Assigning Meanings to Models

Antonio Vallecillo
Francisco Durán, José E. Rivera
Atenea Research Group

MtATL 2009. Nantes, July 8 2009
“A description or specification of a system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language.” [MDA guide (V1 and 2, ab/2003-01-03, 23 January, 2003)]

"A model represents some concrete or abstract thing of interest, and does so with a specific purpose. A model is related to the thing it represents by explicit or implicit correspondences between the elements of that model and the parts of that thing. This correspondence enables understanding the intended meaning of that model.” [MDA Guide (V3, ormsc/05-11-03, 30 November 2005)]

"A model captures a view of a physical system. It is an abstraction of the physical system, with a certain purpose. This purpose determines what is to be included in the model and what is irrelevant. Thus the model completely describes those aspects of the physical system that are relevant to the purpose of the model, at the appropriate level of detail." [UML Superstructure 2.1.1 (formal/2007-02-05)]

"A description of (part of) a system written in a well-defined language." (NOTE: Equivalent to specification.) [Kleppe, 2003]

"A representation of a part of the function, structure and/or behavior of a system” [Model Driven Architecture (MDA) ormsc/2001-07-01]

"A set of statements about the system." (Statement: expression about the system that can be true or false.) [Seidewitz, 2003]

"M is a model of S if M can be used to answer questions about S" [D.T. Ross and M. Minsky, 1960]
Model

1: obsolete : a set of plans for a building
2: dialect British : copy, image
3: structural design <a home on the model of an old farmhouse>
4: a usually miniature representation of something ; also : a pattern of something to be made
5: an example for imitation or emulation
6: a person or thing that serves as a pattern for an artist ; especially : one who poses for an artist
7: archetype
8: an organism whose appearance a mimic imitates
9: one who is employed to display clothes or other merchandise
10 a: a type or design of clothing b: a type or design of product (as a car)
11: a description or analogy used to help visualize something (as an atom) that cannot be directly observed
12: a system of postulates, data, and inferences presented as a mathematical description of an entity or state of affairs; also: a computer simulation based on such a system <climate models>
13: version
14: animal model
Meaning

1 a: the thing one intends to convey especially by language: purport
1 b: the thing that is conveyed especially by language: import

2: something meant or intended: aim
   <a mischievous meaning was apparent>

3: significant quality; especially: implication of a hidden or special significance
   <a glance full of meaning>

4 a: the logical connotation of a word or phrase
4 b: the logical denotation or extension of a word or phrase
Semantics

1: *the study of meanings* [Merrian-Webster]

a: the historical and psychological study and the classification of changes in the signification of words or forms viewed as factors in linguistic development

b (1): semiotic (2): a branch of semiotic dealing with the relations between signs and what they refer to and including theories of denotation, extension, naming, and truth

Formal semanticists are concerned with the *modeling of meaning in terms of the semantics of logic*. In computer science, where it is considered as an application of mathematical logic, *semantics reflects the meaning of programs or functions* [wikipedia]
There are only 10 types of people in the world: Those who understand binary, and those who don’t.
“There are only 10 types of people in the world: Those who understand binary, and those who don't”
Same model for different concepts

There are only 10 types of people in the world: Those who understand binary, and those who don’t.

Your meaning goes here…
Different models for the same concept
What does this model mean?
What does this model mean?
import $[[M]]$ $\cong$ purport $\{\{S\}\}$

$M$ $\cong$ $S$
Why do I need to assign meanings to models?

What do I need models for?

- **Describe** the system
  - Structure, behaviour, ...
  - Separate concepts at different conceptual levels
  - Communicate with stakeholders
- **Understand** the system
  - If existing (legacy applications)
- **Validate** the model
  - Detect errors and omissions in design ASAP
    - Mistakes are cheaper at this stage
  - Prototype the system (execution of the model)
  - Formal analysis of system properties
- **Drive implementation**
  - Code skeleton and templates, complete programs (?)

[Selic, 2003]
How do we assign meaning?

How do we express the meaning of

- Structure?
- Behavior?
- Time-dependent functionality?
- QoS properties?
- ...

Which is the best **notation** for each of those aspects?

