

Component

PxM-10en “Modus: Praxeme methodology”

## Guide of the semantic aspect

**Objective** Praxeme identifies nine aspects of the Enterprise System. The second of those is the semantic aspect. The semantic model describes the core business knowledge, independently of the way the business is run. This guide discusses the importance of the semantic model and describes the appropriate modeling technique.

**Content**

- Definition and objective of semantic modeling
- Products: repository and model
- Positioning in the production chain
- Procedures and methods of semantic modeling

**Author** Dominique VAUQUIER

**Translator** Joanne TOWARD; for the appendix: Jean POMMIER, Anthony JERVIS

**Reviewers** Praxeme Institute (see p. iii)

**Version** 1.99.2, July 10, 2011

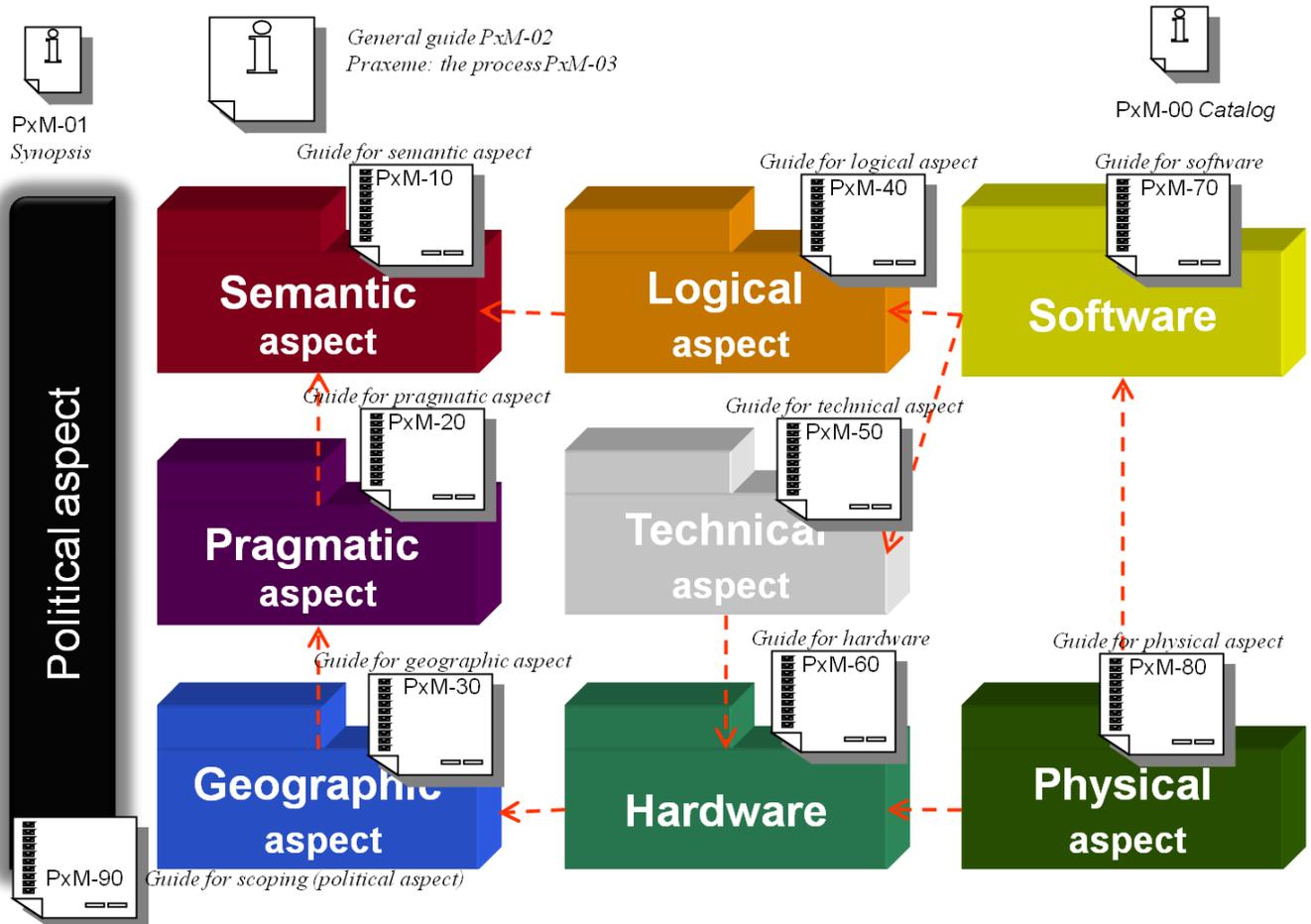
## Configuration elements

### The position of this module in the methodology

#### Situation in the documentation

The methodology Praxeme is based on and structured by the “aspects” and “topology of Enterprise Systems”. The methodological guide book – PxM-02– explains this approach.

Figure PxM-10en\_1. Architecture of the methodological corpus



#### Owner

The Praxeme methodology results from the initiative for an open method. The main participants are the enterprises SAGEM and SMABTP, and the French army<sup>1</sup>. They combined their forces to found a public ‘open’ method. The Praxeme Institute maintains and develops this joint asset.

Any suggestions or change requests are welcome (please address them to the author).

#### Availability

This document is available on the Praxeme website and can be used if the conditions defined on the next page are respected. The sources (documents and figures) are available on demand.

<sup>1</sup> See the website, [www.praxeme.org](http://www.praxeme.org), for the full list of contributors.

## Configuration elements (cont.)

### Revision History

Version	Date	Author	Comment
	March 2004	DVAU	First Writing (Dromos: method Sagem for enterprise architecture of the IT drone system)
	November 2005	DVAU	Extended version (Amos: the IT SMABTP method; SOA approach)
<b>1.0</b>	27/04/06	DVAU	Generalized for submission to the “circle of experts”
<b>1.99.0</b>	July 2009	Joanne TOWARD	Translation
<b>1.99.1 &amp; 2</b>	January, July 2011	DVAU	Review of the translation
<b>1.99.2</b>			Current version of the document

## License

### Conditions for using and distributing this material

#### Rights and responsibilities

This document is protected by a « [Creative Commons](#) » license, as described below. The term “creation” is applied to the document itself. The original author is:

- The association *Praxeme Institute*, for the entire methodology Praxeme.
- Dominique VAUQUIER, for the document;

We ask you to name one or the other, when you use a direct quotation or when you refer to the general principles of the methodology Praxeme.

This page is also available in the following languages :

[български](#) [Català](#) [Dansk](#) [Deutsch](#) [English](#) [English \(CA\)](#) [English \(GB\)](#) [Castellano Castellano \(AR\)](#) [Español \(CL\)](#) [Castellano \(MX\)](#) [Euskara](#) [Suomeksi](#) [français](#) [français \(CA\)](#) [Galego](#) [עברית](#) [hrvatski](#) [Magyar](#) [Italiano](#) [日本語](#) [한국어](#) [Melayu](#) [Nederlands](#) [polski](#) [Português](#) [svenska](#) [slovenski jezik](#) [简体中文](#) [華語 \(台灣\)](#)



**CC creative commons**  
COMMONS DEED  
Attribution-ShareAlike 2.0 France

**You are free:**

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

**Under the following conditions:**



**BY: Attribution.** You must attribute the work in the manner specified by the author or licensor.



**Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

**Your fair use and other rights are in no way affected by the above.**

This is a human-readable summary of the [Legal Code \(the full license\)](#).

## Content

Configuration elements.....	ii
The position of this module in the methodology	ii
Revision History	iii
Conditions for using and distributing this material	iv
Introduction.....	1
Semantic modeling: business fundamentals	1
Semantic modeling: products and procedures	2
Definition and objective of semantic modeling.....	3
Introduction to the semantic aspect	3
A focus on the essentials ensures a stable core	4
Task under the control of the pertinence criterion	5
Semantic modeling terms	6
Correspondence to natural language	8
The meta-model for the semantic aspect	9
Objects of a semantic nature	10
Products: repository and model .....	11
Semantic repository	11
The semantic model	12
Semantic model requirements: quality factors	13
Semantic model requirements: quality criteria	14
Semantic model requirements: documentation	15
Thesaurus-index	16
Positioning in the production chain .....	17
A very stable model, contributing to each project	17
Dynamics between system and project	18
Semantic modeling: analysis and design tool	18
Semantic model verification	19
Ulterior use of the semantic model: neighboring aspects	20
Derivation channels	21
Ulterior use of the semantic model: other aspects	22
Communication by the models	23
Procedures and methods of semantic modeling.....	24
Representation techniques: UML contribution	24
Representation techniques: class diagram	24
Representation techniques: object diagram	25
Representation techniques: state diagram	26
Representation techniques: other techniques	27
One procedure	28
Semantic modeling activity requirements	29
Encapsulation principle of constraints	30
Abstraction principle	31
Sharing principle	33
Recommendations and best practices	34
Principle of decomposition in object domains	35
Three axes of modeling	36
<i>Structural</i> modeling	37
<i>Functional</i> modeling	38
<i>Contractual</i> modeling	39
Other recommendations	41
Appendix: illustrating semantic modeling.....	42

---

## List of figures

Figure PxM-10en_1. Architecture of the methodological corpus .....	ii
Figure PxM-10en_2. The consequences of abstraction .....	4
Figure PxM-10en_3. Table of correspondence between UML and natural language. ....	8
Figure PxM-10en_4. Semantic modeling terminology (extract from the Praxeme meta-model). ....	9
Figure PxM-10en_5. Example of a class diagram.....	10
Figure PxM-10en_6. Semantic model uses .....	13
Figure PxM-10en_7. Ishikawa diagram of quality criteria (cause and effect diagram) .....	14
Figure PxM-10en_8. The aspects in relation to the semantics .....	20
Figure PxM-10en_9. Derivation channels from the semantic model .....	21
Figure PxM-10en_10. Spontaneous expression.....	25
Figure PxM-10en_11. Correct model .....	25
Figure PxM-10en_12. Object diagram .....	25
Figure PxM-10en_13. Example of a semantic model structured in object domains. ....	35
Figure PxM-10en_14. Three axes of modeling .....	36
Figure PxM-10en_15. Example of a state machine polluted by organizational elements.....	39

---

## Epigraph

*“Herein lies the secret of the whole method: in all things, look carefully for that which is the most absolute.”*

René Descartes, *Rules for the direction of the mind* (rule VI)

*“It is not because reality is ambiguous that our concepts should be confused.”*

Max Weber

*“A permanent confrontation between theory and experience is a necessary condition for creative expression.”*

Pierre Joliot, *La recherche passionnément*

---

## Introduction

---

### Semantic modeling: business fundamentals

---

#### Praxeme

Praxeme is an enterprise methodology and, as such, focuses on how business knowledge is represented and managed. This knowledge, free from organizational and technical concerns, makes up the semantic aspect of the enterprise or organization.<sup>2</sup>

---

#### Position in the overall methodology

The fundamental position of Praxeme consists in isolating the relatively independent *aspects* of the system analyzed, and in providing as complete a description of them as possible.

The semantic aspect is the first to appear in the Enterprise System topology. Its position is essential, with a view to restructuring the systems, improving performance and reducing the impact of change. Indeed, semantics capture that which is most stable in the universe analyzed. It is, therefore, necessary to isolate the semantics, in one way or another, in the solution under development.

Semantic modeling takes place upstream from organizational projects, process design and information technology (IT) developments. The logical model can then be derived from the semantic model, in accordance with the transformation rules set out in the PxM-40 and PxM-41 guides, covering services and data respectively.

There is additional scope for the semantic model, notably as a tool for training. It is the starting point for *knowledge management*. In addition, the semantic aspect is the ideal ground for radical thinking about the business. It can absorb part of the strategic directions, precisely those which redefine the business and its content.

---

#### Objective of the Guide

This document, the first of the guides focusing on an aspect, shows how to design a good semantic model, i.e., one that is independent of organizational and IT choices.

---

#### Domain and circumstances of application

The modeling process described here, applies:

- at a local business, domain or software (in a project context) level;
- at a global “System” level (activities linked to strategy formulation, enterprise organization or information system urbanization).

In the first case, we refer to the semantic model; in the second, to the business repository (e.g., Insurance Repository).

Two particular circumstances justify the semantic modeling effort: the innovation surrounding organizational processes and the overhaul of IT systems.

---

<sup>2</sup> For an overview of the methodology, see PxM-01 (Praxeme at a glance) or PxM-02 (General Guide).

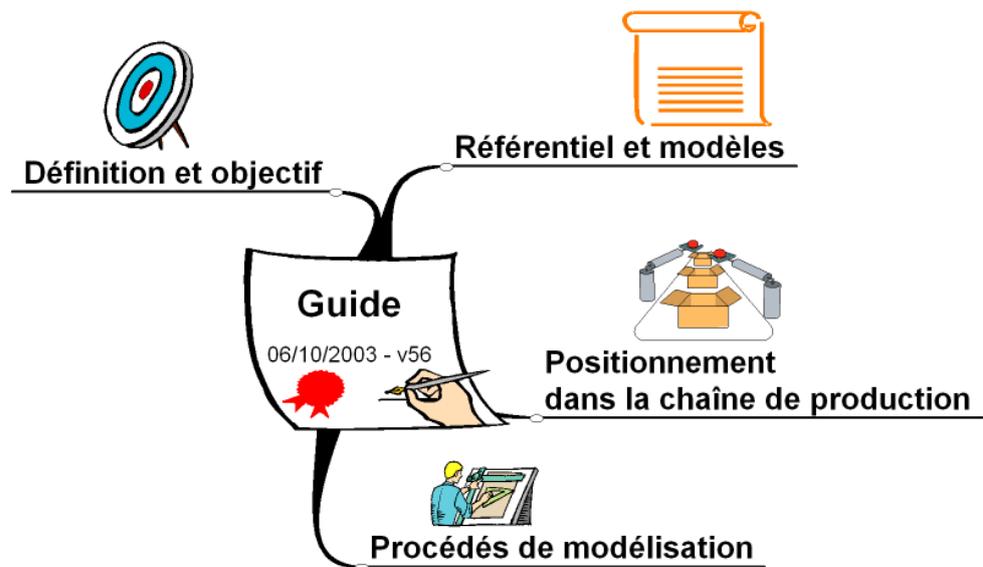
## Introduction (cont.)

### Semantic modeling: products and procedures

#### Guide Contents

After defining semantic modeling and raising awareness of its stakes, the guide looks at three dimensions:

- products;
- processes;
- procedures.



**Products** Semantic modeling products gather together the necessary information on the part of the reality targeted by the solution. There are two types of products, depending on their level:

- the semantic repository;
- the local models.

The guide specifies the types of information these products must contain.

**Processes** Without prejudging the reference process chosen for the activity<sup>3</sup>, the guide assesses the impact of the semantic approach, with regards to the development dynamics.

**Procedures** The last part of the guide proposes ways of doing things and best practices for carrying out semantic modeling. It also includes recommendations for using Unified Modeling Language (UML).

<sup>3</sup> Indeed, it is advisable to respect the orthogonality between the steps (or management method) and the approach (or representation technique). The first is left to the initiative of the project managers. The second is the subject the Praxeme initiative gives priority to.

