Semantic Aspect Approach

Subject	**"Product" dimension**
Purpose of the Guide	Define the semantic aspect of the Enterprise System and describe its content. The semantic model formally describes the business knowledge. It plays a key role in enterprise transformation.
Key words	semantic aspect, business, knowledge, modeling, Product, UML

Reference	PxPRD-20	*Status*	Proofreading in progress
Version	1.0.0	*Date*	18 August 2013
Authors, contributors	Dominique VAUQUIER	*Translator*	Joanne TOWARD
Proofreader	Thierry BIARD		

Summary

Methodological reminders

The Praxeme methodological corpus is composed of:

- Guides, which provide the fundamentals of a subject or a domain;
- Procedures and methods, defined as "ways of doing something, operational modes to execute a task"[1];
- Processes, which describe the sequence of activities and the organizational measures that accompany them.

Outside of the method, the Praxeme open corpus contains models and pedagogic material.

To follow the developments of the open method

- Mailing list
- LinkedIn Group
- Twitter
- the wiki

To participate in the work of the Praxeme Institute

- Become a member of the Praxeme Institute

http://wiki.praxeme.org/index.php?n=Chorus.Join

Document protection

The initiative for an open method rests on voluntary work and the pooling of investments between contributors. It aims to develop and disseminate an open, royalty-free method. Its dynamics only works if this spirit is maintained in the way the documents, which have been made available to the public, are used. This is why the documents are protected with a "creative commons"[2] license, which authorizes the use or reuse of all or part of a document from the Praxeme corpus, the only condition being that the source is quoted. The same conditions should also apply to any documents likely to be derived from t. They must refer to the "creative commons" and feature the appropriate symbols:

Updates to this document

To obtain the latest version of this document, please go to the Praxeme Institute wiki, page: http://wiki.praxeme.org/index.php?n=Modus.PRD20Semantic.

Document history

Index	Date	Author	Content
0.1.0	30/04/2013	DVAU	First draft
0.2.0	31/05/2013	DVAU	Integrated feedback from Thierry BIARD
1.0.0	16/08/2013	JT	Translation by Joanne TOWARD, review by DVAU
1.0.0	18/08/2013		Current version of the document

[1] Cf. Thesaurus section on the Praxeme Institute website: http://wiki.praxeme.org/index.php?n=Thesaurus.Procedure.

[2] See the philosophy and license detail at: http://creativecommons.org/.

1. Introduction

1.1 Definition of the semantic aspect

We see the enterprise as a complex system. Such a system can only be approached through several aspects, both distinct and carefully articulated. The Enterprise System Topology is the conceptual framework that Praxeme proposes to approach the reality of the enterprise.[3].

The semantic aspect isolates the knowledge of the business fundamentals.

By "business fundamentals", we mean the essential knowledge that enables an enterprise to act in its environment, knowledge that is free from any reference to the organization and means implemented.

The content of this knowledge, therefore, has an outward focus: it covers the environment of the system and the interactions between the system and its environment. This point provides a delimitation criterion that semantic modeling will turn to and adopt.

The essential knowledge is expressed across concepts such as: product, person, physical object, target, contract, event… These notions are independent of the internal mode of organization and equipment choices. They build the perception the enterprise has of its environment as well as the interactions that it seeks to develop (its "value proposition").

In contrast, the enterprise processes, its organization, the means it has at its disposal… do not belong in the semantic aspect. The quality of the semantic model depends, in part, on the fact that these elements have been removed from it.

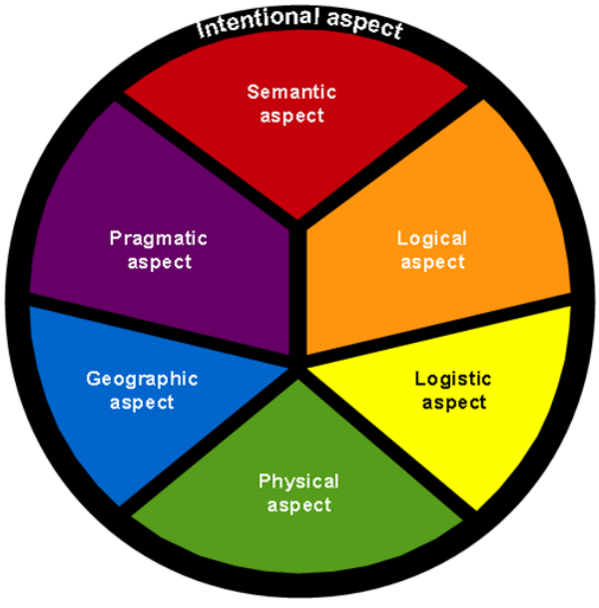
The semantic aspect is the focus of one or several semantic models.

A semantic model expresses the business knowledge, independently of how the activity is conducted.

Such a model covers the business essentials, the minimum upon which agreement is needed for us to understand the business of the enterprise and its required behavior in its environment.

Figure PxPRD-20_1. The Enterprise System Topology

1.2 Positioning of the semantic aspect



The definition and the content of an aspect only appear in their position within the framework applied to the Enterprise System. This position determines not only the outline of the analysis and the design of an aspect, but also the decisions made in the modeling details.

a. Upstream: the intentional aspect

As regards the order of the aspects in the Enterprise System Topology, the semantic aspect is second to the intentional aspect[4] to which it refers, and is where we find:

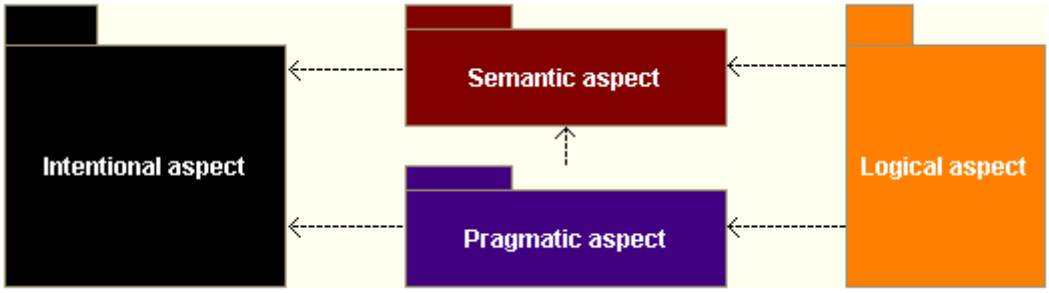
- the enterprise values that the semantic model must conform to;
- the enterprise objectives and requirements, to which the semantic aspect contributes in part;
- the metrics, some of which are outlined in the semantic model;
- the vocabulary, which provides part of the definitions for the elements of the semantic model.

[3] See the "Enterprise System Topology" guide, ref. PxPRD-01.

[4] See the "Intentional aspect approach" guide, ref. PxPRD-10.

In this way, the intentional aspect is one of the sources of semantic modeling. This situation gives rise to the procedure by which semantic modeling begins: "Identify and class the objects and concepts"[5].

Figure PxPRD-20_2. The neighborhood of the semantic aspect



b. Downstream: the pragmatic aspect

The semantic aspect is "above" the pragmatic aspect, one abstraction level higher. This means that we can describe part of the business and the reality of the enterprise, ignoring its organization and its activity. This organization and the business processes can, on the contrary, be described by referring to the more fundamental semantic aspect. Based on this feature, Praxeme proposes an innovative process design procedure[6].

c. Downstream: the logical aspect

The Enterprise System Topology inserts an intermediary aspect, the logical aspect, between the business and its logistic resources. The semantic aspect is thus in contact with the logical aspect. From here come the derivation rules that guide the logical design. For example, the semantic model is the starting point for the identification and detailed design of the services at the heart of the information system[7].

1.3 Stakes of the semantic approach

By virtue of its position in the framework and the abstraction effort it requires, the semantic aspect plays a determining role in understanding and transforming the enterprise.

Running counter to an activity-based approach of the enterprise, semantic modeling seeks the essentials, what we must know without fail to ensure the running of the enterprise, whatever its choice of organization or equipment.

The results of semantic modeling go from accumulating knowledge to transforming the enterprise.

a. Facilitate the understanding of the business

Adding to the semantic repository enables you to accumulate knowledge on the domain. As such, it is a key instrument in preserving the intellectual heritage of the enterprise. It can also serve as a training tool. It provides the starting point for knowledge management.

b. Simplify the business

The abstraction effort enables you to uncover the essentials, to go beyond the diversity of practices and to clear the way for a process of simplification to begin.

The simplicity of this description frees the imagination and later enables the designer to have a wider choice regarding the organization, logistics and technologies.

---

[5] See the procedure data sheet PxPCD-22.

[6] See the procedure data sheet "Innovating with processes", PxPCD-33.

[7] These rules of passage or derivation rules are discussed in the logical aspect data sheets (PxPCD-5#). See also chapter 4 of the present document, paragraph "Derivation", p. 26.

c. Re-examine the business

The semantic aspect is also the place for a radical rethink about the business. It can absorb part of the strategic directions, specifically those that lead to a redefinition of the business and its content. Indeed, the semantic model with its focus on the essentials, is a support for conceptual innovation[8].

d. Share and unify knowledge

Strengthened by the object-oriented approach recommended by Praxeme, the modeling effort pushes for a generic semantic model. As this model goes beyond what is happening internally in the enterprise, its vocation is to state the universal. While ways of working will always differ from one enterprise to another, business notions and representations of the reality can, on the other hand, be more easily shared. A sufficiently stable and generic semantic model can be shared between several organizations. It normalizes the terminology and formalizes the business fundamentals in which partners will be able to recognize themselves.

