Knowledge Reuse

Modeling and metamodeling in Model Driven Development

What is a model: syntax and semantics

Warsaw, May 14-15th 2009

Gonzalo Génova

ggenova@inf.uc3m.es

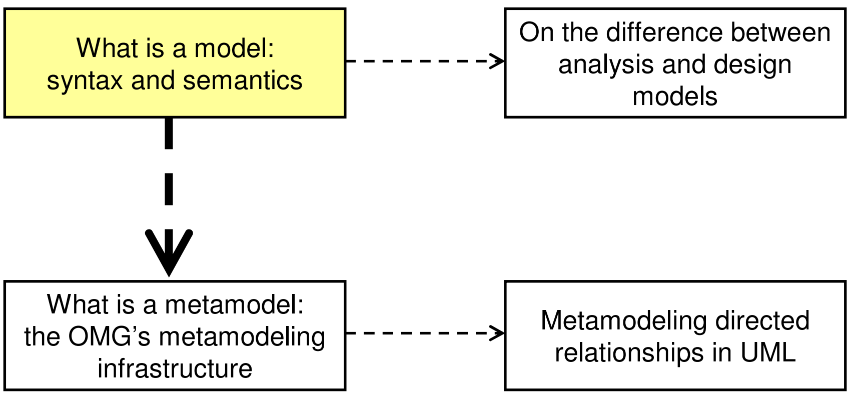
http://www.kr.inf.uc3m.es/ggenova/

Knowledge Reuse Group

Universidad Carlos III de Madrid

Knowledge Reuse

Structure of the seminar



Sources

- Ed Seidewitz.
  - ***What Models Mean.***
    - IEEE Software 20(5): 26 - 32, Sep-Oct 2003.
- David Harel, Bernhard Rumpe.
  - ***Meaningful Modeling: What's the Semantics of "Semantics"?***
    - IEEE Computer 37(10): 64-72 (2004).
- My own ideas and elaboration.

Table of contents

1. Introduction: what is a model
2. Semantics: the meaning of a model
3. Syntax: the structure of a model
4. The full definition of a modeling language
5. Conclusion: taking diagrams seriously

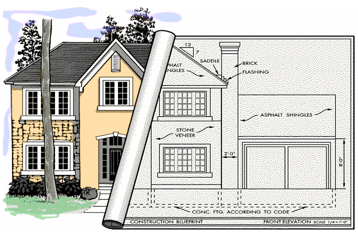
Knowledge Reuse

Introduction: what is a model

Knowledge Reuse

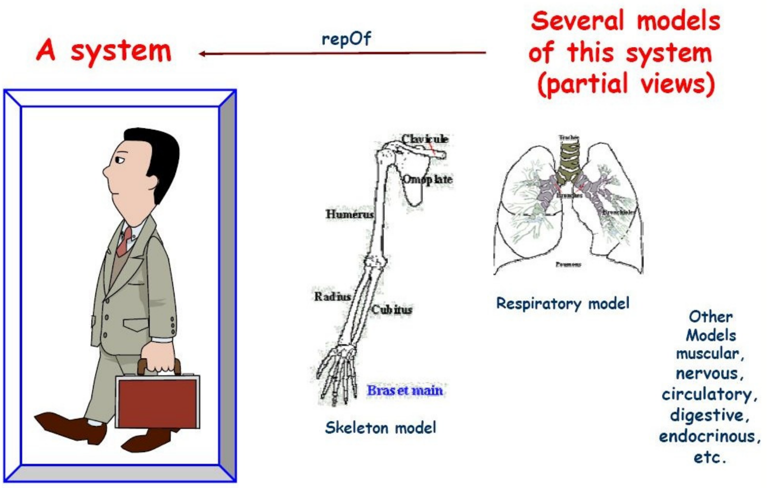
What is a model: a simplified representation

- A model is ***a simplified representation*** of certain reality.
  - The “system under study” (SuS): a software system, or whatever.
  - Simplification (or abstraction): retaining only the relevant aspects.
- A model is expressed in ***a modeling language***.
- A model must be:
  - abstract, understandable, precise, predictive, cheap...



represents

What is a model: a partial view



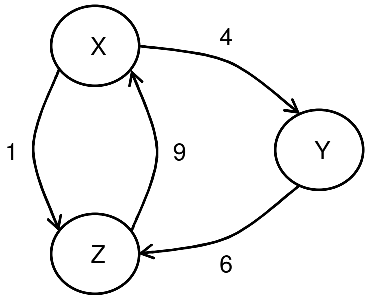
Jean Bézivin, Model Engineering for Software Modernization, 2004.