---

## Definition and objective of semantic modeling

---

### Introduction to the semantic aspect

---

#### Definition

In its semantic aspect, the System representation targets:

- the notions, concepts and objects from the domain being studied,
- the information they carry,
- the behavior they are capable of,
- the associations connecting them in an efficient network of meaning,
- the rules constraining them.

This representation is free from organizational and technical circumstances. Its value-add lies in the abstraction which leads to simplicity. The semantic model captures the essence; that is what makes it simple and stable.

---

#### The stakes

The simplicity of this description frees the imagination, which then enables the designer to broaden his palette of choice regarding the organization, logistics and technologies.

The abstraction effort allows the essentials to be rediscovered, throws aside the diversity of practices and paves the way for simplification. Strengthened by the object-oriented approach that we recommend, this effort pushes for the genericity of the semantic model, to the point of its being universal. The stakes are high: a sufficiently stable and generic semantic model can be shared between several organizations. It normalizes the terminology and formalizes the business fundamentals that partners will be able identify with. From then on, the semantic model forms an essential base to support a network of enterprises. These stakes are highlighted in the context of partner relations, mergers and acquisitions, as well as within those groups where several separate companies or managements coexist.

Adding to the semantic repository also enables an organization to capitalize on the knowledge about the domain. In this way, it is a powerful tool in preserving the intellectual assets of an enterprise.

---

#### Principles

Several strong principles apply to the semantic aspect:

- the abstraction principle.
- the constraint encapsulation principle.
- the object domain decomposition principle.

A description of them can be found in the last part of this document.

## Definition and objective of semantic modeling (cont.)

### A focus on the essentials ensures a stable core

#### Attitude

The modeler whose task it is to capture the semantics, deals with reality, without *a priori*. Contrary to appearances, this attitude is not spontaneous. It requires a special and continually renewed effort, on the modeler's part, to disregard organizational and technical circumstances. The quality of the semantic model depends on the modeler's ability to move away from current practices and the existing solution.

In addition, far from getting bogged down by the apparent complexity of the domain, he must capture the essentials and isolate the fundamental core.

The modeler will have to defend the simplicity of his model faced with the general tendency to complicate matters. One response will be to demonstrate how this essential model restitutes the reality and how it can be instantiated to take into account the diversity of concrete situations.

#### Consequences

If we insist on the necessary, and sometimes troubling, simplicity of the semantic model, it is because the resulting consequences are of importance, from the point of view of information system (IS) urbanization or process simplification. They are summarized in the following schema.

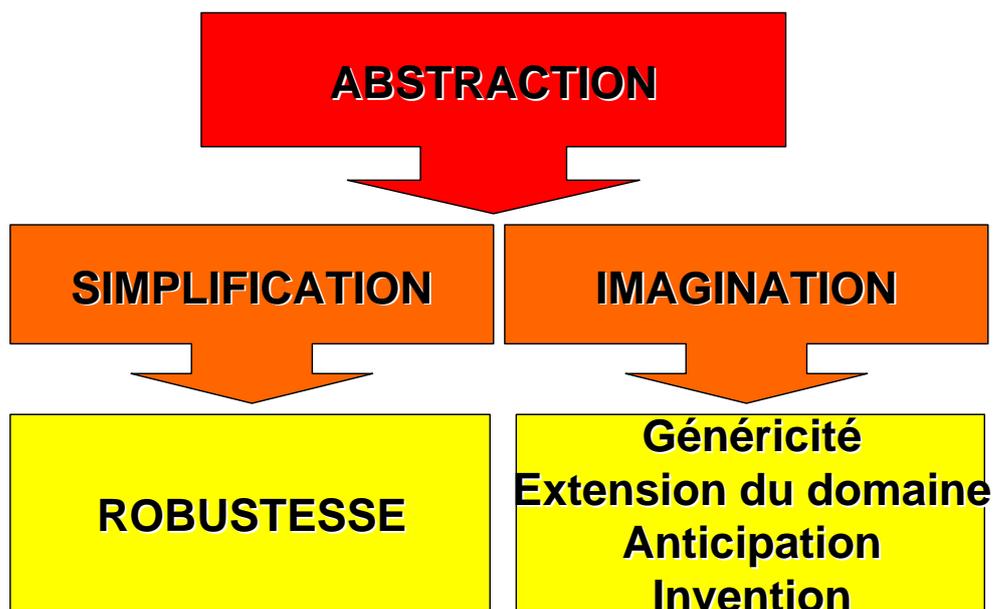


Figure PxM-10en\_2. The consequences of abstraction

Through the abstraction work, and while adhering to the reality targeted, the model simplifies itself. It enables a stable core to be isolated, which is a necessary condition to absorb any development requests: robustness follows on from simplicity. Furthermore, abstraction pushes for genericity, which allows for a wider scope of application. At the same time, the model becomes a tool for forward thinking, anticipating change or future requests. The adjacent aspect guides (pragmatic and logical) present the procedures and methods that enable radical innovation (on the procedures and IS architecture respectively).

## Definition and objective of semantic modeling (cont.)

### Task under the control of the pertinence criterion

#### Exploiting the semantic model

The semantic model will fuel the following tasks:

- It can contribute to an overhaul of process delivery and organization design, providing that we adopt a new approach to process delivery<sup>4</sup>.
- It provides the starting point for a large part of the logical components, mainly the logical services of the internal stratum<sup>5</sup>.
- It enables part of the strategic directions and IS policy to be set.
- It contains the semantic class test cases.
- It can be directly computerized, independently of the target architecture, to simulate the model or as a library of components to put in the logical architecture<sup>6</sup>.

#### Necessary skills

Semantic modeling does not require any IT knowledge. Experience has shown that having an IT background can even be a handicap: the modeler tends to load the semantic model with choices and expressions linked to IT jargon and preconceived ideas.

On the other hand, semantic modeling requires the combination of two sets of skills:

- the in-depth knowledge of the modeled domain (“universe of discourse”);
- the ability to use formal expression.

These two skills, which are completely separate from one another, can be respectively carried out by different people: the domain expert (functional or business expert) and the modeler (the semantician).

#### Semantic modeling limits

Reality is inexhaustible! There is a continual risk that modeling will become a never-ending process. It is important to keep it in check (**effort weighting**). The model does not claim to *totally* reconstitute reality. The criterion of pertinence is applied: the model is built with a particular development in mind (reforming process delivery, preparing architecture, building software, guiding communication, etc.).

In addition, for any given project, the objective may lead to focusing on only one part of the domain (**perimeter reduction**).

<sup>4</sup> This new approach to process delivery, set against the functional approach, is discussed in the Pragmatic Aspect Guide (reference PxM-20).

<sup>5</sup> Cf. PxM-40.

<sup>6</sup> Generally speaking, decisions relating to structure need to be made before IT solutions are implemented. There is usually no room for such decisions in the semantic model. Consequently, opting to computerize the semantic model directly, almost always results in its controlled downgrading. Current technologies do not allow existing semantic models to be automated, at least not on a large scale.

---

## Definition and objective of semantic modeling (cont.)

---

### Semantic modeling terms

---

#### Terms

Which terms does the modeler use to capture the semantic aspect? Which syntax units are used to describe the “problem universe”, the application domain and the reality?

The following categories answer these questions.

By 'terms', we mean the modeling syntax or representation categories. We can settle for a minimal syntax tool for semantic modeling, one that maps the natural grammar as closely as possible. We remove all external constraints that may risk distorting the approach to reality<sup>7</sup>.

---

#### Semantic class

The basic unit in semantic modeling (in the object approach) is the class. Let us call it the “semantic class”, to emphasize its intention: to capture the semantics linked to business objects. This precaution will ensure that there is no confusion with the software aspect class.

The semantic class enables the semantics attached to a real object, a set of similar objects or a concept, to be restituted.

Candidate classes are those classes that the modeler is considering writing into the model, to capture part of the meaning. They often correspond to a substantive used in speech.

#### *Conditions for retaining classes*

Not all candidate classes will be retained. Many will be eliminated because they do not carry any characteristic properties. A class retained by the semantic model must have one of the following conditions:

1. It is an information support.
2. Its instances behave in a specific manner or supply services.
3. The class constitutes an indispensable node in the model's structure.

---

#### Class properties

The class carries properties that are informative, active or cooperative (structural) in nature.

The data or information applies to each instance of a class or to all instances of a class.

#### *Informative properties*

They are translated in UML in the form of an attribute, if necessary, derived (and so created by a software operation).

#### *Active properties*

They presuppose an action or a transformation of the object. They are written in UML, in the form of an operation. If the object is transformed by the action requested, the operation will be managed by a state machine.

#### *Structural properties*

They are the classification and association relations (including composition).

---

<sup>7</sup> This will not be the case with the other aspects, especially from the logical level onwards.

## Definition and objective of semantic modeling (cont.)

### Semantic modeling terms (cont.)

#### Association

The association is an essential element in capturing the semantics. It enables the classes to be joined.

The link (between objects) is the instantiation of an association (between classes).

UML notation enables several constituent elements of the model to be written, all of great value, both for the quality of expression of the semantic model and its future use. These elements are:

- the name of the association (in theory, a binary association could have two names, according to the direction that it is read in);
- the cardinalities (on each “connector” of the association);
- the role names (ditto);
- an associative class, when it is necessary for the object to correspond to each link (reified association);
- one or more qualifiers, in the case of the qualified association.

**Recommendations** Semantic modeling fully exploits these possibilities of expression, in order to reconstitute the semantics of the domain as closely as possible, and capture its rich content. A semantic model will therefore seem richer than many of the so-called UML models that are encountered on development projects.

Semantic modeling takes only a rare interest in the direction of an association (direction of navigation). Indeed, to direct an association means restricting the navigation of the model, prohibiting the “passage” from one class to another. Such restrictions are rarely within the domain itself<sup>8</sup>.

#### Packages

When the domain analyzed is composed of numerous classes – several dozen – it becomes necessary to structure the model. UML provides a tool for this, the package, sometimes known as “category” on the design plane.

In Praxeme, the package used in semantic aspect models is called “object domain”. This name is set against the more classical “functional domain” and underlines the decomposition criterion choice (see “Principle of decomposition in object domains”, page 36).

#### Automaton or state machine

A class can be seen as an object factory. The object is perceived as a micro-machine, managing its internal state and supplying services, in certain operative conditions. This is one of the key points of the object approach, which has to guide modeling with a view to quality. To do this, we use state machines.

#### Event

The event or signal is an exchange unit within the system.

<sup>8</sup> They will intervene, on the other hand, in the following aspects.

## Definition and objective of semantic modeling (cont.)

### Correspondence to natural language

#### Principle

The expected quality of a semantic model lies in its exactness with the universe of discourse: i.e., the accuracy of its verbal description of the reality. It is shown by the semantic model's ability to capture the maximum number of units of meaning present in this universe.

#### Syntactic correspondence

UML has a slightly more formal syntax than natural language (though not completely formal). The correspondence between the syntactic categories of the two languages offers the modeler a practical guide, as well as proof of the accuracy of the model. The table below aims to do this.

Figure PxM-10en\_3. Table of correspondence between UML and natural language.

Category or function object (test)	Natural language category	Modeling element	Recommendation
<b>“Reference” for extension and intention</b>	Noun, noun phrase in subject or object position.	Class	Beware of substantive forms. They hide the action. Better to retain the verb, in this case.
<b>“Juncture”</b>	Noun, noun phrase in complement or preposition position.	Association role	The connection between two objects must be restituted by an association, rather than an attribute name.
<b>Information</b>	Noun characterizing variable values.	Attribute	Avoid class attributes (with the exception of utility classes).
<b>Qualifier</b>	Qualifying adjective (associated with a substantive used by a class).	State in a state machine	Core business objects often have a life cycle. A model without a state machine is a dubious one.
<b>Action</b>	Verbs of action	Operation	The question is “where should the operation be placed?”: on the object which acts (subject) or on the recipient (direct object).
<b>“Juncture”</b>	Verbs of state (constitute, to be composed of).	Association, often composition; sometimes state.	Composition or assembly does not prohibit the naming of the association.
<b>Qualifier</b>	Auxiliary “to be”, followed by a past participle, adjective.	State	
<b>Action and Qualifier</b>	Auxiliary “to be”, followed by a present participle.	State with an action.	Reserved word “do” in the state drawing.
<b>“Juncture” or qualifier</b>	Auxiliary “to have”	Attribute or assembly	According to the object.
<b>Classifier</b>	“is one”, “is a sort of”	Inheritance	Raises several problems: multi-criteria classification, mutation...

Category or function object (test)	Natural language category	Modeling element	Recommendation
Coordinator	Coordinating conjunctions	Synchronization in a process or state machine.	“and”, “or”, “so”. Do not reify processes.

## The meta-model for the semantic aspect

### Presentation

The class diagram below is the synoptic of the Praxeme meta-model for the semantic aspect.

Other diagrams, as well as numerous comments, are provided in the overall meta-model. It contains, in particular, the way in which this metamodel exploits the UML standard one. Indeed, purely on the semantic aspect, the Praxeme meta-model does not bring a great deal compared to UML. Rather, it selects and emphasizes certain categories, shaping them for semantic modeling. It is also a simpler, more compact meta-model, which clarifies the representation categories for use by the semantician.

### Comment

The central notion is the semantic class. This meta-class leverages the “Class” in the UML meta-model.

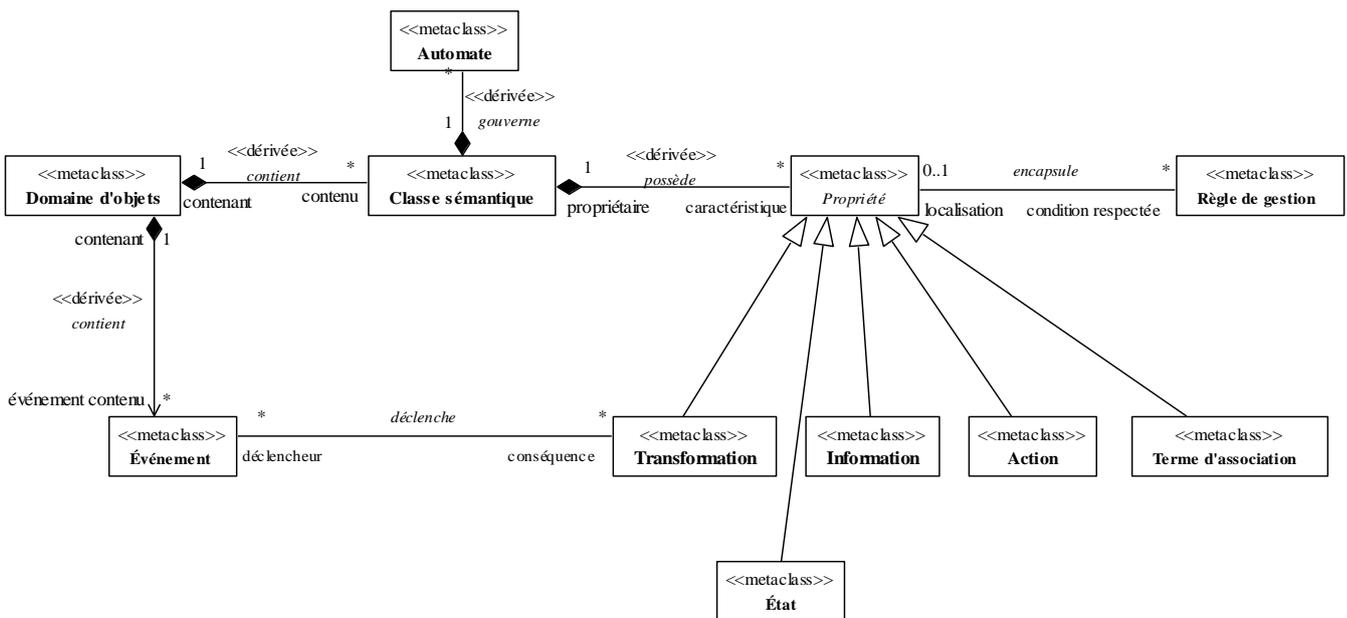
The properties are grouped in quite a different way to the UML meta-model. The cardinalities of the “encapsulate” association impose that a business rule can be located in only one area of the model. This requirement takes up one of the precepts of modeling.