Consequently, the semantic model forms an essential basis to support a network or a federation of enterprises. This stake comes into its own not only in partnership relations, mergers and acquisitions, but also in the coexistence of several companies or management teams inside a group or within the civil service.

1.4 Purpose and content of this guide

This document describes the semantic aspect, its content, its rules, as well as how to represent it. As a guide in the "Product" dimension, it is only concerned with the substance of the aspect and not the way to approach it. It introduces the procedures and methods, described elsewhere.

The aspect is first dealt with as such: the "Content of the semantic aspect" chapter analyzes its substance and fixes the categories in which we can find this aspect.

The "Description of the semantic aspect" chapter introduces the representation techniques that are adapted to this aspect.

Finally, the last chapter lists the uses of semantic models.

2. Content of the semantic aspect

Any approach of the reality supposes a theory of knowledge. Such a theory provides us with the intellectual tools and operational concepts that shape both our perception and knowledge, and casts them into particular forms – "in-forms" them. More often than not, this theory of knowledge is mainly unconscious, carried by language and rooted in the metaphysics that underlies culture. The modeler gains by elucidating this theory and objectifying these tools. Quite a lot of delicate questions find easier solutions or sometimes disappear by their own accord when we uncover these knowledge mechanisms. This subject is tackled in this chapter.

2.1 Business fundamentals

a. Definition and delimitation

The term "fundamentals" has, here, a very precise meaning which can be better understood by contrasting it with everything that it excludes: the organization, the employee activities and the solutions and tools at their disposal. The criterion for deciding whether one element of knowledge can be considered as "fundamental" is: do we change the business if we change this element? "Business" refers, here, to the business of the enterprise, its production, its value proposition.

Changing a "business" process, modifying the organization chart, overhauling the information system… does not fundamentally change the "business" of the enterprise. These actions help the enterprise to do its job better. Significantly changing a semantic model does, on the contrary, lead to the business description being modified, by taking an interest in other elements of the surrounding reality or by changing the catalog or quality of that proposed by the enterprise externally.

[8] An example is given further on, regarding customer focus (p. 25).

For the modeler, the boundary between the semantic aspect (conceptual) and the pragmatic aspect (organizational) must be absolutely clear or, in other words, between the "Business Objects" sphere and that of the "Business Activities". The quality of the modeling and the level of expected benefits are at stake.

b. How the semantic aspect is viewed

The semantic aspect is defined in terms of knowledge. This is what we aim to reach, isolate and preserve, like the foundations upon which the edifice of the enterprise rests. The question is: knowledge of what? Knowledge is in the human mind… where it coexists with all manner of other things that will not be retained in our semantic models.

A way out would consist in saying that we are, first and foremost, seeking to know the objects of the reality, the real objects. In many cases, this precept helps us. For example, when we model physical or socio-technical systems (transport systems, armament systems, energy production…), the primary objects obviously stand out. However, even in these cases, our knowledge is built around concepts. On the one hand, there are concepts built as sets of objects, which enable us to regroup and arrange them; on the other hand, there are derived or abstract objects which describe the reality well, but which we can no longer associate with physical objects (for example, a production log). Moreover, in certain sectors of activity, the everyday business no longer has any connection with physical objects but with abstractions, resulting from a process of representation and convention: contracts, accounts, guarantees, rights…

Nonetheless, it is customary to speak of objects in all these cases. The representation techniques presented below respect this use and even take it literally as they turn to and adopt the object-oriented approach. To reflect further on this, we would have to delve deeper into the theory of knowledge. This is not our aim. We are simply looking to clarify the nature of the semantic aspect and the criteria that will help the modeler decide if an element should or should not belong to it.

c. Objects of a semantic nature

It is precisely because we are on the fundamentals that this question is difficult: what is an object of knowledge? Another question follows: How can we identify *all* the objects of knowledge?

It is easier to approach these questions with examples and counterexamples:

- In a transport system, to consider the train as an object of a semantic nature seems to be self-evident. However, if we introduce it into the semantic model, we will lock down the thought process and assign the enterprise to reproduce its destiny: what we have written in stone in the semantic model, is only one solution to a need for mobility. The semantic modeler will prefer the term "vehicle" or even "mobile" to that of "train", paving the way for multimodal offers.
- In insurance, the contract is a central object, well situated in the semantic model; the same with the claim. But what about the "claim file" so dear to the specialists? As soon as we speak about files, there is a strong likelihood that we will have moved to the pragmatic aspect. How can we be sure? The business expert will swear up and down that the "claim file" is a central element in his/her job, of course. But, does this expression *speak* to the customer? Absolutely not. The customer understands well "contract", "guarantee", "claim", but "claim file" is not something he/she perceives, unless forced to enter into the specialist's domain. This is one test that can be used to remove the elements, which come from work habits, from the semantic aspect!
- At a more detailed level, the semantic model retains action properties such as "declare", "evaluate", but will not keep terms like "save" or "delete", if they only refer to database manipulations. Such elements will come in due course: they belong to other aspects. It would be to weigh down the semantic model and reduce its power of expression if we were to include them.
- Apparently obvious notions, such as invoice or customer, are more delicate to handle. If we include the notion of customer in the semantic model (with information like name, age, status, address, etc.), we will be heading towards serious difficulties. The same person can be known in the enterprise as an employee and outside as a policyholder, beneficiary, user, etc. Thus, the model is no longer shareable: the same element of reality is known under different representation terms, ruining all hope of sharing and interoperability.

Conclusion: customer is not a semantic notion, at least not as a central concept; the notion of person is preferable: the world is populated by people, some of whom assume the role of customer[9].

These considerations are not mere quibbles of purist modelers: they have practical and considerable economic consequences because the capability of organizations to harmonize their perception hinges on them.

In conclusion, the semantic aspect retains only the essentials of the business, beyond any variations in actual practice.

d. Characteristics of the semantic aspect

Providing that this spirit is respected, the semantic model presents characteristics that will play a vital role in transforming the enterprise:

- **stability**, because the semantic aspect is upstream from variations linked to the organization, practices and tools;
- **compactness**, because only the fundamental business concepts are retained in the model and expressed by applying a principle of economy;
- **universality**, because the focus is on the reality and we seek the most general and shareable expressions (as in the example about the person).

The semantic aspect is undoubtedly not completely protected from variations: it is subject to changing regulations and may be the basis of conceptual innovation. On the one hand, these changes remain exceptional. On the other hand, we must take care to isolate the points of variation and to find an ad hoc process for them[10].

The compactness of the semantic model results from the application of modeling precepts. It often appears as a drawback as it leads to considerable difficulty in communicating the model[11]. It does, however, represent, an advantage: on the one hand, redundancy has been removed from the model; on the other hand, the more compact the model, the far greater the scope. This characteristic has inestimable economic consequences.

These three characteristics are not independent: they reinforce each other. Seeking universality – for example, for interoperability ends – pushes you to remove complications, often of a non-semantic nature, and to retain generic terms. This increases stability and compactness. The more compact the model – that is to say able to say a great deal with a minimum of terms – the more stable and universal it will be.

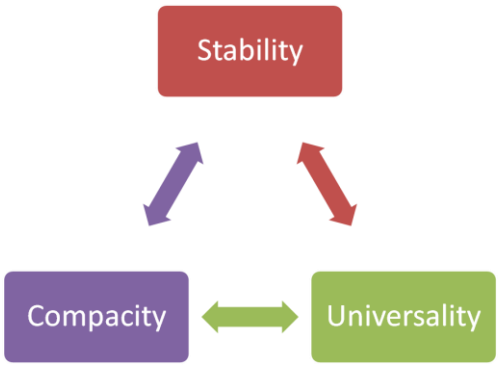


Figure PxPRD-20_3. The characteristics of the semantic aspect

2.2 Categories of representation

In accordance with the triangulation principle[12], the model must analyze the semantic aspect in all its dimensions:

- structural (what is it made of? the *to be*, the substance),
- functional (what does it do there? the *to do*),
- contractual (how do its elements and the system overall behave? the *to become*).

[9] The question is not one of vocabulary (instead of "Person", we could find "Party"…) but of concept: the concept of person cannot carry a "reference number" attribute; it eclipses the concepts of employee, customer, partner, etc. (in fact, roles). Its presence requires the whole of the model to be rearranged on the scale of the enterprise.

[10] A good example is that of the pricing rules in a catalog. See the guide "The agility of the Enterprise System", PxPRD-03, and procedure data sheet "Isolating the points of variation", PxPCD-03.

[11] Cf. chapter 4 of this document.

[12] See the guide "General rules of architecture and modeling", ref. PxPRD-02.

In what terms does the modeler express the semantic aspect? What units does he/she use to describe the "universe of discourse", the application domain and the reality? The following sections answer these questions.

These "terms" are syntactic categories of modeling or categories of representation. For semantic modeling, whose aim is to fix knowledge, we look for a minimal syntactic tool, one that is as close as possible to the grammar of natural language and able to build conceptual systems. In the semantic aspect approach, the method is wary of any external constraint that may distort the approach of the reality[13].

We have to be able to express:

- the notions, concepts and objects of the domain studied, which make up the knowledge of the Enterprise System,
- the information contained in these concepts and objects,
- the behavior that they are capable of,
- the relations that connect them in an efficient network of significance,
- the rules that constrain them.

a. Semantic class: concept and object

The class is the basic unit of semantic modeling. To avoid all ambiguity, we will call it the "semantic class".