- It depends on the **purpose** of the model...
- ...and must have a **precise** meaning
Domain Specific Modeling Languages
Each notation is more apt for a task

\[
\begin{array}{c}
MCMLXVII \\
+ DLXXIX \\
??? \\
\end{array}
\rightarrow
\begin{array}{c}
1.967 \\
+ 579 \\
\end{array}
\]
Each notation is more apt for a task

\[
\begin{align*}
MCMLXVII + DLXXIX & = 1.967 \\
??? + 579 & = 2.546
\end{align*}
\]
Each notation is more apt for a task

MCMLXVII + DLXXIX = MMDXLVI

1.967 + 579 = 2.546
A 40-years-old man has a daughter and a son. If the difference of age between the kids is 4 years, and the sum of their ages is half of the age of the father, how old are they?

\[
\begin{align*}
  x - y &= 4 \\
  x + y &= 20 \\
\end{align*}
\]

\[
2x = 24
\]

\[
x = 12
\]

\[
y = 8
\]

**Solution:** the older is 12 and the younger is 8
An invariant through the history of mature disciplines is the search for notations that allow formulating problems in a language that allows their easy solution.

\[
\frac{\partial f}{\partial x_i}(a_1, \ldots, a_n) = \lim_{h \to 0} \frac{f(a_1, \ldots, a_i + h, \ldots, a_n) - f(a_1, \ldots, a_n)}{h}.
\]

\[\vec{F} = \frac{d}{dt}(m\vec{v}) \quad \int_{-N}^{N} f(x) \, dx \quad \sum_{n=1}^{\infty} \frac{1}{n^2} \quad \Box (p \to q) \to (\Box p \to \Box q)\]

\[
\rho \left( \frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} \right) = -\nabla p + \nabla \cdot \mathbb{T} + \vec{f},
\]

http://en.wikipedia.org/wiki/Temporal_logic
Visual DSMLs

(a) Abstract syntax (object diagram)

(b) Concrete syntax
We need more than syntax...

- **Describe** meaningful models
  - At the appropriate level of abstraction
  - In a correct, complete and accurate manner
  - Using a notation natural to the target domain engineer

- **Animate** models
  - Explicitly define behavioral semantics of DSLs so that models can be understood, manipulated and maintained by both users and machines
  - Add Non-Functional Properties (Time, QoS,...) to DSLs
  - Make models amenable to simulation

- **Analyse** models
  - Connect DSLs to Analysis tools
Anatomy of a DSML (II)

- **DSML**
  - **Semantics**: 0..* connection
  - **AbstractSyntax**: 1 connection
  - **ConcreteSyntax**: 1..* connection

- **MetaModel**
  - **+specification**: 1 connection
  - **+source**: 1 connection
  - **+target**: 1 connection

- **ConcreteSyntax Mapping**
  - **+specification**: 0..1 connection
Semantic (or “Meaningful”) Domains

The meaning of a model $M$ in a given domain $D$ is defined by its interpretation in a meaningful semantic domain $D'$. Each Meaningful Domain has

- Precise semantics
- A set of (equivalent) notations
- A set of Analysis Tools
- Underlying logic
Bridges between Semantic Domains
Bridges between Semantic Domains
How to implement Semantic Mappings?

As Model Transformations!!!

Types
- Domestic
- Horizontal
- Vertical
- Abstracting
- Refining
- Pruning
- Forgetful
- ...

The relationship between domains $D$ and $D'$ is defined by a model transformation $T: D \rightarrow D'$.

$$[[M]]_D := [[T(M)]]_{D'}.$$
How do we analyse models?

Crossing the bridges!!!
Models to connect (analysis) tools!

**Design Tools**

**MCAD Tools**
- CATIA, NX, Pro/E®, ...

**Analysis Building Blocks** *(ABBs)*
- 1D Linear Elastic Model
- Continuum ABBs:
  - Extensional Rod
  - Torsional Rod

**Analysis Templates of Diverse Behavior & Fidelity** *(CBAMs)*
- Margin of Safety
- Allowable Stress
- Margin of Safety (> case)
- Allowable Stress (> case)

**Analysis Solvers (via SMMs)**
- FEA
  - Ansys
  - Abaqus®
  - MSC Nastran®
  - MSC Patran®
  - NX Nastran®

**Materials Libraries**
- In-House, ...

**Parts Libraries**
- In-House®, ...

* = Item not yet available in toolkit—all others have working examples 2007-04

---

**Legend**

- Tool Associativity
- Object Re-use

**General Math**
- Mathematica
- Matlab®
- MathCAD®

---

MtATL 2009. Nantes, July 8, 2009

[Borrowed from Russell Peak presentation at OMG, 2007]
Assigning Meanings to Models