The level of accuracy of a model is necessarily limited.

Semantics: the meaning of a model

Knowledge Reuse

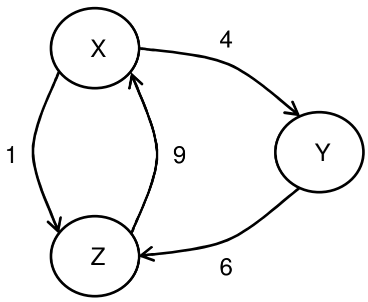
The meaning of a model



- What does this model mean? Many possible answers:
  - ***State transitions*** after a specified time interval in seconds.
  - ***Migration flows*** between pairs of countries in millions of people.
  - ***Debts*** in euros between pairs of people.
  - ...

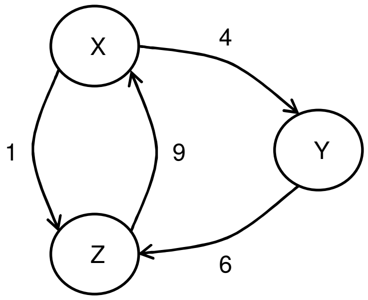
Knowledge Reuse

The meaning of a model: interpretation



- We need a ***mapping*** between each symbol or model element and the entity it represents:
  - Circles: states (or countries, or people, or...).
  - Arrows: transitions (or flows, or debts, or...).
- This mapping gives the ***interpretation*** of the model, its ***meaning***.

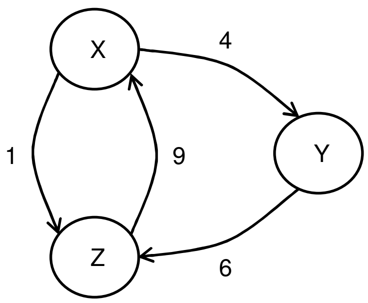
A model is a compound expression



X, Y and Z are people.
X owes 4 € to Y.
X owes 1 € to Z.
Y owes 6 € to Z.
Z owes 9 € to X.

- The ***model as a whole*** represents a certain reality, and ***each element*** represents one small part of that reality.
- A model is ***a set of statements*** about some system under study (SuS).
  - Each statement can be true or false.
- The model is true ***iif*** every statement is true.
  - In the usual convention (there can be other rules for composing statements).

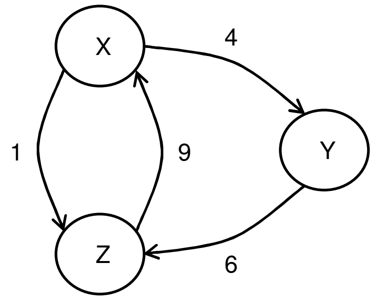
Truth and falsity of a model



X, Y and Z are people.
X owes 4 € to Y.
X owes 1 € to Z.
Y owes 6 € to Z.
Z owes 9 € to X.

- The interpretation mapping determines the ***truth of the model***.
- ***Description models:***
  - A statement is true if it ***corresponds*** to actual observations in the reality.
  - Is it true that “X owes 4€ to Y”?
- ***Specification models:***
  - The model is true ***by principle***, it is reality that must conform to the model.
  - The system is valid if it conforms to the model, otherwise it is invalid.

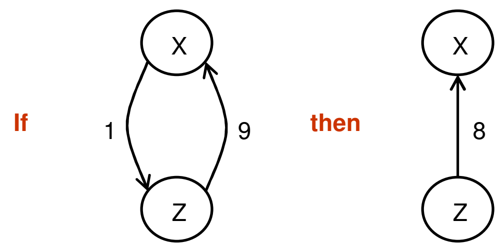
Meaning is something more than interpretation



What can we learn from this model?

- When we ask for the meaning of a model, we are asking in fact for two different things:
  - What does the model represent? Its relation to ***the things being modeled***.
  - What can we learn from the model? Its relation to ***other models*** derivable from it (***implicit truth***).