The class is a "set of knowledge objects brought together through the presence of common characteristics, corresponding to a concept or a notion"[14].

Semantic modeling consists in identifying the objects and concepts that come under the domain studied, pinpointing their common characteristics and organizing them in the most economic way possible.

The semantic class enables all the semantics connected to a real object, a set of similar objects or a concept to be restored.

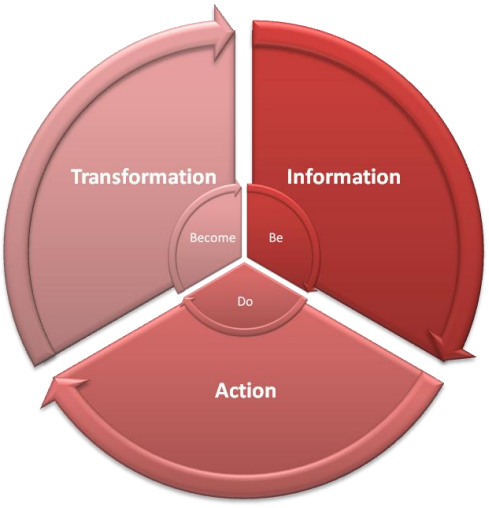
Candidate classes are classes that the modeler is considering including in the model, to capture part of the meaning. They often correspond to a substantive used in speech.

b. Class properties

Figure PxPRD-20_4. The three types of properties of the semantic class

The class contains the properties that are common to all the objects it regroups. We can distinguish:

- the informative properties: a concept or set of objects are known through the information that can be specified for each object or that applies to the set[15]; this information is applicable to each instance or all instances of the class;
- the active properties: certain concepts or certain objects can act or undergo actions; in some cases, these actions are sufficiently linked to the concept that they can be included in the semantic class that is modeling them[16];
- the transformative properties: objects change (their form, their informational content, their action capability… change), but these changes obey certain rules, they can happen under certain conditions and the actions can be affected by this; it is essential that the model captures these phenomena, which can be the source of complexity and value.



[13] This will not be the case with the other aspects, particularly from the logical level.

[14] Cf. *Le Grand Robert* French language dictionary, publication led by Alain Rey.

[15] For example, information about the car color and total number of cars is connected to the concept of car. These informative properties are part of the semantics of the Car class.

[16] Let us take the example of the projection or development of a geometric figure.

These properties are included in the semantic class.

Added to this are the structural properties, discussed in the following section. They are similar to informative properties but they involve other classes.

c. Relations

The relation of the semantic class – the concept – to a specific object is the instantiation. The object can also be referred to as an "instance of the class". This relation can be seen as one of an object as a class member, class being taken as a set of objects that come under the concept. We can also describe the class as being the "mold" which makes the objects or the representations of objects, according to whether they are abstract objects or physical objects[17].

The inverse of the object to class relation is subsumption. This is the modeler's decision to "see" or represent one part of the reality through a given concept.

Classes are classified according to a generalization / specialization logic. The properties are distributed on the class hierarchy. Specific classes (the "child/subclass") share properties with generic classes (the "parent/superclass") and have characteristic properties (distinctive features).

The objects have links between themselves. These links are modeled as associations between classes. The associations express connections between concepts.

d. Object domain

When the field of study is fairly broad and rich, it can only be described through several tens, or even hundreds or thousands of classes. It is then necessary to structure the model. The criterion used to decompose the semantic aspect has serious consequences. The modeler regroups the semantic classes in coherent sets, that we will call "object domains" in order to emphasize the type of criterion to be used.

It is important to note that the object domains do not actually belong to the reality observed; they result from the modeler's decision to structure the knowledge.

Decomposing the semantic aspect reveals dependencies between the domains, which constrain the movement between the objects arranged in this way. It therefore introduces constraints which were not necessarily present in the knowledge, even less so in the reality. We can go from one concept to another and back again, with no particular limit. From the moment that the semantic aspect is divided into object domains, additional structural constraints are imposed on the model itself. So it is an act of architecture and one that should be weighed up with caution[18].

e. Object lifecycle

Part of the complexity comes from changes in object behavior. These changes can be analyzed in terms of the objects' internal states. Therefore, it is advisable to isolate these changes and to represent them as such.

The object lifecycle is all of its states and possible connections between these states. A state is defined as a configuration of the object's properties in the value domains, a configuration within which the object behaves in a precise manner. For example, in one particular state, the object will be able to do a particular thing or respond to a particular stimulus, whereas in other states it would not be able to.

From any given state, the object can go to another state: this is known as a transition.

The lifecycle is connected to the semantic class. It expresses the transformative properties of the class.

f. Event

The event (or signal) is a unit of exchange within a system. The event tells us that something has been produced.

---

[17] In the case of an abstract object or a conventional object like a contract, the class instance containing the information is no less real than the paper on which the information is printed. The reality of the contract is not in a written document but in the agreement and institutional conditions that guarantee its enforcement. On this issue, see in particular, *The construction of social life*, by John Searle.

[18] The decomposition of the semantic aspect is the subject of procedure PxPCD-25.

Objects can communicate among themselves via events, for example to let the others know about their changing states.

The event is also, and especially, for the Enterprise System, the way in which it can inform itself about its environment. It is the unprocessed form by which it receives information.

g. Rule and constraint

In addition to the lifecycle that constrains the transformation and behavior of objects, constraints of all kinds can affect objects.

A constraint is a proposition formulated in other terms of the aspect, completed by logical operators, which limit the configuration or behavior of an object or set of objects.

The term "rule" is synonymous with constraint.

We must be careful to only include in the semantic model the rules that depend solely on concepts, excluding rules linked to the organization or other choices. We refer to them as semantic rules or conceptual rules[19].

A semantic (or conceptual) rule is never the result of a choice: it is imposed on the model, either because it logically belongs to the concept or because it comes from external regulations[20].

A constraint always refers to one or more elements from previous categories.

We can distinguish:

- the structural constraints, which are concerned with values of the information and the arrangement of objects;
- the functional constraints, which limit actions and their links;
- the transformational constraints, which control the transformations of objects and sets of objects.

Constraints are concerned either with one object or a group of objects. In either case, they form the essential elements of the semantic reality.

2.3 Metamodel for the semantic aspect

a. Summary of the categories of representation

The categories of representation described above enable the whole semantic aspect to be covered, in the three dimensions of modeling: be, do and become[21].

In the figure below, the outside circle represents sets of objects[22]. Several categories enable us to comprehend these sets: object domains, relations and events.

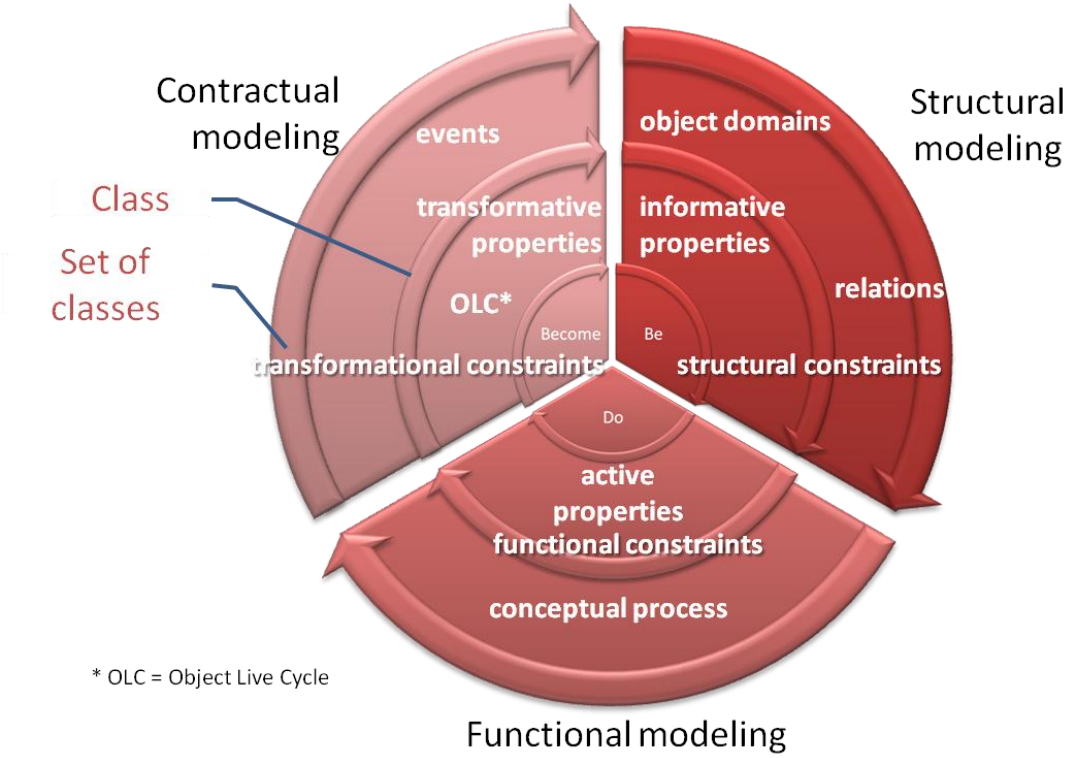
[19] The label "business rule" is marred by the ambiguity specific to the term "business". Indeed, because the Praxeme framework recognizes several aspects that contribute to the business description (semantic, pragmatic and geographic), a "business" rule requires qualification: it can be conceptual or organizational. According to its type, it will be directed towards the semantic aspect or the pragmatic aspect. There are also other types of rules, bound for other aspects.

