Meta-modeling: concepts, tools and applications


Saïd Assar ¹,²

¹ Associate Professor
Telecom Ecole de Management, Department of Information Systems
Evry, France

² Invited Researcher
Centre de Recherche en Informatique, University of Paris 1 La Sorbonne
France
1. Introduction & background (cont'd)

- **INSTITUT MINES - TELECOM**: 4 leading schools in engineering and management
1. Introduction & background (cont'd)

- Since 1999, long-term research collaboration with CRI at Paris 1 Sorbonne
Learning objectives

✓ To understand **what is a meta-model**, its historical background, when it is needed and **how it can be defined and exploited**.

✓ To **witness and participate in building concrete meta-models** and exploiting them using existing meta-modeling technical infrastructures.

✓ To be **aware of problems and challenges in meta-modeling** and the actual state-of-the-art regarding these issues.
1. Introductory examples

2. Abstraction levels and the instantiation problem

3. Computerized tools for meta-modeling

4. Illustration
   a) A Domain Specific Language (DSL) for IoT v3.0 Framework
   b) A DSL to model software development processes
   c) A DSL to specify web applications

5. Research challenges

6. Conclusion
1. Introductory examples – what is a model?

“A model (M) for a system (S) and an observer (O) is any kind of representation which can help O in answering questions and reasoning about S.”

- It is an abstraction, a reduction of reality to its most relevant aspects
- It is the result (i.e. the product) of some specific process
- It is built using some kind of notation (formal and/or graphical)
- Descriptive vs. prescriptive models
- Models are artifacts, i.e. they can also be modeled
Let’s consider this very simple process model with **two concepts** (event, function) and **one link** among concepts (event – function).
1. Introductory examples – what is a meta-model?

- A **meta-model** would be a textual, graphical, and/or formal **representation of the concepts and how they are linked**

Questions

- How to represent a meta-model?
- How can a MM be leveraged in IS engineering?

1. Introductory examples – generic models (1)

Suppose a trans-national car tracking system
✓ interoperability issue
✓ system evolution issue

Are all these plates special cases of a car plate model?
1. Introductory examples – generic models

- Or is there a generic vehicle ID model from which these models - and may be others - can be derived?

Interactive question: In terms of system evolution, these two modeling solutions are similar or very different?
1. Introductory examples – generic models

In terms of system evolution, these two modeling solutions are similar or very different?

What about the country code, how to model it?
1. Introductory examples – generic models (2)

For a software development company with different methods
✓ evolution issue
✓ enactment issue

Method M1 for simple projects
- Planning
- Design
- Development
- Implementation

Method M2 for large projects

Agile method M3 for risky projects

General method

IsA ?

General method structure

InstanceOf ?

For a software development company with different methods
✓ evolution issue
✓ enactment issue
1. Introductory examples – IoT framework

In the field of Internet of Things (IoT), a large set of heterogeneous objects are expected to interact and be part of a huge information network …

➤ Is it possible to have a general model for all interacting objects?
1. Introductory examples – IoT framework

A reference model of IoT is proposed by an FP7 project …

➢ What this model is helpful for?
➢ Is it possible to provide computerized support for using this model?

Source
Language syntax definition (a toy language)

<postal-address> ::= <name-part> <street-address> <zip-part>
<name-part> ::= <personal-part> <last-name> <opt-suffix-part> <EOL>
                |<personal-part> <name-part>
<personal-part> ::= <first-name> | <initial> "."
<street-address> ::= <house-num> <street-name> <opt-apt-num> <EOL>
<zip-part> ::= <town-name> "," <state-code> <ZIP-code> <EOL>
<opt-suffix-part> ::= "Sr." | "Jr." | <roman-numeral> | ""
<last-name> ::= [A-Z|a-z]
<first-name> ::= [A-Z|a-z]
1. Introductory examples

2. Abstraction levels and the instantiation problem

3. Computerized tools for meta-modeling

4. Illustration
   a) A Domain Specific Language (DSL) for IoT v3.0 Framework
   b) A DSL to model software development processes
   c) A DSL to specify web applications

5. Research challenges

6. Conclusion
2. Abstraction levels – instanciation vs. generalisation

CLASS

• Attribute1
• Attribute2
• ...

