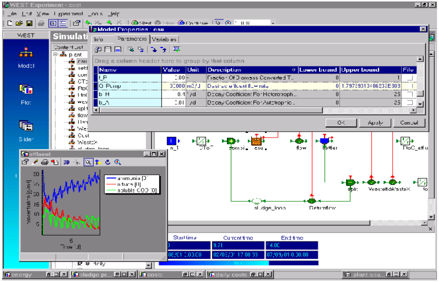
Promises and Challenges of Model-Based Design

Hans Vangheluwe

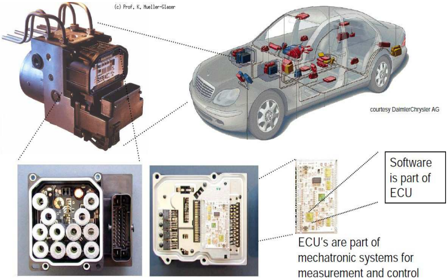
29 April 2012

CAMPaM workshop, Bellairs Research Institute, Barbados


WEST Experiment - cost
Model Properties : asu
Simulati
effluent


TANK WARS
EA


htc


(c) Prof. K. Mueller-Glaser
courtesy DaimlerChrysler AG
Software is part of ECU
ECU's are part of mechatronic systems for measurement and control


MAMMOTH


McGill
icri
LEUVEN
MSDL
EMC

Dealing with Complexity

Dealing with Complexity

Model Everything . . . Explicitly

Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)

Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)

The spectrum of uses of models

- Documentation

Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)

Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)

Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)
- Test Generation

Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)
- Test Generation
- Simulation (one model, one behaviour)
  . . . calibration, optimization, . . .

Dealing with Complexity

Model Everything . . . Explicitly
for **design** (Engineering) and **analysis** (Science)

The spectrum of uses of models

- Documentation
- Formal Verification of Properties
  (all models, all behaviours)
- Model Checking of Properties
  (one model, all behaviours)
- Test Generation
- Simulation (one model, one behaviour)
  . . . calibration, optimization, . . .
- Application Synthesis (mostly for models of software)

Requirements ("What?")

- Detached or Semi-detached
- Style (classical, modern, . . .)
- Number of Floors
- Number of rooms of different types (bedrooms, bathrooms, . . .)
- Garage, Storage, . . .
- Cellar
- . . .

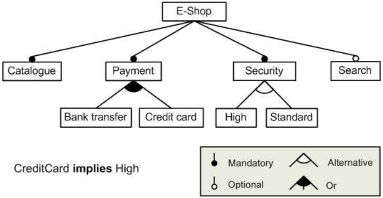
Requirements ("What?")

- Detached or Semi-detached
- Style (classical, modern, . . .)
- Number of Floors
- Number of rooms of different types (bedrooms, bathrooms, . . .)
- Garage, Storage, . . .
- Cellar
- . . .

Design ("How?")

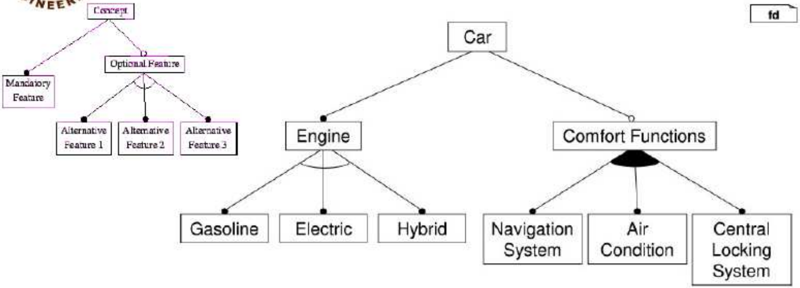
Requirements (“What?”)

- Detached or Semi-detached
- Style (classical, modern, . . .)
- Number of Floors
- Number of rooms of different types (bedrooms, bathrooms, . . .)
- Garage, Storage, . . .
- Cellar
- . . .

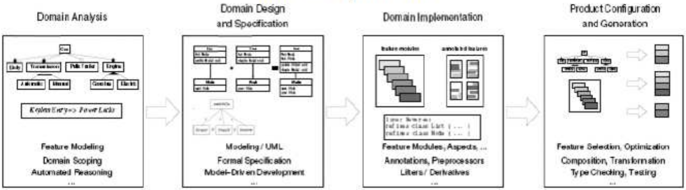


Design (“How?”)

Feature Diagrams and Product Families



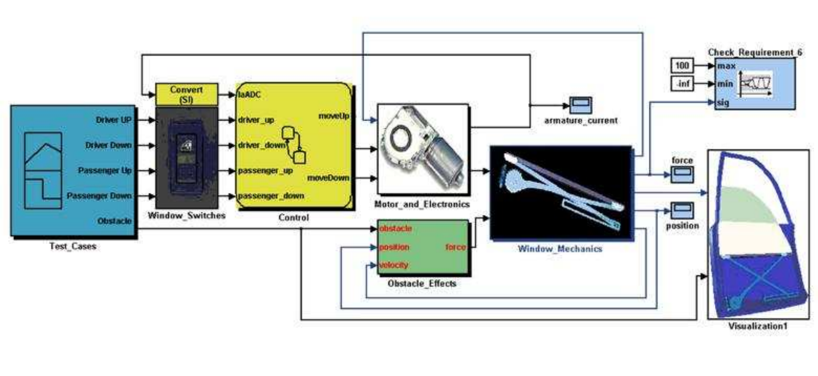
basis for **synthesis**



System Boundaries

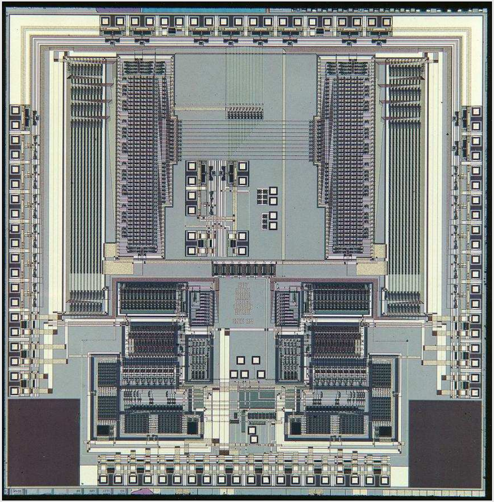
- **System** to be built/studied
- **Environment** with which the system interacts

System vs. "Plant"



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

Number of Components – hierarchical (de-)composition

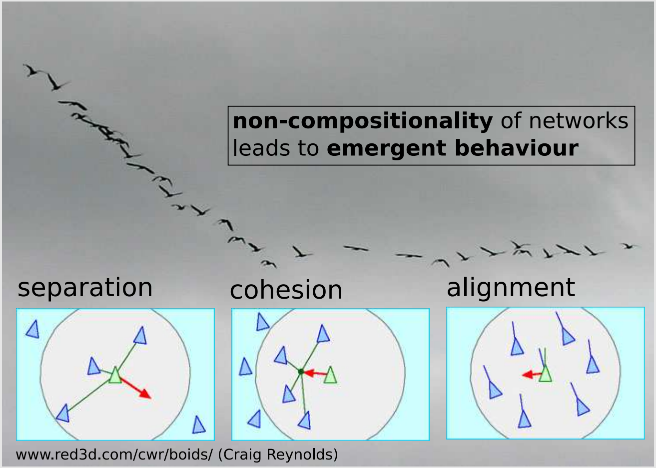


Crowds: diversity, interaction

www.3dm3.com

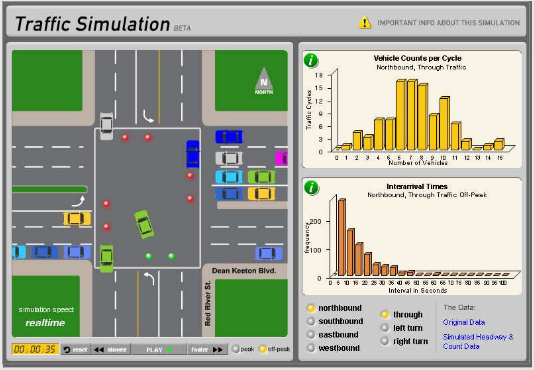
Diversity of Components: Power Window

Non-compositional/Emergent Behaviour



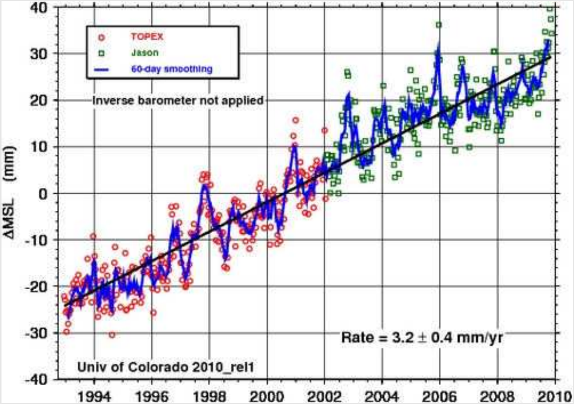
Uncertainty

Often related to level of abstraction:
for example continuous vs. discrete

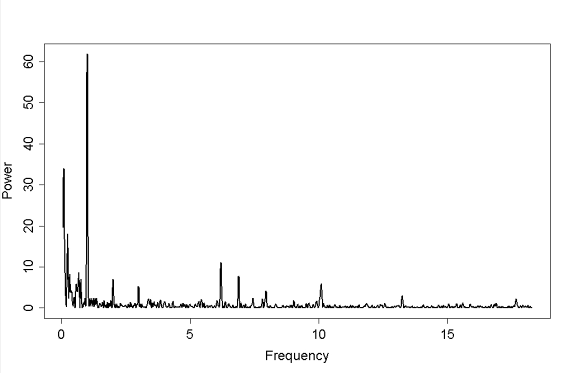


www.engr.utexas.edu/trafficSims/

Question: is the deviation from the trend periodic?