[20] In the first case, an example is reaching maturity at 18, authorizing certain behaviors. In the second, the VAT rate to be applied depends on a categorization fixed by the legislature.

[21] Cf. PxPRD-02.

[22] An example of a set concept is a catalog. It must not be represented as a specific class. The Product or Offer class, in its extended form, in its set dimension, expresses the notion of catalog. The properties of the catalog are the properties of the class scope, that is to say the properties that take their value for the overall class and not for each of its individual instances.

Figure PxPRD-20_5. Triangulation of the semantic aspect



The notion of conceptual process must be handled with care. It will only be discussed below when we have extracted the "internal rule" of the semantic aspect (see p. 17).

The intermediary circle circumscribes the categories connected to the semantic class: the properties, including the object lifecycle (OLC) that assembles the states and transitions.

The constraints concern both these levels: class and set classes.

b. Relations between the categories of representation

The semantic class is the basic unit of semantic modeling. It shapes the concept of knowledge.

The properties, relation and lifecycle (state and transition) categories are connected to the class.

The object domain is a set class.

In the semantic aspect, events are emitted or received by a class property.

Constraints are connected to one or several elements of the previous categories.

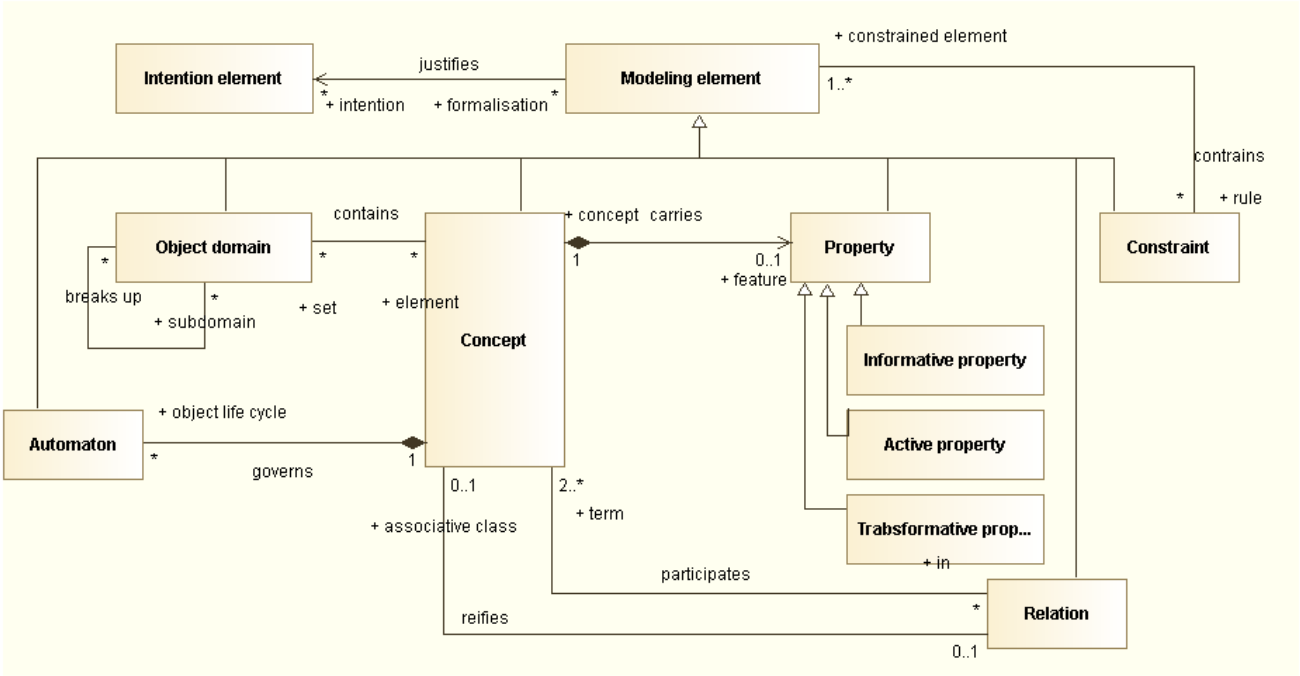
c. Metamodel

The class diagram presented below represents only the surface of the Praxeme metamodel. Indeed, missing from it are details about the properties and references to the underlying metamodel. The Praxeme metamodel is based on the UML (Unified Modeling Language) metamodel. On the one hand, this means we avoid having to reformulate a lot of the details (for example: detail about building a state machine or description of properties). On the other hand, a practical implication is that it is easy to tool the Praxeme approach: its metamodel is easily translated as a UML profile.

It is important to note that several of the categories presented here are not specific to the semantic aspect. This remark conditions the structure of the metamodel. We can notably find classes and state machines in several other aspects of the Enterprise System. We will not deal with this question here either[23].

[23] The Praxeme metamodel is presented in the document PxMDS-05.

Figure PxPRD-20_6. Diagram presenting the categories of the semantic aspect with their relations



Comment

The root "Modeling element" covers all the modeling elements in all the aspects. The mechanism presented here is typical of the enterprise methodology. It summarizes the particular position of the intentional aspect, whose root "Intention element" controls all the elements that guide or justify the modeling choices.

The constraint is a modeling element that can therefore refer back to one or several intention elements. It is a requirement of the model that a constraint be necessarily connected to at least one modeling element. A constraint of the metamodel (not represented graphically) excludes that one constraint can be connected to another.

Chapter 3 gives the correspondences between the categories of the semantic aspect and the types of elements defined by UML.

2.4 Internal rule of the semantic aspect

The categories of representation supply the pieces for a game of chess with their usage rules. All that remains is to define the general operating rules – the aim of the game. Indeed, applying these categories is not enough to guarantee the quality of the semantic approach. Overall principles are required to fix the logic of this approach[24].

We use the term "principle" in the strongest sense of the word: "prescription that cannot be deduced from other elements of the domain studied"[25]. In methodology, a principle is therefore in the same position as an axiom in a formal system.

To guide the semantic approach, the following principles are required:

- abstraction principle;
- encapsulation principle;
- factorization principle;
- cooperation principle.

[24] Whether these principles are specific to the semantic aspect, or linked to the chosen categories and techniques for modeling the knowledge, is a theoretical question that we will not examine here. What is certain, though, is that it is only by applying these principles that the practitioner will be able to correctly model the semantic aspect.

[25] See the definition and the discussion in the Thesaurus section of the website: http://wiki.praxeme.org.

These four principles complete the list of the categories of representation that define the logic of the semantic approach. This first enables the semantic matter to be circumscribed; the second encourages you to take advantage of the categories of representation; the third specifies the parsimony principle (or Ockham's razor) that applies to both modeler and scientist; the last one helps you to apprehend the system dimension, beyond the unit represented by the class[26].

a. Abstraction principle

This principle teaches you to "disregard" (abstraction) any non-conceptual element.

The objects and concepts at the heart of the activity are modeled for what they are, disregarding organizational or technical circumstances.

The "core of the business" is very stable. It changes only when the enterprise modifies its mission, its product or service offerings, or when external elements (legislation, regulations) force it to adapt. In the case of physical systems, the stability of the description is even more self-evident as we position ourselves on the concept plane, upstream of the solutions. It is therefore in our interest to protect this aspect from the introduction of decisions that are more likely to change.

b. Encapsulation principle

Classes are the basic units of the semantic aspect. All other categories are connected to them. This is obvious with properties and also with the set categories of object domain and conceptual process. There are two remaining categories to deal with: constraint and event. The instances of these categories can appear and define themselves in an independent manner, as independent texts, as we generally do with "business rules". The encapsulation principle stipulates that the elements of these categories must be connected to at least one class.

All design constraints are incorporated in the classes.

Applying this principle may lead the modeler to modify the structure.

The notion of constraint is broadened to that of proposition. For example, a calculation formula (a person's age, an indicator…) or a definition, are propositions on one or some objects. These propositions must not remain hanging in thin air, floating within the semantic aspect. Following the same encapsulation principle, they must also be connected to a class, as an ad hoc property.

All events emitted by the system studied are done so from a class.

As for events received from the outside, this rule does not apply. As mentioned above, the event is the means for the system to inform itself about its environment[27].

c. Factorization principle

The abstraction principle enables us to select the material that pertains to the semantic aspect and to exclude anything that may risk polluting it. The encapsulation principle leads us to fix the elements of knowledge around the basic units that the semantic classes constitute. We still need one more principle so as to reduce the semantic expression, a guide to obtaining "well formed" models. This is the factorization principle. It is a principle of economy according to which:

A same element of meaning must only be formulated once in the model.

---

[26] These principles are not to be confused with modeling precepts, even if they can inspire them. The precepts appear in the Procedures dimension, whereas the principles condition the material, in the Product dimension. Nor are they to be confused with the modeling or architecture choices or decisions.

[27] In object logic, encapsulation also refers to the fact of hiding the realization details behind the properties. From outside the class, we can only see the properties displayed, not their content. This notion intervenes in semantic modeling, although it cannot be formulated in these terms. There is not, strictly speaking, any realization within the semantic aspect: its material is the knowledge. To take an example, surface is a property of the Geometric figure class. The way of calculating the figure is internal to the property. The requester does not need to know the formula. It is encapsulated. To be even more rigorous, we must ensure that the requester does not have to consider how to obtain the surface: for him/her, it is just a piece of information, regardless of whether it is available immediately or calculated. The encapsulation principle is extended by the "uniform reference principle" formulated by Bertrand Meyer.