The meta-model situates the notion of object domain and positions the state machine, which governs the class.

The derived associations encapsulate the subjacent connections between the meta-classes.

For more detail, please see the full metamodel.

Figure PxM-10en\_4. Semantic modeling terminology (extract from the Praxeme meta-model).



## Definition and objective of semantic modeling (cont.)

### Objects of a semantic nature

#### Examples and counterexamples

Manipulated objects: claims; clients; contacts; dossiers; database...

Actions on the objects: declare; save; evaluate; calculate; search...

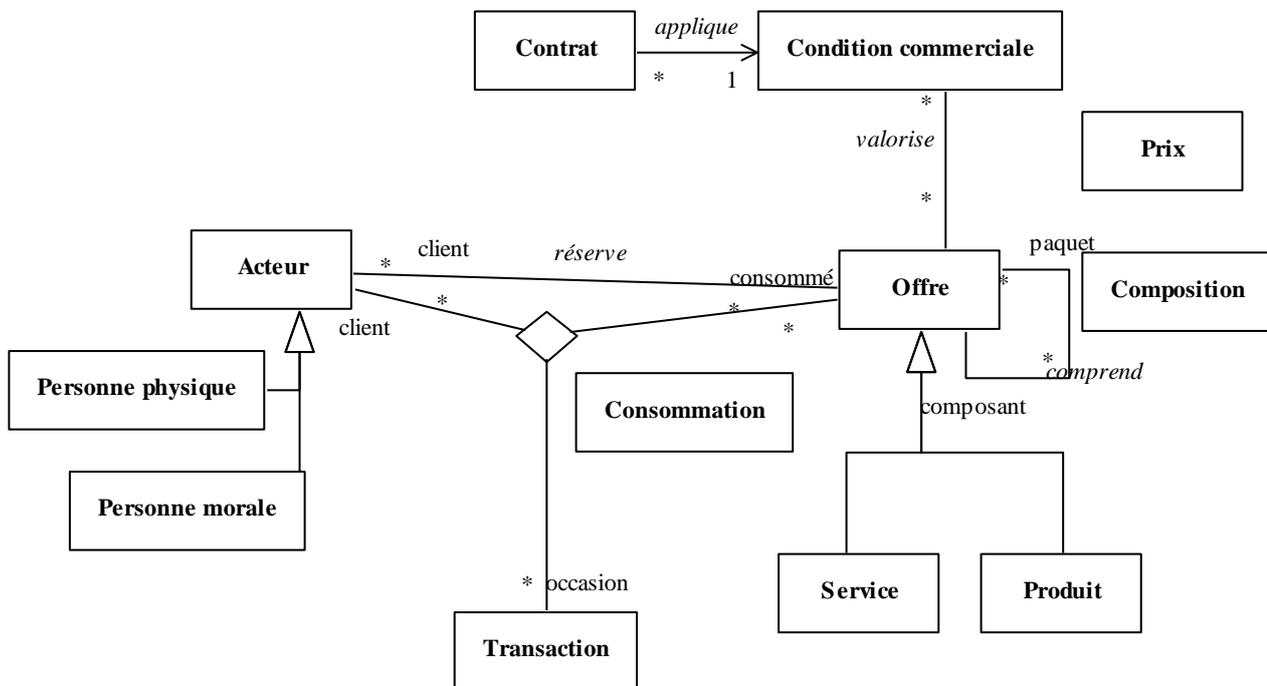
#### Comments

- The database is a technical element of the system. It is only a technical means. It is not part of the semantic aspect, which is based on the reality, independent of the system.
- A “dossier” is an object of an organizational nature; it groups objects that have semantic value, in view of a particular action or process.
- The notion of “client” is, without doubt, part of the universe of discourse. Does that make it an autonomous concept? It will be modeled as a role in an association of person with contract.
- Terms such as “declare” and “evaluate” carry real semantic content in the Insurance universe. “Save” and “search” are implicit operations, spontaneously provided by automation; there is no need to weigh the semantic model down with these factors.
- Semantics only keep the incontrovertible things, above and beyond variations in practice.

The diagram below shows some of the possibilities of expression offered by UML for semantic modeling. It is too complex to be representative of the diagrams shown in a model. It does not show class properties.

#### Example

Figure PxM-10en\_5. Example of a class diagram



## Products: repository and model

---

### Semantic repository

---

**Objective** The semantic or business repository stores the descriptions of an organization's perception of reality.

The repository progressively builds up a complete and detailed description of reality, which is then leveraged by the different projects. It describes the core enterprise know-how.

It accumulates and coordinates the semantic models, each designed to meet the needs of successive projects. The repository is a key tool in IT policy, enabling a clear dividing line to be drawn between the part of the reality covered by urbanization and the part not.

**Perimeter** The semantic repository describes, in theory, the whole Enterprise System. The information system and IT system can only translate part of it.

As an example, semantics are attached to market segments, products and services combined. Excluded from the semantic model are the contingent choices: organizational choices, practical details of implementation and, of course, IT solutions...

### Content

---

**Collected data** The semantic repository is a result of the modeling work:

- firstly, at a general level, to define the general structure and set the scope;
- then, at a detailed level, to capture the concepts and how they work.

The repository also acts as a glossary or, even better, as a thesaurus (see further on). Thanks to which, it gathers the diversity of discourse: synonymy, definitions and variants, use in context, specialized jargon<sup>9</sup>...

The model itself brings this diversity back to a more concise subset, where there is no place for ambiguity. It is however, necessary to conserve the input terms, both for reasons of traceability and to justify the model.

**Model depth** At the beginning, the semantic repository shows, above all, the decomposition in object domains. It identifies the principal objects and situates them within the overall set. It then gives a definition and a description to the root classes.

It is enriched, over time, by project contributions, to provide a complete description of the semantics. This description comprises all the details, including operations and constraints.

---

<sup>9</sup> In tool terms, this is the “dictionary” function. Each term is linked to the standard term, which refers, in turn, to an element of the model.

---

## Products: repository and model (cont.)

---

### The semantic model

---

#### Objective

The semantic model fully represents the part of reality defined by the scope of the project.

Its first objective is to **contribute to the smooth running of a project**, clarifying the notions of a domain and structuring the data and behavior associated with these notions. From this point of view, it constitutes an important milestone in the project approach.

The semantic model fulfills a second purpose: to enrich the semantic repository.

The arbitration between these two purposes is discussed further on (page 18).

---

#### Perimeter

For a software development project, the perimeter is defined, more often than not, in functional terms. The same applies to process modeling projects and to the majority of enterprise investments. Such a perimeter needs to be converted into valid terms for the semantic aspect. The modeler must engage in a projection exercise – projecting the functional into the framework imposed by the repository. The latter is structured in object domains, irreducible to functional domains. This projection work needs to take place in the early stages of the project, as it conditions its organization and dimension.

---

#### Content

**Collected data** The data is the same as that of the semantic repository, described on the previous page. Among the input materials conserved will be:

- minutes from interviews and creative sessions with domain experts;
- statutory instruments, standards, results from competitive intelligence, applicable to the domain;
- decisions relating to the model's structure, mainly in connection with the distribution of the notions within the repository (after consultation with the urbanist);
- business glossaries, etc.

**Model depth** The model should be complete and detailed, to meet the project needs.

This constraint does not mean that elements, not fully documented by the project, cannot be included. Such elements may provide anchorage for future modeling efforts or pave the way for more global thinking.

## Products: repository and model (cont.)

### Semantic model requirements: quality factors

#### Model usages

The quality factors are inferred from the model usages. There are four main usage categories:

1. **Communicate:** in particular with domain experts, but also with representatives of the system users, general contractors, partners, clients, sales force... as well as with the stakeholders, who participate in the strategy formulation.
2. **Control:** the model is more able than the input material to verify certain requirements, necessary for successful project completion.
3. **Produce:** the model is destined to be used by other projects, or even other teams.
4. **Learn:** the model is one of the knowledge capitalization tools.

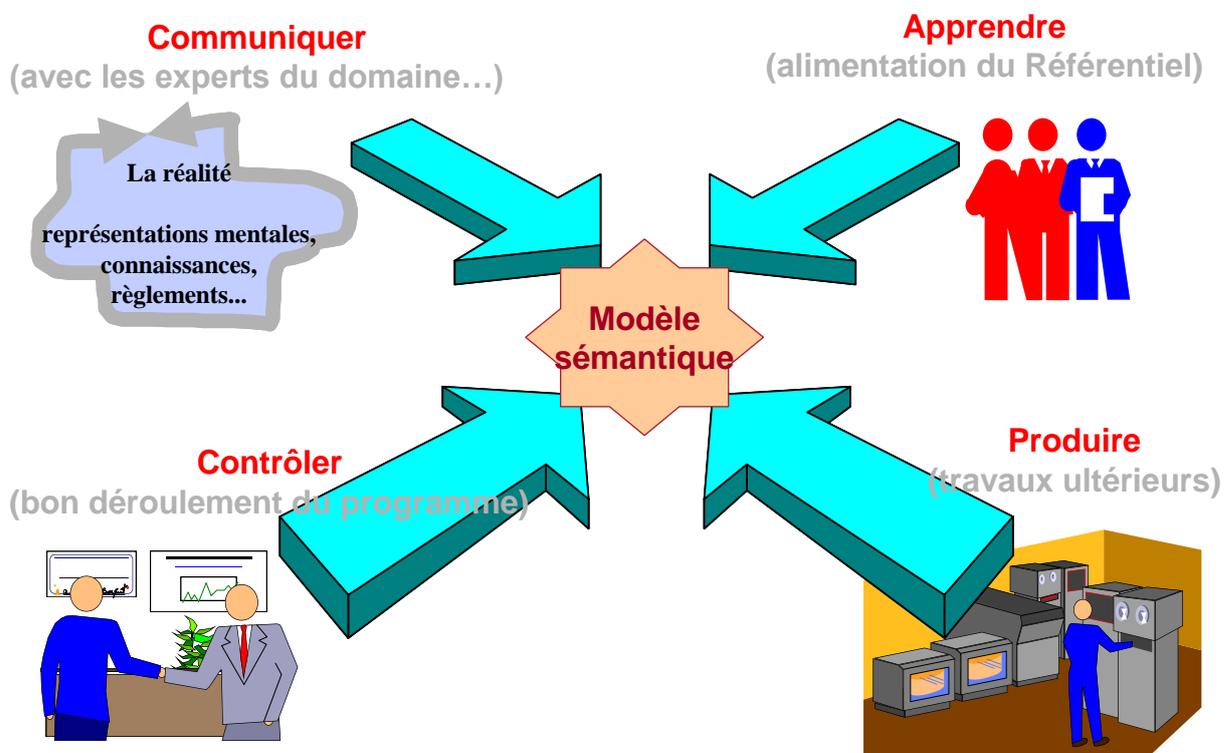


Figure PxM-10en\_6. Semantic model uses

#### Quality factors

From these uses, we infer specific quality factors on the semantic model:

- exactness: degree of closeness to the real world and to the representations of this reality made by the system's actors; in other words: quality of content (coverage, exhaustiveness);
- communicability: the model's contribution to communication (readability, auto-justification...);
- formal quality: respect of form constraints, which contribute to other factors themselves;
- reuse (capitalization) and exploitability (production).

## Products: repository and model (cont.)

### Semantic model requirements: quality criteria

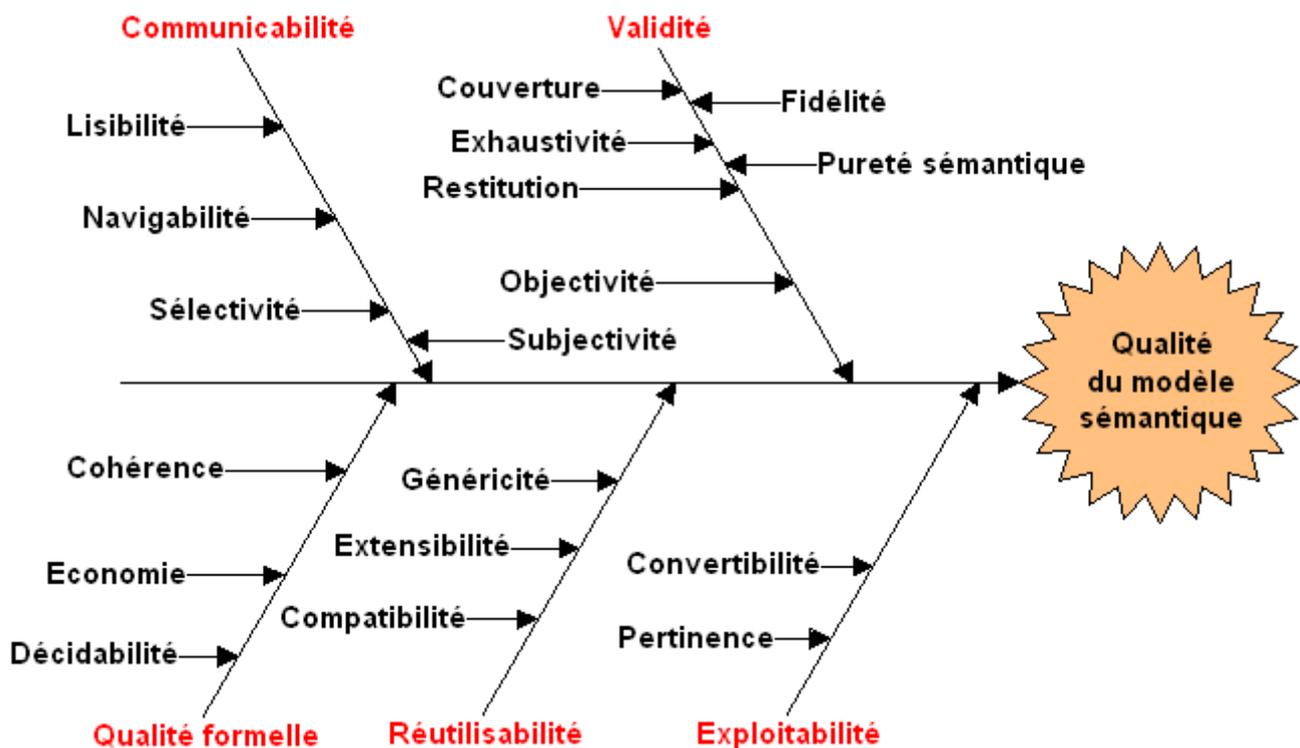
#### Criteria in relation to factors

Quality factors express requirements in terms of usage, from an external perspective. They have the advantage of being concerned with essentials and even with end objectives – which should be a constant preoccupation for any quality approach.