CLASS : a structure, a pattern

CLASS A

• Attribute a1
• Attribute a2
• ...

CLASS B

• Attribute b1
• Attribute b2
• ...

Object

• "Value1"
• "Value2"
• ...

ToObject

InstanceOf

Object: identifier, values

IsA
2. Abstraction levels – the meta-modeling language issue

✓ A meta-model is described using a meta-modeling language

➢ **How to describe the meta-modeling language ... a circular issue.**
2. Abstraction levels – instantiation levels

• Meta-modeling framework by OMG: Meta Object Facility (MOF)

=> Strict instanciation
2. Abstraction levels – instantiation levels

- Instantiation levels in the **Meta Object Facility (MOF)** by OMG

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta-metamodel</td>
<td>The infrastructure for a metamodel architecture. Defines the language for specifying metamodels.</td>
</tr>
<tr>
<td>Metamodel</td>
<td>An instance of a meta-metamodel. Defines the language for specifying a model.</td>
</tr>
<tr>
<td>Model</td>
<td>An instance of a metamodel. Defines a language to describe an information domain.</td>
</tr>
<tr>
<td>User objects (user data)</td>
<td>An instance of a model. Defines a specific information domain.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta-metamodel</td>
<td><em>MetaClass, MetaAttribute, MetaOperation</em></td>
</tr>
<tr>
<td>Metamodel</td>
<td><em>Class, Attribute, Operation, Component</em></td>
</tr>
<tr>
<td>Model</td>
<td><em>StockShare, askPrice, sellLimitOrder, StockQuoteServer</em></td>
</tr>
<tr>
<td>User objects (user data)</td>
<td><em>(Acme_SW_Share_98789), 654.56, sell_limit_order, (Stock Quote Svr_32123)</em></td>
</tr>
</tbody>
</table>

*Source: OMG*
2. Abstraction levels – the “Powertype” approach

Where does meta-modeling come from?
• Artificial intelligence (i.e. frame concept, Smalltalk, ObjVLisp)
• Databases (i.e. meta-database)
• Object oriented modeling (i.e. Classes are objects too => MetaClasses)

How to simultaneously create a generalization link AND an instantiation link
How to define attributes to Classes
How to cross the strict instantiation levels

➢ One of the earliest work on these issues is by P. Cointe in 1987

Cointe, Pierre: Metaclasses are first class: The ObjVLisp Model. SIGPLAN Notices. 22(12), 156–162 (1987).
2. Abstraction levels – the “Powertype” approach

- How to **simultaneously** create a generalization link AND an instantiation link
- How to define attributes to Classes
- How to cross the strict instantiation levels

**Illustration** (Source: Gonzalez-Perez & Henderson-Sellers, *Metamodelling for software engineering*, p.38)

- How to model species-specific characteristics, e.g. Breed?
2. Abstraction levels – the “Powertype” approach

Illustration (Source: Gonzalez-Perez & Henderson-Sellers, *Metamodelling for software engineering*, p.39)

- Using specialization link to model species-specific characteristics

![Diagram showing entities and attributes with specialization link]

- Pet
  - PetName
  - DateOfBirth
  - Owner

- CatSpecie
  - Breed

- "Valentina"
  - 8-sep-2008
  - "César"
2. Abstraction levels – the “Powertype” approach

Illustration (Source: Gonzalez-Perez & Henderson-Sellers, *Metamodelling for software engineering*, p.39)