This principle will inspire several precepts or operating rules, notably in the terms chosen in the model, in its structuring and the decision to resort to particular modeling techniques like polymorphism[28].

d. Cooperation principle

Finally, we also need a guide to tackle, beyond the class, the large sets and overall behaviors of a system. This principle is provided by the cooperation of objects:

A behavior (a function) of a system results from the cooperation between the objects that it comprises.

This principle can be read negatively, banning the writing of long accounts, describing how something is processed or how it functions with long action sequences. It is quite the reverse. The actions are distributed on the classes, respecting their semantics. The behavior of the system is achieved by letting the objects "live" and extend their activity.

The cooperation principle is, without a doubt, the most troubling of object logic. However, it reveals the power of object logic in approaching complexity.

2.5 External determination of the semantic aspect

As mentioned in the abstraction principle: the semantic aspect is completely independent of organizational and technical circumstances. A model that would take on board elements from these subsequent aspects would simply be a bad model. It would limit the ability to design new solutions, on all levels.

What, then, determines the semantic aspect? Where does its content come from? What influences are exerted on it?

a. Determination by the reality

First, the semantic aspect is the seat of fundamental knowledge of the Enterprise System. This knowledge is primarily concerned with the reality that surrounds the enterprise. It is not too strong a word: it really is about the reality, a reality that is in our best interests to grasp and comprehend as is, as far as possible. This remark has important consequences: the semantic aspect must not be expressed using internal vocabulary terms, but ones from the external reality.

Obvious is it not? Not so sure. Let us take an emblematic example: the client. There is a temptation to include, in the semantic model, a Client class (or User, Beneficiary, etc.), and even to give it a central place as – and everyone agrees – it is an essential notion for the business. However, it is precisely what one must avoid doing. Client is not a direct notion, extracted from the reality, but a relative notion: our way of seeing a person. Person is the true notion. The discussion is not limited to the choice of word. It is the whole structure of the model that risks being affected. For example, a Client class would carry the client number, which has no meaning outside of the enterprise. Consequently, the model would no longer be shareable.

This is why it is important to insist on determining the semantics by the reality. As much as this advice seems obvious, human psychology and culture conspire to distract us from this fact. The modeler must be aware of this. The universality of semantic models depends on it, as does our ability to share them and ensure the interoperability of our systems.

b. Determination by the intentional aspect

The Enterprise System Topology allows for the semantic aspect to be dependent on the intentional aspect. It makes the references from the modeling elements possible towards:

- the enterprise values;
- the objectives and requirements;
- the metrics (indicators);
- the vocabulary.

The procedures detail how these sources are utilized[29].

[28] See chapter three, the paragraph on classification.

c. Determination by the environment

Part of the environment impacts on the Enterprise System through its intentional aspect. However, sometimes the semantic aspect of interacting systems is in direct contact with the semantic aspect of the system studied. This is, incidentally, a condition for ensuring the interoperability between partner systems.

In that case, there are two options available:

- The opposing systems can communicate part of their semantics through events.
- Parts of their semantic aspect are shared.

These options are detailed in the multisystem approach[30].

Finally, we should also mention the generic models, which can act as a starting point to model the semantic aspect.

3. Description of the semantic aspect

The previous chapter talked about the semantic aspect, its nature and its content. Of course, part of our perception finds its way into this reality[31]. But we have taken care only to present the knowledge tools – chiefly, the concept – and to avoid imposing, from the outset, the choices of representation. This chapter will discuss these choices.

3.1 Choice of the notation

We are looking, therefore, for a representation technique that covers all the syntactic categories listed above and that articulates them in accordance with the relations that we have identified. The notation must be sufficiently formal to help us:

- check the syntactic conformity;
- produce "well formed" expressions of knowledge with a good coverage rate.

Formalisms such as those of the entity-relations methods, ontologies, terminology or less formal approaches must be ruled out. Such formalisms can be useful in the approach of other aspects, but they have proved to be insufficient for the semantic aspect. Indeed, they only cover one part of the categories of representation.

The notation currently retained by Praxeme, to model the semantic aspect, is UML (Unified Modeling Language) from the OMG (Object Management Group). The reasons are as follows[32]:

- UML gives an expression to each of the categories of representation of the semantic aspect. If we are not averse to using the levels of sophistication it has, its power of expression is perfectly suited to formalizing the knowledge.
- UML enables correct coverage of other aspects of the Enterprise System, which allows the semantic elements to be linked with elements in the contiguous aspects, thus implementing an MDA (Model Driven Architecture) approach.
- The level of formalism sought is guaranteed by the UML metamodel.
- The notation is widely used and sufficiently tooled.

3.2 Correspondence between the categories of representation and types of elements

UML proposes a large quantity of types of modeling elements, all rigorously defined in its metamodel. This section shows how the semantic categories are translated into UML terms.

---

[29] The procedures of the intentional aspect implement the "projection" of these elements towards the other aspects. The procedure "Identify and class objects and concepts" (ref. PxPCD-22) explains how to exploit these sources.

[30] Ref. PxPCD-07.

[31] What is within us always escapes our knowledge.

[32] The PxPCD-02 document on notations provides a more detailed argument.

a. Semantic class

The semantic class, naturally, is represented as a UML class. Even if this language was designed for software modeling, the notion of class is, nevertheless, far earlier than software engineering itself. The class, as described in UML and more widely in the object-oriented approach, corresponds perfectly to the concept of knowledge and enables it to be modeled in a pertinent manner.

b. Informative properties

The informative properties are translated either as attributes – in the UML sense – or as operations. In most cases, the attribute will suffice (for example: date of birth). Fairly frequently, the modeler will turn to more sophisticated forms:

- the calculated attribute when the information depends on other information or when it is the result of a calculation or a search (for example: age);
- the class scope attribute, when the information applies to a set of instances of the class (for example, the VAT rate on a Product class or, more generally, the number of instances of the class).

In certain cases, the information depends on the parameters. For example, we want a value in a certain unit. We therefore have to translate the information as an operation, whose signature specifies the parameters. This is almost always the case with the indicators included in the semantic model.

Consequently, it is regrettable that the informative properties are separated into two compartments: the attributes and the operations. What keeps them together is the way they are named, in contrast with the active properties.

c. Active properties

They correspond to actions of, or on, the object.

They are always translated as an operation. Their name begins with a verb. Their signature can contain one or several parameters and their possible result has distinctive features, like an attribute.

Like an attribute too, an operation is said to be of an instance scope or a class scope. This does not, in any way, change its behavior, but adds a point to its manner of functioning.

Unlike an attribute, an operation has a body: the explanation, simulation or realization of the action, in text, algorithm or code format. In the body of the operation, we isolate pre-condition and post-condition parts. Together, and with the signature, they make up the contract of the operation[33].

d. Transformative properties and lifecycle

UML contains an extraordinarily rich notation to describe all state machines. This tool is perfect for translating the objects' lifecycle. The states can be enriched by attributes and behaviors (action, emitting or receiving events). The transitions link the events and the operations.

A class can be governed by several state machines[34].

e. Associations

UML expresses the class structural properties through association and classification.

The association enables objects or concepts to be linked and the dependencies of associated concepts to be expressed. Cases include:

- binary association, the most frequent, between two classes;
- assembly and composition, a particular case of binary association with a pre-established semantics;
- reflexive association, of one class on itself (for example: relationship between two people, that is to say, two instances of the same Person class);
- n-ary association, where 'n' is the number of terms of the association, essential to express complex concepts, determined by several others;

[33] The description of the class properties is the subject of the procedure data sheet PxPCD-22a.

[34] For further information, see the data sheet PxPCD-24.

- qualified association, concerning one or several attributes which qualify the link.

The association is a means of fixing an important part of the semantics. It reproduces, fairly often, the sentence of natural language. This is why it is often named by a verb in the present indicative (present simple). Its power of expression does not stop there. Each term of the association can be named, which enables us to specify its role in the concept built this way. The association cardinalities contribute to specifying the signification and structure. Finally, an association itself can be considered as a property-carrying concept. In that case, it is reified, that is to say we attach a class to it of which each instance only exists if linked to an instance of each of the classes that are part of the association. In this way, the model can grasp the associated concepts and unravel the most complex configurations[35].

f. Classification

Classification consists in building a class hierarchy by distributing the features in the most economical way. The object-oriented approach is inheritance. The generic properties are included on the parent classes (or superclasses), the specific properties on the child classes (or subclasses). This procedure brings the factorization principle into play and makes the modeling ideal possible: expressing a single element of meaning, in a model, once and once only.

Other notions complete the mechanism around inheritance:

- abstract class (from which we can draw no instance, because it is too generic);
- virtual operation (defined on an abstract class, but only described at its child class level);
- polymorphism (mechanism taking advantage of the previous notions).

Armed with these notions, the modeler can undertake a task similar to cladistics and provide a representation of the system studied that is both complete and economical, without any redundancy.

g. Structuration

Decomposing the semantic aspect is achieved by creating object domains. In UML, the type of element that corresponds to the object domain is packaging, a very general mechanism for regrouping modeling elements.

Packages are linked by dependencies.

h. Cooperation

In semantic modeling, cooperation between objects refers to:

- direct calls between objects (via their properties);
- events,
- conceptual processes.