Deriving models from models

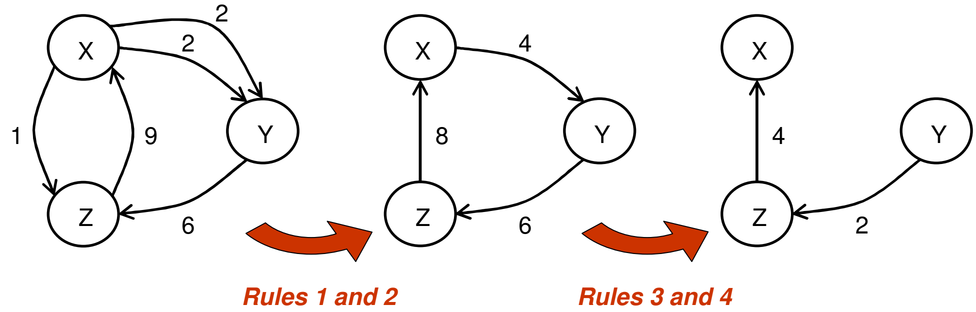


- A ***theory*** is a way to deduce new statements about an SuS from the statements already in some model of the SuS.
- Derivation is made according to a certain theory, i.e. a set of ***inference rules***.
- The derived expression can be considered a ***new model***, or an ***extension*** of the existing model.
- The theory can be itself ***right or wrong***, whether deductions correspond to actual observations.

A theory of debts

- ***Rule 1 (addition):***
  - Two arrows with the same source and target can be replaced by a single arrow labeled with the sum.
- ***Rule 2 (difference):***
  - Two arrows with opposite source and target can be replaced by a single arrow labeled with the difference.
- ***Rule 3 (cycle):***
  - The arrows that form a cycle can be all incremented or decremented by the same amount.
  - Rule 2 is a particular case of Rule 3, redundant in a minimal set of rules.
- ***Rule 4 (null arrow):***
  - An arrow labeled with 0 can be suppressed.

Applying the theory of debts



- The ***theory must conform*** to the modeled reality, too.
  - These rules make no sense for state transitions.
  - Only partially useful for migration flows (essential information is lost).
  - They are not valid if we want to represent the history of debts, instead of summary debts.

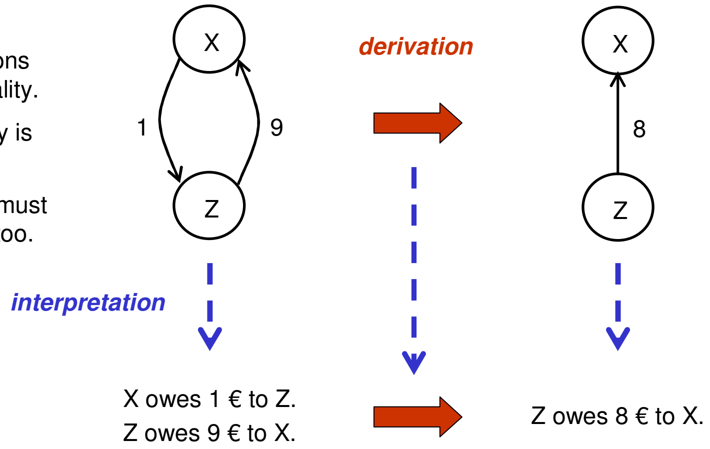
Knowledge Reuse

Interpretation and derivation are orthogonal

Inference rules can be used to make predictions about the modeled reality.

(If the derivation theory is right, of course.)

The derivation theory must correspond to reality, too.



Knowledge Reuse

Two aspects of the meaning of a model

- Summing up, two aspects of the meaning of a model:
  - Its relation to the things being modeled (***semantic mapping***).
  - Its relation to other models derivable from it (***semantic derivation***).
- These to aspects (jointly referred to as **semantics**) are an essential part in the definition of a modeling language.
  - Semantics comes from the greek ***semeion*** (sign, meaning).
  - Semiotics is the ***science of signs***, the science of meaning.
- ***Language semantics defines:***
  - The kinds of things we can represent in the models in that language.
  - How we can make deductions between models.
- The semantics of a modeling language determines the meaning of the models in that language.
- The other essential part in the definition is **syntax**.