(Using model transformations)
Our proposal

- Specify the **structure** of Models/Metamodels with **usual** modeling notation and tools (Ecore, EMF, GMF, ...)
- Specify the **behavior** of models using **visual** languages (including Time and QoS aspects)
- Specify the **structure** of models and metamodels using a **formal** system, i.e., a precise Semantic Domain (e.g., Maude)
- Specify the **behavior** of models using a **formal** system, i.e., in a Formal Semantic Domain (e.g., Maude)
- Define **mappings** between the **visual** and **formal** notations (the latter provides the meaning for the former)
- Make use of the **analysis tools** in the target domain to reason about the visual models
A Production System Example

conformsTo

Machine

Container

Part

Handle

Head

Conveyor

Assembler

Tray

User

Hammer

conformsTo

p

hag

c1

t1

capacity = 4

hag

c2

a

c3

capacity = 4

t2

MtATL 2009. Nantes, July 8, 2009
Specifying dynamic behavior

- Use of In-place Transformation Rules (e.g., graph transformations)
- Completely Independent from the underlying semantic framework (e.g., Maude)

\[ l: [\text{NAC}] \times \text{LHS} \rightarrow \text{RHS} \]
Adding time to behavioral specifications

- Part of the e-Motions modeling notation
- Rule duration
- Periodicity, soft scheduling
- Ongoing rules
- Access to the Global Time Elapse
  - Time stamps, scheduled actions

\[ l: [NAC] \times LHS \xrightarrow{t} RHS \]
Adding action executions

Specification of action properties

Without the need of unnaturally modifying the metamodel
Bridges between Semantic Domains

- Precise semantics
- A set of Analysis Tools
- Underlying logic

Structure
(Metamodel)

Behavior
(In-place transformations)

ATL Model Transformations

Rewriting Logic
(Reachability analysis, model checking…)

Petri Nets
(Termination, Confluence…)

Semantic Domain N
...

MAUDE

MAUDELING
ProductionSystem {
  < 'p : Plant | els : 'heg 'hag 'c1 'c2 't1 'a 'c3 't2 'u >
  < 'hag : HandleGen | in : null, out : 'c2, xPos : 1, yPos : 1 >
  < 'heg : HeadGen | in : null, out : 'c1, xPos : 1, yPos : 3 >
  < 'c1 : Conveyor | parts : nil, out : 't1, xPos : 2, yPos : 3 >
  < 'c2 : Conveyor | parts : nil, out : 't1, xPos : 2, yPos : 1 >
  < 't1 : Tray | parts : nil, capacity : 4, xPos : 3, yPos : 2 >
  < 'a : Assembler | in : 't1, out : 'c3, xPos : 4, yPos : 2 >
  < 'c3 : Conveyor | parts : nil, out : 't2, xPos : 5, yPos : 2 >
  < 't2 : Tray | parts : nil, capacity : 4, xPos : 6, yPos : 2 >
  < 'u : User | parts : nil, xPos : 6, yPos : 3 >
}
Representing Metamodels with Maude

```plaintext
op ProductionSystem : -> @Metamodel .
op PS : -> @Package .

sort PositionedEl .
subsort PositionedEl < @Class .
op PositionedEl : -> PositionedEl .

op xPos : -> @Attribute .
op yPos : -> @Attribute .

sort Container .
subsort Container < PositionedEl .
op Container : -> Container .
op parts : -> @Reference .

sort Machine .
subsort Machine < PositionedEl .
op in : -> @Reference .
op out : -> @Reference .

... eq isAbstract(Machine) = true .
... eq type(in) = Tray .
eq lowerBound(in) = 0 .
éq upperBound(in) = 1 .
... eq type(out) = Conveyor .
eq opposite(out) = null .
eq lowerBound(out) = 1 .
eq upperBound(out) = 1 .
```
Model management

- Model difference
- Model subtyping
- Model metrics

http://atenea.lcc.uma.es/index.php/Main_Page/Resources/Maudeling
Model difference: Comparison process

Matching
- Finding different objects from both models that represent the same element
- Model as a result
- Persistent identifiers vs. structural similarities

Differencing:
- Makes use of matching models to detect modified elements
- Model as a result

- Self-contained
- Compact
- Independent of the metamodel of the source models
A Model Difference Example

(Subtrahend Model)

< 'SM : StateMachine | initialState : 'ST1, containedStates : ('ST1, 'ST2) >
< 'TR : Transition | name : "Tr", src : 'ST1, target : 'ST2 >

(Minuend Model)

< 'SM : StateMachine | initialState : 'ST1, containedState : ('ST1, 'ST2) >
< 'TR : Transition | name : "Tr", src : 'ST1, target : 'ST2 >
< 'TR2 : Transition | name : "Tr2", src : 'ST2, target : 'ST1 >

