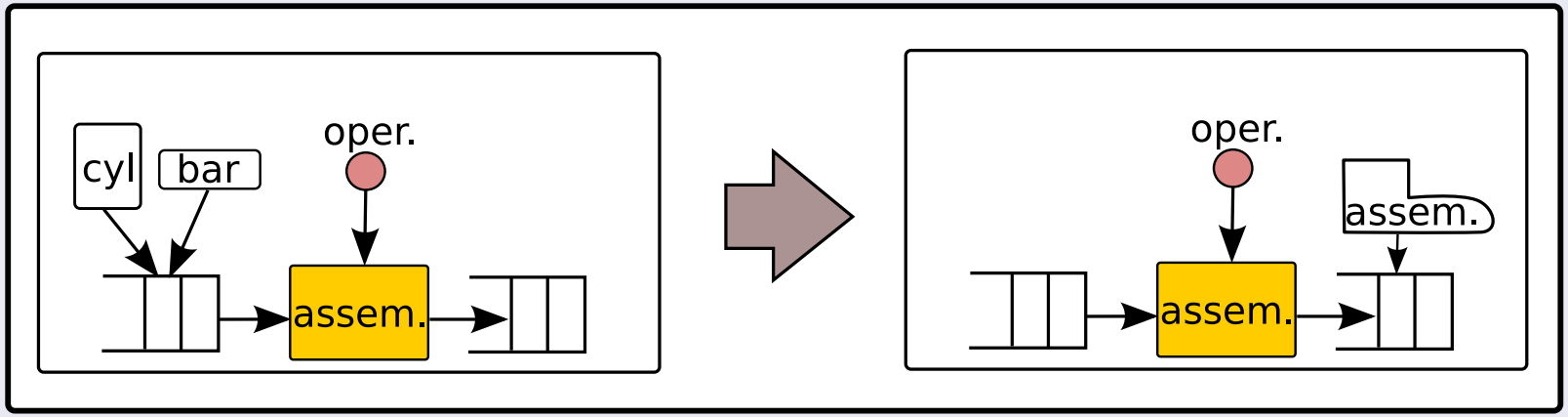


Meta-Modelling Challenges

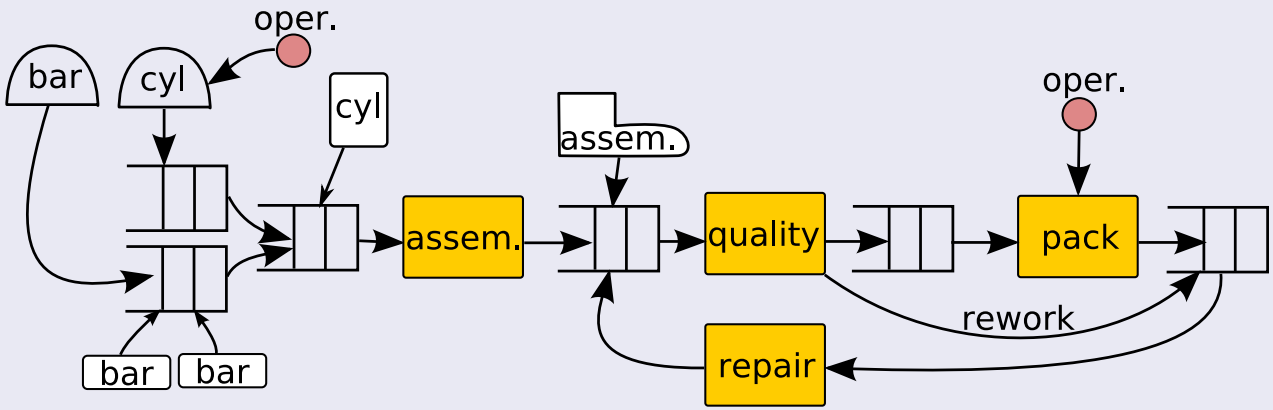
- scalability of (meta-)models
- meaningful model version control
- (meta-)model evolution
- deal with concrete syntax
  (mix textual/visual) in a unified manner