On the other hand, they have the major drawback of being unable to be made objective, and even less to be measured.

It is for this reason that they are analyzed in terms of “quality criteria”, which contribute to the factors and are more easily observed.

Figure PxM-10en\_7. Ishikawa diagram of quality criteria (cause and effect diagram)



## Products: repository and model (cont.)

### Semantic model requirements: documentation

#### Documentary requirements

In addition to the requirements outlined above and the requirements which are common to all models, the semantic model takes on some special requirements:

- upstream traceability,
- restitution,
- semantic purity,
- delivery of quantitative indications.

**Traceability** A traceability mechanism must allow the elements of the semantic model to be justified in comparison to its input:

- conceptual requirements, when they exist,
- functional requirements (traditional needs elicitation),
- interviews with domain experts or user representatives,
- statutory or institutional documents,
- possibly, current database models...

Note that the model only answers upstream traceability. There is no reference to the downstream (in accordance with the Enterprise System Topology)<sup>10</sup>.

**Restitution** It must be possible to reinterpret the semantic model diagrams in natural language and to restate the domain's verbal descriptions, from the model. To do so, it is necessary to increase the expressivity of the model and to conserve the synonyms associated with the modeling elements. This second point is handled by the thesaurus.

**Purity** Purity is the characteristic of a model, expressing only the semantic aspect and excluding all other considerations.

**Quantitative indications** However the semantic model is not an “abstract” one in the sense of it representing an ethereal world. It describes reality. As such, it is interested in the quantitative information that can be gleaned at this stage.

The documentation, covering the key notions of the model, consists of:

- number of instances (minimum, maximum, possibly: behavior over time);
- volume of data (for principal classes and their secondary classes)<sup>11</sup>.

<sup>10</sup> A semantic model reference towards the other aspects would be heresy: the aim of the topology schema is to put the data to be processed in sequence, along the production chain.

<sup>11</sup> This information can be saved in the form of *tagged values*, by the UML tool, as defined in the standard.

---

## Products: repository and model (cont.)

---

### Thesaurus-index

---

**Need** It is important to show how the model restitutes the discourse of the actors in a system. This is all the more important as semantic modeling, along with the restructuring effort, rearranges these descriptions.

**Solution** The ideal solution to deal with this is the thesaurus-index. It is set up in the same database as the models. It is made up of dictionaries and sub-dictionaries, structured to reflect the state of the mentalities and fields. For example, there can be as many glossaries as there are different occupations or projects in the enterprise.

All the terms employed, with – if possible – their definition and origin, are saved here. These terms are bound together (thesaurus function) and, above all, sent back to the modeling elements.

In this way, the thesaurus acts as an *air lock* passing from current practices to the new, as defined by the models.

Furthermore, the mechanism has an “index” function, which lists all the terms in alphabetical order.

**Pre-modeling** In the production chain, the thesaurus is positioned alongside requirements management. It is the result of pre-modeling activity. It bridges, on the one hand, the initial, intuitive perception of the business and, on the other, the formal, demonstrable model.

## Positioning in the production chain

### A very stable model, contributing to each project

#### About stability

The semantic model is very stable, providing that its “purity” is respected. Evolution in the semantics can be absorbed with little problem, on the condition that the model's structure has been built in a sufficiently generic and concise manner.

In addition, the majority of the causes of change are not of a semantic nature. Isolating the semantics leads to identifying the system's stable core and protecting it from change.

Semantic modeling thus introduces a stage in the development process.

#### Repository and models

The dynamics imposed by the global-local dichotomy lead to interactions between the projects and the instance responsible for the vision of the system.

#### Process delivery

The action sequence is the following:

1. At the start of the project, the specific request is positioned within the scope of the system. The terms relative to the request need to be identified (candidate classes) and **positioned on the global IT system map**.
2. This analysis leads to **evaluating the proximity** between the local request and the global vision. The local response can be adapted to increase its contribution to the repository. Conversely, and in the extreme, the two logics can be decoupled and the project removed from the urbanization.
3. In a favorable case, the **zones of compatibility** between the model and the repository need to be specified. The importance of this will become clear in the last stage and, in the meantime, will restrain the modeler's work.
4. The useful elements for the project can then be **extracted from the repository**.
5. When the local semantic model is ready, it is appropriate to review it. The urbanist – or system vision owner – puts forward the global point of view. The method engineer can intervene regarding the formal quality of the model.
6. When the model is considered to comply with the rules and to have been endorsed by the methodology and urbanization plan, it can be **updated in the repository**. The “compatibility zones” identified beforehand, will be extracted.

#### Search for convergence

In this balancing act between general interest and short-term perspective, there is one overriding factor: the force of constraint which the management decides to exercise in building the system, to comply with an urbanization target. It is all the more important and decisive that the management be operationally involved in the system enhancement policy, a policy which is a direct result of the enterprise vision and strategy.

---

## Positioning in the production chain (cont.)

---

### Dynamics between system and project

---

#### Repository and models (cont.)

This dynamic results in an important exercise taking place: the arbitration between target levels. There are two conflicting points of view:

- The project's role is to provide an immediate response to a specific request.
- The system, on the other hand, is built with the long term in mind; its chief concerns are universality and openness towards partner systems.

This contradiction is resolved through reuseability: the repository provides the project with elements which fit its objectives.

#### *Arbitration*

The conclusions from the arbitration are shared over a range of scenarios, of which the most extreme are:

- complete compatibility: everything the project delivers can be picked up by the repository and used by others;
- complete decoupling: when the request is too divergent or urgent, it would be detrimental to force the project to follow the urbanization rules.

In between these two extremes lie numerous possibilities. The divergence may apply to only part of the project. Subsequent recoveries can be planned for: the immediate model puts down anchorage points that more system-compliant elements will attach themselves to later etc.

---

## Semantic modeling: analysis and design tool

---

### Semantic modeling moments

It would be a mistake to reduce semantic modeling to a mere analysis activity. Indeed, as with any of the aspects, the modeler can adopt both a design and analysis **posture**.

The moments, during which modeling is called for, are the following:

#### *Analysis of the existing*

On the one hand, the existing system can be a starting point for semantic expression, more cost-effective than a series of interviews. On the other hand, a diagnosis of the existing system will clarify certain choices relating to the future solution. A focus, for example, on how data is organized, redundancy levels, volumetrics...

#### *Needs elicitation*

During the needs elicitation, semantic modeling isolates the fundamental concepts, which makes lighter work of formulating the functional requirements<sup>12</sup>.

#### *Semantic design*

Semantic modeling becomes inventive when it incorporates a new concept or proposes a new structure of concepts.

#### *Design in detail*

Decisions relating to structure, concerns about genericity, factorization... lead to decisions which take semantic modeling beyond a simple review. In the same way, considering objects as autonomous and cooperating machines, results in producing a model which can appear very innovative with regards to current representations.

---

<sup>12</sup> It is not necessary to rewrite the same constraints or the same list of information for several functional requirements, when they have been factorized or encapsulated in the semantic model, either before or in parallel.

---

## Positioning in the production chain (cont.)

---

### Semantic model verification

---

#### Verification

The semantic model is, first and foremost, a description of the real system (and not a description of the software system). Before continuing with the work, the validity of the model should be checked. It is necessary to verify the quality and exactness of this model, before deriving it, in the following aspects.

---

#### Tests from the model

The semantic modeling file is attached to the model and contains the test case descriptions that can be gathered from it. It is a “unitary conceptual test”. The model contains the description of autonomous micro-machines (classes) which work (state machines, message publishing, etc.). It is important to be in a position of being able to verify the behavior of these micro-machines.

The design procedure and method for test cases is largely based on contractual modeling.

Designing tests from the model should not be kept for the last phase of the project, but should be conducted consecutively and by the same people as for the model itself. There are two reasons this type of planning:

- firstly, it is cost-effective, as it leverages the competence of those who know the semantics best, to ascertain its exactness;
- secondly, it generates feedback on the model. The modeler, in designing the test cases, has to check the exactness and clarity of his/her model.

To further improve this mechanism, the best option is to opt for cross-checking: each modeler designs tests for someone else's part of the model.

---

#### Model simulation

Simulating the semantic model involves subjecting it to critical rereading in workshops, either by automating it or “by hand”.

Automatic simulation can be seen as quality assurance. It relies on having a simple, but fairly powerful, environment to hand, to limit the amount of development work necessary to transform the model into executable prototypes. Such an environment must be able to provide quasi-automatic responses to the following needs:

- persistence (in order to rerun the simulation or test scenarios);
- automating state machines;
- message and event publishing in the system.

These points will, of course, be the focus of the technical architecture design. There is no need to wait for the target architecture to simulate the semantic model. On the contrary, it is preferable to be equipped with an *ad hoc* environment, for example an object-oriented database and transformation *patterns* or rules which will mechanize the above mentioned elements.

## Positioning in the production chain (cont.)

### Ulterior use of the semantic model: neighboring aspects

#### Immediate neighborhood

What happens to the information in the semantic model, in the rest of the aspects?

Two aspects refer directly to the semantics. Their relationships with the semantic model are very different.

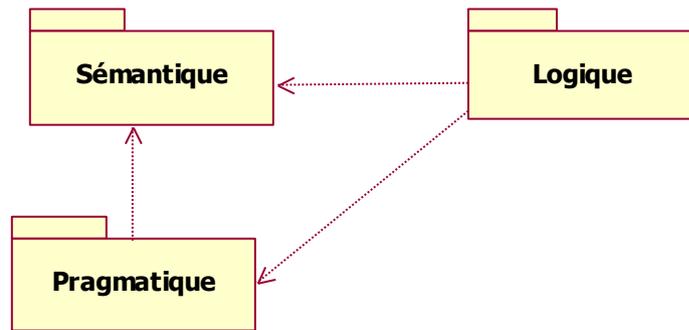


Figure PxM-10en\_8. The aspects in relation to the semantics

**Pragmatic aspect** Elements of the pragmatic model, such as the operations, processes, use cases and type of actors, refer to elements from the semantic model, in particular: classes and their properties, state machines and events.

The pragmatic model shows how, following the organizational choices made, the actors manipulate objects from the semantics. This manipulation is reduced to sending messages, in accordance with the encapsulation principle.

The pragmatic model formulates operation modes, as it were, *around* the semantic model.

**Logical aspect** The logical model is positioned totally differently. One of its functions is to specify how the semantic model elements will be translated into the IT solution.

To this end, there are two key options available to the logical architecture:

- to take up the semantic model as is, specifying how to extend it to the heart of the system<sup>13</sup>;
- to transform the elements from the semantic model into logical components, restraining them by applying rules specific to the logical architecture.

In the current state of technologies and competences, the second option is the one most often retained. One speaks of “derivation”, as the semantic modeling elements are not transformed: they remain unchanged in the Enterprise Repository. They are derived, i.e., their descriptions are used to produce other elements – the logical components.

<sup>13</sup> This option is obtained through a *framework* or, better still, through a virtual machine.

## Positioning in the production chain (cont.)

### Derivation channels

#### Derivation

The semantic model offers several immediate contributions:

- the formal expression of business fundamentals (capitalization of knowledge, *knowledge management*),
- the abstraction effort (more genericity, and so a more compact model),
- universality (a pathway to convergence).

The semantic model is also the starting point for several derivation channels, which take up its modeling elements, either as a design support for other aspects, or to design other modeling elements.

#### Three channels

Praxeme foresees three derivation channels:

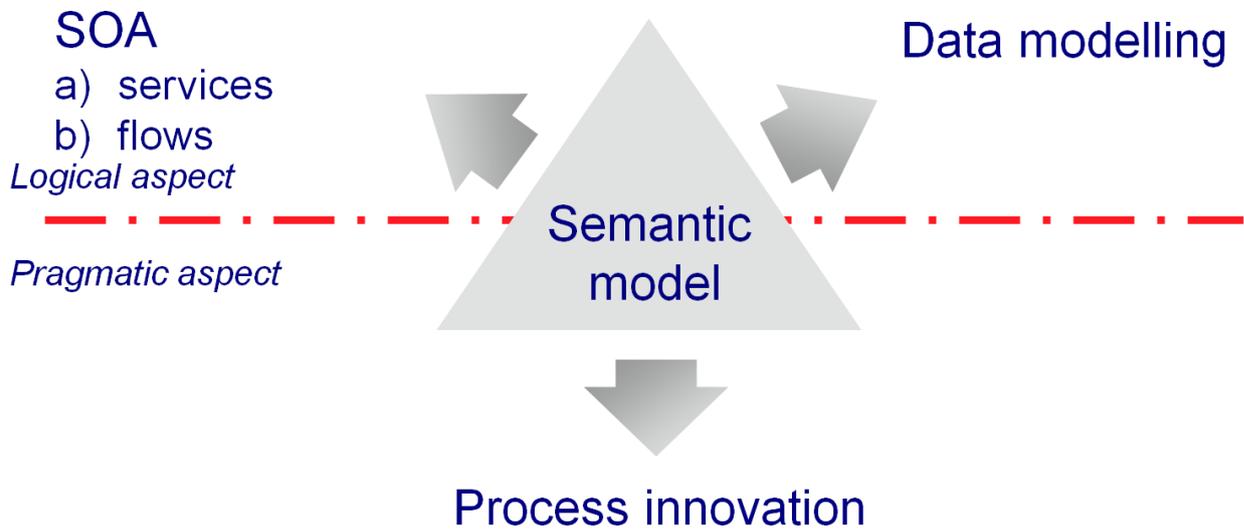
- one towards the pragmatic aspect and innovative process delivery design,
- two others towards the logical aspect for data modeling and service design.



This approach relies on the Model Driven Architecture (MDA) standard.

The pragmatic and logical aspect guides detail the derivation and design procedures, linked to these channels.

Figure PxM-10en\_9. Derivation channels from the semantic model



---

## Positioning in the production chain (cont.)

---

### Ulterior use of the semantic model: other aspects

---

#### Beyond neighboring aspects

Semantic model elements, transformed or simply passed from one aspect to another, will inundate the whole production chain up to the physical level.

The following points are of note:

1. The description of operations is part of the semantic model, at least for those operations which have a meaning at this level. It provides the algorithm for these operations to be realized in the software.
2. The state machines will be taken up either by all the operations involved, or by a general mechanism which will enable their management or execution<sup>14</sup>.
3. Events are a very useful category of representation in the semantic model, which can also be found in the pragmatic model. This category is not necessarily leveraged by the technical architecture<sup>15</sup>. It must, therefore, be converted into a valid, system-wide mechanism.
4. In the same way, it is sometimes necessary to transform the inheritance, when the target environment does not provide an equivalent one.
5. Persistence is indicated for semantic classes. This information, associated with the volumetrics and structure of the model (associations, cardinalities, roles...), will become part of generating the logical, and then physical, data support schemas. It is not surprising that this data on persistence and object volumetrics meets in the semantic model. They are both part of the semantics: before being reformulated in technology terms, they belong to the business reality. The same cannot be said, for example, of the choice of option for deriving the inheritance tree<sup>16</sup>.