Answer: transform to make the solution obvious

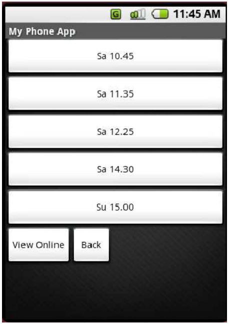


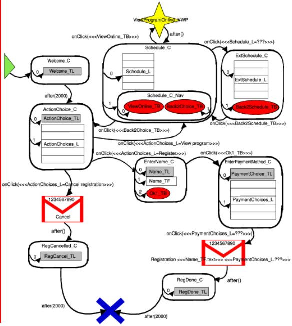
Guiding principle (∼ physics: principle of minimal action)

minimize **accidental complexity**,
only **essential complexity** remains

Fred P. Brooks. No Silver Bullet – Essence and Accident in Software Engineering. Proceedings of the IFIP Tenth World Computing Conference, pp. 1069–1076, 1986.

`http://www.lips.utexas.edu/ee382c-15005/Readings/Readings1/05-Broo87.pdf`

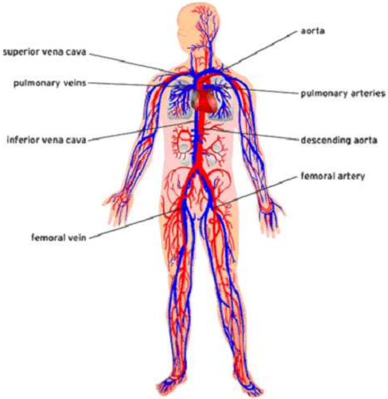


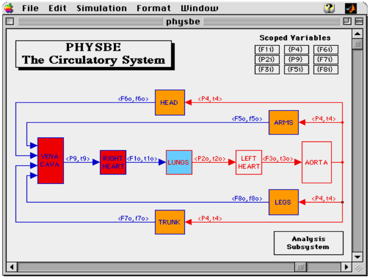


Dealing with Complexity: some approaches

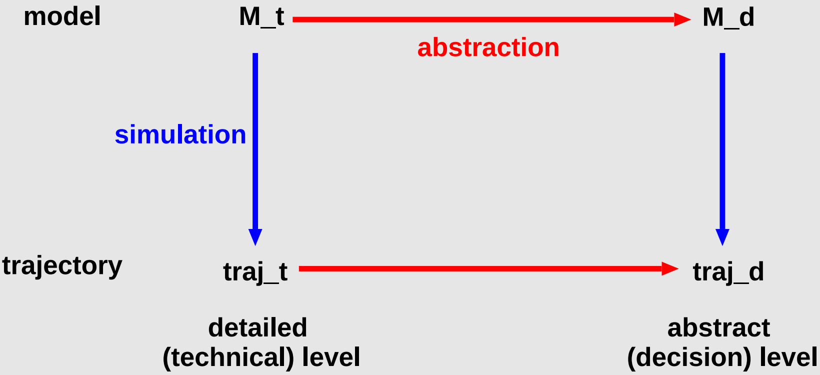
- multiple abstraction levels
- optimal formalism
- multiple formalisms
- multiple views

Different Abstraction Levels – properties preserved





Levels of Abstraction/Views: Morphism

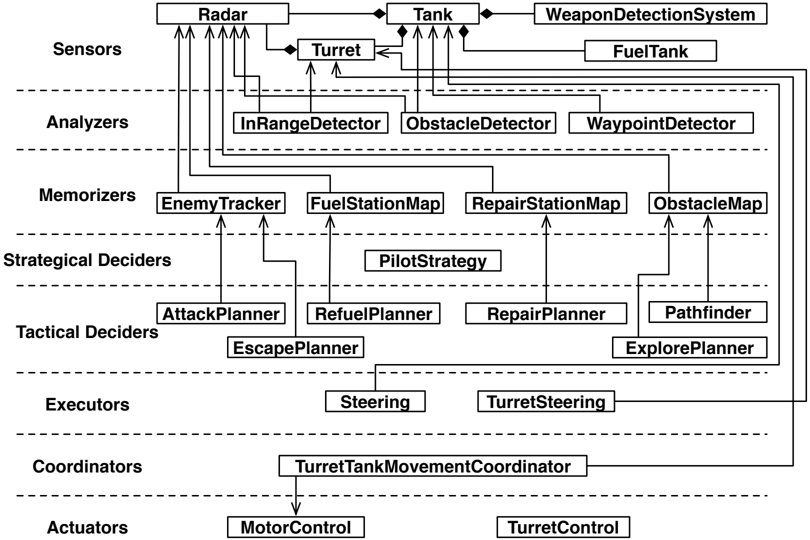


Most Appropriate Formalism (Minimizing Accidental Complexity)

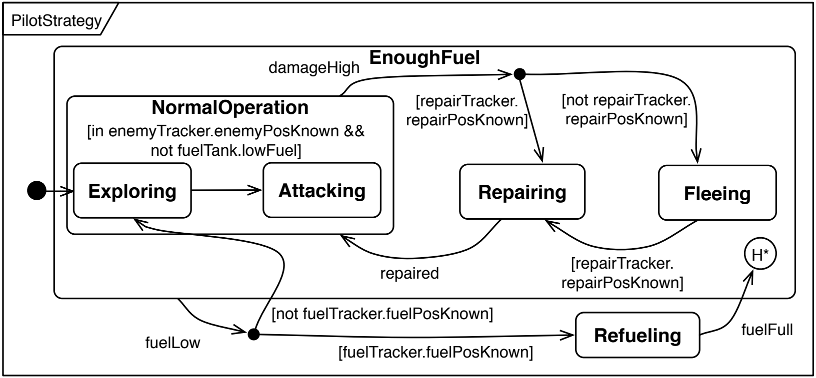
www.planeshift.it

Massively Multiplayer Online Role Playing games need Non-Player Characters (NPCs)

TankWars: high level

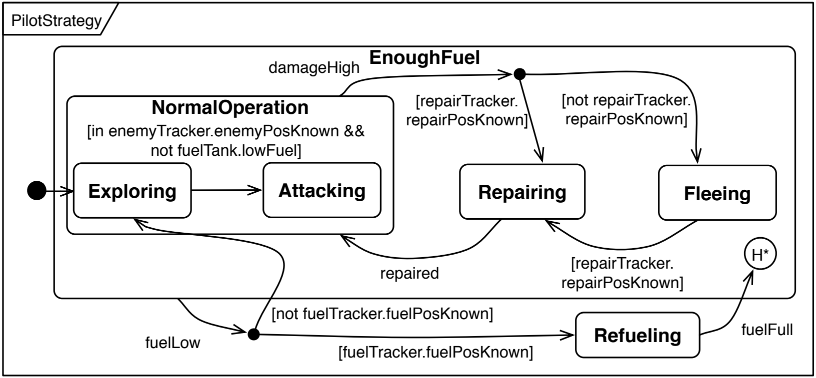


Strategic Deciders – High-level Goals



Jörg Kienzle, Alexandre Denault, Hans Vangheluwe. Model-Based Design of Computer-Controlled Game Character Behavior. MoDELS 2007: 650-665

Strategic Deciders – High-level Goals

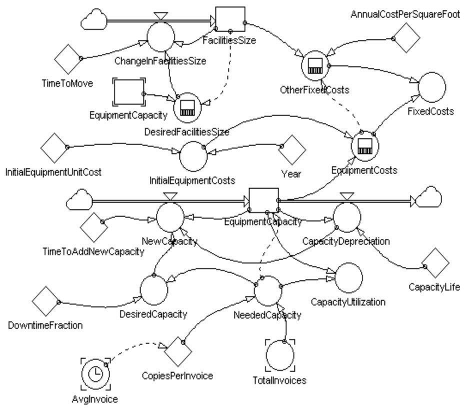


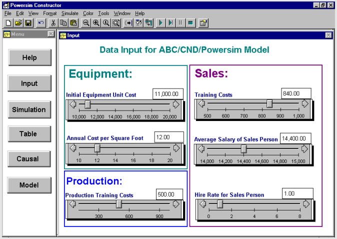
Jörg Kienzle, Alexandre Denault, Hans Vangheluwe. Model-Based Design of Computer-Controlled Game Character Behavior. MoDELS 2007: 650-665

