The Role of Foundational Ontologies in Deep Modeling

Colin Atkinson

MODELSWARD 2014
Keynote Lecture
January 8th, 2014

Software Engineering Group

UNIVERSITÄT MANNHEIM
Modeling Everywhere ...

- Modeling is a now key activity in virtually all IT projects
  - today is performed in many different languages

- General-purpose structural languages
  - UML, OWL, ERA, RDF ...

- General-purpose behavioural languages
  - UML activity diagrams, Petri nets, ...

- Associated textual languages
  - OCL, ATL, QVT, Xtext

- Domain specific language
  - ......
Basic Tension in Language Design

- Since models are used to communicate properties of the real world between humans they should be able to -
  - accurately and unambiguously represent the real world (representational adequacy [GGO5])
  - at any desired level of detail
  - in the simplest and most concise way possible (lucidity, ontological clarity, construct redundancy [GGO5])

- There is a basic tension between expressive power and simplicity
  - rich, expressive languages often complex and verbose
  - simple, concise languages often lack expressive power

- Related to the question
  - General-purpose versus domain specific?
# UML 1.x Four Layer Model Architecture

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>meta-metamodel</td>
<td>The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.</td>
<td><em>MetaClass, MetaAttribute, MetaOperation</em></td>
</tr>
<tr>
<td>metamodel</td>
<td>An instance of a meta-metamodel. Defines the language for specifying a model.</td>
<td><em>Class, Attribute, Operation, Component</em></td>
</tr>
<tr>
<td>model</td>
<td>An instance of a metamodel. Defines a language to describe an information domain.</td>
<td><em>StockShare, askPrice, sellLimitOrder, StockQuoteServer</em></td>
</tr>
<tr>
<td>user objects</td>
<td>An instance of a model. Defines a specific information domain.</td>
<td><em>&lt;Acme_Software_Share_98789&gt;, 654.56, sell_limit_order, &lt;Stock_Quote_Svr_32123&gt;</em></td>
</tr>
</tbody>
</table>

(from UML Semantics Version 1.1)
In an n-level modeling architecture, $M_0, M_1 \ldots M_{n-1}$, every element of an $M_m$ level model must be an instance-of exactly one element of an $M_{m+1}$ level model, for all $m<n-1$ and any relationship other than the instance-of relationship between two elements $X$ and $Y$ implies that $\text{level}(X)=\text{level}(Y)$.

- every element has exactly one type, except those elements in the top level
- instance-of relationship used to define levels
- model elements allocated to levels according to their location in the type hierarchy
- M₀ level contains the real-world elements (the subject of the model)
- M₁ level contains model representation of types and instances
- M₂ level contains linguistic types of the M₁ elements
- M₃ level contains the linguistic types of the M₂ level concepts

(from UML Infrastructure Specification v2.4.1)
The Powertype Problem

- X is a power type of Y, if the instances of X are subclasses of Y
  - power types are classes whose instances are also subclasses
- therefore, according to the strict modeling tenet, powertypes must be at the M2 level
  - but they are represent domain concepts
  - the *isPowerTypeOf* relationship crosses level boundaries
- occur very frequently in practice

```
Video
  title: String
  title = "2001: A space Odyssey"
Book

Product
  :ProductType

ProductType
{disjoint, incomplete}

1
* 

Product

:Video

:ProductType
```
Deep Characterization

- usually a type only describes its instances
  - e.g., every dog has an age
- sometimes, however, a type needs to define constraints on the properties of the instances of its instances
  - e.g., every instance, of a breed instance, has an age
- such transitive influences cannot be directly expressed using traditional instanceOf relationships
- with UML must resort to -
  - constraints
  - powertypes
  - stereotypes
  - .....

Diagram:
- Breed
- Poodle
- Fido
- Dog
two distinct forms of classification organized in different, dimensions

- linguistic and ontological
- Strict (meta)-modeling in each dimension

- So called “linguistic/ontological metamodeling paradox” [EHA13]
Deep (Multi-Level) Modeling

- Orthogonal classification architecture
- Unified class/object model element
  - Clabject
- Level-agnostic mechanisms for representing “typeness” of clabjects
  - Potency (deep instantiation)
Different Views of the Infrastructure

Linguistic (Meta)model

Tool Developer's View

Modeler's View

90°
Essence of a Deep Linguistic Metamodel

- abstract syntax for a Level-Agnostic Modeling language (LML)
Classes and Objects