---

<sup>14</sup> A Gang of Four (GoF) *pattern* exists, which takes care of that, even if it is costly.

<sup>15</sup> With the exception of *workflow* solutions.

<sup>16</sup> These considerations on the semantic nature, or not, of the data, intervene when developing the tooling and detail of operation modes. The guidance is simple: the semantic model must only contain data of a semantic nature, independently of technical choices and IT design decisions. This simple, common sense requirement is driven by the need to classify the data as best as possible, across the whole production chain. It is sometimes thwarted by tooling constraints.

---

## Positioning in the production chain (cont.)

---

### Communication by the models

---

#### Problem

The perception categories that IT workers apply to the information system – which they view mainly as being the IT system – are not those of the “users”<sup>17</sup>. These categories are imposed by the confrontation with different technologies, as relayed by the vendors, and shape their way of seeing things.

The contracting owner would not know how to adopt the same vision. He/she carries the stakes and knowledge of the business and views the system as a tool, not as an end in itself.

These cultural differences cannot be overlooked, just as separating the production chain into project management and contracting owner parts is considered natural. However, they are detrimental to communication and introduce a breakdown in communication which can be fatal.

---

#### Recourse to models

By construction, in addition to being a design means, a model is a communication instrument. Or at least, it can be. For it to take on this role, its end aim needs to be fixed, in accordance with the communication needs of the project and the cognitive domains of the different actors.

This is the native idea of the MDA standard, which reiterates the necessity of having models that are independent of technical and other dependents. It is a way of going back to the idea of levels of abstraction. MDA adds a passage technique from one model to another, thanks to the UML profiles. MDA does not define the models. Praxeme does.

Two factors promote communication between the contracting owner and the project manager:

- the meticulous definition of models from the System Topology aspects;
- the modeling requirement and verification techniques which enable a complete and coherent model to be delivered to the project manager, so that he/she only needs to be concerned with the functional content.

To this is added the articulation of the aspects and the possibility of automating the derivation rules from one aspect to another.

---

<sup>17</sup> The term “user” is a telling one, with regards to the ethnocentric view of the IT worker!

## Procedures and methods of semantic modeling

### Representation techniques: UML contribution

#### UML contribution



Although the primary vocation of UML has not explicitly been upstream modeling and, on the contrary, notation has been pulled constantly downstream, it provides some good tools for expressing semantics.

The modeling element types listed on page 6 generously cover the needs of semantic modeling.

For each modeling element, the model provides a clear definition, the exactness of which depends on the category and progress level of the modeling.

For example:

#### **Class example**

A class will own a definition. If it is a building block on how the reality is perceived, or immediate data about conscience, it will be difficult to produce a formal definition. If it is a subordinate or secondary class, the definition could be expressed in relation to a central notion and completed by the mention of responsibilities vis-à-vis the other classes.

#### **State example**

In the documentation of a state within a state machine, the modeler first has, at his/her disposal, a state name, making relative sense with the class name. He/she will progressively specify the value of this state as the other states are identified. It will be possible, later on, to associate a formal description to the state, for example, in terms of a range of values for the object attributes or the presence of other objects in its environment.

### Representation techniques: class diagram

#### Indispensable

The class diagram is an indispensable part of object modeling. The diagram is not the model, but a partial, biased illustration of it. Once finished, the class model represents the whole substance of the system, both data and processes<sup>18</sup>.

Each diagram is realized with a communication goal in mind. It only presents the elements which contribute to this objective. Diagrams must be readable: size will be limited to a sheet of A4 paper and rules such as the famous *magic number seven*<sup>19</sup> can be applied.

However, at least for the modeler or developer's needs, a class diagram can exceed these limits and even contain the whole of the model. It follows that this diagram will then be known as the class model. It will not necessarily be part of the file, but constitutes a tool to find one's way around the model.

<sup>18</sup> These two notions of data and processing do not belong to the semantic aspect but burgeon over IT terminology. In the semantic aspect, preference is given to the terms: information, action, transformations and associations of objects.

<sup>19</sup> Rule summarizing the work of G. Miller in experimental psychology: a good structure – from a presentation point of view – is made up of about seven elements (give or take two).

## Procedures and methods of semantic modeling (cont.)

### Representation techniques: object diagram

#### A complement to illustrate the model

The object diagram is used each time it is necessary to illustrate particular points in the class diagram. In particular, the sophistications and subtleties of notation call for this type of clarification, for example: associative classes, n-ary associations, networks...

Two types of use:

- In the files, object diagrams are inserted to increase readability. It is a technique to reconstitute the vocabulary or the user representations, especially when the model moves away from them (see the example below).
- The modeler may feel the need to carry out object diagrams as a rough draft, to help clarify ideas or test representation choices. Such diagrams are not necessarily conserved; in any event, they will not be included in the final documentation.

**Example** The below object diagram follows on from an initial class diagram, modeling the current way that the user expresses things. The modeler proposes a more concise model (the “right” solution if we apply object logic). The object diagram is the tool of choice, to link them both.

Figure PxM-10en\_10.  
Spontaneous expression

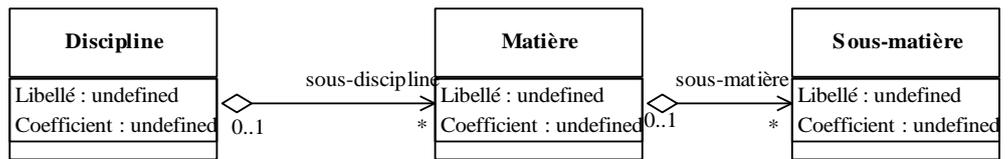


Figure PxM-10en\_11.  
Correct model

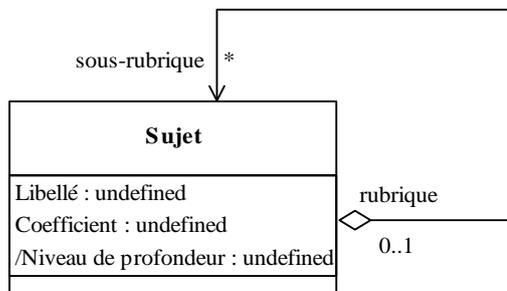
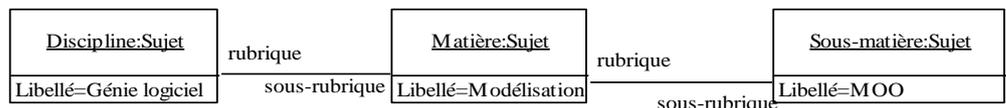


Figure PxM-10en\_12. Object diagram



## Procedures and methods of semantic modeling (cont.)

### Representation techniques: state diagram

#### At the heart of the object-oriented approach

The UML state diagram takes the different representations of state machine or state automaton back.

This approach, contractual modeling, is an essential tool to push the object logic further, i.e., to design classes as truly autonomous machines, responsible at any given instance for their internal state and accepting responsibility for missions proposed externally.

#### When should the state diagram be used?

The modeler needs to ask him- or herself the following questions, for each class:

- Do the objects in this class follow a life cycle?
- Do their transformations obey any constraints or a concatenation?
- Is their behavior conditioned by their internal state, or the stage at which they find themselves at a given time?

If the answer to any of these questions is yes, the modeler is justified in representing the phenomenon in the form of a state machine.

A contract, a claim, an operation... have a life cycle with different states marked out. They will behave differently depending on the value of the state. The classes representing them are therefore equipped with state machines.

#### How to proceed?

The notions of status, milestones, stages... as well as the adjectives or present participles bracketed to the class name, constitute starting points for isolating states.

Once the states have been identified, valid transitions are represented (without doubt more than in a nominal object mechanism, all too often reduced to a linear list...).

To consolidate the state machine, it is important to consider how an object is stimulated and how it behaves under stimulation from each state. Backward steps are added to absorb any disruption the object may be subjected to, during its existence (life cycle regression).

Then, inscribed on the transitions are:

- trigger events or conditions,
- operations which carry out the transition,
- emit events.

The state machine is a good way of freeing up the operations of a semantic nature.

#### Notation

When the modeling is finished, the transitions carry the operations inscribed on the class that the state diagram is made for.

---

## Procedures and methods of semantic modeling (cont.)

---

### Representation techniques: other techniques

---

#### Semantic fields

Before being in a position to design a first model, the analyst can take a less formal approach. As an initial approach, the semantic field technique allows him/her to gather and organize the terminology of the domain analyzed. This very simple technique is less formal. It involves jotting down on paper, the terms present in the universe of discourse and linking them according to their semantic proximity. In this first step, there are only two categories of representation:

- terms;
- relations between terms (arrows, if necessary accompanied by a label which expresses the nature of the relation between the two terms).

The semantic field tolerates synonyms. It is a pre-modeling technique, akin to the thesaurus, bridging in this way the raw materials and the first model. It enables a first take at organizing the material, which will be leveraged by the modeler afterward. The passage from semantic fields to model, requires the modeler to decide on each term, eliminate synonyms and redundancy, determine the most suitable type of modeling element...

---

#### Textual analysis

The table on page 8 introduces the hypothesis of the correspondence between the natural language categories and those of the object approach. On the evidence of this observation, textual analysis enables an almost mechanical passage from the texts collected towards the model. Several tools are based on this principle. The modeler can proceed manually, highlighting elements of text with different colors, according to their category: substantives (candidates for a class), relation verbs (associations), action verbs (operations), adjectives and participles (states), etc.

This technique is a good starting point for a modeler who feels a little lost when faced with the complexity or novelty of the analyzed domain. It encourages the domain descriptions to be collected or written.

#### *Writing rules*

The preparatory writing of these descriptions complies with some style rules: simple phrases (subject, verb, object); clearly expressed modalities; articulations without any ambiguities...

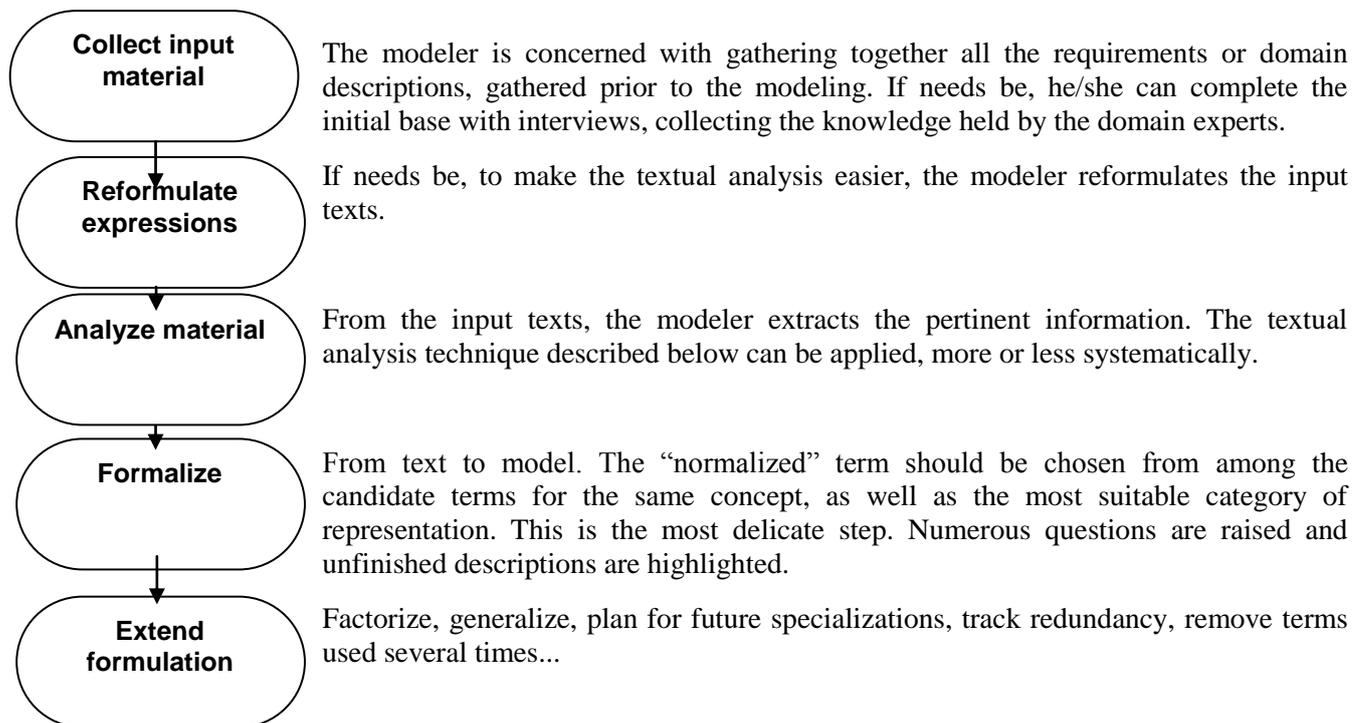
- Go from the passive to the active form.
- Make shorter sentences.
- Clarify singular objects, deictics (words whose meanings are dependent on context: e.g., “this”, “this one”, pronouns, certain adverbs...) and implicit meanings...

## Procedures and methods of semantic modeling (cont.)

### One procedure

#### Introduction

The “procedure” given here is in the region of the process... in other words, it is applicable at an individual level: guiding a modeler's task for scaled-down elements.



#### Other techniques

The following two procedures are of particular interest for the semantic aspect, being easy to tool:

- dictionary, if possible in its advanced thesaurus form;
- traceability.

**Dictionary** As far as possible, all pertinent, modeling input elements are stored in the documentary base and organized in glossary form, more substantial than the model itself.

**Traceability** It is important, sometimes even necessary, to prove or justify the “coverage” of a model. The modeler places traceability links between the modeling elements and the input material.

When a dictionary exists, upstream from the model, it acts as an intermediary for the traceability between input material and model.

---

## Procedures and methods of semantic modeling (cont.)

---

### Semantic modeling activity requirements

---

#### Level of detail

A frequent confusion is to bring the semantic value of the model back to its generality: it is a mistake. The semantic model can be general, like architecture, if so decided; but, a semantic model is only complete when it has restituted everything that needs to be known about a reality to act on it. As a consequence, the semantic model is every bit as detailed as any other. Simply, the details that are added to it do not come from the technical solution, but, from a thorough description of reality. Operations and their description, as well as constraints, are found in the semantic model.

---

#### Principles

The requirements linked to the product (to the model) were discussed in the second section (page 14). The requirements discussed here are those that influence the modeler's activities. They can be expressed in terms of principles (when they are of general relevance) or techniques (when they concern a specific action):

- the encapsulation principle of constraints;
- the principle of abstraction;
- the incorporation of state machines;
- the decomposition in object domains.

These principles and techniques are discussed hereafter. Generally speaking, the quality of the modeling relies on the object logic being interiorized by the modeler.

---

#### “Object” quality

There has been little coverage given to the quality characteristics of an object model in literature. There is still a way to go before reaching the “normal forms” of classical methods. However, models which present only a few operations, little or no state machines, few associative classes... should be regarded with suspicion. Their absence is more likely to reveal the incompetence of the modeler, than the state of reality. Rare indeed, are the pure descriptions of reality that can do away with such “sophistications”.

---

## Procedures and methods of semantic modeling (cont.)

---

### Encapsulation principle of constraints

---

#### Enunciation of the principle

#### Constraints are encapsulated in “classes”

Obviously, the semantic model only takes into account the semantic constraints in the sense of “business rules”, constraints strongly linked to the semantics, the mechanism of “business” objects...