Modelling Operational Semantics in the form of Rules

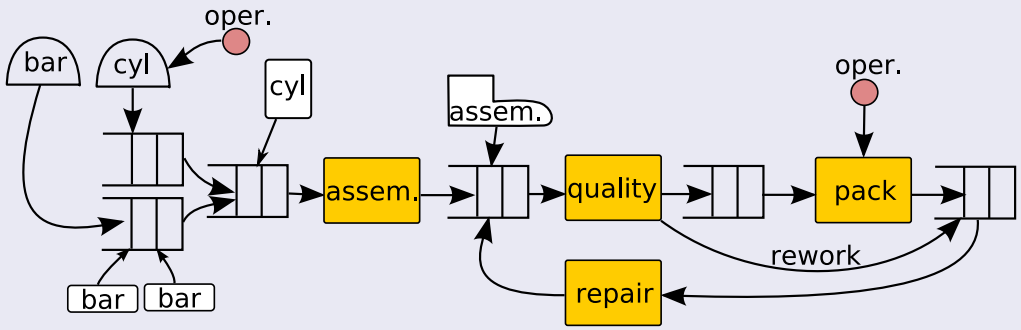
assemble

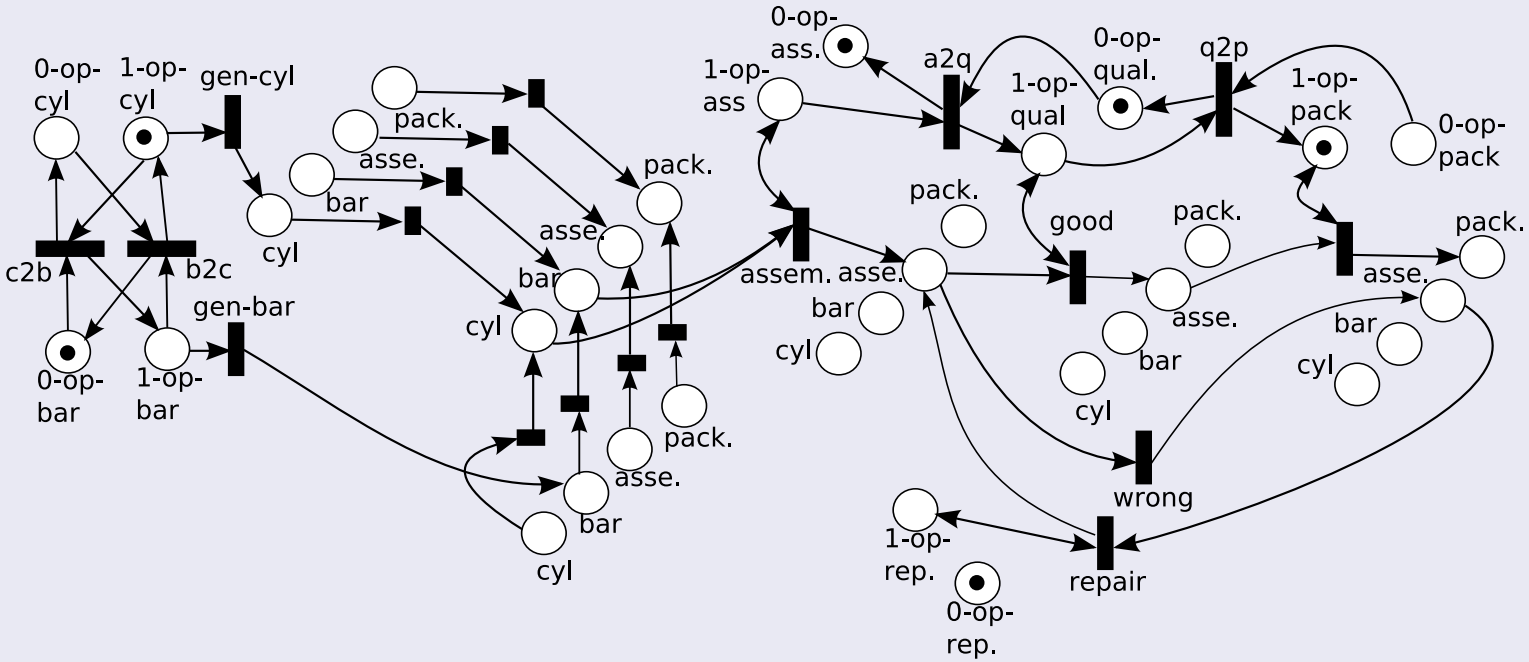


Note the use of **concrete** syntax !

