(Domain-Specific) Modelling Language Engineering

Hans Vangheluwe

http://msdl.cs.mcgill.ca/
at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling processes

Enabler: (domain-specific) modelling language engineering
including model transformation
Henk Vanhooren, Jurgen Meirlaen, Youri Amerlinck, Filip Claeys, Hans Vangheluwe and Peter A.Vanrolleghem.
http://www.mikebydhi.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


DS(V)M Example in Software Domain
smart phones, the application

MetaEdit+ (www.metacase.com)
DS(V)M Example: smart phones, the Domain-Specific model
Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

model → transformation → app

small modification

model' → transformation → app'
Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

small modification

small modification in model may lead to large change in app
~ choice of formalism (e.g., Statecharts)
Statecharts
Model-Based Development:
Modify the Transformation
(e.g., target platform changes, or optimization)
Building DS(V)M Tools Effectively...

- **development cost** of DS(V)M Tools may be prohibitive!
- \(\Rightarrow\) need **Modelling Language Engineering**
Dissecting Modelling

## Model Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapping feature</td>
<td>A model is based on an original.</td>
</tr>
<tr>
<td>reduction feature</td>
<td>A model only reflects a (relevant) selection of an original’s properties.</td>
</tr>
<tr>
<td>pragmatic feature</td>
<td>A model needs to be usable in place of an original with respect to some purpose.</td>
</tr>
</tbody>
</table>
REALITY

Real-World entity

only study behaviour in experimental context

System S

experiment within context

Experiment Observed Data

MODEL

Base Model

GOALS

Model Base a-priori knowledge

Model M

within context

simulate = virtual experiment

Simulation Results

Modelling and Simulation Process

Model Validity
Range

\[ F = -kx \]
Abstraction Relationship

**foundation:** the *information* contained in a model $M$. Different *questions* (properties) $P = I(M)$ which can be asked concerning the model. These questions either result in true or false.

*Abstraction* and its opposite, *refinement* are relative to a *non-empty set of questions* (properties) $P$.

- If $M_1$ is an *abstraction* of $M_2$ with respect to $P$, for all $p \in P$: $M_1 \models p \implies M_2 \models p$. This is written $M_1 \sqsupseteq_P M_2$.
- $M_1$ is said to be a *refinement* of $M_2$ iff $M_2$ is an *abstraction* of $M_1$. This is written $M_1 \sqsubseteq_P M_2$. 
“All non-trivial abstractions, to some degree, are leaky.”

Joel Spolsky
http://www.joelonsoftware.com/articles/LeakyAbstractions.html

Alexandre Muzy, David R. C. Hill. What is new with the activity world view in modeling and simulation?: using activity as a unifying guide for modeling and simulation. Winter Simulation Conference 2011: 2887-2899.

Token Models
Rôles a Model may Play
Ontological vs. Linguistic Conformance/Instatiation
Linguistic Conformance: morphism
strict Model PetriNets@1 {
    abstract Node NamedElement{
        name : String {id};
    }
    Node Place : NamedElement {
        tokens  : int = 0;
        outTrans : Transition[*] {ordered,unique};
        inTrans  : Transition[*] {ordered,unique};
        minTokens: $self.tokens>=0$
    }
    Node Transition : NamedElement {
        inPlaces : Place[*] {ordered,unique};
        outPlaces: Place[*] {ordered,unique};
    }
    Edge ArcPT (Place.outTrans, Transition.inPlaces) {
        weight: int = 1;
    }
    Edge ArcTP (Transition.outPlaces, Place.inTrans) {
        weight: int = 1;
    }
    minWeight(ArcTP, ArcPT): $self.weight>0$
    minPlaces:$Place.allInstances().size()>0$
}

load "PetriNets"

