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
Causes of complexity?

Different Abstraction Levels (need morphism)
Large Number of Components
Multiple Formalisms: Power Window
Complexity

Language Engineering

Domain-Specific Modelling (DSM)

Conclusions

Causes of complexity?

Components in Different Formalisms

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

**View: Events Diagram**

- **Wireless Event**
  - **CTRL Data**
    - **Video**
      - **Audio**
        - **Image**
          - **Open**
          - **Close**
          - **Sync**
          - **Fatal**
          - **STOP**
          - **Quick**
          - **Eject**
          - **Play**
          - **ON**
          - **OFF**
          - **FFWD**
          - **REWIND**
View: Protocol Machine
Non-compositional/Emergent Behaviour

non-compositionality of networks leads to emergent behaviour

separation  cohesion  alignment

www.red3d.com/cwr/boids/ (Craig Reynolds)
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 ...

- Influent: $f_{\text{influent}}$
- Mixer: $f_{\text{mixed}}$
- Aeration tank: $f_{\text{processed}}$
- Aeration fraction: $0.9$

Complexity
Language Engineering
Domain-Specific Modelling (DSM)
Conclusions
...its meaning (steady-state abstraction): Causal Block Diagram (CBD)
The Need for Transformations

Meaning of the CBD

\[
\begin{align*}
\text{f}_\text{influent} &= \text{C}_\text{influent} \\
\text{f}_\text{bacteria} &= \text{C}_\text{bacteria} \\
\text{f}_\text{mixed} &= \text{f}_\text{influent} + \text{f}_\text{bacteria} \\
\text{aeration}_\text{fraction} &= \text{C}_\text{aeration} \\
\text{f}_\text{processed} &= \text{aeration}_\text{fraction} \times \text{f}_\text{mixed} \\
\text{settling}_\text{fraction} &= \text{C}_\text{settling} \\
\text{negated} &= -\text{settling}_\text{fraction} \\
\text{one} &= 1 \\
\text{dump}_\text{fraction} &= \text{one} + \text{negated} \\
\text{f}_\text{dump} &= \text{f}_\text{processed} \times \text{dump}_\text{fraction} \\
\text{f}_\text{out} &= \text{settling}_\text{fraction} \times \text{f}_\text{processed}
\end{align*}
\]
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

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

- Syntax
  - Concrete
    - Textual
    - Visual
  - Modelling Language (Formalism)
    - Abstract
    - Semantic Mapping
    - Semantic Domain
A Production System Model

[Diagram showing a production system model with nodes labeled 'bar', 'cyl', 'assem.', 'quality', 'pack', 'repair', and arrows indicating flow and operations like 'oper.' and 'rework.']
Modelling Languages as Sets of Models
Meta-Modelling
Modelling Abstract Syntax: Meta-Model

- **Operator**
  - 0..1
  - 1

- **Machine**
  - input
  - output
  - rework

- **Conveyor**
  - *

- **Piece**
  - Cylinder
  - Bar
  - Assembled
  - Packed

- **Generator**
  - GenCylinders

- **Repair**
  - GenBars

- **Assembler**
- **Quality**

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

assemble

Note the use of **concrete** syntax!
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
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

Trajectory Formalism

Point
+current

+next

+Timestamp: Real
+Value: Real

5: Simulation

Graphs showing:
- Picked Input
- Load Height
- Height metering
WWTP Domain-Specific Modelling Environment

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

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/transformations 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 ...
Eat Your Own Dogfood!
A Tool for Multi-formalism and Meta-Modeling

Even our logos are modeled!

Attributes:
- T :: String
Constraints:
> T

Model Everything! (to deal with Change)

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