Business rules are the constraints that have an impact on the behavior of “business” objects. They should not be overrun by operational rules and choices, nor by technical constraints.

---

#### Justification

The expected benefits of the object-oriented approach only come to fruition if the object logic is respected. Structuring rules lead to autonomous units, responsible for their state, being built. These units assemble and protect the data, processes, internal states and, of course, rules.

In this way, a single business rule is written only once into the overall model.

---

#### Practical consequences

- Dossiers which list constraints independently of the model or in an appendix form are not accepted.
- Encapsulating rules is only possible if there is some intervention on the model, with the missing anchorage points being added (operations, “reified” associations).

When looking to encapsulate constraints, the modeler sometimes has to modify the structure of the model. In the most common cases, an operation will be added to a class, in order to localize the rule there. Sometimes, the modeler will have to add an associative class, so that a rule, affecting several objects, can be localized. The impact on the model is an important one, so much so that modeling cannot be deemed finished until the constraints have been localized.

---

## Procedures and methods of semantic modeling (cont.)

---

### Abstraction principle

---

#### Enunciation of the principle

**Objects and concepts at the heart of the business are modeled for themselves, abstraction of organizational and technical circumstances.**

---

#### Justification

The “heart of the business” is very stable. It changes only when the enterprise modifies its product and service offering or when external factors (legislation, regulation) require it to adapt. In the case of physical systems, the stability of the description is even more self-evident. It is, therefore, important to safeguard this aspect from the introduction of decisions more susceptible to change.

When designing the semantic model, the modeler eliminates those decisions that he/she is entitled to question, as they do not belong to the semantic aspect as previously defined. These decisions can concern practices, organizational choices, the IT solution and even the current state of products sold. The objective is to rediscover the simplicity of the objects at the heart of the operations. The semantic model is all the more better because it is simple and “speaks” to the user of the system. The modeler expels the artificial objects, the pseudo-concepts resulting from the fossilization of administrative habits, or the bureaucratic perception of things, from the model.

The semantic model only retains those objects and concepts which indisputably belong to the business fundamentals. The criterion to recognize them is their universality. The semantic classes will later be restituted as they are, in the IT system. They will take the shape of very stable units, which will last and will not be impacted by changes to the other aspects.

In addition, this modeling basis enables existing practices to be revisited and processes to be redesigned and simplified, by centering them on the “natural” life cycle of business objects.

The modeling effort focused on this aspect has another important consequence: it enables the common points between the different practices to be picked out. Indeed, at this level of design, the variations that may exist between one enterprise and another are glossed over: there is no place for them in this plane. Semantic modeling is, therefore, a powerful instrument for system convergence.

Other justifications for the abstraction principle:

- to contribute to the quality of the system;
- to facilitate the dialogue between the actors of the systems;
- to facilitate the novice's comprehension of the enterprise world.

---

## Procedures and methods of semantic modeling (cont.)

---

### Abstraction principle (cont.)

---

#### Practical consequences

1. Each development is based on a “semantic” model, in other words, a business object representation, free from organizational and technical references.
2. The quality of the semantic model is subject to verification as planned in the development approach. Formal verification focuses on the conceptual “purity” of the model.
3. The simplicity of the semantic model is the result of abstraction and genericity. It can sometimes disconcert the user when it leads to concepts and objects being restructured.

---

## Procedures and methods of semantic modeling (cont.)

---

### Sharing principle

---

#### Enunciation of the principle

**Business objects that concern several actors, operations or processes in the system are shared via the “repository”, accessible to all.**

---

#### Justification

At the semantic level, the system is structured in “object domains” so that everything that can be shared is: information, objects, rules, associated documentation, requirements, etc.

The sharing principle allows:

- major concepts<sup>20</sup> to be unified and system redundancy to be reduced;
- the exploiting of information to be increased.

This principle is reinforced by the precept that a single term appears only once in the whole model.

The sharing principle is applied, without derogation, on the semantic plane. It does not prejudge in any way, the denormalization decisions that will be taken in the logical architecture or at a later stage.

---

#### Practical consequences

1. Each local model (produced by a project) flows into the enterprise repository. Here, the structure and articulations will be checked with the rest, as specified by the urbanization target.
2. In return, projects draw on the modeling repository for part of their models and all the interface notions. They evolve these models to meet their specific requirements.
3. In this way, the repository is progressively consolidated to reflect the business reality, with all its details, even if the applications cannot yet be structured according to the same sharing principle: what is feasible on paper, in upstream models, takes longer to filter into the reality of the IT system.

---

<sup>20</sup> In the existing system, a single concept can be processed by several applications.

## Procedures and methods of semantic modeling (cont.)

### Recommendations and best practices

#### Identify principal objects

The modeling work is organized from the principal objects, which must be identified very early on.

Then, for each of these principal objects, an exhaustive description will be provided and secondary objects adjoined.

#### Structure the model

Exploring the reality quite quickly uncovers dozens of concepts, represented by as many classes. The question is then raised of how to organize this material, to manage it and to distribute the work. This structuring must happen at the semantic level: it does not apply the same strict rules as does the logical architecture, but provides it with a rough draft.

To structure the semantic model, the criterion of decomposing into object domains is applied (as opposed to functional domains). This recommendation is detailed on the next page.

#### Keeping modeling efforts within reasonable limits

Reality is inexhaustible; signification protean. Semantic modeling can be taken a long way, far beyond the needs of any one project. Such “**meta**” **drifting** should be condemned – it consists of simplifying the model excessively and making it so generic that it is no longer possible to know what it can be applied to.

The following actions keep us from the “genericity frenzy”<sup>21</sup> :

- When the class diagram becomes too compact, open it out into the form of object diagrams; explain how it restitutes the user or the domain expert's perception.
- Write the attributes and operations on the classes (classes are not simple concepts, but micro-machines which deliver services).
- Only retain operations of a semantic nature, picked up from the state machine during the encapsulation of the constraints<sup>22</sup>.
- Express the class responsibilities.

The model – even semantic – is not just a drawing, but a communication tool and, even more so, the plan of a machine which one must be able to get up and running, at least in spirit.

<sup>21</sup> As there is a risk, during the elicitation of needs, of a “specification frenzy”, against which the methodological tradition warns us to be on our guard.

<sup>22</sup> The perpetual operations “save”, “modify”... or “visualize” bring absolutely nothing to the semantic model. They are not semantic. When the model shows only these types of operations, it is a sure sign that the modeling of the business is far from complete.

## Procedures and methods of semantic modeling (cont.)

### Principle of decomposition in object domains

#### Enunciation of the principle

**On the semantic plane, the criterion used to decompose the system is the “semantic” object.**

#### Justification

Enterprise System).

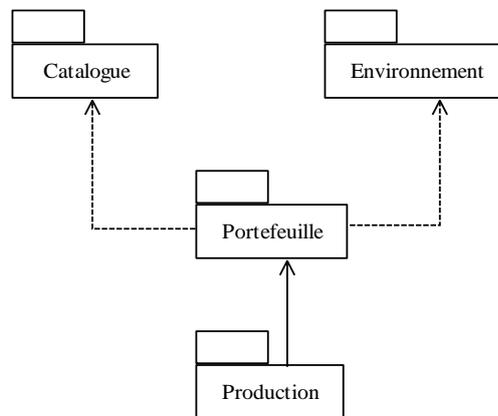
The global model of the “heart of the business” is structured in object domains, i.e., in areas which are organized around the principal objects of the described reality (the

This principle is set against a functional domain structure, as used by past methods. The functional approach was shown to bring significant redundancy. A change in approach allows, at least at the design level, a more economical representation, banishing redundancy.

#### Illustration

The package diagram below, gives an example of the decomposition of a system in object domains. In real-life cases, the structure is not that much more complicated: it consists of half a dozen object domains which may have a lot of dependencies. At this stage, coupling is not a concern. It will become one when dealing with the logical architecture.

Figure PxM-10en\_13. Example of a semantic model structured in object domains.



This change brings about a redistribution of the substance of the system, in accordance with a truly new architecture. The “logical architecture graph” gives an idea of this.

Each of the so-defined areas, is placed under the responsibility of a development entity within the IT department. This change in approach will also impact the organization of developments and definition of responsibilities.

## Procedures and methods of semantic modeling (cont.)

### Three axes of modeling

#### Principle

The object-oriented approach constitutes a whole. Trying to split it between different phases, for example by distributing different components of the description of a same phenomenon, would ruin its interest.

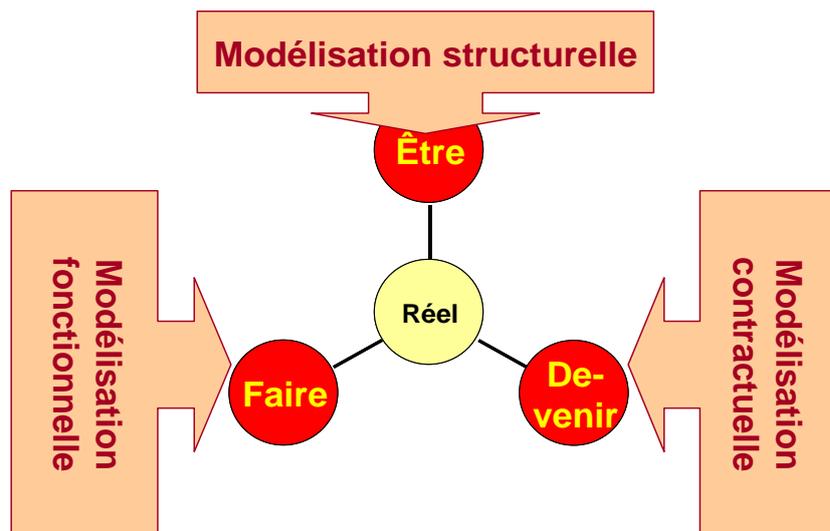
Object logic can be applied to each of the aspects. The complete UML toolbox may, therefore, be needed during different moments of the process.

Inversely, each object which retains the attention of the modeler must be examined in all the dimensions of our perception. Our approach to reality can be summarized by the three dimensions of reality (or, more specifically, the three axes of our knowledge): to be, to do, to become.

No matter which aspect the model covers, it must be envisaged in these three dimensions.

#### Axes

Figure PxM-10en\_14. Three axes of modeling



#### **Structural modeling**

Structural modeling identifies elements and organizes them in stable structures.

#### **Functional modeling**

Functional modeling is applied to different levels: from the global behavior of a system to the algorithm for operations. The operation – in UML terms – links up structure and function: it is part of the dynamics and fits into the structure.

#### **Contractual modeling**

The object approach provides the tools to describe the system as a set of micro-machines, responsible for their state and transformation. This third angle of attack is, without doubt, the most innovative.

---

## Procedures and methods of semantic modeling (cont.)

---

### *Structural modeling*

---

#### **Practical consequences**

This representation of the responsibility of modeling along three axes, albeit theoretical, expresses a radical requirement which will be seen in the reality of the models. The consequences that this will have on the quality of the deliverables and the project follow up, will be given in the following paragraphs.

---

#### **Object model**

The aim of the semantic model is to fully express the business fundamentals. In order to do this, the base unit is the “semantic class”, which represents a set of business objects. The semantic class picks up the semantics of the business object:

- its information (attributes, derived attributes);
- its actions (operations);
- its transformations (state transitions);
- its associations.

The classes are ordered in structures, through available relations in the notation: generalization, composition, association with all the sophistications which enable the richness of the discourse or reality to be restituted (associative class, qualified association, ternary or derived association...).

---

#### **Object domains**

When the scope of study is sufficiently vast, it becomes necessary to order these classes into sets. This act has serious consequences. UML provides a means for this – packages – but does not provide the criterion to delimit them. For the semantic model, this criterion is specified by Praxeme: it is the object domain – and most definitely not the functional domain. The object domain is built up around one or two principal objects, by “neighborhood extension”.

These domains are very coherent. Their coupling is closely studied (mutual relations are prohibited). Their division into object domains prepares some of the logical architecture decisions. This division greatly conditions the rest of the project and directly influences the appearance of the future system.

---

#### **Hoped-for improvements**

This semantic model, with its stable structure and detailed content, will be the source of the application core.

It is stable because, due to the abstraction required by semantic modeling, it is free from organizational or technical circumstances. Talking only of business fundamentals – and of this alone – its vocation is universal: services which derive from it will be shareable on a wide scale.

---

## Procedures and methods of semantic modeling (cont.)

---

### *Functional modeling*

---

#### **In the semantic model**

The functional approach in the semantic model is only readable at an operations level. It consists in specifying their algorithm. It can be completed, if need be, by dynamic diagrams which show the behavior of the global system.

## Procedures and methods of semantic modeling (cont.)

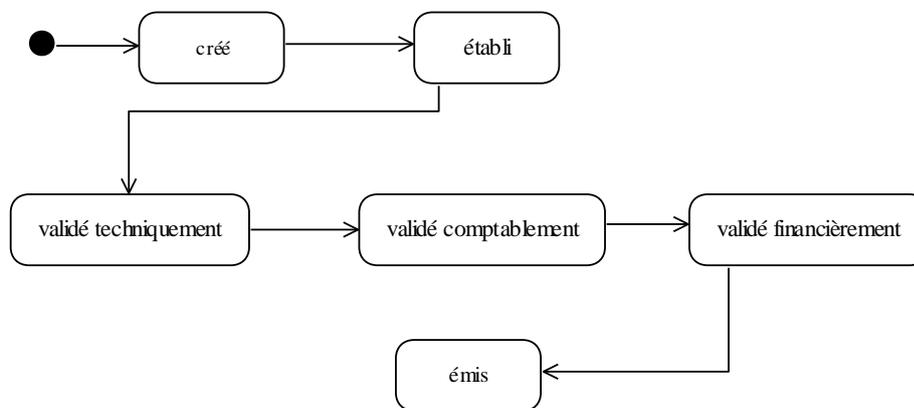
### Contractual modeling

#### Presence in input documents

Traces of object behavior or life cycle can sometimes be found in the modeling input. The difficulty here will lie in dissociating the transformations linked in depth to the business object semantics with those coming from the operations. The latter are legitimate but, being linked to organizational choices, are subject to change. The natural place for them is in the pragmatic model, not the semantic model.

This is the case with the example below. The different stages of validation are not part of the business fundamentals; the evidence being that they would not be found at competitor enterprises in the sector. The semantic model says too much.

Figure PxM-10en\_15. Example of a state machine polluted by organizational elements



This state diagram – incomplete – is overloaded with organizational constraints. Such a state machine should never rule a semantic class, but could have a place within the pragmatic model, on an administrative type object.

#### Contractual modeling means of expression

The most obvious form of the contractual modeling approach is the state automaton (or state machine); it takes the form of one or several state diagrams.

Contractual modeling covers the pre- and post-conditions of “classifiers”, i.e., classes and use cases.

The pre- and post-conditions need to be formed not only in natural language, but also in a thorough way: i.e., expressed in the terms of the model.

In the same way, all states of the state machine must, at the very least, have a definition and, at a more advanced stage, a more formal expression.