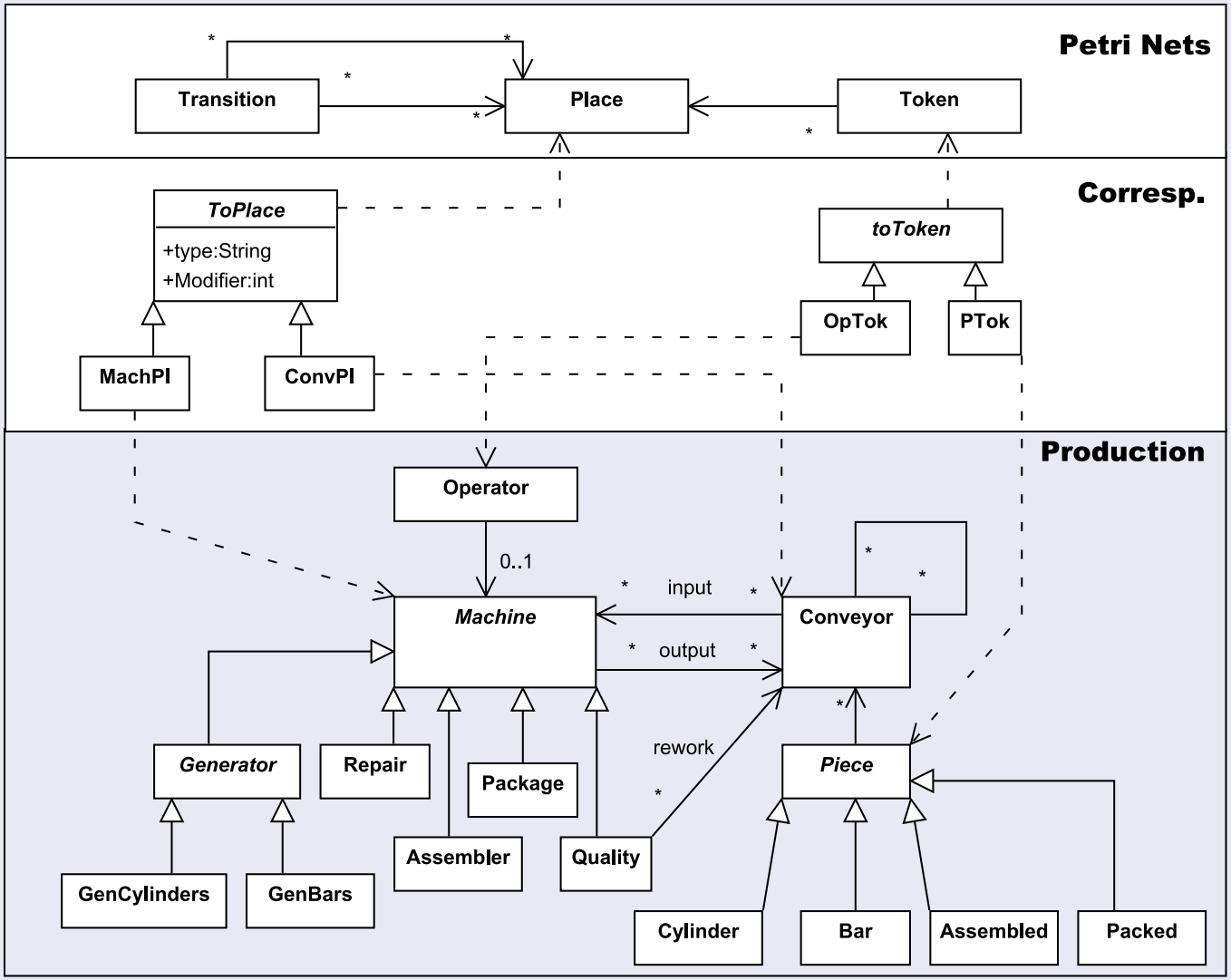


Denotational Semantics

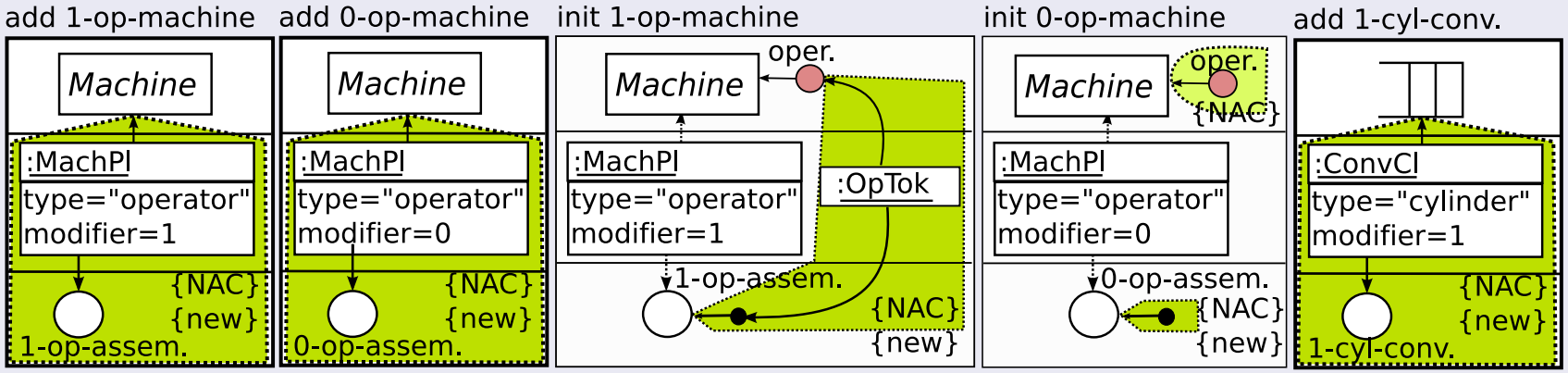




Modelling Denotational Semantics



How: Transformation Triple-Rules (bi-directional!)



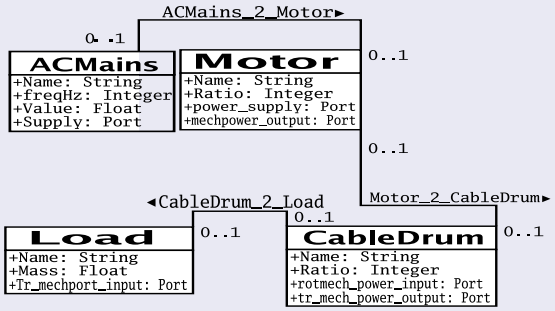
Model Transformation Challenges

- standardization/interoperability
- scalability (expressiveness and performance)
- automated testing (of models and model transformations)
- trace-ability (backward links), debugging
- from transformations to relationships (consistency)
- higher-order transformations