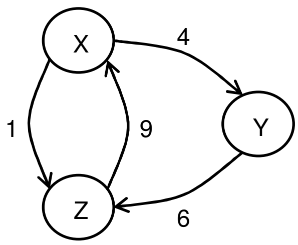
Syntax: the structure of a model

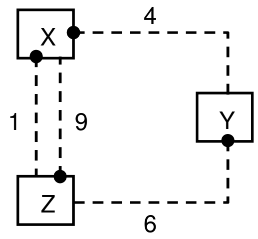
Model and diagram

- Often used interchangeably, but in fact they are not the same.
- A ***model*** is the representation of a certain reality.
  - A model is a ***logical set of elements*** that represent something.
  - A model is expressed in a certain modeling language.
  - The model itself can be represented in various forms (graphical, textual).
- A ***graphical modeling language*** is an artificial language that serves to express a model with symbols.
- A ***diagram*** is the visual representation of a set of interconnected elements belonging to a model.
  - A diagram is a ***partial view*** of a model.
  - A diagram is a linguistic expression made up of interconnected graphical symbols (***a graphical statement***).
  - A diagram represents the model using a graphical notation.

Knowledge Reuse

Concrete syntax



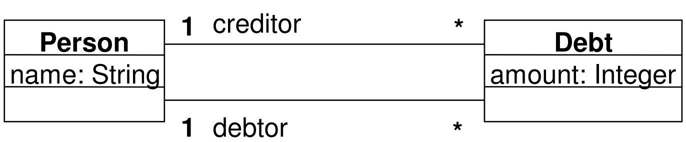


X, Y and Z are people.
X owes 4 € to Y.
X owes 1 € to Z.
Y owes 6 € to Z.
Z owes 9 € to X.

- The notation can vary significantly, while the underlying model remains the same: ***meaning is independent of notation***.
  - But the definition of notation cannot ignore social constraints.
- There is something more substantial to syntax than the graphical disposition of symbols: the ***logical combination of elements***.
- How can we specify the allowed combinations?

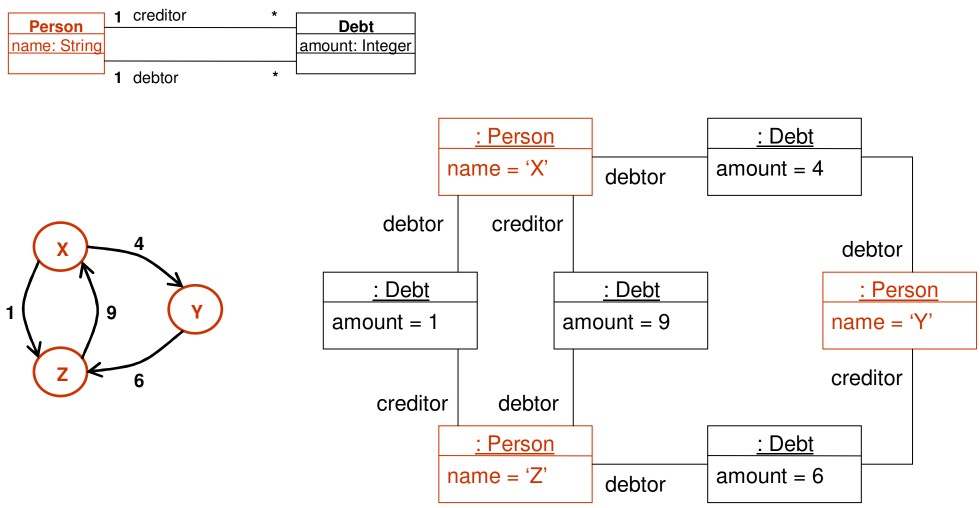
Knowledge Reuse

Abstract syntax



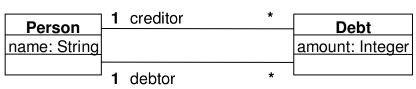
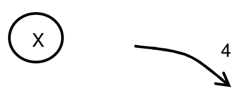
- Imagine we are developing ***a tool to draw debt-diagrams***.
- What would be the ***data structure*** of a diagram?
  - The data structure specifies all possible combinations of elements.
- We can express the data structure as a graphical model.
  - This is a ***metamodel***: a model of the abstract syntax of diagrams/models.
  - A metamodel is ***the "class" of all possible models*** in a given language.