The first point was discussed earlier.

The notion of event exists in UML. The events are represented as classes. We can arrange them in order thanks to inheritance. The event often carries information. Attributes are therefore associated with it[36].

The conceptual process has not been retained as a fully-fledged category, because it does not exist per se: it is the result of the cooperation between objects. Moreover, there would be a danger in instilling, in the semantic aspect, an approach and decisions that of a pragmatic (or organizational) nature. Once this warning has been understood, it is possible to represent object cooperation on a grand scale. This is what we call a conceptual process. UML offers several means of representing them: the activity diagram, sequence diagram and communication diagram.

[35] Cf. procedure PxPCD-23, "Expressing complex or associated concepts".

[36] This richness is almost excessive. In any event, it raises a question: from which moment does the description of an event become that of an object? This point is discussed in the procedure data sheet on the overall behavior of the system (ref. PxPCD-26).

i. Constraints

UML contains the notion of constraint, which is a note attached to one or several elements of the model. The content of the note can be expressed with varying levels of formalism, from natural language to formal languages, even programming languages.

3.3 Semantic models

a. Differences with data modeling

A semantic model incorporates a conceptual data model, but says a great deal more. Its aim is to express all the semantics of the system studied, in three dimensions: structural, functional and contractual. A data model only describes information, and even then not all the information is described, as it does not use the derived informative properties (example: age or the indicators). Another weakness of the data model concerns the class scope properties: they can only be represented as variables detached from the concept (for example, the VAT rate will be handled separately from the Product entity, which will only retain the instance scope properties).

Thus, the conceptual data model is shown as being insufficient to represent all informative properties. It is, of course, totally incapable of representing the other types of properties.

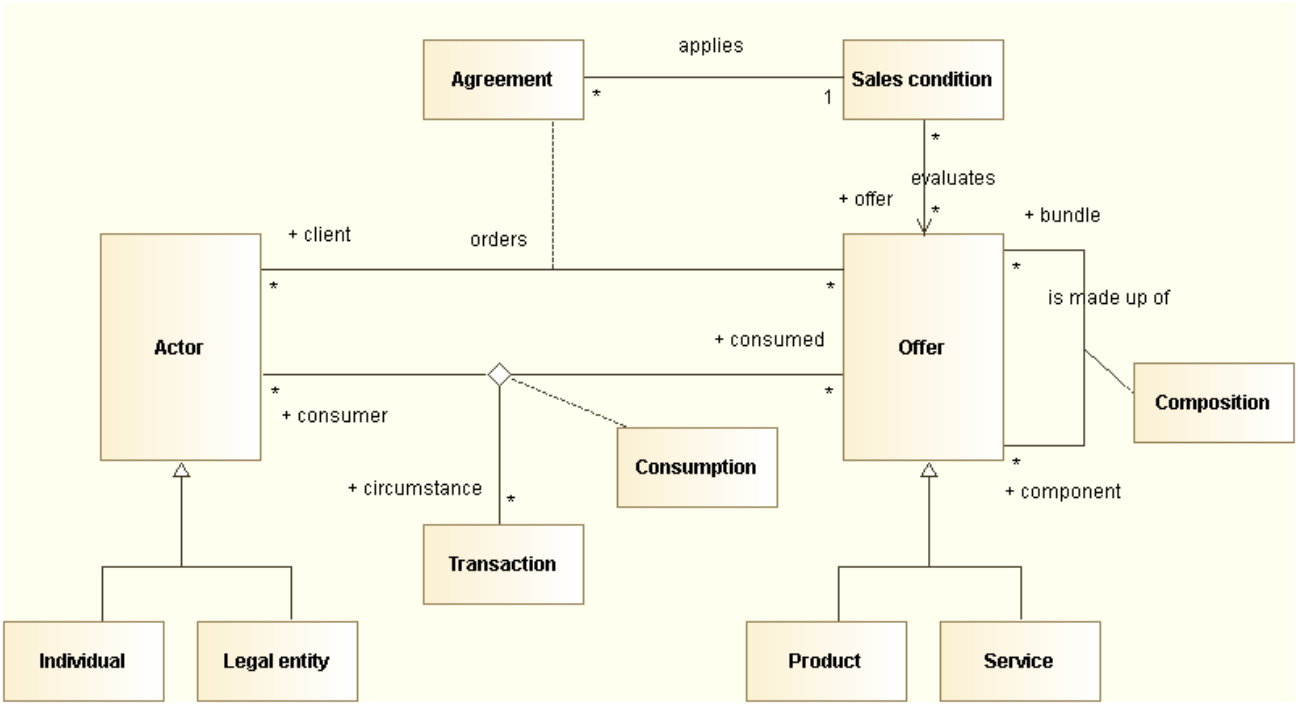
It is always possible to extract that data model from the semantic model. In conclusion, the conceptual data model becomes useless when we have a true semantic model.

One noteworthy point is that the modeling effort results in different structures, according to whether we are interested in data or whether we adopt a semantic approach. Indeed, by taking all the semantic properties into account and applying the principles, this approach leads to very different structuring decisions.

b. Example

The diagram below shows some of the possibilities of expression offered by UML for semantic modeling. It is not representative of the diagrams that must be shown in a model, as too complex (too many elements are represented). It does not show the class properties. It is only a very partial view of the semantic model.

Figure PxPRD-20_7. An example of a class diagram



The notion of Actor has been introduced to generalize the relations with the enterprises and individuals. The lines shared by both these concepts, including the associations, have been "brought up" to this class. The contract is an associative class: each contract instance is attached to an Actor-Offer couple (note that, in this

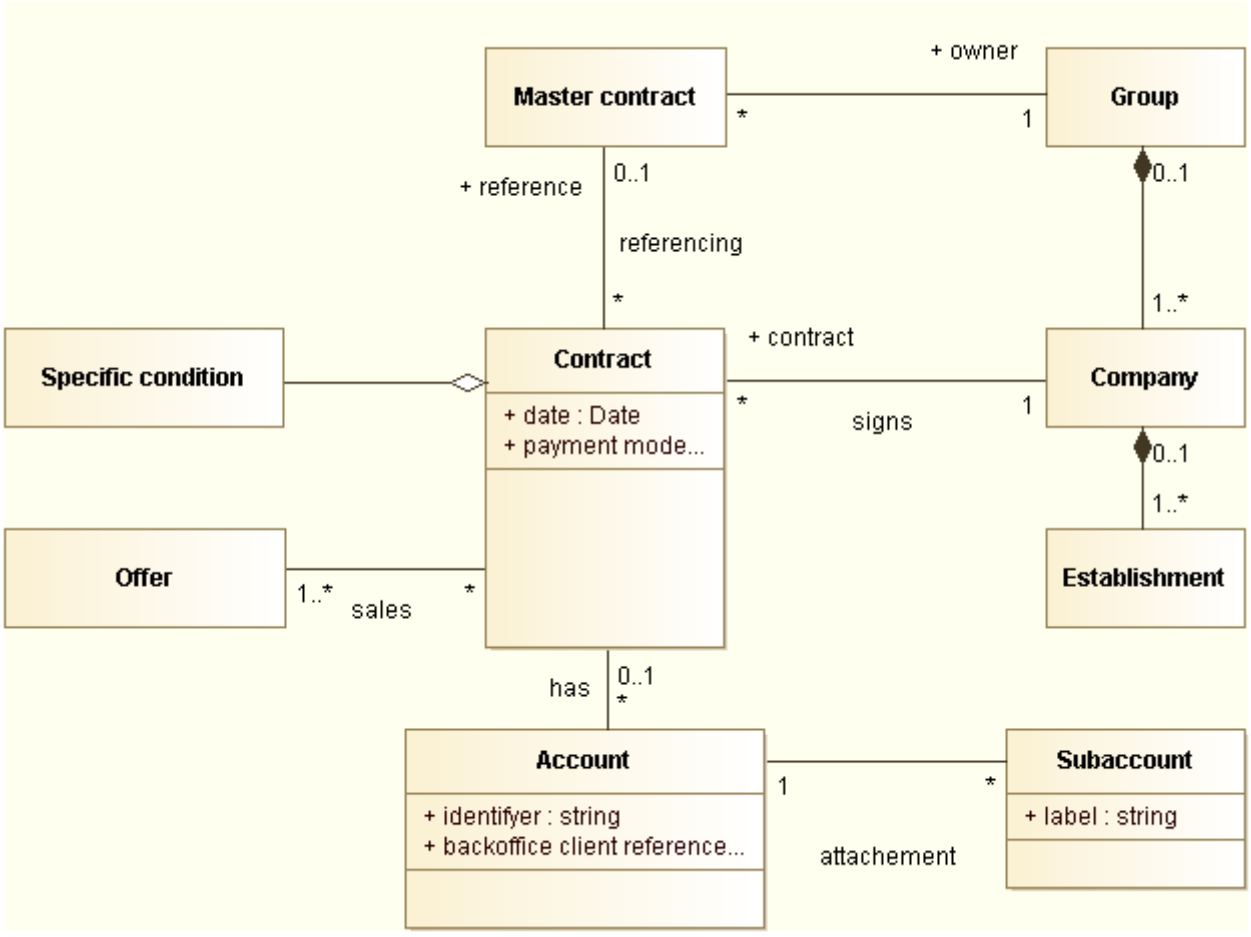
model, we can only create a single contract for a given couple). The notion of Product has been completed by that of Service and abstracted from the Offer concept. This results in an elegant solution to deal with "bundles" and other "packaged offers", dear to marketing: the reflexive association on Offer, with the related Composition class which enables the packaging conditions to be detailed. An associative class also processes the price of an offer, which depends on the sales conditions. The ternary association enables the consumption to be taken into account, as a concept determined by the consumer, the product consumed and the period. Modeling this way has the advantage of visualizing the dependencies between concepts, which is a noted contribution of the semantic model. One of its consequences is that most cardinalities are of the '*' type (for "several").