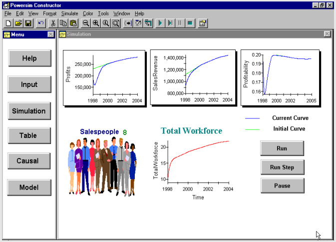
Could have used production rules instead of Statecharts

Eugene Syriani, Hans Vangheluwe: Programmed Graph Rewriting with DEVS. AGTIVE 2007: 136-151

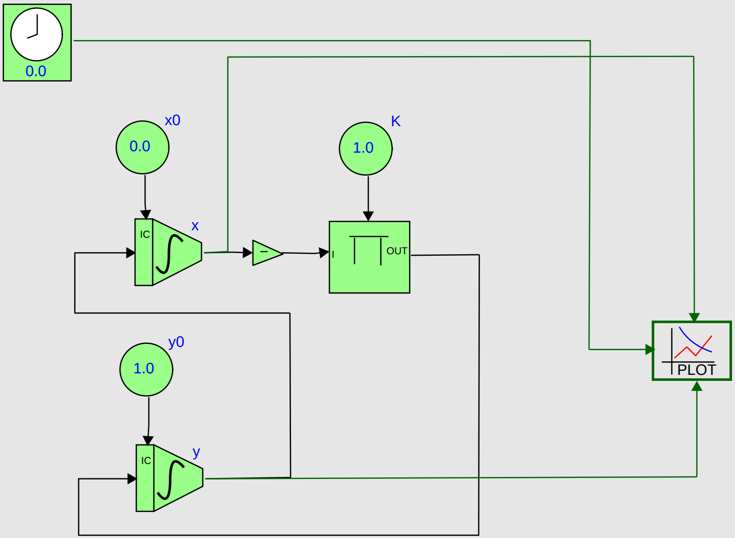
"Management Flight Simulator" using Forrester System Dynamics model



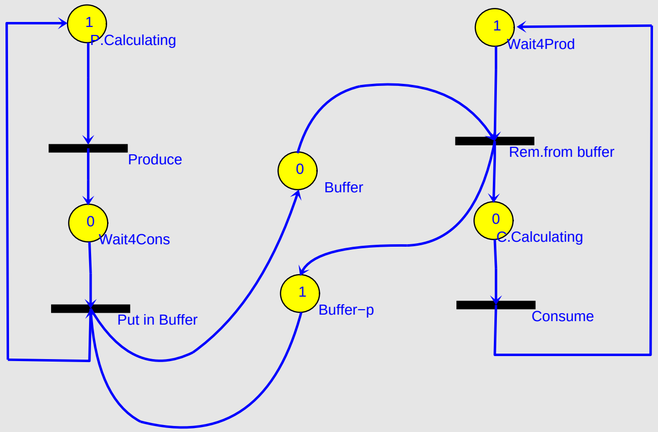




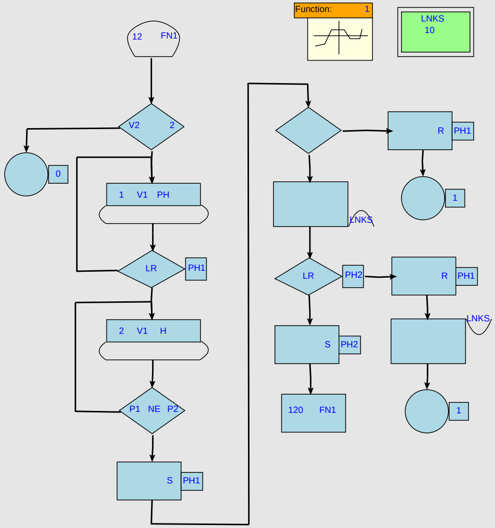
Causal Block Diagram model of Harmonic Oscillator



Petri Net model of Producer – Consumer

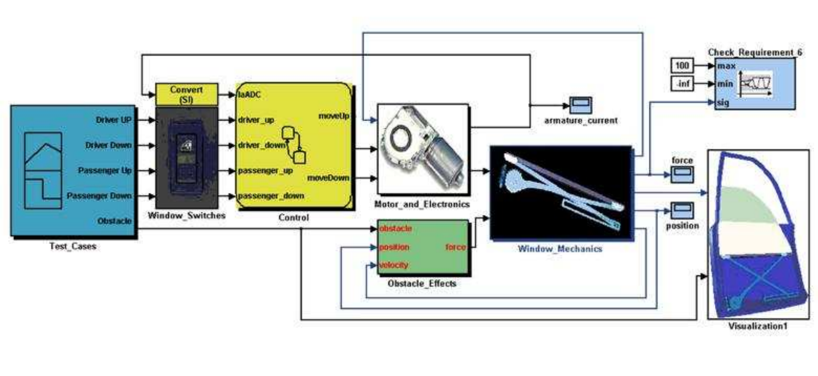


GPSS model of Telephone Exchange



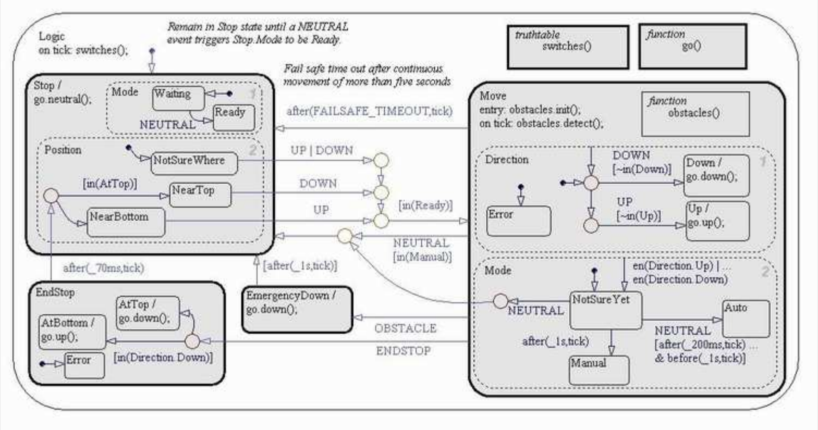
Multiple Formalisms: Power Window

Components in Different Formalisms

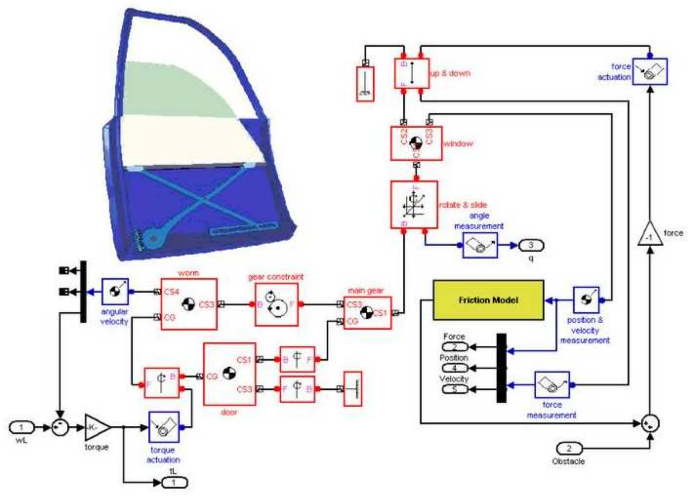


www.mathworks.com/products/demos/simulink/PowerWindow/html/PowerWindow1.html

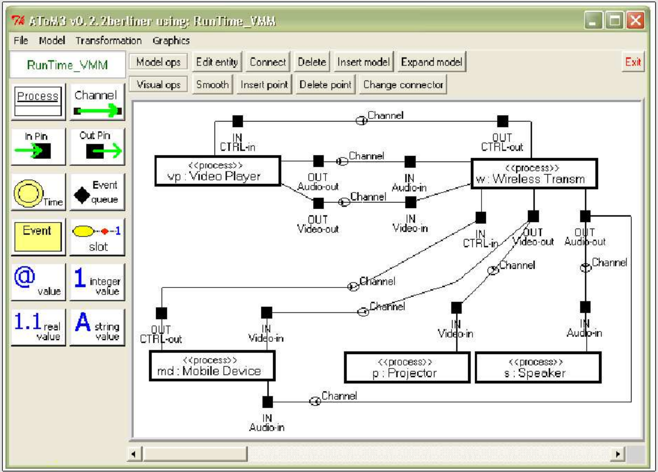
Controller, using Statechart(StateFlow) formalism



