Multi-Paradigm Modelling and Language Engineering

Hans Vangheluwe

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

MODPROD Workshop
4 February 2009
Linköping, Sweden
Dealing with Complexity
Different Abstraction Levels (need morphism)
Multiple Formalisms: Power Window
Components in Different Formalisms

Multiple Views in Different Formalisms (need consistency!)
Causes of complexity?

View: Structure Diagram
Cause of complexity?

View: Events Diagram
View: Protocol Machine
Causes of complexity?

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.


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?

- influent
- mixer
- aeration_tank
- settler
- effluent
- f_influent
- f_mixed
- f_processed
- f_out
- f_bacteria
The Need for Transformations

Transformation from WWTP to ...
The Need for Transformations

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

\[
\begin{align*}
f_{\text{influent}} &= C_{\text{influent}} \\
f_{\text{bacteria}} &= C_{\text{bacteria}} \\
f_{\text{mixed}} &= f_{\text{influent}} + f_{\text{bacteria}} \\
aeration_{\text{fraction}} &= C_{\text{aeration}} \\
f_{\text{processed}} &= aeration_{\text{fraction}} \ast f_{\text{mixed}} \\
settling_{\text{fraction}} &= C_{\text{settling}} \\
\text{negated} &= -settling_{\text{fraction}} \\
\text{one} &= 1 \\
dump_{\text{fraction}} &= \text{one} + \text{negated} \\
f_{\text{dump}} &= f_{\text{processed}} \ast dump_{\text{fraction}} \\
f_{\text{out}} &= settling_{\text{fraction}} \ast f_{\text{processed}}
\end{align*}
\]
The Need for Transformations

Other uses of transformation

- operational semantics (as opposed to denotational)
- requirements → design → application
- requirements → tests
- optimization
- ...
- multi-formalism modelling
The Need for Transformations

Multi-Formalism Modelling: Formalism Transformation Graph

- **Process Interaction Discrete Event**
- **3 Phase Approach Discrete Event**
- **Event Scheduling Discrete Event**
- **Timed Automata**
- **Activity Scanning Discrete Event**
- **Statecharts**
- **Petri Nets**
- **Bond Graph causal**
- **DAE causal sequence (sorted)**
- **Transfer Function**
- **DAE causal set**
- **System Dynamics**
- **Causal Block Diagram**
- **KTG**
- **DAE non-causal set**
- **Bond Graph a-causal**
- **PDE**
- **Cellular Automata**
- **scheduling-hybrid-DAE**
- **DEVS&DESS**
- **Difference Equations**
- **state trajectory data (observation frame)**
Dissecting a Modelling Language

- Modelling Language (Formalism)
  - Syntax
    - Concrete
      - Textual
      - Visual
    - Abstract
  - Semantics
    - Semantic Mapping
    - Semantic Domain
A Production System Model
Modelling Languages as Sets of Models

Concrete Formalism $F$

Concrete Syntax $C$

Language (Abstract Syntax) $L$

Semantic Domain $S$

Semantic Mapping $M$

$K(m)$

$K$
Meta-Modelling
Modelling Abstract Syntax: Meta-Model

not shown: local and global constraints
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

Note the use of **concrete** syntax!
Denotational Semantics
Modelling Denotational Semantics

Petri Nets

Transition * Place * Token

Corresp.

ToPlace
+type: String
+Modifier: int

MachPI

ConvPI

toToken

OpTok

PTok

Production

Operator

0..1

Machine

* input

* output

* rework

Conveyor

Piece

* *

* *

* *

Cylinder

Bar

Assembled

Packed

Generator

Repair

Package

Assembler

Quality
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

Bond Graph Modeling Formalism

Modelica Object-Oriented Modeling Formalism

BNF Grammar of Modelica 2.2
Design-Space Exploration

- Domain-Specific
- Bond Graph
- Differential Equation
- Trajectory
WWTP Domain-Specific Modelling Environment

[Image of the WWTP Domain-Specific Modelling Environment]

www.hemmis.com/products/west/

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)
  ⇒ easier to learn
  ⇒ 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.

DS(V)M example application, the PhoneApps Domain-Specific model
DS(V)M example application: conference registration (Google Android)
Only transform ...

ConferenceRegistrationApps

1

PhoneApps

2 3

StateCharts AndroidAppScreens

4 5

AndroidAppFiles

6

Actual files (AndroidManifest.xml, PhoneApp.java, PhoneAppStateChart.java, screen_*_.xml)
Eat Your Own Dogfood!
A Tool for Multi-formalism and Meta-Modeling

Even our logos are modeled!

Model Everything! (to deal with Change)

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