(Difference Model)

< 'ST1@MOD : ModifiedElement | element : 'ST1@NEW, oldElement : 'ST1@OLD>
  < 'ST1@NEW : State | incoming : 'TR2 >
  < 'ST1@OLD : State | incoming : empty >
< 'ST2@MOD : ModifiedElement | element : 'ST2@NEW, oldElement : 'ST2@OLD >
  < 'ST2@NEW : State | outgoing : 'TR2 >
  < 'ST2@OLD : State | outgoing : empty >
< 'TR2@ADD : AddedElement | element : 'TR2@NEW >
  < 'TR2@NEW : Transition | name : "Tr2", src : 'ST2, target : 'ST1 >
Difference related operations

**Operation *do***
- \( \text{do}(M_s, M_d) = M_m \)
- Applies to a model all the changes specified in a difference model

**Operation *undo***
- \( \text{undo}(M_m, M_d) = M_s. \)
- Reverts all the changes specified in a difference model

\[
\text{undo(} \text{do}(M_s, M_d), M_d) = M_s \quad \text{do(} \text{undo}(M_m, M_d), M_d) = M_m
\]

**Sequential composition of differences**
- "Optimize" the process of applying successive modifications to the same model
Model type

- Essentially its metamodel

Model subtyping

- Model operations reuse (megamodeling)
- Type safety
- Polimorphism in MDSD
- Model bus, metamodel matchmaking, metamodel evolution
Model subtyping

- **Metamodels** $M', M$: $M' \leq M \iff$
  \[ \forall K \in \{M.\text{package}\} \exists K' \in \{M'.\text{package}\} \bullet (K' \leq K) \]

- **Packages** $K', K$: $K' \leq K \iff$
  \[ \text{isRelated}(K'.\text{name}, K.\text{name}) \land \forall C \in \{K.\text{class}\} \exists C' \in \{K'.\text{class}\} \bullet (C' \leq C) \]

- **Classes** $C', C$: $C' \leq C \iff$
  \[ \text{isRelated}(C'.\text{name}, C.\text{name}) \land (C'.\text{isAbstract} \implies C.\text{isAbstract}) \land \forall C \in \{C.\text{superTypes}\} \exists C' \in \{C'.\text{superTypes}\} \bullet (C' \leq C) \land \forall S \in \{C.\text{structuralFeatures}\} \exists S' \in \{C'.\text{structuralFeatures}\} \bullet (S' \leq S) \]

- **Attributes** $P', P$: $P' \leq P \iff$
  \[ \text{isRelated}(P'.\text{name}, P.\text{name}) \land (P'.\text{type} \leq P.\text{type}) \land (P'.\text{isUnique} = P.\text{isUnique}) \land (P.\text{lower} \leq P'.\text{lower}) \land (P'.\text{isOrdered} = P.\text{isOrdered}) \land ((P.\text{upper} = P'.\text{upper}) \land (2 \leq P.\text{upper} \leq P'.\text{upper})) \]

- **References** $R', R$: $R' \leq R \iff$
  \[ \text{isRelated}(R'.\text{name}, R.\text{name}) \land (R'.\text{type} \leq R.\text{type}) \land (R'.\text{isUnique} = R.\text{isUnique}) \land (R.\text{lower} \leq R'.\text{lower}) \land (R'.\text{isOrdered} = R.\text{isOrdered}) \land ((R.\text{upper} = R'.\text{upper}) \land (2 \leq R.\text{upper} \leq R'.\text{upper})) \land (R'.\text{opposite} \leq R.\text{opposite}) \]
Representing **Behavior** with Maude

```
rl [Carry] :
ProductionSystem {
  < p : P:Part | xPos : XPOS, yPos : YPOS, SFS >
  < c : Conveyor | parts : (p PARTS), out : t, SFS' >
  < t : Tray | xPos : XPOS', yPos : YPOS', parts : PARTS', SFS'' >
  OBJSET }
} =>
ProductionSystem{
  < p : P:Part | xPos : XPOS', yPos : YPOS', SFS >
  < c : Conveyor | parts : PARTS, out : t, SFS' >
  < t : Tray | xPos : XPOS', yPos : YPOS', parts : (p PARTS'), SFS'' >
  OBJSET }
```

MtATL 2009. Nantes, July 8, 2009
Precise Semantics of Timed Rules

Defined by a Semantic Mapping to Real-Time Maude

This makes models amenable to formal analysis using the Real-Time toolkit!
Reasoning about temporal specifications