---

## Procedures and methods of semantic modeling (cont.)

---

### *Contractual modeling (cont.)*

---

#### **Contractual modeling contribution**

State machines offer numerous advantages:

- They are an irrefutable way of freeing up the operations (those operations which will become logical services).
- They avoid over-complicating formal descriptions. They are a more natural means of apprehending change.
- They incite the modeler to make the system more flexible by planning for life-cycle disruption. This flexibility will increase the system's reaction possibilities and the user's comfort.

Whether it be on the classes or the use cases, formulating the pre- and post-conditions prepare the service contracts that will be defined in the logical model.

## Procedures and methods of semantic modeling (cont.)

### Other recommendations

#### Aggregated data

How can indicators or consolidated data be modeled? All that are consolidated data, statistical results, steering information, etc. are often pushed to one side of the model. Yet, UML offers an elegant and economical means of integrating them in the model: these pieces of information take the form of attributes or operations on a class level.

It is not only a tip for filing the “upstream” model documentation. Indeed, these attributes or operations will be taken up when deriving the logical model from the model. They carry the seeds of the services which will position themselves on the collection accessors, planned for in the Praxeme procedure for Service Oriented Architecture (SOA).

#### Some modeling precepts

Still with the objective of presenting the Praxeme philosophy and demonstrating how far the modeling requirements can go, listed below are some precepts to guide the modeler:

- A term can only appear once in a model<sup>23</sup>. This precept clarifies the economy requirement of the model. The model is at its best when simple and compact.
- Business rules and all forms of constraints must be part of the model. As long as there are rules which have not been incorporated into the model, it remains unfinished<sup>24</sup>.
- Requirements must be qualified before being localized in the models (a functional requirement can be a business rule or an operational rule; in the first scenario, it will be oriented towards the semantic model, in the second towards the pragmatic model).
- A model must be readable, or at least, restate clearly and as exactly as possible, the language (consequence: on associations, the expressive possibilities offered by UML will be used, association and role names).
- “The map is not the territory”<sup>25</sup>. A model is not a set of diagrams: it is a complete and pertinent representation of a reality. It is comprised, above all, of its modeling elements. The diagrams are only the most visible part. The model must also contain comments: definition of elements, explanation of its mechanism, justification of modeling decisions...
- The simpler a model is, the better; but also, the more difficult it then is to explain. Hence the need to “unfold” it, using object diagrams to illustrate the more difficult points.

<sup>23</sup> In the opposite case, two things are one: either the term is polysemous, in which case the ambiguity must be resolved; or it expresses the same thing, in which case there is one modeling element too many.

<sup>24</sup> Incorporating business rules is another way of making operations of a semantic value appear. It is not an insignificant act: it can modify the structure of the model.

<sup>25</sup> Famous quotation by Korzybski, *General Semantics*

## **Appendix: illustrating semantic modeling**

# Semantic modeling

## An overview of the modeling techniques of the public Praxeme methodology

Dominique Vauquier

Translated by Jean Pommier, VP Methodology, ILOG; reviewed by Anthony Jervis, Accenture, and Joanne TOWARD..

*Semantic modeling is promised a bright future, although maybe under other names. Indeed, whatever we undertake, common sense and judgment lead us to think at this level of abstraction:*

- *To build content-rich services (like in SOA), you must first model the business.*
- *To merge systems from different organizations or corporations, the only solution is to identify the essential core and isolate the business fundamentals.*
- *To radically innovate and change processes and organizations, you must escape from current practices, get rid of any bias and identify a new starting point.*
- *To encourage and sustain ergonomic design, it is better to map GUIs to business objects rather than operations, etc.*

*Therefore, whatever our goal is and in order to do the right thing, we must first look at the reality and then step away from our usual habits or our cultural conditioning. This is what semantic modeling has to offer.*

## What is semantic modeling used for?

### An anecdote

When we opened an account for the Praxeme Institute<sup>26</sup>, our Treasurer and I were astonished by the following: In order to store our names and addresses in the system, the bank teller had no other option than to open dummy accounts for each of us, putting one euro in each. At least we had earned something! Yet, this is a trivial proof that our IT systems do not always match the reality.

### Generalization

It would be easy to find other stories to emphasize our point. We all know too well the status of our IT systems in this area: complication rather than complexity, redundancy rather than rationalization. But take a look at the reasons. By default, functional design has driven IT development for the past decades. Due to our culture and education, at least in the West, we think of the world in terms of actions. IT systems and applications have therefore been built as a set of procedures and functions. In the previous story, the procedure is the opening of a bank account. The opening has then been decomposed in sub-functions, in a very Cartesian way. Evidently the designer had never looked at the particular needs of an association and its relations with persons and the roles they play in this organization, etc.

Another factor has reinforced, amplified and dramatized the shortcomings of such a functionalist approach: the typical organization selected for IT development activities, that is to say that the Project mode, has been widely adopted for software development. Of course this mode has tactical, psychological and economical benefits. But it also has a major drawback: a project addresses a short-term goal, a goal which is itself expressed in terms of actions for business users and stakeholders. Naturally, a functional approach fits well with this mode and actually reinforces its limitations.

### What can we do?

Today we must simplify our systems, address increasing ambitions and master technologies in order to bend them so they can serve the enterprise. Not only do we have to redesign systems but also clusters and systems of systems. In addition to its shortcomings for building legacy systems, the functionalist approach is powerless to address the complexity of tomorrow's mega systems.

How can we elaborate components both content-rich and reusable? How can we merge systems from several companies, entities, parties? How can we continuously adjust the organization and processes of the enterprise without being limited by the complexity of IT systems? How can we enable and facilitate organizational design? How can we properly leverage technology, i.e., not at the expense of the business reality and context?

<sup>26</sup> See <http://www.praxeme.org>

We are facing huge challenges to preserve the agility and innovation capability of our enterprises, putting their survival at risk.

To address these challenges, we cannot keep using the same approach and thinking or pretending that they will work, eventually. Why would we be more successful than previous generations when they, as opposed to us, could rely on proven processes and formal thinking?

## A new way

The public methodology, Praxeme<sup>27</sup>, brings a new frame of reference. The framework is called the Enterprise System Topology. It includes semantic modeling as the initial and most upstream formal representation activity<sup>28</sup>. To support this semantic aspect of the enterprise, Praxeme proposes a process based on object orientation. This is the topic of this article. Although we envision addressing this semantic aspect with other techniques in the future, this is currently out of scope.

## What is semantic modeling?

---

### What it is not

What is not a semantic model? A semantic model is not a process or activity model: it lies upstream, in a world that humans do not live in. In which we don't see actors, organizational structures, habits and common practices. Speaking of semantic models, we use terms like *business* objects. Yet, many so-called *business models* or *business object models* end up by being simply limited and poor conceptual models of data. They only capture a small and truncated part of the business. Even their structure is potentially erroneous as we cannot be sure of their stability until the modeling activity has included operations and rules.

Let's immediately put a confusing myth to rest: when you look at a UML class diagram, with classes and attributes, this is not automatically a semantic model. In order to satisfy the encapsulation requirement, operations should be part of the diagram. When you see links – associations – which do not have a name, it means that the semantics have not been expressed. Bottom line, the use of UML or object orientation does *not* guarantee the semantic aspect of a model.

Furthermore, *semantic* (or *conceptual*) is not at all synonym of *general*. A good and finalized semantic model must be precise, formal, detailed and... annotated with comments. Otherwise it cannot be used efficiently.

### What it should be

What is a semantic model? The semantic model represents the objects and concepts used by the enterprise while conducting its business, independent of the means<sup>29</sup>.

The process of semantic modeling elaborated in Praxeme is based on the object-oriented approach and fully leverages the object paradigm. For instance, objects and concepts are represented as classes. Class does not correspond to the concept used in software and programming languages, but the general concept commonly used by philosophers since Aristotle. This is our way to describe our perception of the reality.

The model expresses all semantics related to these objects and concepts: information, actions and transformations. The class connects these three modeling dimensions. Simply looking at the information side will lead to a data model, which is not enough. Information needs to satisfy some constraints and is used in calculations. The model encapsulates these constraints and the operations triggering the computations. The classes are grouped into classifications through inheritance and into networks via associations. The process leverages the UML expressivity: associative classes, ternary association, qualified association, derivation, etc.

The semantic modeler must master all UML features. Otherwise he/she may produce a limited representation of the business. However, semantic modeling does not equal UML: the UML standard, which has a software connotation, is used as an instrument. What the semantic modeler is after is not Java classes, attributes and operations to implement in an IT system. He/she looks for information, actions and transformations as semantic

---

<sup>27</sup> See <http://www.praxeme.org>

<sup>28</sup> There are other activities and work products occurring before semantic modeling, but they are not models. They are expressions, more or less formal: goals and objectives, requirements, glossaries. But that is another discussion.

<sup>29</sup> In Praxeme, Enterprise has a generic meaning: any type of activity, organization, weapon system, system of systems, etc. It is a recursive concept of course. It also leverages the dual meaning of organization (firm) and action (endeavor).

characteristics of real life objects and concepts. To achieve this goal, he/she leverages the UML meta-model, reusing some meta-classes to map his/her own representation categories. The UML attribute, possibly a derived one, captures a piece of information. The UML operation captures an action, *on* or *of* an object. State diagrams will capture transformations as a sequence of operations and transitions.

## Illustration

### Simplified model of legacy application

Let's start with a draft of a model using a common approach.

Issuer

Figure 16. Quick draft (Merise)

This *model* will be used as a starting point for the discussion below. Boxes are extremely simplified views of the model. The goal of this article is to illustrate the benefits of semantic modeling based on an object-oriented approach. It highlights its positive impact on the quality of the system structure and design.

### Genericity

Object orientation fosters model genericity. For instance, it leads to a simplification of the model through the creation of generic classes such as Offer and Actor, hiding irrelevant additional details about these two key concepts.

Here are the benefits:

- Removal of redundancy: the properties (information and behavior) which are shared by Services and Products are only represented once in the model, on the upper class, Offer.
- Structure simplification: the number of associations decreases since the reservation, processing and pricing of an offer apply to both Products and Services but are represented only once, on the upper class.
- Power of the model: the model is more compact and concise; the process of selecting an offer covers both product and service selection. That is a better approach than separately developing a solution for product selection and another one for service selection. Savings are substantial.

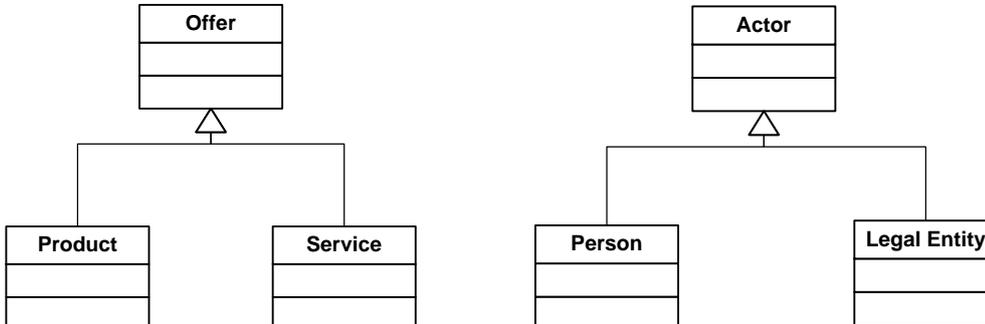


Figure 17. Examples of generic models

Nevertheless, the modeler should not push genericity too far. We too often see models full of inheritance graphs, with dozens of classes which only contain their name. This is a misuse of the notation: in this case the model is used as a semantic network in order to capture all the business vocabulary, inappropriately.

## Offer design

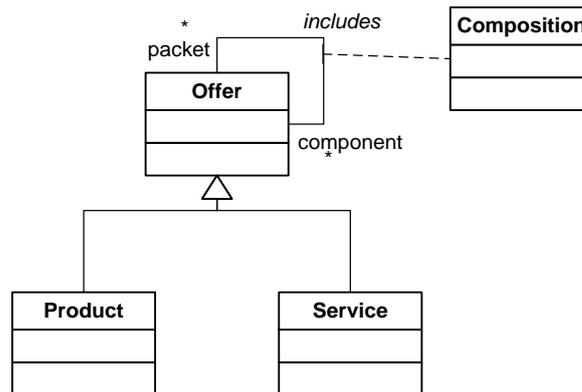


Figure 18. Solution to model the combination of offers

In Figure 3, the class diagram captures some of the semantics of the Offer through the mechanism of composition (reflexive association categorized by the associative class *Composition* to attach information to the relation). Thanks to this model and the *includes* association in particular, an offer can be either a product, a service or a combination of both (several products, a set of services or an offer combining several products and services). To ease the reading of the model, the modeler named the roles of the association which read as follows: “a packet includes components.”

Multiplicities capture the scalability of the model and show how concepts can be combined to represent real situations. In order to remain as universal as possible, semantic models usually use the most generic multiplicity, i.e., « \* » for many<sup>30</sup>. In the event the model needs to be restricted to take into account specific constraints in an enterprise, the best solution is to implement a MDM solution (master data management<sup>31</sup>) rather than altering the semantic model.

The constraint which prevents an offer from being a component of itself, or the one preventing the creation of cycles, is encapsulated into the model. No need to express everything graphically: the model is a whole and does not end with the diagram. Structural constraints are actually sometimes better expressed as pre-conditions rather than on the class diagram. All the constraints and business rules should be captured, or at least identified, otherwise, the model is incomplete.

The reflexive association *includes* is represented as a class, *Composition*. This class is called an *association class*. A semantic model includes many association classes. This formalism allows the modeler to establish a formal definition of pertinent concepts. In the above example, a composition describes a pair of offers, one being the container or packet, the other a component. The structure of the model represents the formal definition of this relation, better than any speech. The association class can, as any other class, bear properties (attributes and operations). It can even be, in turn, associated to other classes.

*This technique leads to the creation of compact models, able to deal with several levels of decomposition, where, in other common approaches, this would have produced several classes and tables.*

## Contract setting

Semantic models must be readable. More precisely, the model must provide a way to capture then reconstitute the context of the verbal description of the reality. UML offers the required expressivity, which the modeler needs to leverage appropriately.

The figure below corresponds to the following statement: an actor selects one or potentially several offers. Within this relation, the actor plays the customer role. Like in the previous example about composition, the *select* association is reified<sup>32</sup>. Indeed, the act of selecting an offer produces some information which needs to be stored and checked. This way, the model introduces the concept of *Contract*. Thanks to this visual representation, the reader knows that a specific contract applies to one and only one customer, and one and only one offer. However, thanks to the composition discussed earlier, a contract can apply to a combination of products and services.

<sup>30</sup> The UML ‘\*’ notation (many) corresponds to [0,n] in other modeling methods such as Merise.

<sup>31</sup> A mechanism which contributes to the agility of the system.

<sup>32</sup> The meaning of the dotted line between the Contract class (rectangle) and the association.