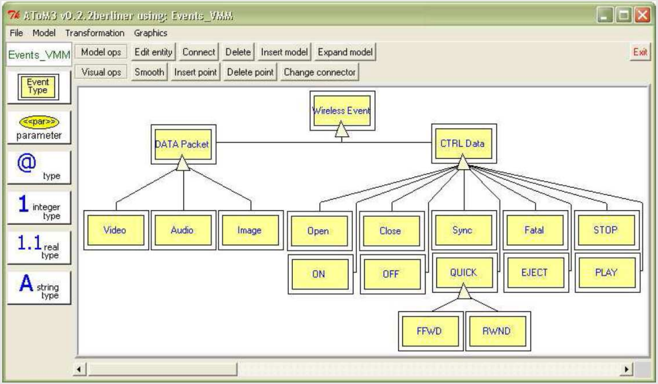
Mechanics subsystem



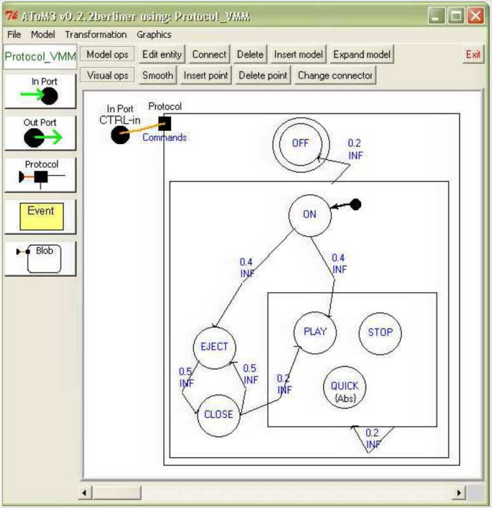
Multiple (consistent !) Views (in ≠ Formalisms)



View: Events Diagram



View: Protocol Statechart



No Free Lunch!

Solutions often introduce
their **own accidental complexity**

- multiple abstraction levels (need **morphism**)
- optimal formalism (need **precise meaning**)
- multiple formalisms (need **relationship**)
- multiple views (need **consistency**)

Multi-Paradigm Modelling
(*model* everything, minimize *accidental complexity*)

- at the most appropriate **level of abstraction**
- using the most appropriate **formalism(s)**
  Class Diagrams, Differential Algebraic Equations, Petri Nets, Bond Graphs, Statecharts, CSP, Queueing Networks, Sequence Diagrams, 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.

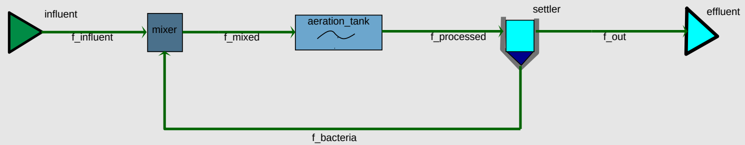
Special Issue: Grand Challenges for Modeling and Simulation.

Waste Water Treatment Plants (WWTPs)

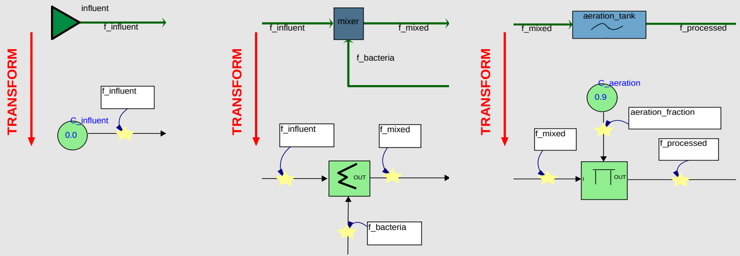
NATO's Sarajevo WWTP

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

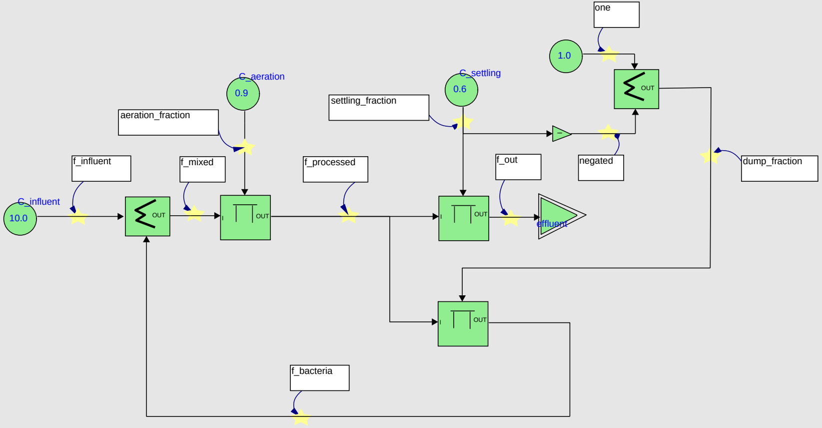
What does this WWTP model mean?



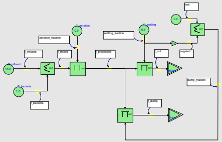
Transformation from WWTP to . . .



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

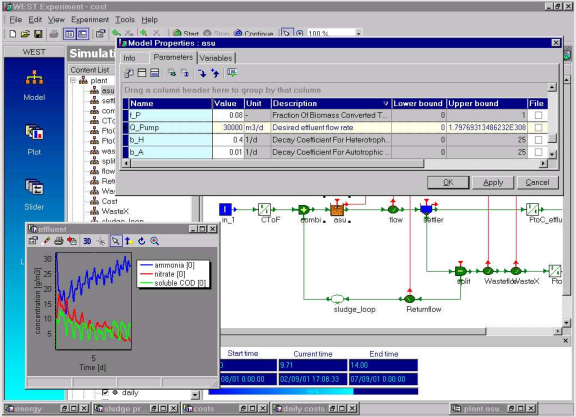


Meaning of the CBD

$[\![$  $]\!] =$

$$
\left\{
\begin{array}{rcl}
f\_influent & = & C\_influent \\
f\_bacteria & = & C\_bacteria \\
f\_mixed & = & f\_influent + f\_bacteria \\
aeration\_fraction & = & C\_aeration \\
f\_processed & = & aeration\_fraction * f\_mixed \\
settling\_fraction & = & C\_settling \\
negated & = & -settling\_fraction \\
one & = & 1 \\
dump\_fraction & = & one + negated \\
f\_dump & = & f\_processed * dump\_fraction \\
f\_out & = & settling\_fraction * f\_processed
\end{array}
\right.
$$

WWTP Domain-Specific Modelling Environment

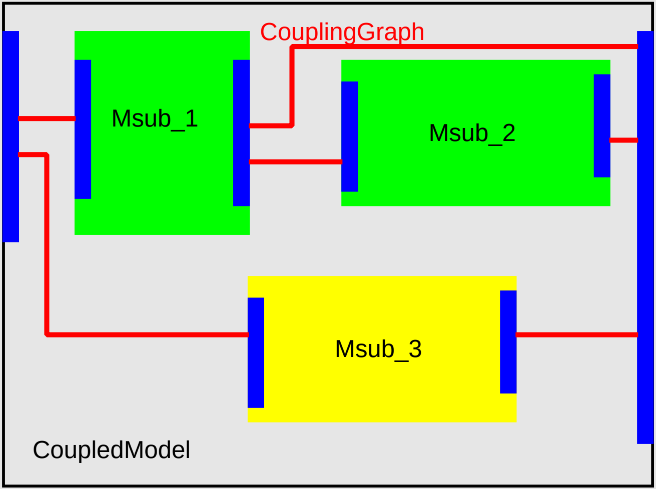


www.hemmis.com/products/west/

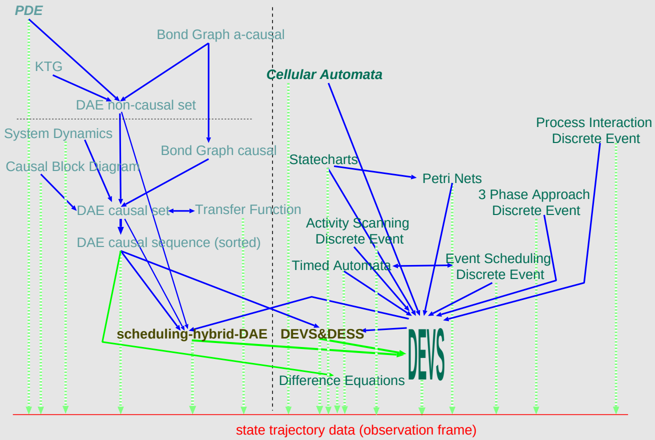
Henk Vanhooren, Jurgen Meirlaen, Youri Amerlinck, Filip Claeys, Hans Vangheluwe, and Peter A. Vanrolleghem.