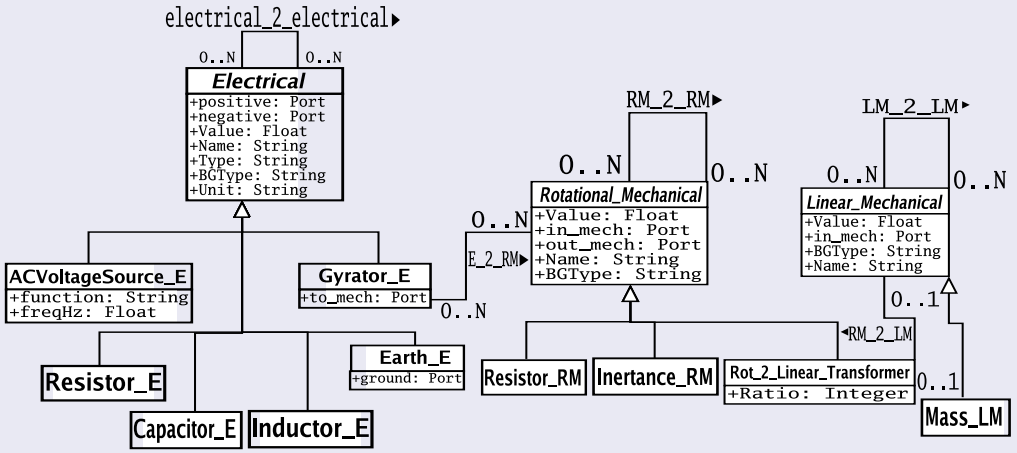
Sets and Transformations

Metamodel

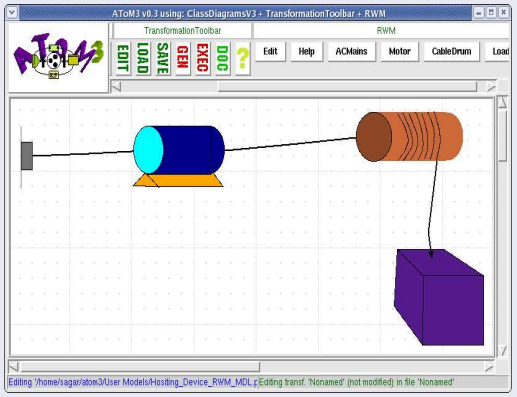
Real World Visual Modeling Formalism



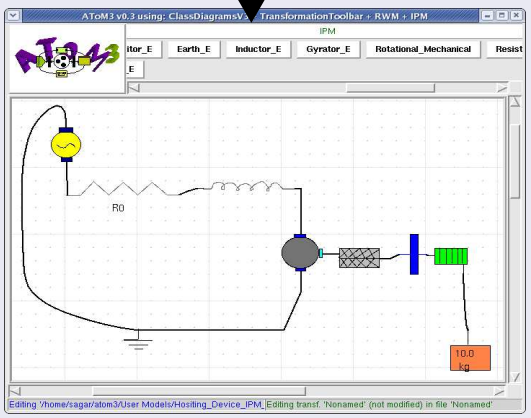
Idealized Physical Modeling Formalism



Generated Visual Modeling Environment



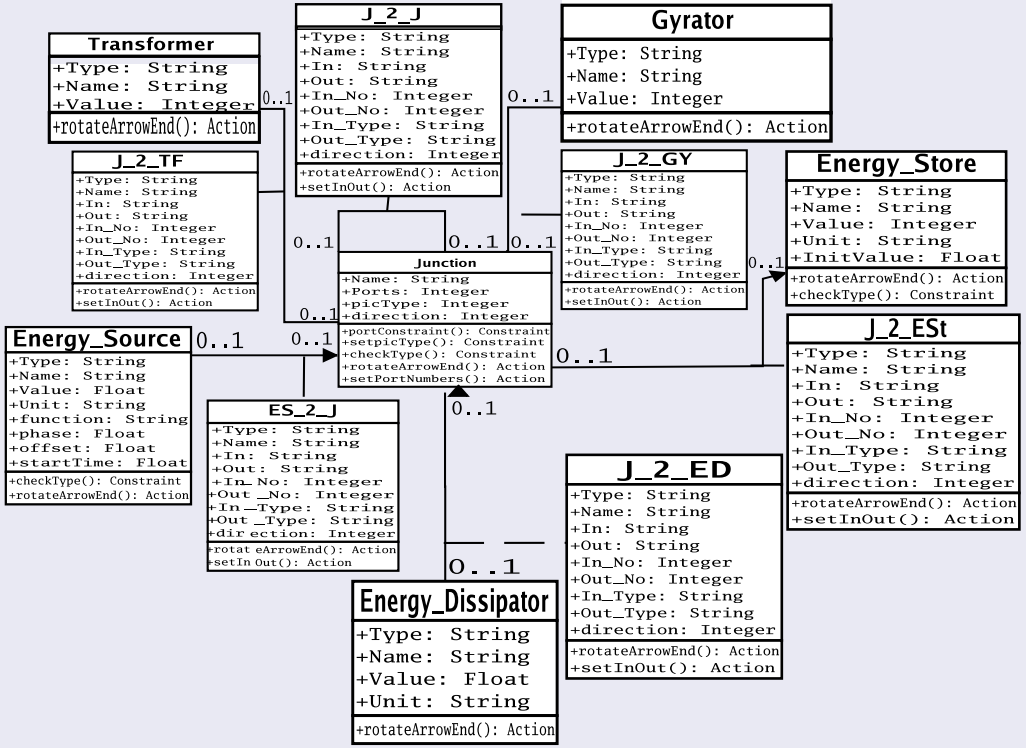
1: GG: RWVM_2_IPM



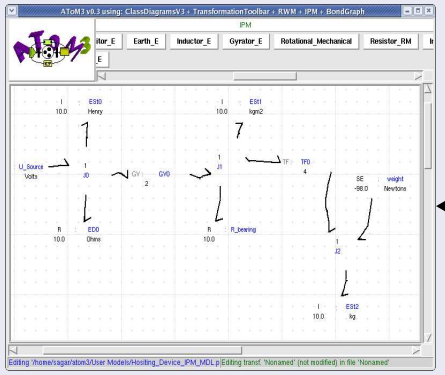