- In LML classes and objects are unified into the notion of “clabjects”
  - clabjects can have ontological attributes that play the role of UML attributes / slots
- ontological attributes must always have a name
  - can also, optionally, have a type and/or a value
Associations / links are also represented by a unified concept in LML – Connections

- conection are also clabjects
  - can have attributes and participate in generalizations
  - can be visualized in an exploded or “dotted“ form
Generalizations are another first class citizen of LML
- can also be view in an exploded or visually “insignificant“ form
- The name of the generalization usually identifies the discriminant
- Usual constraints such as disjoint and complete also supported
Stereotypes and stereotyping supported by metamodeling

- Stereotypes and stereotyping supported by metamodeling

```
«Metaclass»
Class

«Stereotype»
Clock
  OSVersion : String
  startOperation : Operation
  POSIXCompliant : Boolean

Stereotyped Element

«Clock»
AlarmClock

«Clock»
  OSVersion = "1.1"
  startOperation = start
  POSIXCompliant = True
```

UML

LML
LML also provides natural support for powertypes

- Power types are metaclasses whose instances are also subclasses.
  - X is a power type of Y, if the instances of X are subclasses of Y

UML diagram:

- Account
  - * account classifier
  - account

AccountType

CheckingAccount
SavingAccount

LML diagram:

- Account
  - AccountType

CheckingAccount: AccountType
SavingAccount: AccountType
Deep Instantiation

- a simple mechanism supporting deep characterization
- allows feature properties to be exactly specified using the notion of potency
- provides a simple but precise definition of instanceOf relationships (principle of application) over multiple classification levels
  - potency takes non-negative integer values
  - an instance of a type always has a potency that is one lower than the type
- although instantiation also lowers the level of a clabject by one,
  - the potency and level of a clabject do not have to be the same
The Power Type Example Revisited

OO

AccountType

TaxCode¹=TX00
TypeIntroductionDate²

O1

Account

balance³

AccountType

CheckingAccount¹:AccountType

TypeIntroductionDate⁰=11/6/2003
TaxCode²=TX34⁰

SavingsAccount¹:AccountType

TypeIntroductionDate⁰=1/1/2000
TaxCode²=TX54⁰

O2

CA776:CheckingAccount

TaxCode²=TX34
balance=2000

SA342:SavingsAccount

balance=500
TaxCode²=TX54

LML
Domain Specific v General Purpose?

GPLs
- E.g. UML, Java, XML etc.
- Lingua Franca
- less concise and efficient
- facilitate inter-domain communication

DSLs
- E.g. BPMN, Circuit Diagrams etc
- optimized for special tasks
- more concise and efficient
- “Tower of Babel“ problem
DSL State-of-the-Art

- DSLs defined in rigid, disjoint phases and environments
- Result of one phase has to be “deployed” in a major compilation step to enable the second phase
- In each phase/environment only one classification level is “soft”
- Only one concrete syntax is available in a given phase.
Many features of UML were not included in the earlier comparison of UML and LML:

- packages, components, composition...

these can be added as DSL features.
**Symbiotic Languages**

- Symbiosis – a relationship in which two things compensate for each other’s weaknesses and reinforce each other’s strengths

- Symbiotic languages
  - coexists and can be mixed in arbitrary ways
  - Melanie supports a symbiosis between general purpose and domain specific visualizations

- Key features –
  - choice between purely general purpose or domain specific representations of the same model
  - symbols can be arbitrarily mixed
    - any symbol can be dynamically toggled from domain-specific to general purpose and back
  - can be applied at multiple classification level (i.e. models) simultaneously
Symbiotic Language Example (1/3)
Symbiotic Language Example (2/3)
Symbiotic Language Example (3/3)
**Foundation Ontologies**

- A foundation ontology (also known as a top-level ontology or upper ontology) describes very general concepts that are common across all knowledge domains.

- Meet the following basic needs [EHA13]
  - Promote reuse in a higher level of abstraction aimed at maximising the reuse of domain models.
  - Produce domain specifications that are truthful to reality.
  - Theoretical foundations for conceptual modelling languages.