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

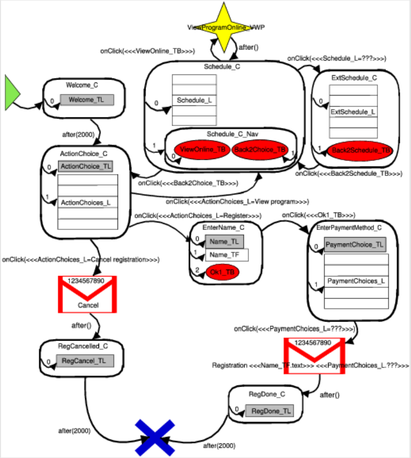
Multi-formalism coupled model: multi-formalism modelling



Formalism Transformation Graph

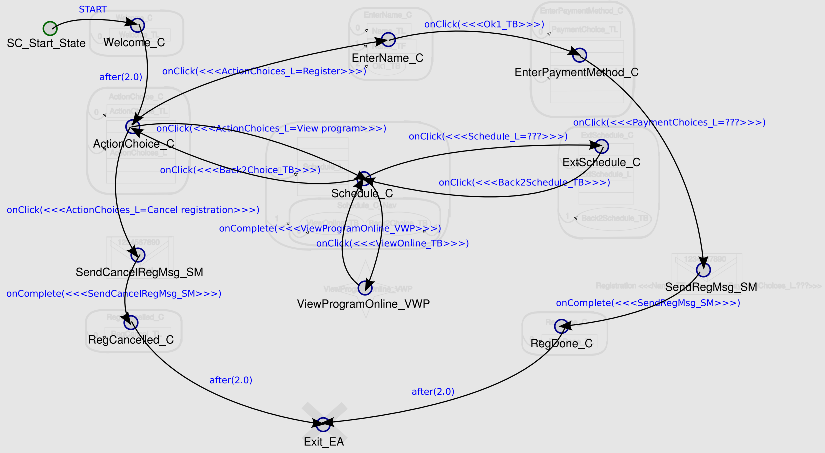


DS(V)M example application, the PhoneApps Domain-Specific model

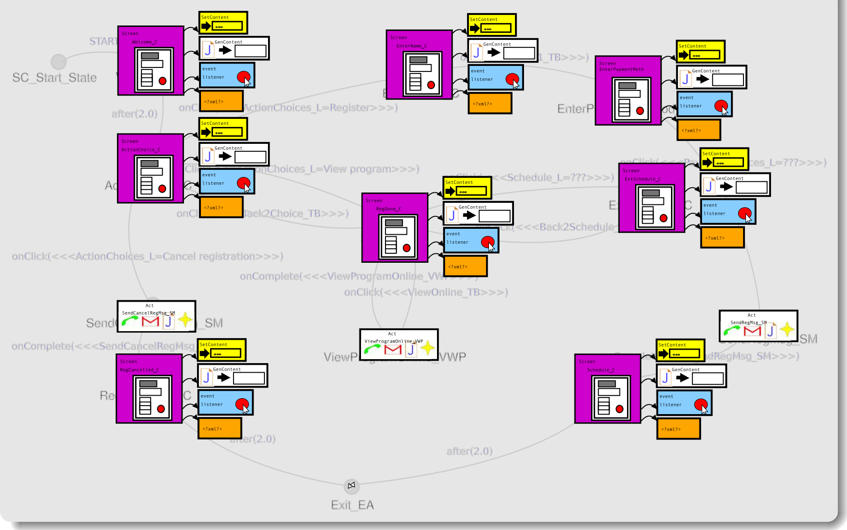


DS(V)M example application: conference registration (Google Android)

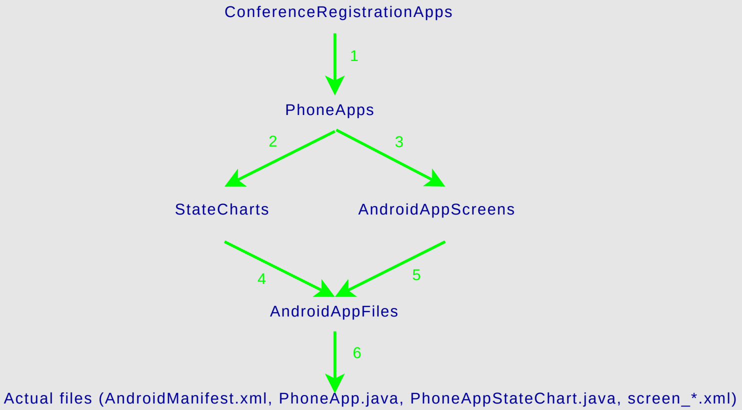
Transformation: extract behaviour (Statechart)



Transformation: extract presentation



Only transform . . .



Raphael Mannadiar and Hans Vangheluwe. *Modular synthesis of mobile device applications from domain-specific models*. Proceedings of the seventh International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MoMPES). 2010.

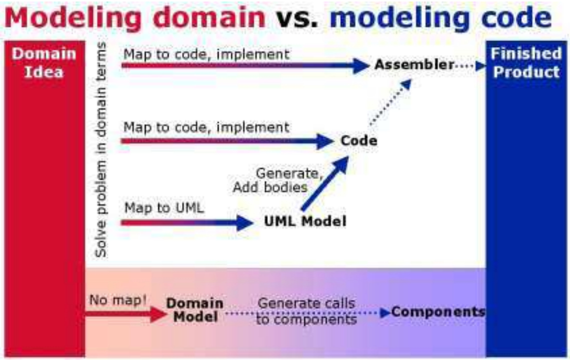
Why DS(V)M ?
(as opposed to General Purpose modelling)

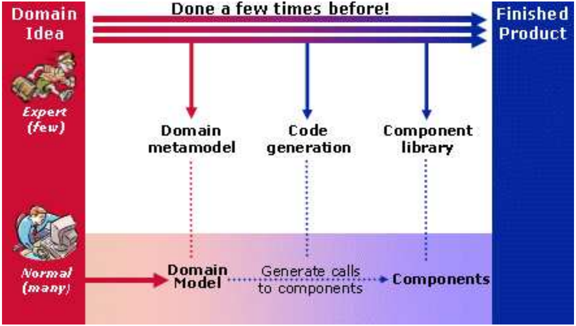
- **match the user's mental model** of the problem domain
- **maximally constrain** the user (to the problem at hand)
  $\Rightarrow$ easier to learn
  $\Rightarrow$ avoid errors
- **separate** domain-expert's work
  from analysis/transformation expert's work
- re-use **transformation** knowledge
  (such as in variations of a Domain-Specific formalism)

Anecdotal evidence of 5 to 10 times speedup

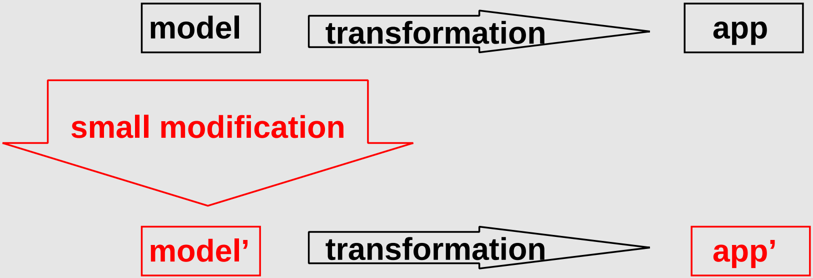
Steven Kelly and Juha-Pekka Tolvanen.

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

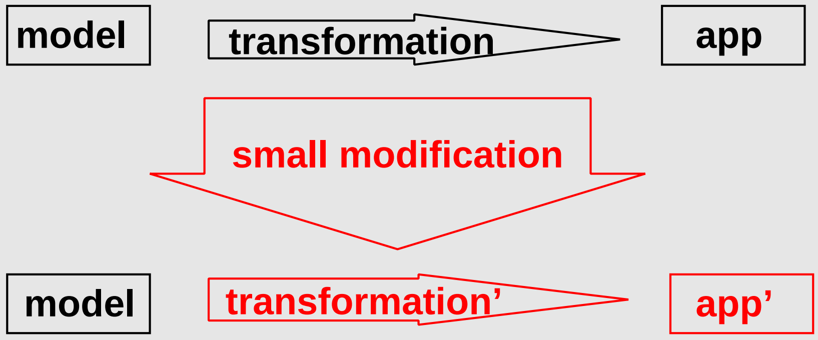

Modeling domain vs. modeling code
Domain Idea
Solve problem in domain terms
Map to code, implement
Assembler
Finished Product
Map to code, implement
Code
Generate, Add bodies
Map to UML
UML Model
No map!
Domain Model
Generate calls to components
Components


Domain Idea
Done a few times before!
Finished Product
Expert (few)
Domain metamodel
Code generation
Component library
Normal (many)
Domain Model
Generate calls to components
Components