2: GG: IPM_2_ABG

Sets and Transformations

Bond Graph Modeling Formalism



2: GG: IPM_2_ABG



3: GG: ABG_2_CBG

4: GG: CBG_2_Modelica

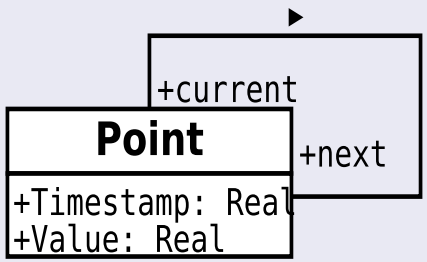
Modelica Code

Modelica Object-Oriented Modeling Formalism

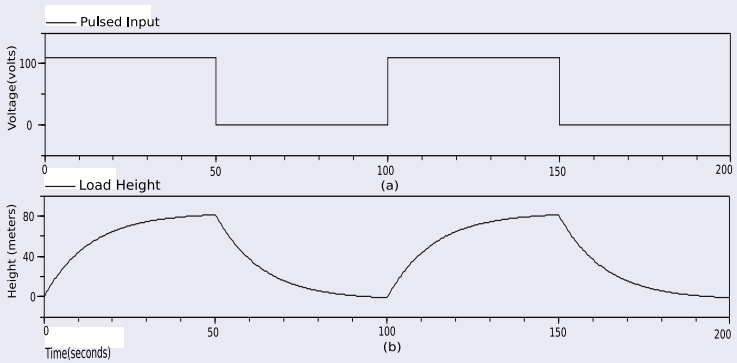
BNF Grammar of Modelica 2.2

Sets and Transformations

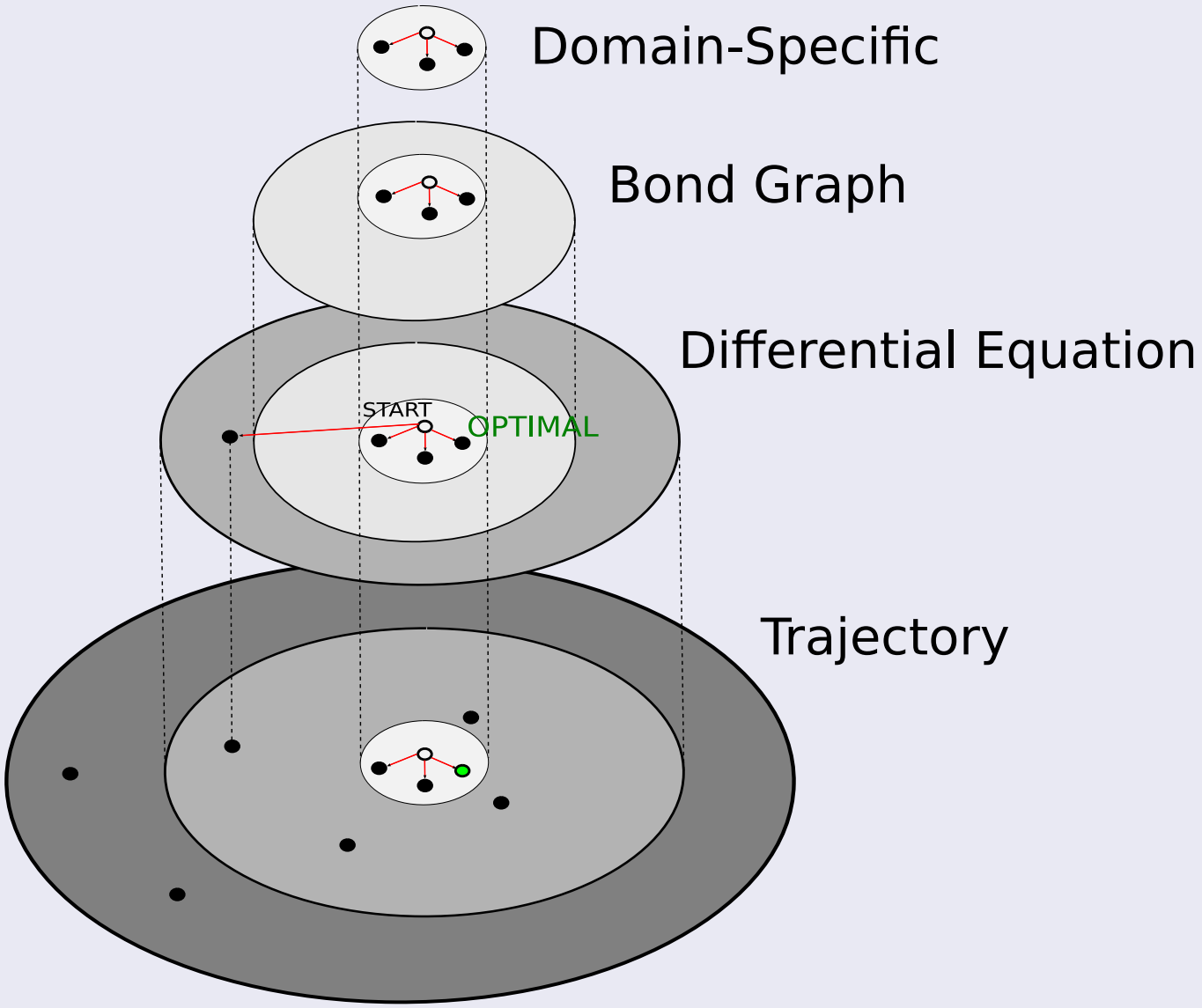