A model is an “instance” of its abstract syntax



Definition of language syntax

- A modeling language must define both aspects of syntax:
  - ***Abstract:*** the rules that specify well-formed expressions of symbols.
  - ***Concrete:*** the set of graphical symbols used to render the diagrams.
- Syntax can be specified in various manners:

	Abstract syntax	Concrete syntax
Textually	A **Person** has a <u>name</u> of type String. A **Debt** has a <u>creditor</u> and a <u>debtor</u> of type Person, and an <u>amount</u> of type Integer.	**Person**: circle with name inside. **Debt**: arrow from creditor to debtor labelled with amount.
Graphically		

The full definition of a modeling language

The full definition of a modeling language

Syntax	**Concrete syntax**	Set of graphical symbols used to render the diagrams.
	Abstract syntax	Rules that specify well-formed expressions of symbols.
Semantics	**Interpretation (semantic mapping)**	Relation to the things being modeled.
	Theory (semantic derivation)	Relation to other derivable models by means of inference rules.

Dependencies between language elements

- Concrete syntax depends on abstract syntax.
  - It must define ***a symbol for each concept*** defined in the abstract syntax, and the symbol must ***adequately express*** the properties of the concept.
  - Typically, entities are redered as ***nodes***, relations as ***arcs*** between nodes.
- Syntax and semantics are closely related, but remain different.
  - Syntax specifies ***allowable expressions***, semantics assigns ***unambiguous meaning*** to them.
  - ***What a language looks like*** does not equate with ***what it means***.
  - Abstract syntax defines a set of ***modeling concepts*** that must correspond with the concepts in the semantic domain (it mimicries the domain).
  - Abstract syntax ***does not properly specify the semantic mapping***, safe for the adequate choice of names for modeling concepts.
  - Abstract syntax says nothing about the derivation rules between models; it specifies well-formed expressions, ***but not how to transform them***.

Syntax is not semantics

- A ***BNF definition*** of the syntax for simple numeric expressions:
  - <Exp> ::= <Number> | <Exp> + <Exp> | succ(<Exp>)
  - Allowed expressions:
    - 5+2, succ(3), succ(4+1), 6+succ(3), etc.
  - Forbidden expressions:
    - succ(+), 5+succ, 4++, 3#1, twist(4), etc.
- What is the meaning of allowed expressions?
  - Meaning: ***semantic mapping*** from expressions to the semantic domain.
- There are many possible meanings for this language.
  - 5+2 = 7 (addition)
  - 5+2 = 52 (why not concatenation?)
  - 5+2 = 10 (why not multiplication? who says '+' means 'addition'?)
- Each symbol must be given an ***unambiguous meaning***.
- A language is not understandable from its syntax alone.

Semantics and behavior

- Semantics is dealing with behavior?
  - Many languages deal with behavior, especially ***reactive behavior***.
  - Semantics must prescribe the system's behavior for each allowed expression (this is the ***behavioral interpretation*** of the language).
  - However, ***structure description languages*** (such as class diagrams) don't talk about behavior, but they still need semantics.
- Semantics is being executable?
  - Not all languages specify behavior.
- Semantics is the behaviour of a system?
  - The way a system behaves is quite different from the semantics of the language used to describe that system.
- Semantics is message passing?
- The ***core of language semantics*** is the theory of inference rules between language expressions.
  - Semantics cannot be reduced to ***dynamic semantics***, expressing behavior.

Conclusion: taking diagrams seriously

Taking diagrams seriously

- If we really want to put diagrams and models at the core of the development process, then we need ***rigorously defined modeling languages***, in both aspects of syntax and semantics.
- The UML specification has ***severe limitations***:
  - It is focused more on syntax than in semantics.
    - Even concrete syntax leaves too much freedom to tool developers.
  - Different level of rigor and accuracy: metamodeling / informal language.
  - Explanations of semantics are spread over the documents, with many inconsistencies.
  - The different kinds of diagrams are only loosely connected.
  - It is not at all easy to reason with diagrams, to derive ones from others.
- UML (and other graphical modeling languages) are a significant step forward, but there is still ***a lot of work to do***.

Questions?