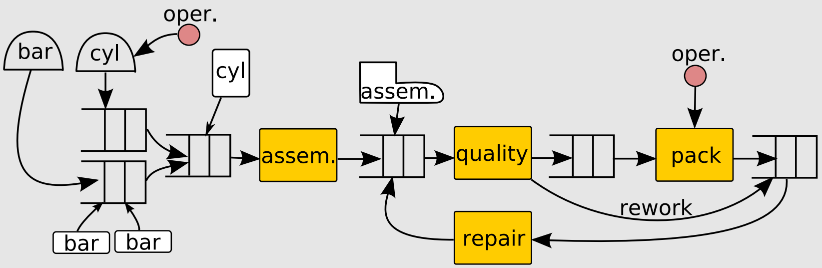
Model-Based Development: Modify the Model



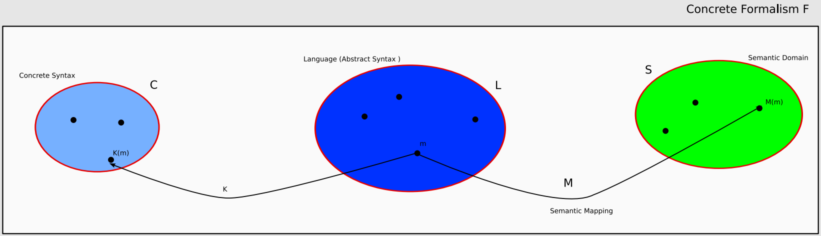
Model-Based Development: Modify the Transformation



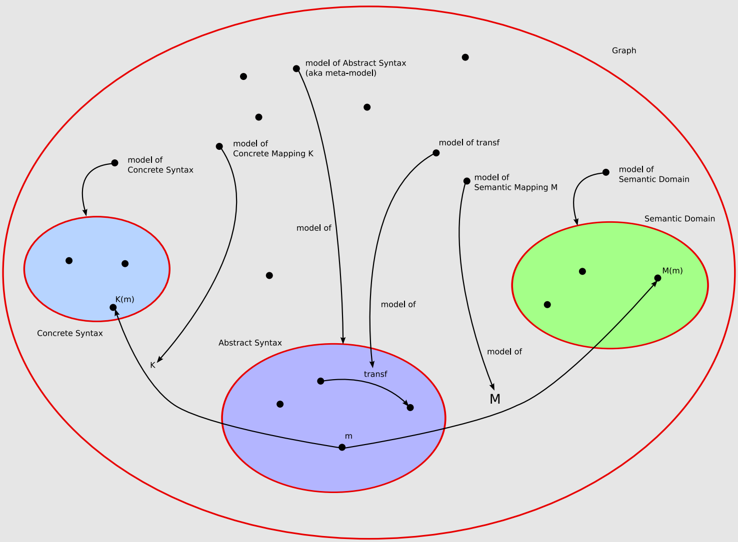
A Production System Model



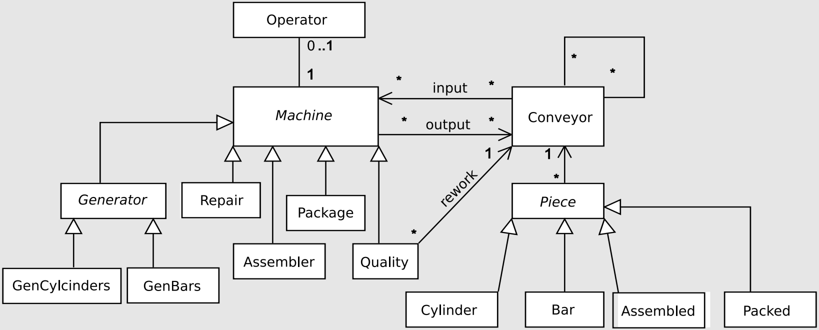
Modelling Languages as Sets of Models



Meta-Modelling . . . and more

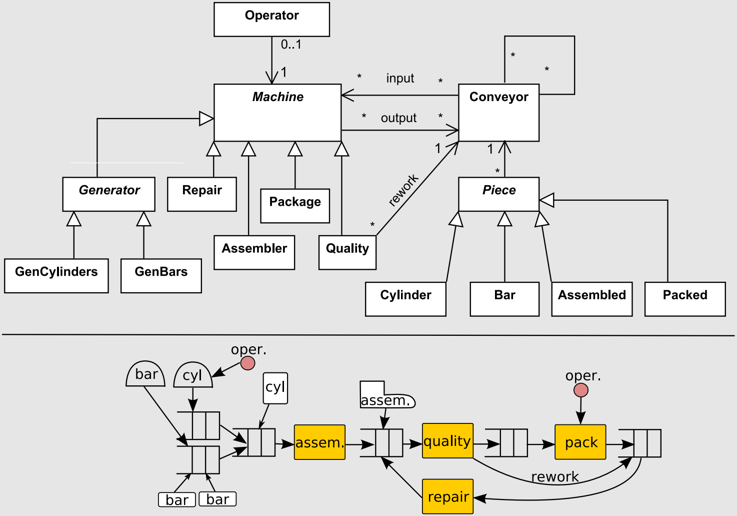


Modelling Abstract Syntax: Meta-Model



not shown: attributes, local and global contraints

Modelling Concrete Syntax (and UI Behaviour)



Meta-Modelling Challenges

- scalability of (meta-)models
- model differencing and meaningful model version control

  Antonio Cicchetti, Davide Di Ruscio, Alfonso Pierantonio. A Metamodel Independent Approach to Difference Representation. Journal of Object Technology 6(9): 165-185 (2007)
- (meta-)model evolution

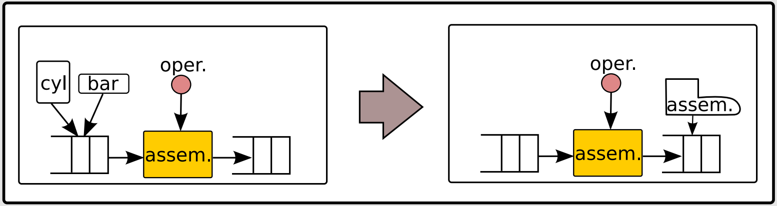
  Bart Meyers and Hans Vangheluwe. A framework for evolution of modelling languages. Science of Computer Programming, 2011. http://dx.doi.org/10.1016/j.scico.2011.01.002.
- deal with concrete syntax
  (mix textual/visual) in a unified manner

  Francisco Pérez Andrés, Juan de Lara, Esther Guerra. Domain Specific Languages with Graphical and Textual Views. AGTIVE 2007: 82-97
- debugging

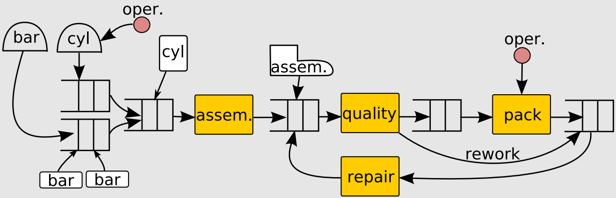
  Raphael Mannadiar and Hans Vangheluwe. Debugging in Domain-Specific Modelling. In The third International Conference on Software Language Engineering - SLE, volume 6563 of Lecture Notes in Computer Science (LNCS), pages 276 - 285. Springer, 2011. Eindhoven, The Netherlands.

Modelling Operational Semantics in the form of Rules

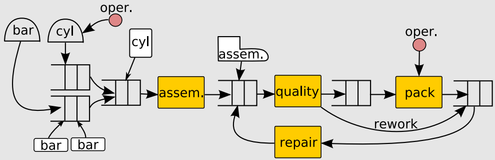
assemble

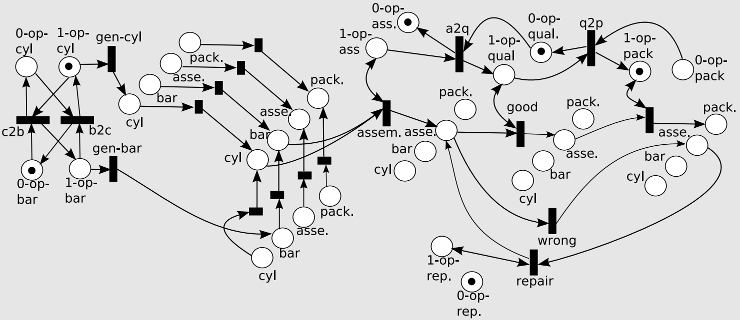


Note the use of **concrete** syntax !