Trajectory Formalism



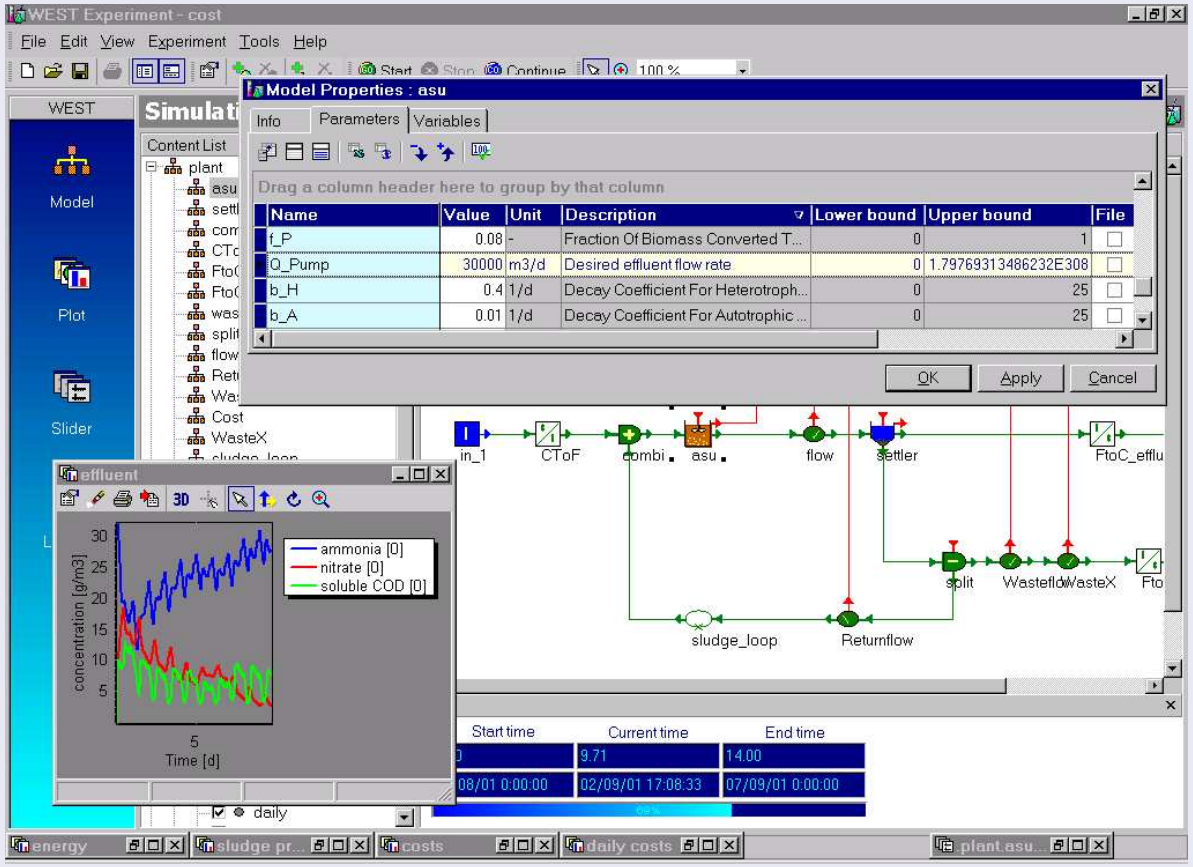
5: Simulation



Design-Space Exploration



WWTP Domain-Specific Modelling Environment



www.hemmis.com/products/west/

Henk Vanhooren, Jurgen Meirlaen, Youri Amerlinck, Filip Claeys, Hans Vangheluwe, and Peter A. Vanrolleghem.

WEST: Modelling biological wastewater treatment. Journal of Hydroinformatics, 5(1):27-50, 2003.