PetriNets Test{
    Place p0{name="p0"; tokens=2;}
    Place p1{name="p1"; tokens=0;}
    Place p2{name="p2"; tokens=2;}
    Transition t1{name="t1";}
    ArcPT pt0(p0,t1){weight=1;}
    ArcPT pt1(p2,t1){weight=1;}
    ArcTP tp(t1,p1){weight=2;}
}
MetaDepth operational semantics

operation main() {
    'Simulating the Petri Net'.println();
    while (Transition.allInstances()->exists(t|t.enabled() and t.fire()) ) {} 
}

operation Transition enabled() : Boolean {
    ('checking enabledness of '+self.name).println();
    return self.ArcPT->forAll(arc| arc.inPlaces.tokens>=arc.weight);
}

operation Transition fire() : Boolean {
    ('Firing '+self.name).println();
    for (arc in self.ArcPT)
        arc.inPlaces.tokens := arc.inPlaces.tokens-arc.weight;
    for (arc in self.ArcTP)
        arc.outPlaces.tokens := arc.outPlaces.tokens+arc.weight;
    return true;
}
Meta-models as Language Definitions
Meta-hierarchy – OMG’s 4 Layer Architecture

Syntax, **Semantics**, and all that Stuff

David Harel, Bernhard Rumpe.

*Meaningful Modeling: What's the Semantics of "Semantics"?*


- “operational” semantics
- “denotational” (transformational) semantics
Operational vs. Denotational (Translational) semantics

NATO’s Sarajevo Waste Water Treatment Plant

www.nato.int/sfor/cimic/env-pro/waterpla.htm
What does this WWTP model mean?

- **Influent** (f_influent) flows into a **mixer** (f_mixed).
- The mixed water then enters an **aeration tank** (f_processed).
- From the aeration tank, it goes to a **settler** (f_out) and then to the **effluent**.
- Additionally, there is a flow of **f_bacteria** from thesettler back to the influent.
...its meaning (steady-state abstraction):
Causal Block Diagram (CBD)
Meaning of the CBD... semantic mapping onto algEqns

\[
\begin{align*}
f_{\text{influent}} & = C_{\text{influent}} \\
f_{\text{bacteria}} & = C_{\text{bacteria}} \\
f_{\text{mixed}} & = f_{\text{influent}} + f_{\text{bacteria}} \\
aeration_{\text{fraction}} & = C_{\text{aeration}} \\
f_{\text{processed}} & = aeration_{frac} \times f_{\text{mixed}} \\
settling_{\text{fraction}} & = C_{\text{settling}} \\
negated & = -settling_{frac} \\
one & = 1 \\
dump_{\text{frac}} & = one + negated \\
f_{\text{dump}} & = f_{\text{processed}} \times dump_{\text{frac}} \\
f_{\text{out}} & = settling_{frac} \times f_{\text{processed}}
\end{align*}
\]
“linguistic” view on Modelling Languages/Formalisms: Syntax and Semantics
Explicit “linguistic” Modelling of Modelling Languages/Formalisms
Meaning of the CBD ... semantic mapping onto algEqns

\[
\begin{align*}
f_{\text{influ}} &= C_{\text{influ}} \\
f_{\text{bact}} &= C_{\text{bact}} \\
f_{\text{mix}} &= f_{\text{influ}} + f_{\text{bact}} \\
a_{\text{aer}} &= C_{\text{aer}} \\
f_{\text{proc}} &= a_{\text{aer}} \times f_{\text{mix}} \\
F_{\text{sett}} &= C_{\text{sett}} \\
\neg &= -F_{\text{sett}} \\
\text{one} &= 1 \\
d_{\text{dump}} &= \text{one} + \neg \\
f_{\text{dump}} &= f_{\text{proc}} \times d_{\text{ump}} \\
f_{\text{out}} &= F_{\text{sett}} \times f_{\text{proc}}
\end{align*}
\]
Causal Block Diagrams (syntax)
logicalTime ← 0

while not end_condition do
  schedule ← LOOPDETECT(DEPGRAPH(cbd))
  for gblick in schedule do
    COMPUTE(gblick)
  end for
  logicalTime ← logicalTime + Δt
end while
Causal Block Diagrams (semantics)
Formalism Transformation Graph (FTG)

- WWTP
- CBD
- (Diff)AlgEqns
- continuous-time signals
Formalism Transformation Graph (FTG)

Bran Selic: “fragmentation problem”

Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe.


https://www.youtube.com/watch?feature=player_detailpage&v=RYtea2BiQ98
at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling processes

Enabler: (domain-specific) modelling language engineering,
including model transformation