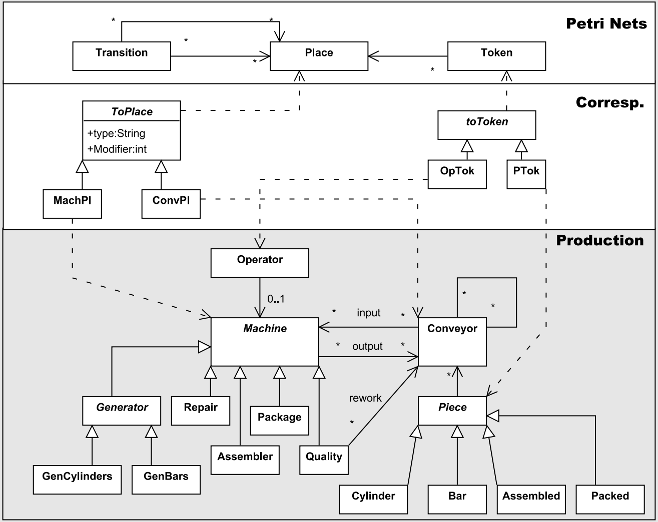


Denotational Semantics

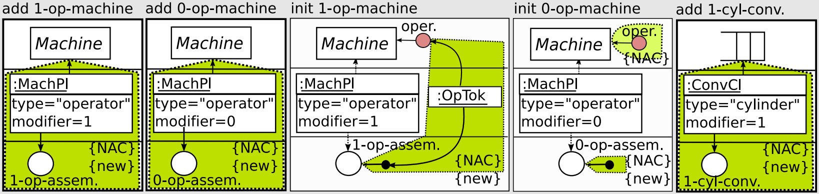




Modelling Denotational Semantics



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



Juan de Lara, Hans Vangheluwe. Automating the transformation-based analysis of visual languages. Formal Aspects of Compututing 22(3-4):297-326 (2010)

Model Transformation Challenges

- precise modelling of transformation languages (including higher-order transformations)

  Thomas Kühne, Gergely Mezei, Eugene Syriani, Hans Vangheluwe, and Manuel Wimmer. Systematic transformation development. Electronic Communications of the EASST, 21: Multi-Paradigm Modeling, 2009. http://eceasst.cs.tu-berlin.de/index.php/eceasst/issue/view/30.
- families of transformation languages Eugene Syriani and Hans Vangheluwe. De-/re-constructing model transformation languages. Electronic Communications of the EASST, 29: Graph Transformation and Visual Modeling Techniques (GT-VMT), 2010. http://eceasst.cs.tu-berlin.de/index.php/eceasst/issue/view/39.
- standardization/interoperability
- scalability (expressiveness and performance)

Model Transformation Challenges ctd.

- analysis of (properties of) model transformations (and of properties of transformed models)

  Levi Lucio, Bruno Barroca, Vasco Amaral. A Technique for Automatic Validation of Model Transformations. MoDELS 2010: 136-150

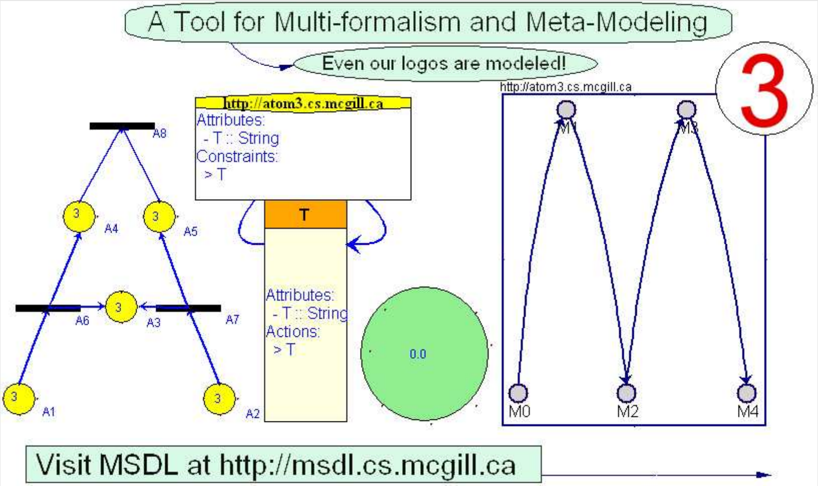
- automated testing of model transformations (and of transformed models)
- debugging

  Raphael Mannadiar and Hans Vangheluwe. Debugging in Domain-Specific Modelling. In The third International Conference on Software Language Engineering - SLE, volume 6563 of Lecture Notes in Computer Science (LNCS), pages 276 - 285. Springer, 2011. Eindhoven, The Netherlands.

- trace-ability (backward links)
- from transformations to relationships (consistency)

Eat Your Own Dogfood!

Model Transformation

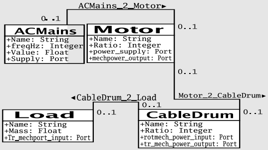


Juan de Lara and Hans Vangheluwe. $AToM^3$: A tool for multi-formalism and meta-modelling. FASE, LNCS 2306, pages 174 - 188. 2002.

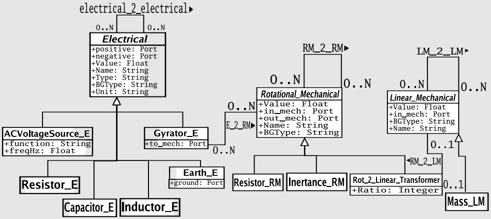
Design-space Exploration

Metamodel

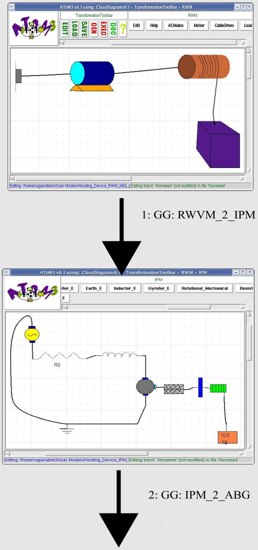
Real World Visual Modeling Formalism



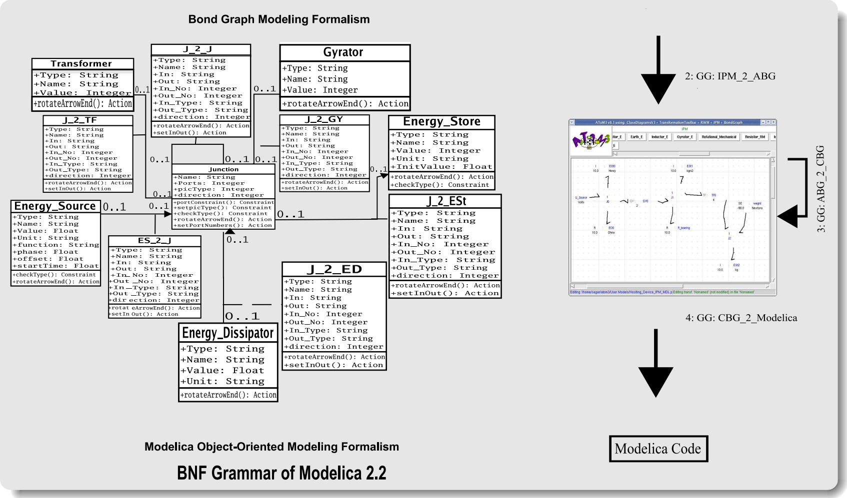
Idealized Physical Modeling Formalism



Generated Visual Modeling Environment

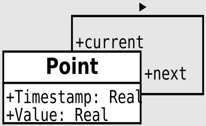


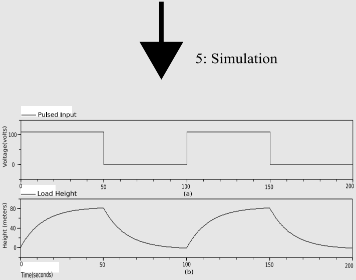
Design-space Exploration



Design-space Exploration

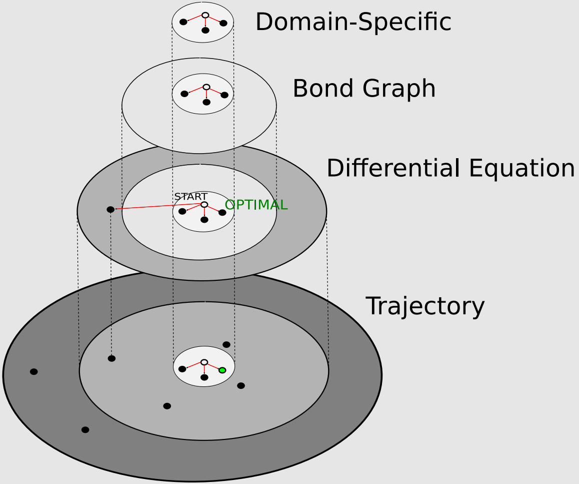
Trajectory Formalism



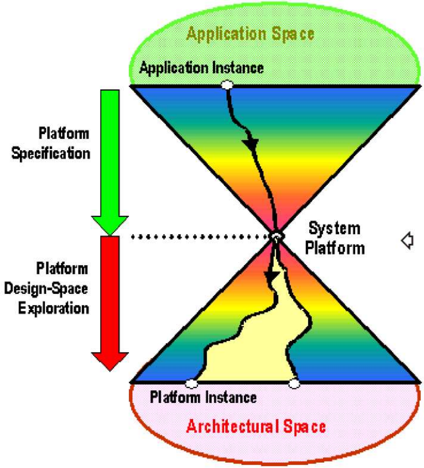


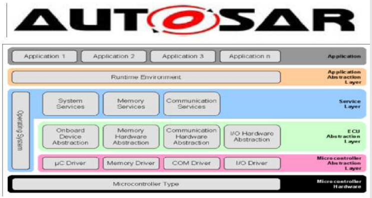
Sagar Sen and Hans Vangheluwe. Multi-domain physical system modeling and control based on meta-modeling and graph rewriting. In Computer Aided Control Systems Design (CACSD), pages 69 - 75, Munich, Germany, October 2006. IEEE.