- Specific representation

```
<table>
<thead>
<tr>
<th>AnimalSpecies</th>
<th>Pet</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpecieName</td>
<td>PetName</td>
</tr>
<tr>
<td>AvgLifeSpan</td>
<td>DateOfBirth</td>
</tr>
<tr>
<td>Vaccinations</td>
<td>Owner</td>
</tr>
<tr>
<td>Diet</td>
<td></td>
</tr>
</tbody>
</table>

"Cat specie" 8
"Triple", "Leukaemia"
"Fish"

"Valentina" 8-sep-2008
"César"
"Siamese"

Class + Object

Illustration (Source: Gonzalez-Perez & Henderson-Sellers, *Metamodelling for software engineering*, p.39)
2. Abstraction levels – the “Powertype” approach

Illustration (Source: Gonzalez-Perez & Henderson-Sellers, *Metamodelling for software engineering*, p.40)

- PowerType and Clabject

<table>
<thead>
<tr>
<th>AnimalSpecies</th>
<th>Pet</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpecieName</td>
<td>PetName</td>
</tr>
<tr>
<td>AvgLifeSpan</td>
<td>DateOfBirth</td>
</tr>
<tr>
<td>Vaccinations</td>
<td>Owner</td>
</tr>
<tr>
<td>Diet</td>
<td></td>
</tr>
</tbody>
</table>

- Cat specie
  - 8
  - "Triple", "Leukaemia"
  - "Fish"

- CatSpecie
  - Breed

- "Valentina"
  - 8-sep-2008
  - "César"
  - "Siamese"
### 2. Abstraction levels – the “Powertype” approach

#### Illustration (Source: González-Perez & Henderson-Sellers, *Metamodelling for software engineering*, p.41)

- **PowerType and Clabject for method engineering**

<table>
<thead>
<tr>
<th>Task Kind</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskName</td>
<td>Start</td>
</tr>
<tr>
<td>Purpose</td>
<td>End</td>
</tr>
<tr>
<td>MinCapLevel</td>
<td>Duration</td>
</tr>
</tbody>
</table>

#### Task: "Code writing"

- **To write code …**
- **MinCapLevel**: 1
- **Start**: 12-sep-07
- **End**: 18-sep-07
- **Duration**: 7d
- **Program Language**: C#
2. Abstraction levels – deep instantiation

• Proposal:
  – Add an indicator (*potency*) of the abstraction level at which the attribute is to be instantiated

```
+ TaskCategory¹
+ TaskDuration⁰

CodeWritingTask

+ TaskCategory: Coding

T1

+ TaskDuration: 10d
```
1. Introductory examples
2. Abstraction levels and the instantiation problem
3. Computerized tools for meta-modeling
4. Illustration
   a) A Domain Specific Language (DSL) for IoT v3.0 Framework
   b) A DSL to model software development processes
   c) A DSL to specify web applications
5. Research challenges
6. Conclusion
3. Computerized tools for meta-modeling

Focus on 3 meta-modeling tools:

(1) eclipse

(2) MetaCase

(3) ConceptBase.cc - A Database System for Metamodelling and Method Engineering
3. Computerized tools for meta-modeling

Other meta-modeling tools:

(4) metaDepth: A framework for deep meta-modelling

(http://astreo.ii.uam.es/~jlara/metaDepth/)

(5) ADOxx Meta Modelling Platform

(http://www.adoxx.org/live/home)

(6) JastAdd (http://jastadd.org) and

JastEMF (https://code.google.com/a/eclipselabs.org/p/jastemf/)
for language engineering
3. Computerized tools for meta-modeling

ECORE meta-modeling framework (MOF based)

Source: http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html
3. Computerized tools for meta-modeling

ECORE meta-modeling framework (MOF based)

Source http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html
3. Computerized tools for meta-modeling

**MetaEdit (1)**

- A meta-CASE tool for automatic **customization** of CASE tools
- Result of research project at Jyväskyla (K. Lyytinen, M. Rossi, S. Kelly et al. - 1990')
- Actually commercialized by MetaCASE
- Targeted towards **Domain Specific Modeling**
3. Computerized tools for meta-modeling

MetaEdit (2)

- Meta-modeling language: GOPRR
  - Graph
  - Object
  - Property
  - Relationship
  - Role
3. Computerized tools for meta-modeling

MetaEdit (3)

- Meta-editing functionalities

A store acts as a repository for data. Thus, a store allows you to define parts of the system where you need data stored between processes. A store is symbolized by a rounded rectangle where the name of the store is shown inside the symbol. Store has two properties: store name and documentation.
3. Computerized tools for meta-modeling

MetaEdit (4)
- Customized editor
3. Computerized tools for meta-modeling

MetaEdit (5)

- MERL: A scripting language for generating code (i.e. flow of characters)