Humans are mortal

*Plato* is human

$\Rightarrow$ *Plato* is mortal

The President of the US is elected every 4 years

*Bush* is the President of the US

$\Rightarrow$ *Bush* is elected every 4 years

[Sowa, 89]
Real-Time Maude used to provide semantics to eMotions
Model Simulation and Analysis with Maudeling

Simulation/Execution of specifications

Reachability Analysis
- Deadlock
- Invariants
- Others

LTL Model checking
- Liveness properties

MtATL 2009. Nantes, July 8, 2009
Semantic Mappings

- **DSL Structure**
  - EMF
  - e-Motions
  - ATL'
  - ECLIPSE

- **Source Domain**

- **DSL Behavior**
  - mOdCL
  - Maudeling
  - MAUDE

- **Target Domains**
  - PETRI NETS
  - ...

**ATL Transformations**

**Extraction/injection**

**Rule**
\[
\text{rule } \text{AnEventOccurs} \{ \\
\quad \text{from} \\
\quad s: \text{StateMachine}, \\
\quad t: \text{Transition}, \\
\quad x: \text{EventOcurrence} \\
\quad \text{to} \\
\quad s': \text{StateMachine}(...)
\]
In addition to time...

- QoS Properties
- Resource consumption
- SLAs
- ...

How to add them to our behavioral specifications?
How to connect them to existing analysis tools?
Other “kinds” of Semantics
Meaning?

Design of a real Retail application
How to understand the system?
How to reason about it?
How to detect design problems or anomalies?
Visual metaphors

An analogy which underlies a graphical representation of an abstract entity or concept with the goal of transferring properties from the domain of the graphical representation to that of the abstract entity or concept [Diehl, 2007]

The representation of a new system by means of visual attributes corresponding to a different system, familiar to the user, that behaves in a similar way [Dürsteler, 2008]

They are also semantic mappings!!!
VIASCO project

A project to visualize component-based systems with the goal of detecting “anomalies”

Defined as a chain of ATL model transformations

Connects several tools
- Parser
- Clustering
- Metrication
- Visualization

Uses Wires*
A project to visualize component-based systems with the goal of detecting “anomalies”

Defined as a chain of ATL model transformations

Connects several tools
  - Parser
  - Clustering
  - Metrication
  - Visualization

Uses Wires*
Epilogue
Summary

Assigning **Meanings** to Models is required
- For building tools
- To explicitly and completely describe behavior
- To disambiguate semantic variation points
- To understand and reason about the systems

( ATL ) Model Transformations can be used to define semantics of models (realizing the “**semantic bridges**”)

\[ [[M]]_D := [[T(M)]]_{D'} \]

We have shown a proof-of-concept, that uses Maude
- To specify models, metamodels and their behavior
- To make use of Maude’s analysis tools
- To provide formal semantics to other visual approaches (based on Eclipse, Graph grammars,...)
Some challenges

**Composition mechanisms and type systems**

**Testing/validating model transformations**
- 10. Fraternali et al. Mutation Analysis for Model Transformations in ATL.

**Enhanced traceability mechanisms**
- 8. Yie et al. Advanced Traceability for ATL.

**Connections to other notations and tools**
- 1. Chenouard et al. Using ATL to define advanced and flexible constraint model transformations.

**Querying models and View synthesis**
- 2. Chiprianov et al. An Approach for Constructing a Domain Definition Metamodel with ATL.
- 12. Vénisse. UMLQualityAnalysis: UML models measurements with ATL.
Some more challenges

- Addition of more Non-Functional Properties to DSMLs

- Specification and development of more Semantic Bridges
  - Specially to semantic domains with powerful analysis tool support (Petri Nets, Alloy, ...)

- Performance
  - Rule-based specs become unmanageable very soon
  - Performance is a big issue when dealing with LARGE models

- Global consistency checking of specifications
  - When dealing with multiple models of the same system...
Thanks!

Acknowledgements:

and, especially, to Jean, Frédéric, and the rest of the AtlanMod team!
When we consider MDE with this unified vision, many well-known situations may be integrated into a more general consideration. To take one additional example, the seminal work of R. Floyd ("Assigning meanings to programs", [12]) that founded the research field of programming axiomatics may be viewed as "decorating" a flowchart model with an axioms model. This may lead us first to consider decorations as models, then to understand the nature of the binding between the flowchart model and the axioms model, this binding being itself some kind of correspondence model. Finally, these considerations may lead us to generalize R. Floyd’s approach and to add on the research agenda a new item on "Assigning meaning to models". Model weaving and model transformation will be essential to the study of this subject.