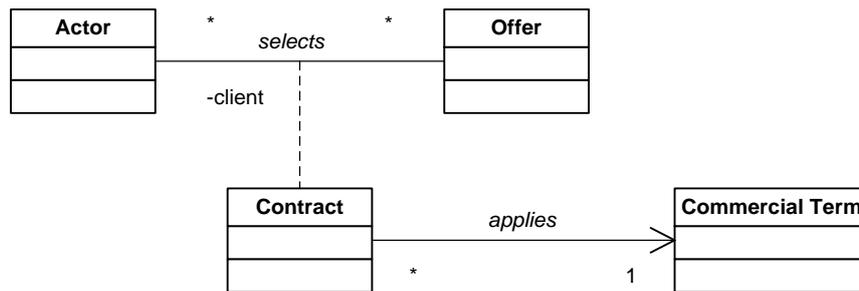


Figure 19. A customer selects an offer

An association class can be linked to other classes, a capability not offered by earlier methods (Merise, SA/SD...). In this example, a contract is subject to certain commercial terms.

In case the modeling exercise identifies structural or content differences between contracts across the various partners and parties, it will be easy to extend the model accordingly with sub-classes of the Contract class. Such extension will not impact what has already been captured around the concepts of actors and offers.

*The model “speaks” to us, it tells a story through the proper naming of associations and the roles on the associations. Moreover, association classes are leveraged to capture information on how concepts relate to each other.*

## Pricing

In legacy systems you often find pricing information spread across numerous tables (recently, although across three countries, I identified no less than 20 fields for this purpose).

Semantic modeling is reluctant to house such redundancy. Based on common sense, a term should appear once and only once in a model. If you see the same term used in two different settings, you are facing one of the two following options:

- Either the term means two different things, in which case the modeler must find another term for one of them;
- Or the term does indeed describe one single concept, and the model needs to be restructured to use and display only one instance of the term.

In our model, price will appear only once, like any other concept or information for that matter. Pricing occurs for the couple: offer and commercial term. This is captured by the *valorize* association.

Multiplicities indicate that one offer can get different pricings, depending on the commercial terms. Conversely, one set of commercial terms (e.g., *standard catalog*) can apply to several offers.

The number of n-ary associations and association classes is proportional to the model expressivity, one quality of a semantic model.

UML allows setting a direction on an association. However, such a feature is rarely used in semantic modeling because it restrains the model navigation. In our example, it would be easy to orient the association following the way we read the diagram; allowing navigation from Commercial Terms to Offer, but not the other way. Such a restriction usually has no semantics associated to it and would only limit the possibilities of a system derived from such a model. Such limitations actually fall into the scope of logical modeling, a subsequent step of the modeling activity chain.

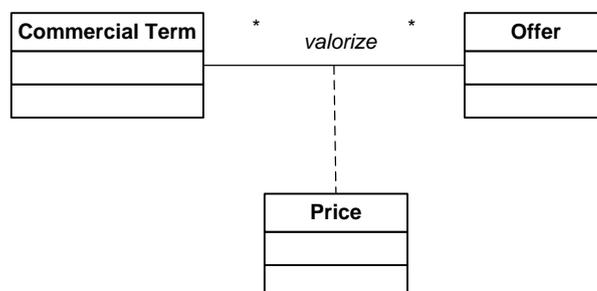


Figure 20. Only one way to price an offer

*Semantic models are characterized by two things: the fact that terms are used only once in a diagram and the conciseness of the model. With this approach, and as opposed to common ones, a term must only be used once.*

## Model generalization

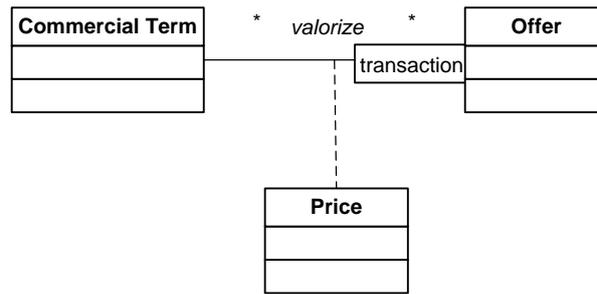


Figure 21. A qualified association

The only difference between Figure 5 and Figure 6 is the box on one branch of the association. In UML terms we call this box a *qualifier*. Qualifiers extend the expressiveness of the modeling language.

For instance, in our example, the *transaction* qualifier can be either *purchase* or *sale*. With that, for a given offer and based on the value of the qualifier, a set of commercial terms will lead to a certain price. This way, the model allows the storage of both the price of an offer and the one at which it was purchased, both for products and services, and splitting these prices in two distinct subsets. Suppliers can be characterized by the *Actor* class and its subclasses. Incidentally, this avoids duplicating information when an actor is both a supplier and a customer: it is recorded only once, with its specific role, as a natural person or a legal entity. The role of supplier or customer is then determined through its structural connections with the other objects.

Such a straightforward outcome of semantic modeling is the complete opposite of a functional approach. The latter would have led to two steps and two forms: “build the catalog (for sales)”, “manage procurement/supply”. Each of these two needs would have likely been addressed by two distinct projects and, unless through extreme precautions, have led to two different databases.

When executed upfront, semantic modeling prevents such divergence from happening and, subsequently, simplifies the derived systems. In addition, it standardizes terminology, glossaries, and representations.

*UML expressiveness is usually not used to its full potential. UML is actually a valuable instrument for semantic modeling and the formal representation of knowledge. Qualified associations in particular are an elegant way to express constraints and increase the applicability of models. Without much overhead and attention, semantic modeling prevents duplication and redundancy of functions; two common flaws of a functionalist design.*

## Utilization

A customer uses products and/or services. The use of customer in this context seems to violate the uniqueness rule for terms. However, and albeit informally, what we express here is the constraint preventing an actor to use a product or a service if he/she is not already a customer, with the appropriate contract in place.

Were the *uses* association to be binary (i.e., only between *Actor* and *Offer*), it would allow only one use, by one (actor, offer) pair. This would quickly lead the company to bankruptcy! To address this, the modeler needs to introduce a third term: *Transaction*, which corresponds to a date or a period... This relation, a ternary association, is represented by a diamond. It means that we do not only create pairs but triplets: an actor uses an offer through a transaction.

The use of multiplicity on ternary associations requires some explanations. From a graphical standpoint, it seems that a transaction could involve several offers and only one actor, but is it really the case? The question can be answered with a constraint (dotted arrow) to which the modeler will attach a note or a formal expression (using OCL<sup>33</sup> for instance).

<sup>33</sup> OCL – Object Constraint Language – is a formal language to express constraints on object-oriented models. It is an OMG standard.

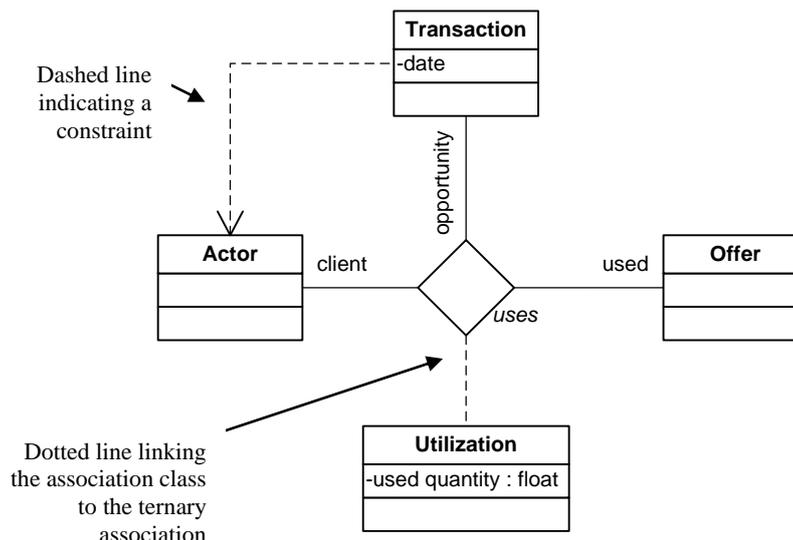


Figure 22. A ternary association

Speech and natural representation involve many concepts, themselves defined by other concepts. Such concepts are formalized with categorized associations, with as many branches as there are parameters. Semantic modeling highlights these parameters: they are essential for a correct understanding of the scope of the model. A less rigorous model would dilute these parameters into groups of binary associations.

## Summary

We briefly introduced the key concepts pertaining to modeling. This model is simple but provides the structure for the overall model. We can now compare it to the original Figure 1, which was the outcome of a common data-driven approach. To make our point, we only looked at the structural dimension of semantic modeling. Beyond additional or secondary classes, the complete model will include the definition and documentation of all the class attributes (information) and operations, as well as all the constraints.

Figure 23. The overall class diagram, result of our modeling exercise

## Additional considerations

The previous section triggers two follow-up discussions:

1. Semantic modeling does not end with a static description of the universe. It encompasses all the core business knowledge. Such knowledge includes information (data) and behavior (actions). It covers business rules which have a semantic nature. It excludes rules which depend on the organization or which are technical. The next section illustrates the dynamic dimension of the model, leveraging the concept of state machines.
2. Around this core or along this spine, the model will grow to capture more business specifics and details. It can grow from a dozen classes, as in our example, to one hundred or more. To address this scalability need, UML provides the concept of package. We will use this feature to structure objects into domains in a subsequent architecture diagram.

## State machines

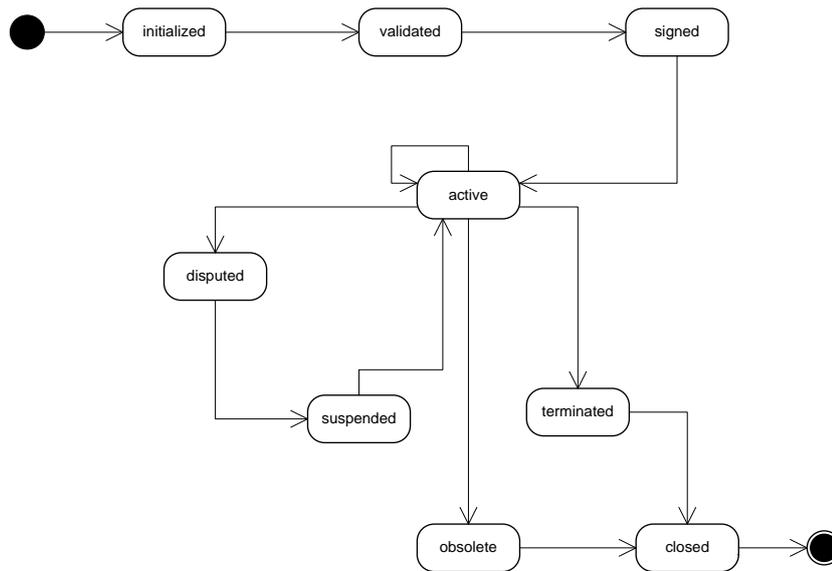


Figure 24. An example of an (informal) state diagram for the Contract class

The above diagram provides an example of an informal use of the concept of state machine. Object orientation aims at designing self-contained and highly coherent components. Concretely, objects are responsible for their state. Whatever action is triggered on an object, we know that the object will behave in accordance with its constraints. Of course, such a benefit is only realized thanks to a rigorous modeling exercise.

We are dealing here with the *contractual* dimension of semantic modeling (the other two are: *structural* modeling which we discussed in the first part of this article, and *functional* modeling, which describes the content of operations and processes). To capture this contractual dimension, UML provides a very powerful paradigm: the state machines represented as state diagrams. This paradigm is largely underutilized especially in non-technical IT applications (e.g., accounting).

State machines are required during modeling in two cases: firstly, when adjectives are used to qualify a concept, or secondly, when we mention milestones or states in an object's life cycle. The modeler first identifies the states which are usually named using adjectives applying to the name of the class. He/she then connects these states using transitions, respecting transition constraints. Leveraging the two dimensions of the diagram, instead of only a one-dimension sequence, the modeler will be more keen to take into account potential disruptions or loops in the object life cycle. Such ability will lead to a more fluid and richer model. Being better prepared to any unexpected event, the model will also be more robust.

Finally, to complete the model, the modeler analyzes the triggering and execution of the transitions. In so doing, he/she highlights events, conditions, and operations. This is one reason why non-technical IT applications have trouble properly identifying the operations at a conceptual level. The state machine is the right way to identify such operations.

*To describe transformations and identify operations, the semantic modeler leverages states machines. Such a new technique has a huge impact because it is a perfect way to take transformations into account. In common approaches, transformations are either ignored or pushed into procedures which then lead to a key source of complication.*

## Object domain decomposition

Decisions about the model structure have long-term consequences. Thus, the modeler must think twice and thoroughly before selecting a decomposition criterion. Because both theory and pragmatism have proven that functional domains lead to redundancy and inadequate coupling, we will not use them as a decomposition criterion, at least not to structure the semantic model or the business frame of reference. Praxeme recommends a structure based on *object domains*, at least for the system's core, for the business layer.

The following package diagram is a draft of the logical architecture. It is an unnecessary constraint in a semantic model because the relations restrict and limit the navigation. For instance, such formalism imposes to set directions on certain associations. However, as the model grows, modelers need this formalism to facilitate team work. The good news though is that such a diagram lays the foundations for the transformation of the semantic model into a logical model, for instance, in preparation for a service-oriented architecture.

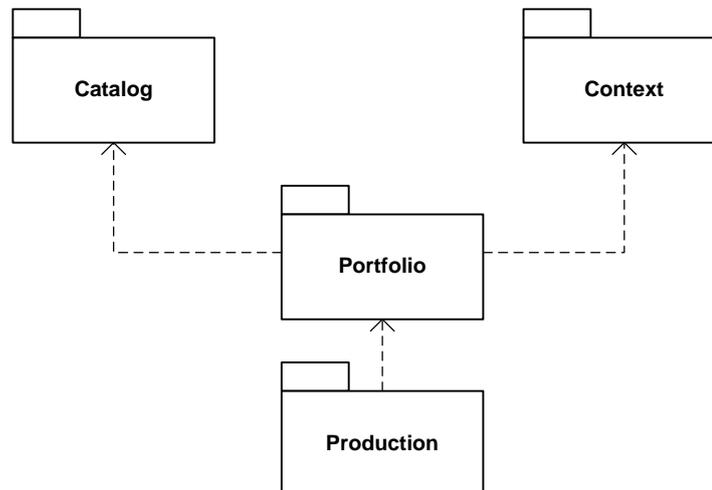


Figure 25. An example of an object domain decomposition

At the semantic level, the modeling scope gets organized into “object domains”. This leads the way to the logical architecture. By introducing object domains in addition to the traditional functional domains, Praxeme brings a radical change to the structure of IT systems.

## Conclusion

The modeling options and techniques highlighted in this article are customary for semantic modeling experts. We limited the scope of the article to the structural quality of the model. One should also consider:

- The definition of the modeling scope through a pre-modeling exercise,
- The encapsulation of business rules,
- The specification of operations at the semantic level,
- The design patterns,
- Etc.

Semantic modeling is a demanding discipline and exercise. It captures the core business concepts and the essence of the business in a way which can be leveraged by the downstream analysis and implementation activities. It translates target market definitions and strategic goals, it instills an innovative way to design processes, and it lays the groundwork for the design of rich and highly-reusable services.

Our ability to create real modeling skills will directly impact the success of tomorrow’s projects and the realization of our ambitions (e.g., IT convergence, agility, reuse, innovation). The inherent cultural change requires the support of the top management. All successful semantic modeling initiatives share one factor: an assertive leadership to mitigate potential resistance, to overcome the obstacles – cultural or psychological – and to valorize the real impact and benefit for the enterprise.

▲