In this example, we have only taken an interest in the structure of the model, as can be shown by a class diagram. The model also contains attributes, operations, state machines and constraints, which are not represented on this diagram.

c. Questions to ask oneself

The figure below is a class diagram, taken in another context, but covering similar notions.

Figure PxPRD-20_8. Second example of a class diagram



At first glance, we can note a different style of modeling. Only binary associations link the classes. We can thus ask ourselves whether some conceptual determinations have not been missed. Let us take Contract: it is in a very different position compared to the previous solution; it is no longer an associative class. Indeed, the cardinality of the "sales" association on the Offer side is "1..*", which means that a contract is drawn up for one or several offers. It is therefore not completely the same notion of contract. In this case, the solution of the associative class is actually not suitable, as it would impose that a contract could only be linked to a single offer.

The Account class carries a reference towards the customer. It would be better represented as an association towards the establishment or company. This incidentally raises the question: what type of customer is it? The modeler could try to unify the notions of Group, Company, Establishment, under the same concept accompanied

by a reflexive association enabling the decomposition of the organizations (in as many levels as required). In such a solution, the name "Client" would appear as a role name on the association going from Account to this new class. This would be one way of answering the question raised.

A framework (or master) contract is concluded across several contracts. The cardinality is powerless to express a constraint: "the contracts linked to the same framework contract are signed by companies belonging to the group that drew up the framework contract". This constraint must be documented. It can be represented graphically on the class diagram, by linking the four associations, exactly as the concepts intervene in the same sentence above.

d. Cut-off criterion

Semantic modeling sets off to conquer the reality: it is a potentially never-ending effort. Consequently the question is raised: How far does one go in this effort to elucidate the reality?

The scope of semantic modeling is circumscribed according to the pertinence criterion, that is to say, by relating this effort to an explicit objective. Yet, it is often necessary to go slightly beyond this objective, especially if it is a project objective. Indeed, to get to the "right" representation more quickly, the modeler has to anticipate future developments or uses of the model.

Another question that the modeler encounters is that of the level of genericity of abstraction of the model. On the one hand, the genericity of the model increases it scope and therefore constitutes an advantage. On the other hand, it complicates the modeler's task and makes it more difficult to communicate the model. This question can only be decided one way or another by analyzing the context and ambition. It implies an architecture vision.

3.4 Architecture of the semantic aspect

It was mentioned earlier, at the enterprise scale, it is necessary to decompose the semantic aspect. This decomposition is more for convenience than for an organization specific to the aspect. It is therefore the result of a decision rather than an observation. Nevertheless, this type of architecture decision must obey some rules. The first is that of the decomposition criterion. The unit used is that of the object domain, in contrast to the functional domain. This means that we adopt the aggregation logic of naturally coupled objects, in contrast to the activity logic. More precise factors guide this architecture decision, notably the objects' pace of life by domain.

These considerations have a considerable impact on mastering the semantic model and, even more so, on how it is used in the transformation chain (see the following chapter). They are the subject of the procedure data sheet "Structuring the semantic aspect" [37], intended for business architects. This data sheet presents a generic architecture of the semantic aspect.

3.5 Quality of the semantic modeling

a. Introduction to the quality of the models

To ensure that the requirement and obligation of semantic modeling is understood, there is no better way than to make the expected quality of the models clear. Before examining the criteria for this, it is important to remind ourselves of some of the rules that guide the modeling practice.

Firstly, **the level of detail**. We oppose the widely held view that a semantic or conceptual model should be a general model and that the modeler is exempt from entering into details. Certainly, a semantic model can remain general, like architecture, if we so decide; but, a semantic model is only complete when it restitutes *all* the semantics of the domain studied, at least all that we need to know about a reality to act on it. Consequently, the semantic model is just as detailed as any other one. Simply, the details that we include do not come under the technical solution but are those of a rigorous description of the reality. Among others, the operations and their description, as well as constraints, are found here. This reminder is essential if the transformation activities are to run smoothly. Each time the semantic modeling effort is interrupted too early, we live to regret it, as it pushes back the task of describing the business at a sufficiently detailed level to other actors.

[37] Ref. PxPCD-25.

Then, the quality of the model depends on applying the principles that make the internal rules of the semantic aspect (abstraction, encapsulation, factorization, cooperation). **These principles are directly transcribed in the model**. An audit of the model can check whether they have been respected. One part of this verification is formal and makes use of indexes such as the n-ary association ratio; however, the second part requires us to take a closer look. For example, redundancy is not always visible at first glance as it can be hidden by different denominations.

Finally, it is important to remember that, according to Praxeme, **a model must contain its proof**. While remaining at the design level, the modeler can run the model "mentally", use modeling elements to design test cases, not to mention the simulation of the model[38].

b. Attitude of the modeler

The modeler tasked with expressing the semantics, approaches reality with no preconceived ideas. Contrary to appearances, this is not a spontaneous attitude. It requires, on the modeler's part, a specific and continuous effort to disregard the organizational and technical circumstances (the abstraction effort). The quality of the semantic model requires us to step away from current practices and existing solutions.

Furthermore, rather than obsessing over the apparent complexity of the domain, the modeler must capture the essentials and extract the fundamental core of business knowledge.

He/she will have to defend the simplicity of his/her model faced with the general trend towards complication. One answer consists in showing how this essential model restitutes the reality and how it can instantiate itself to take the diversity of actual situations into account.

If we insist on the required – and sometimes troubling – simplicity of the semantic model, it is because important consequences follow from this model, with the prospect of the urbanization of information systems or the simplification of processes.

Through the abstraction effort, the model simplifies itself, while remaining close to the targeted part of the reality. It enables a stable core to be isolated, a necessary condition to buffer future requests for change: the robustness follows from the simplicity. In addition, abstraction encourages genericity, which allows the scope to be broadened. In the same movement, the model becomes a tool to lead forward thinking and to anticipate changes and future requests. The guides of the adjacent aspects (pragmatic and logic) present procedures that enable innovation.

c. Quality factors

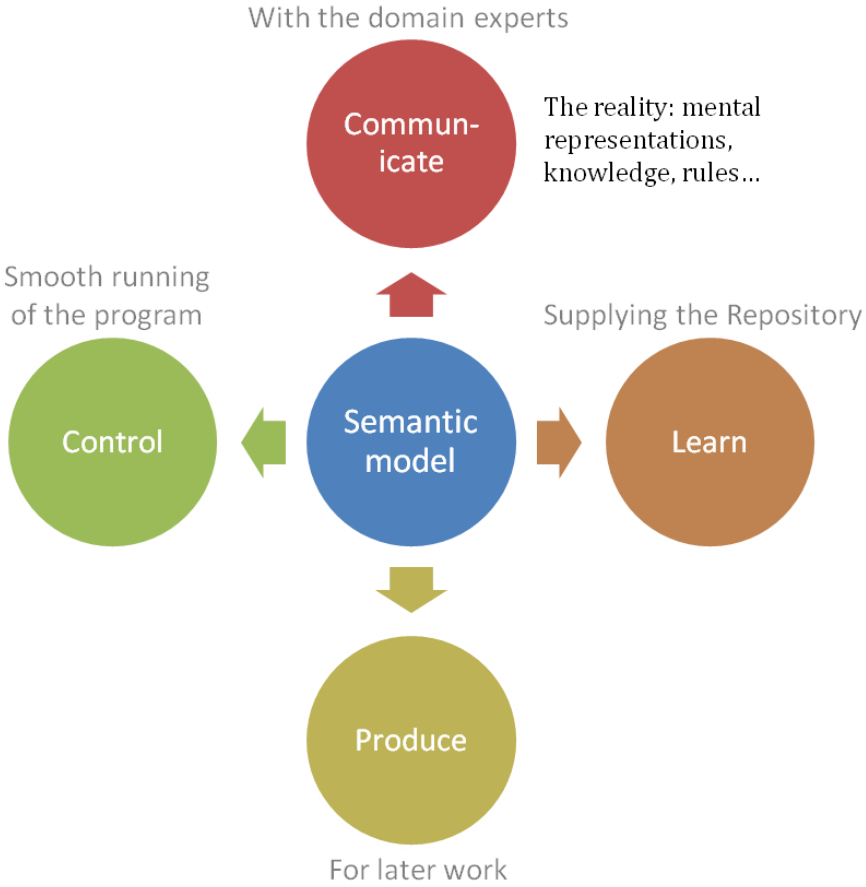
The requirements that the semantic aspect has to bear are deduced, first, from its utilizations. It is from this angle that we will discover the quality factors.

We can identify four important categories of use:

1. **Communicate**: particularly with business experts, but also representatives of the actors of the enterprise, contracting owners, partners, clients, sales reps… and also with the stakeholders who participate in developing the strategy.
2. **Control**: the model, better than the input material, enables you to verify certain requirements for the successful completion of the project, from a transformation standpoint.
3. **Produce**: the model is intended for use by other projects, even other teams.
4. **Learn**: the model is one of the tools used to capitalize on knowledge.