```sql
REPORT 'create related entity FKs'
/* USED FOR: ~Entity part or .Relationship */
/* INPUT: $targetTable : the for which the FK's are being created */
/* OUTPUT: foreignDeclarations (the foreign key sql declarations) */

$foreignDeclarations=''
dowhile .Entity {
    dowhile .ERAttribute; where :Constraints =~ 'PRIMARY' {'
        :Attribute name; $var; ' ' :
        :Type of data; ' ' /* Assumption: FK's are never null */
        'NOT NULL /* Foreign Key */'

        variable 'foreignDeclarations' append
        'ALTER TABLE ' $targetTable; newline
        'ADD FOREIGN KEY (' :Attribute name; $var; '] ' ' REFERENCES ' :Entity name; $var; ' ] ' '

        newline
    close
    ', ' newline
} }, ' newline
endreport
```
3. Computerized tools for meta-modeling

MetaEdit (6)

- Example in MERL for generating SQL code from E/R schema

```sql
CREATE TABLE Buy_order (
    Delivery_date DATE,
    Order_date DATE,
    OrderID REAL NOT NULL PRIMARY KEY,
    WorkerID REAL NOT NULL /* Foreign Key */);

CREATE TABLE Department (
    Address CHAR,
    DepartmentID INTEGER NOT NULL PRIMARY KEY,
    Name CHAR NOT NULL UNIQUE);

CREATE TABLE Product (
    Colour INTEGER,
    Part INTEGER NOT NULL PRIMARY KEY,
    OrderID REAL NOT NULL /* Foreign Key */);
```
3. Computerized tools for meta-modeling

Conceptbase (1)

- A deductive database supporting *object-centered* TELOS modeling language
- Result from research project DAIDA (J. Mylopoulos, M. Jarke et al. - 1990')
- Continued as an open source project (M. A. Jeusfeld)
- Targeted towards **Model Engineering**

**TELOS modeling language**

- Any element (class, object, attribute, link, constraint ...) is internally represented as a **predicate** in a deductive DB
- Attributes are first order objects i.e. they can have properties
- Attributes correspond to links between two elements
- Any object can be instantiated
  - Multiple instantiation ('Paul' in 'Employee' and 'Paul' in 'Student')
  - Multiple levels of instantiation
  - Multiple Generalisation/Specialisation (i.e. heritage)
  - Generalisation/Specialisation can co-exist with instantiation
3. Computerized tools for meta-modeling

Conceptbase (2)

• Model Illustration

Person in Class with attribute
  ID: Integer;
  name:String
end

Client isA Person with attribute
  reduction:Real
end

Employee isA Person with attribute
  function: String
end

Student isA Person with attribute
  age:Integer
end
3. Computerized tools for meta-modeling

Conceptbase (3)

- Multiple instantiation illustration

John in Student, Employee, Client with
  ID
    id_john: 44555
  name
    name_j: "John Legrand"
  reduction
    r1: 30.0
  function
    f1: "Sale manager"
  age
    age1: 24
end

Client with constraint
  reduction_OK:
    $\forall c/Client \ x/Real (c \ reduction\ x) \Rightarrow (x \geq 5.0)$
end

QueryClass AllPeople isA Person with
  constraint
    Allpeople_c: $\exists s/String (this\ nom\ s)$
3. Computerized tools for meta-modeling

Conceptbase (3)

- Implementing Class attributes and the “PowerType” concept

```
AnimalSpecies in Class, MetaClass with attribute
   SpecieName: String;
   AvgLifeSpan: Integer;
   Vaccinations: String;
   Diet: String
end