Why DS(V)M ?
(as opposed to General Purpose modelling)

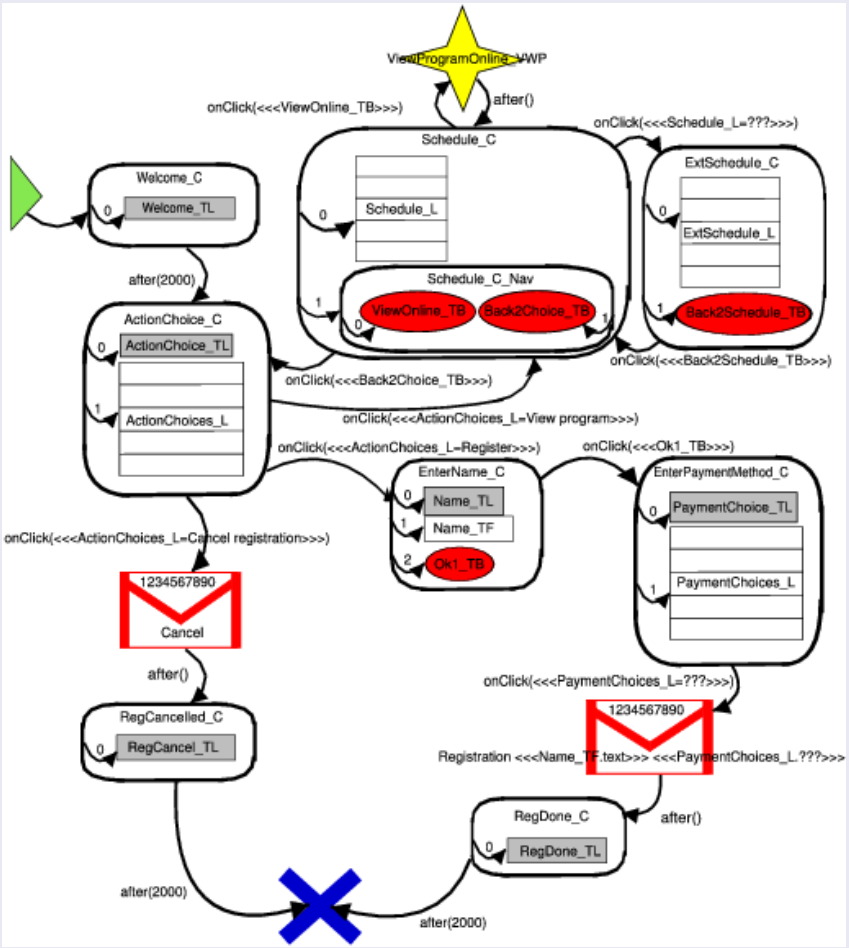
- **match the user's mental model** of the problem domain
- **maximally constrain** the user (to the problem at hand)
  $\Rightarrow$ easier to learn
  $\Rightarrow$ avoid errors
- **separate** domain-expert's work
  from analysis/transformation expert's work

Anecdotal evidence of 5 to 10 times speedup

Steven Kelly and Juha-Pekka Tolvanen.

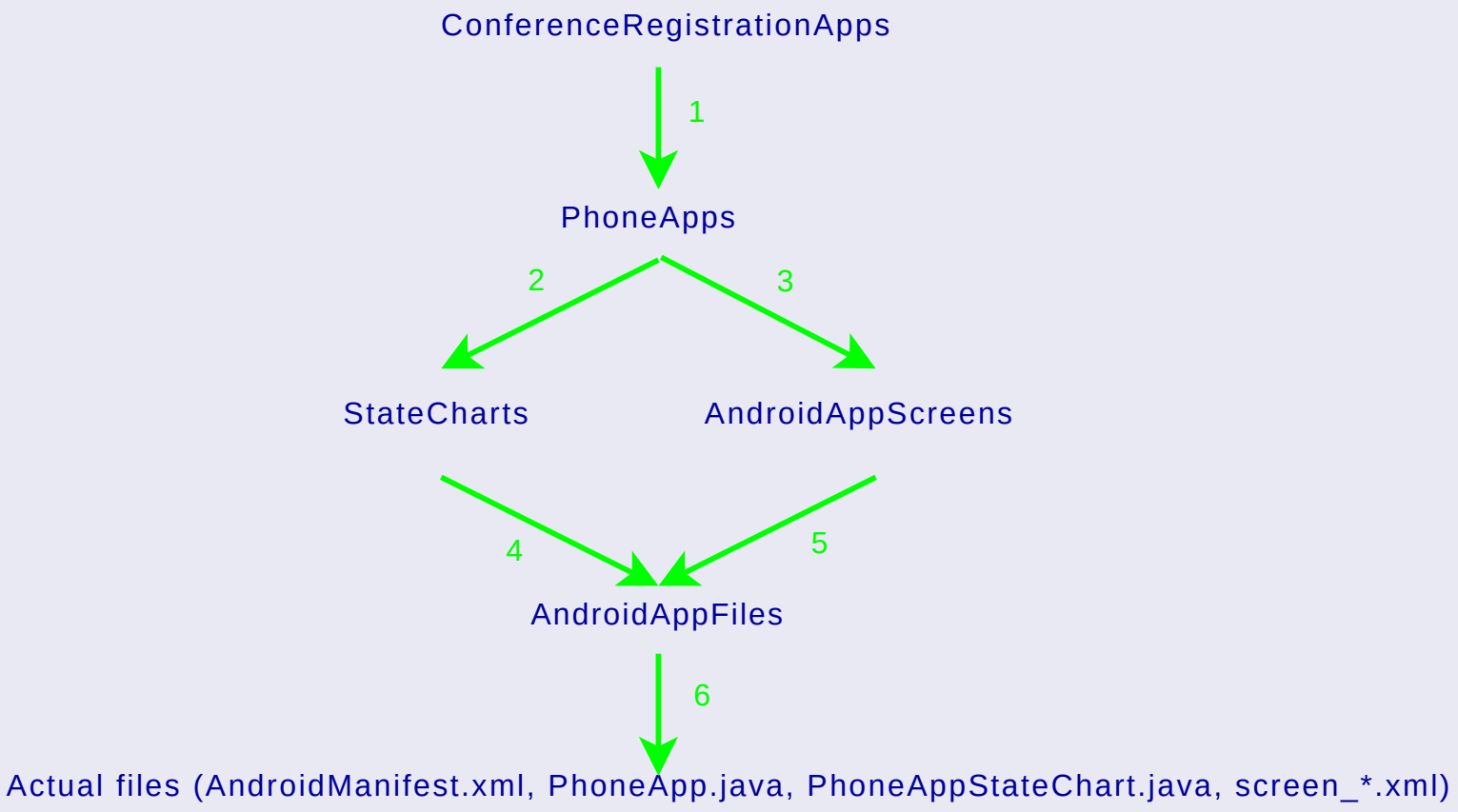
Domain-Specific Modeling: Enabling Full Code Generation. Wiley 2008.