Exploring the Design Space

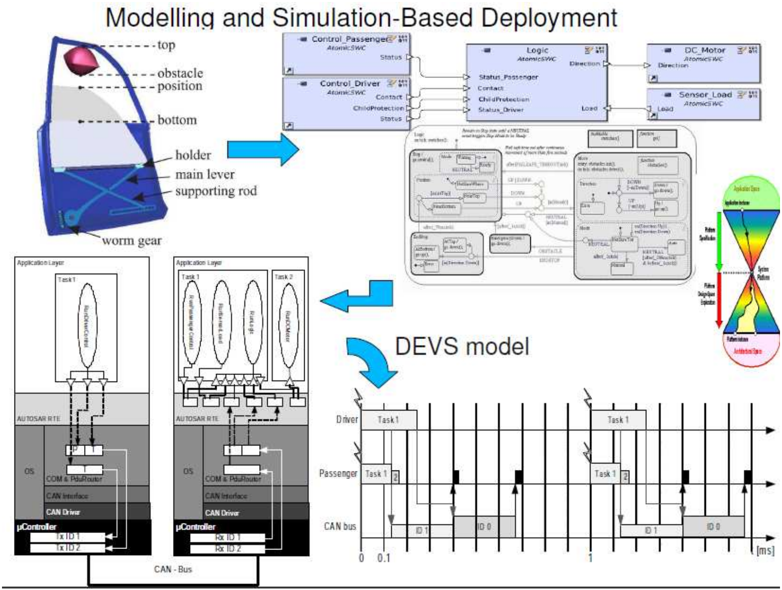


Deployment Space: Platform-Based Design (Alberto Sangiovanni-Vincentelli)

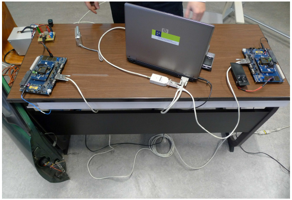


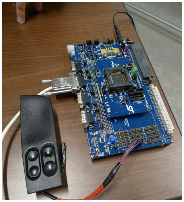


Deployment-space Exploration



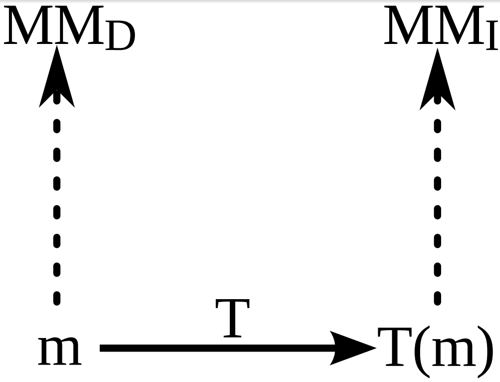
Deployment-space Exploration





Joachim Denil, Hans Vangheluwe, Pieter Ramaekers, Paul De Meulenaere, Serge Demeyer. DEVS for AUTOSAR platform modeling. Symposium On Theory of Modeling and Simulation. Boston, MA. 2011.

Co-evolution starting point



Bart Meyers and Hans Vangheluwe. A framework for evolution of modelling languages. Science of Computer Programming. 2011. (in press)

Model (instance) evolution

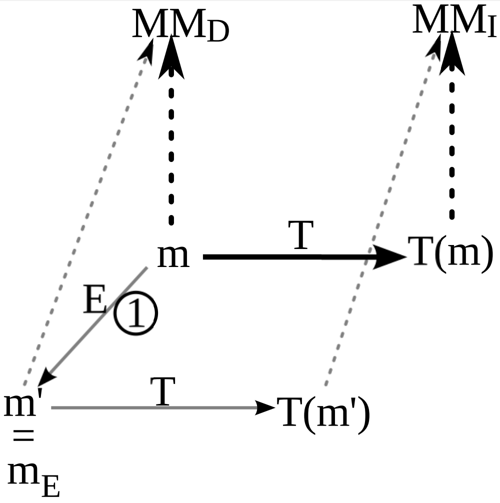
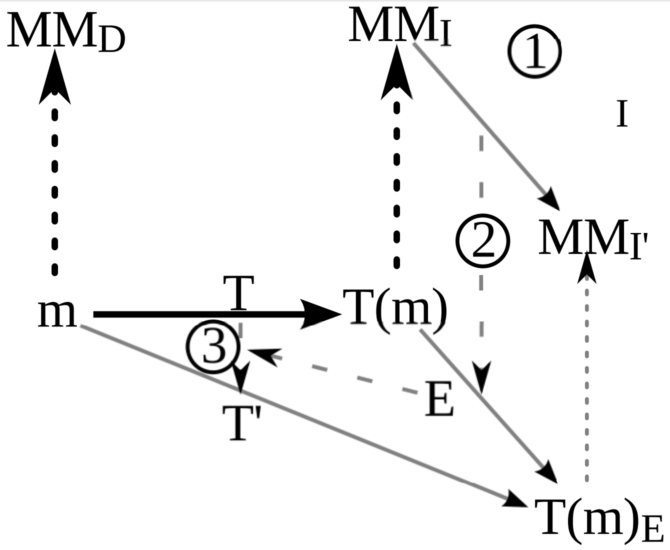
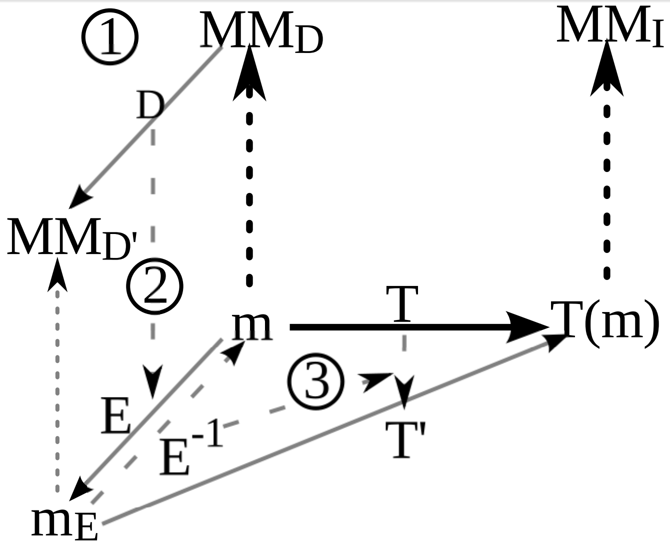


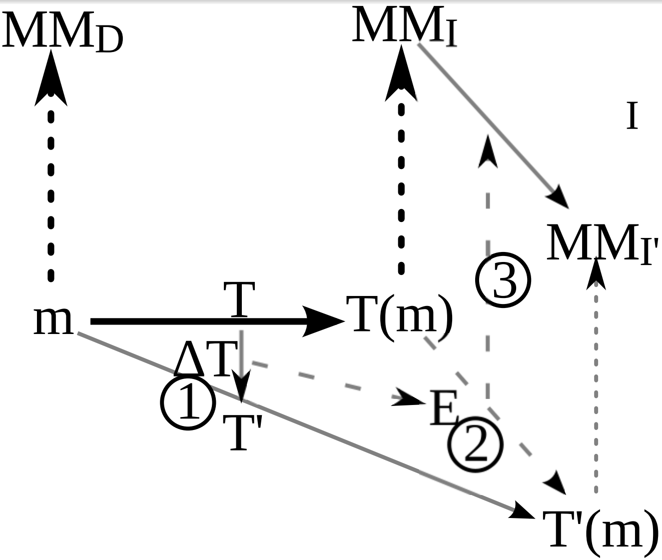
Image evolution



Domain evolution



Transformation evolution



Conclusions

model everything !

⇒ ability to manipulate knowledge

Conclusions

model everything !

$\Rightarrow$ ability to manipulate knowledge

- Causes of Complexity
- Dealing with Complexity
- Multi-Paradigm Modelling
- Domain-Specific Modelling
- Language Engineering
- Language Evolution
- Design-space Exploration
- Deployment-space Exploration

Questions ?