Pet in Class, MetaClass with attribute
   PetName: String;
   PetDateOfBirth: String;
   PetOwner: String
end

CatSpecie in AnimalSpecies, Class with attribute
   Breed: String
SpecieName
   sn: "Cat specie"
AvgLifeSpan
   als: 8
Vaccinations
   v1: "Triple"; v2: "Leukaemia"
Diet
   d1: "Fish" end

MyCat in CatSpecie, Pet with
   Breed
      MyCatBreed:"Siamese"
PetName
   p1: "Valentina"
PetDateOfBirth
   d1: "08-Sep-2008"
PetOwner
   pow: "César“ end
```
3. Computerized tools for meta-modeling

Conceptbase (3)

• Implementing Class attributes and the “PowerType” concept
1. Introductory examples
2. Abstraction levels and the instantiation problem
3. Computerized tools for meta-modeling
4. Illustration
   a) A Domain Specific Language (DSL) for IoT v3.0 Framework
   b) A DSL to model software development processes
   c) A DSL to specify web applications
5. Research challenges
6. Conclusion
Exercise 1: Internet of Things framework

The IoT ARM is exemplified with a use-case scene: A load carrier is equipped with sensors and can communicate with other devices in terms of wireless radio technology. With this hardware, every load carrier continuously measures its environmental parameters and sends all measurements via the embedded event service to the mobile phone of the truck driver.
Exercise 1: Internet of Things framework

“For example, Ted is a truck driver transporting highly sensitive orchids to a retail store. After loading the orchids on his truck, he attaches an array of sensors to the load carriers in order to measure the temperature. While having lunch, Ted forgot that by turning off the engine, air condition for the transported goods highly sensitive orchids shuts off, too. The temperature inside the truck starts rising and when it reaches a predefined critical level inside, one of its sensors notices this and its node sends an emergency signal to Ted’s IoT-Phone. On the IoT-Phone’s display, Ted can now see that the orchids are in danger so he rushes back to the vehicle and turns the air condition on.

The IoT-Phone also keeps track of any alert messages it receives from the load carriers and saves this message history for future inspection in a way that cannot be altered. When the truck reaches the retail store for delivery, the sensor history is transferred to the store’s enterprise system and the sensors authenticate themselves as being un-tampered.”

How to model this system using the IoT framework?
Exercise 2: Method engineering

The design and development team in a company has defined a set of engineering methods according to the nature of projects. Among others, and in the case of small size projects (number of end-users <= 10, number of data tables <= 15, number of business process <=5), they have defined the following simple method:

“The Requirements Engineering team (1 or 2 engineers) uses administrative documents and conduct interviews with end-users to elicit general requirements, i.e a natural language description of the required system (including NF requirements), and process definitions represented as BPMN diagrams. The design team (2 to 4 engineers) uses the process models together with the general description to sketch a system architecture, i.e. an UML class diagram, use case diagrams and sequence diagrams. The development team (3 to 5 developers) build the system according to the BPMN models and the system architecture, and deliver an operational implementation composed of Java code for web services, a relational database and a workflow script. The test team (2 to 3 persons) verify the quality of the system using a set of test cases they have designed and in accordance with the NF requirements”.

How such method can be modeled using a subset of SPEM and the MetaEdit+ workbench?

Forms-based software for data collection has found application in various areas, including scientific surveys, online course-ware and guidance material to support the auditing process. As an overall term for this kind of software applications we use the term "questionnaire". In this exercise, the goal is to create a simple DSL, called QL, for describing questionnaires. Such questionnaires are composed of sequential forms, each form is characterized by conditional entry fields and (spreadsheet-like) dependency-directed computation. The following example presents a possible textual representation of a simple questionnaire with only one form:

```plaintext
questionnaire HouseOwning
form Box1HouseOwning
{
    hasSoldHouse: "Did you sell a house in 2010?" Boolean
    hasBoughtHouse: "Did you by a house in 2010?" Boolean
    hasMaintLoan: "Did you enter a loan for maintenance/reconstruction?" Boolean
    if (hasSoldHouse)
    {
        sellingPrice: "Price the house was sold for:" money
        privateDebt: "Private debts for the sold house:" money
        valueResidue: "Value residue:" money(sellingPrice - privateDebt)
    }
}
```
4. Meta-modeling illustration examples


This simple form should generate into a GUI which allows the following user interaction:

- Did you sell a house in 2010? [ ] Yes
- Did you buy a house in 2010? [ ] Yes
- Did you enter a loan for maintenance/reconstruction? [ ] Yes
- Price the house was sold for: [ ]
- Private debt for the sold house: [ ]
- Did you enter a loan for maintenance/reconstruction? [ ] Yes
- Did you buy a house in 2010? [ ] Yes
- Price the house was sold for: [ ]
- Private debt for the sold house: [ ]
- Value residue: [ ]

How such language can be designed and implemented?
Agenda

1. Introductory examples
2. Abstraction levels and the instantiation problem
3. Computerized tools for meta-modeling
4. Illustration
   a) A Domain Specific Language (DSL) for IoT v3.0 Framework
   b) A DSL to model software development processes
   c) A DSL to specify web applications
5. Research challenges
6. Conclusion
6. Research challenges in meta-modeling

Behavior meta-modeling

- *Behavior* perspectives are generally missing in software engineering meta-models

  ⇒ Important knowledge about modeling languages is lacking

  ⇒ Essential for **tools designers** and **method engineers**

  ⇒ For a process modeling language, "behavior" perspective inquire on its **executable/operational semantics**
6. Research challenges – behavior modeling

Research goal

- How to express the operational semantics for a modeling language?
- How to design and build enactment engines for a given modeling language?

➤ *Maintainability and portability are central issues*

![Diagram of behavioral semantics specification](image)

**Fig. 1.** Specification of the behavioral semantics

[Source: T. Mayerhofer et al., “xMOF: Executable DSMLs Based on fUML”, SLE’2013]
6. Research challenges – behavior modeling

Behavior meta-modeling

- Context: MAP process modeling notation

Example
6. Research challenges – behavior modeling

Event-based behavior meta-modeling

C1: target intention = Stop
C2: target intention /= Stop
F1: pour toute sectionInstance retournée par l'algorithme de calcul de candidates
6. Research challenges in meta-modeling

Behavior meta-modeling

- Other approaches

⇒ KERMETA

For an example, see [http://en.wikipedia.org/wiki/Kermeta](http://en.wikipedia.org/wiki/Kermeta)
6. Research challenges in meta-modeling

Behavior meta-modeling

$\Rightarrow$ xMOF (Mayerhofer et al., 2013)

**Fig. 6. Specification of the Petri net DSML**
1. Introductory examples
2. Abstraction levels and the instantiation problem
3. Computerized tools for meta-modeling
4. Illustration
   a) A Domain Specific Language (DSL) for IoT v3.0 Framework
   b) A DSL to model software development processes
   c) A DSL to specify web applications
5. Research challenges
6. Conclusion
6. Conclusion

• No well established and recognized standard for meta-modeling
• Tool support is complicated
• … OMG's **MOF** and IBM **Eclipse** emerge as "market" leaders for UML & Java
• **MetaEdit**: leader for DSM engineering tools
• **Conceptbase**: most powerful and theoretically sound modeling language
• **Semantics** for process meta-models still to be defined
• Beyond DSM, few empirical studies on meta-modeling applications
References


Links

- Kermeta: [http://www.kermeta.org](http://www.kermeta.org)
References

References (cont’d)

THANK YOU FOR YOUR ATTENTION!

Questions? Comments? Insights?

Homepage  http://www-public.it-sudparis.eu/~assar/
Linkedin  http://fr.linkedin.com/pub/said-assar/4/68a/66a
Slideshare  http://fr.slideshare.net/SaidAssar/
Twitter  https://twitter.com/Said_Assar/