DS(V)M example application, the PhoneApps Domain-Specific model



DS(V)M example application: conference registration (Google Android)

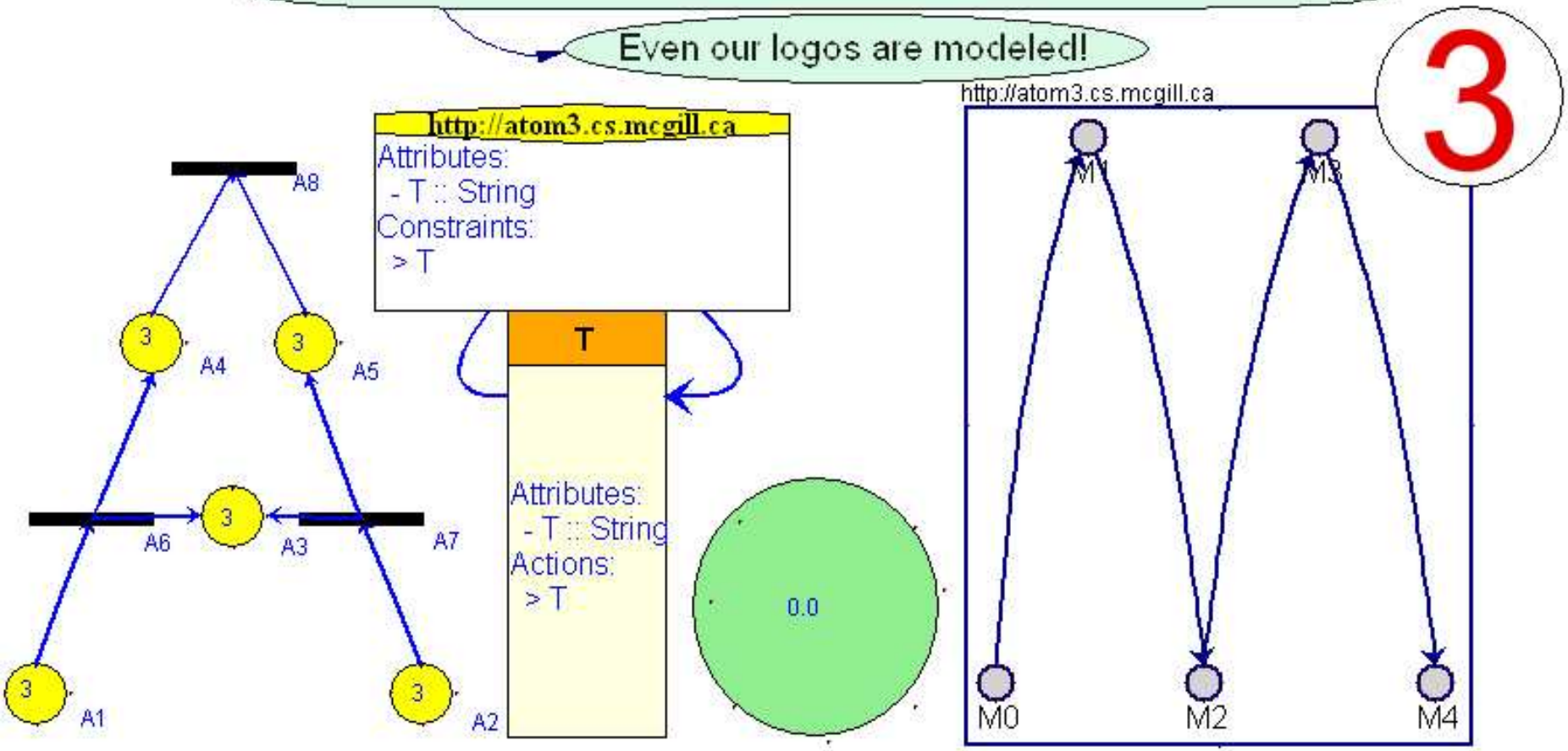
Only transform . . .



Eat Your Own Dogfood!

A Tool for Multi-formalism and Meta-Modeling

Even our logos are modeled!



Visit MSDL at http://msdl.cs.mcgill.ca

Model Everything! (to deal with Change)

- at most appropriate **level of abstraction**
- using most appropriate **formalism(s)**
- with **transformations** as first-class models