- Examples
  - Bunge-Ward-Weber (BWW), Basic Formal Ontology (BFO), General Formal Ontology (GFO), Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), Suggested Upper Merged Ontology (SUMO),
  - Unified Foundation Ontology (UFO) [GG05]


Ullmann’s Triangle

- basis for understanding how symbols represent concepts in a language
- “the relation between language and reality is always intermediated by a certain conceptualization” (Baldinger)

Guizzardi’s Square

- characterizes relationship between conceptualizations, models, languages and specializations [GG05]
**Concrete Incarnation**

- In practice, boils down to a detailed taxonomy of modelling constructs capturing and philosophically well-founded concepts

- Based on a traditional (two-level) class / object dichotomy
  - Aristotle’s four category ontology

- Most widely used and practical incarnation is OntoUML
  - a UML profile capturing the UFO taxonomy
  - supported by a tool developed by NEMO (Ontology & Conceptual Modeling Research Group) in Vitoria, Brazil

- Arguments for validity and utility are philosophical
  - the necessity or correctness of modelling features can not be proven per se
  - often the information they convey can be modelling in another way (albeit perhaps less elegant)
  - phases can be modelled by attributes of enumeration types
Aristotle’s four category ontology

Second chapter of Aristotle’s Categories [GG05]
Aristotle’s four category ontology

“are in time”
• persist over time
• same individual can exist at two times
• e.g. a house, a pile of bricks

“happen in time”
• extend in time accumulating temporal parts
• same individual cannot exist at two time intervals
• e.g. a race, a party

Second chapter of Aristotle’s Categories [GG05]
Aristotle’s four category ontology

“can exist independently”
- objects or amounts of matter
- e.g. house, pile of sand

“inhere in other individuals”
- attribute, properties, tropes
- cannot exist independent
- e.g. age, reliability

Second chapter of Aristotle’s Categories [GG05]
Taxonomy of Substantial Universals
"have a sense of identify and type"
• support classification (principle of application) and identification (principle of identity)
• basically correspond to classes in UML
• e.g. person, video

“dispersive universals”
• represent things with a particular quality
• e.g. rational entity, strong entity
**Taxonomy of Substantial Universals**

A type whose instances cannot cease to be instances of the type without ceasing to exist
- if I is an instance of T in one possible world, then I must be an instance of T in every possible world
- e.g. person

A type whose instances can cease to be instances of the type whilst continuing to exist
- if I is an instance of T in a given world, there is another possible world in which I is not an instance of T
- e.g. student,
Example (OntoUML) [GG05]
At first sight FOs and deep modelling are fundamentally incompatible
- UFO based on the fundamental universal / instance dichotomy
- deep modelling exists to precisely to overcome this limitation

UFO does not explain some concepts that are important in IT
- abstract classes
- interfaces
- power types
- stereotypes?

but UFO can be regarded as a DSL in deep modelling
- provides exactly the same expressive power as OntoUML
**UFO as DSL**

[Diagram showing various entities and relationships within a software engineering context, including domain elements, features, attributes, methods, and connections.]
**Big Question**

- In an optimal deep modelling environment, what features in (e.g.) UFO be moved to the linguistic (meta)model?
  - would the notion of rigid or antirigid clabjects be useful?
  - would it be useful to introduce the notion of role clabjects?

- For example EHA13 argue that Dog is a substantial class while Breed is a moment type
  - but what if Dog were not in the model?

- The top ontological model and the linguistic model need to be aligned and optimized to support deep, FO based modeling
ontological model elements are symbols
real world entities, including concepts, are things
higher level model elements are concepts
a concept can be an instance of another concept (counter to [EHA13])
ontological model elements are symbols
real world entities, including concepts, are things
higher level model elements are concepts
a concept can be an instance of another concept (counter to [EHA13])
Deep Interpretation of Guizzardi’s Square

- Model elements at one level are a language for the model specification at the level below.
- A model language represents a conceptualization.
- A model, an instance of a conceptualization, is a model specification.
Deep Interpretation of Guizzardi’s Square

- model elements at one level are a language for the model specification at the level below
- a model language represents a conceptualization
- a model, an instance of a conceptualization, is a model specification
Deep Interpretation of Guizzardi’s Square

- Model elements at one level are a language for the model specification at the level below.
- A model language represents a conceptualization.
- A model, an instance of a conceptualization, is a model specification.
Conclusion

- FOs and deep modelling are at one level fundamentally incompatible, but at another level they are synergetic
  - an FO can be supported as a DSL in deep modelling
  - FO semantic models map well to a potency-based OCA (Ullmans’s triangle, Guizzardi’s square)
- “Linguistic” and “ontological” are highly misleading labels for the different forms of classification
  - infrastructural versus conceptual or domain
- The modeling community would benefit from a fundamental unification of FO and deep modelling
  - FOs needs to move beyond the two-level world view
  - Infrastructural model needs to be enhanced with FO concepts
- Deep modelling environment from Uni. Mannheim
  - www.melanee.org
Deep, SUM-based Environments