[38] The procedure "Proving the quality of the semantic model" (ref. PxPCD-27) goes further in the quality analysis and links it with designing tests.

Figure PxPRD-20_9. The main categories of use of the semantic model



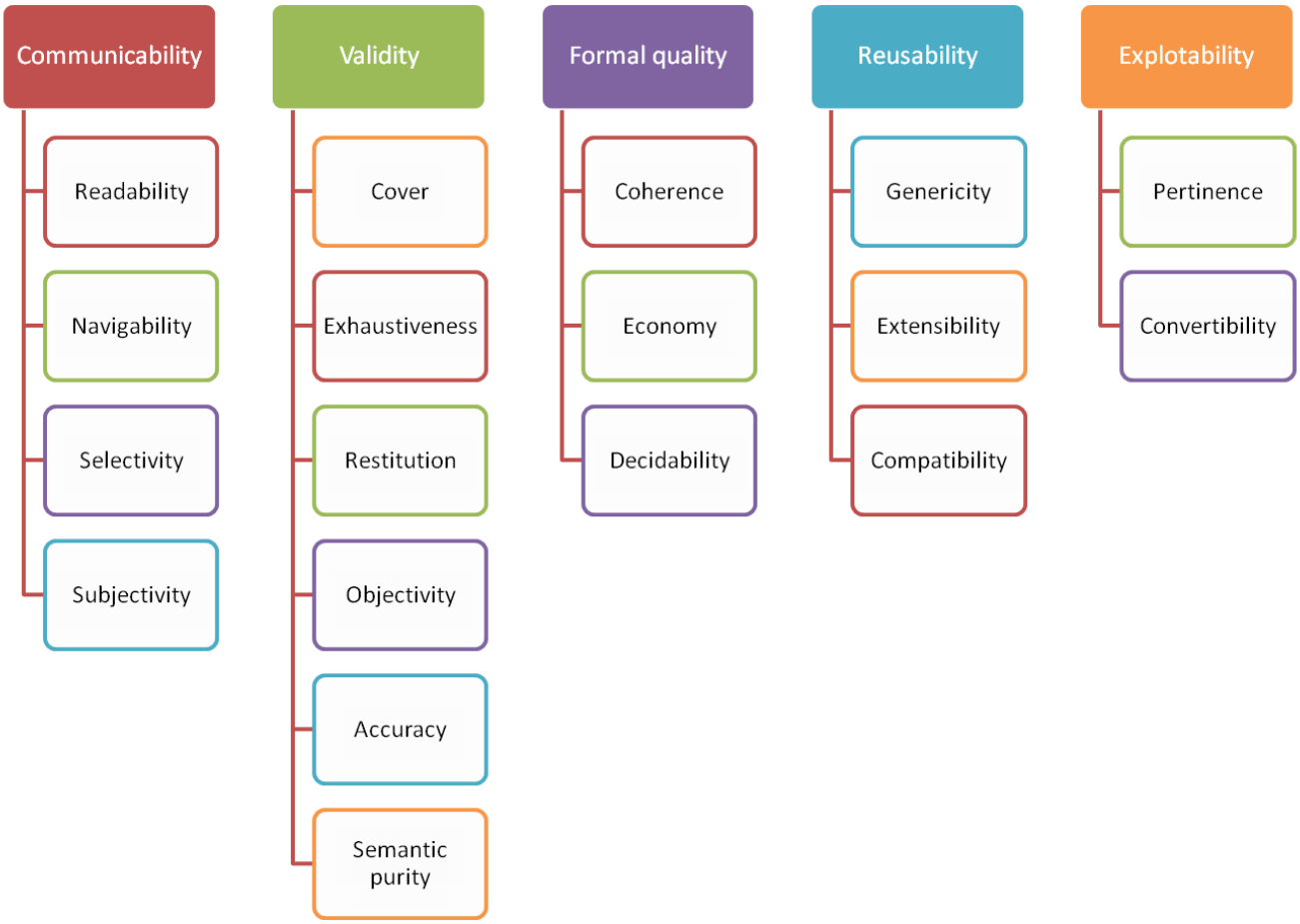
The quality factors, specific to the semantic model, can be deduced from these uses:

- accuracy: appropriateness to reality and the representations made of it by the actors of the system; more widely: the quality of the content (cover, exhaustiveness, precision);
- communicability: contribution of the model to the communication (readability, self-justification…);
- formal quality: respect of form constraints, which contribute to other factors themselves;
- reusability (capitalization) and exploitability (production).

d. Quality criteria

After the results in usage terms, the quality analysis continues with the study of the features shown by the model. The quality factors express requirements in use terms; it is the external point of view. They have the advantage of being concerned with the aims themselves – which should be the constant concern of the quality approach. Their major flaw, on the other hand, is that they cannot be directly objectified and, less still, measured. This is why we have to go from quality factors to quality criteria. The latter contribute to the factors and are more easily observed.

Figure PxPRD-20_10. The quality criteria for a semantic model



e. Documentation

The model also has to satisfy documentary requirements. They consist of:

- the traceability upstream,
- the restitution,
- the semantic purity,
- providing quantitative indications.

A **traceability** mechanism must enable the semantic model elements to be justified regarding their input:

- conceptual requirements, when they exist,
- functional requirements (statement of need, traditional),
- interviews with domain experts or user representatives,
- regulatory, strategic or institutional documents,
- current terminology,
- potentially, existing database models…

In accordance with the metamodel, the traceability chains move from the semantic aspect towards the intentional aspect. The case of the reference to technical or physical types of material (such as existing database schemas) does not contradict this logic. It brings the "curve of the sun" into play [39]: we have two versions of the Enterprise System to deal with[40].

[39] About the curve of the sun, see "Elements of methodology" in the Process dimension (ref. PxPCS-01).

[40] Managing several version of the ES (and of its description) is a delicate task. The transformation dynamics cannot avoid it. This point is discussed in the Process documents (see PxPCS-03).

Restitution is the ability of the model to retranslate its formal elements into everyday terms, understood by the stakeholders. The diagrams of the semantic model must be able to be reinterpreted in natural language, without resulting in contradictions or ambiguities. We must be able, from the model, to restitute the discourse on the field of study. To this end, it is necessary to increase the expressive power of the model and to keep the synonyms related to the modeling elements. The thesaurus ensures this second point[41]. The modeler will use object diagrams or sequence diagrams to unfold the model and explain, using scenarios, the more subtle points.

Purity is the feature of a model that only explains the semantic aspect, excluding all other types of considerations. It is a result of applying the principle of abstraction. In the documentation, we will check, for example, that there are no traceability links to intentional elements (requirements, terms…) that are themselves directed towards other aspects.

However, the semantic model is not an "abstract" model in the sense that it would represent an ethereal world. It describes the reality, even if it is through concepts. For this reason, it takes an interest in the quantitative information that we can glean at this stage. The documentation consists, for the principal notions of the model, of:

- the number of instances (minimum, maximum, potentially: distribution in time);
- the volumetric information (for the principal classes and their satellites)[42].

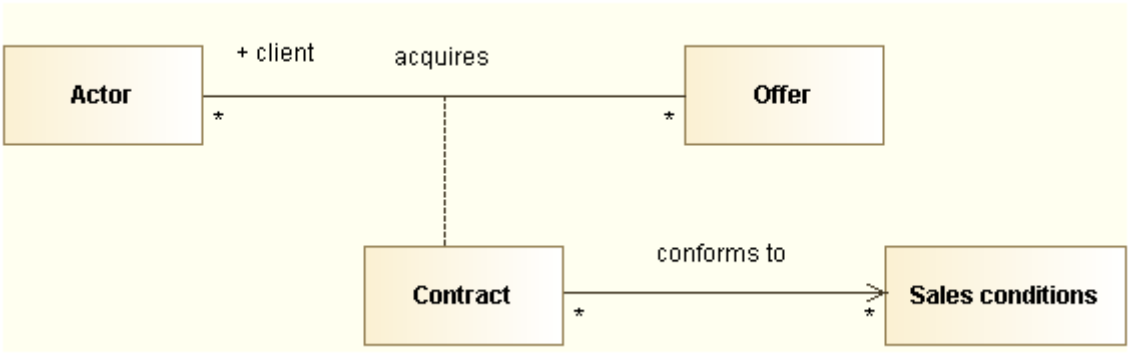
This information will be of use later, when we will need to size up the organizational, logistics and physical settings.

f. Expressivity of the model

Among the many criteria, we emphasize the expressivity, which plays a key role in the quality of the model. The model must, as far as possible, reproduce the universe of discourse, the business knowledge. It has to be "readable", that is to say the key business expressions must be found within.

The constraints must also be found within. Thus, a contract is drawn up for one offer and one client. If we link the Contract class by two binary associations, one to Actor and the other to Offer, we lose the structural definition. In the solution below, the reader can find the natural sentence: "an actor reserves an offer". This act constitutes the contract and the actor assumes the role of client in it.

Figure PxPRD-20_11. An example: concluding a contract



Here, the modeler has taken care to arrange the elements of the diagram in such a way as to respect the order: subject (Actor), verb (reserves), direct object (Offer). This is not the way it is read if we take the direction indicated on some associations, as with "conforms to", on the same example. The associations' direction does not show which way round it is read but the navigability, which results from the dependencies between the domains in which the classes are split[43,44].

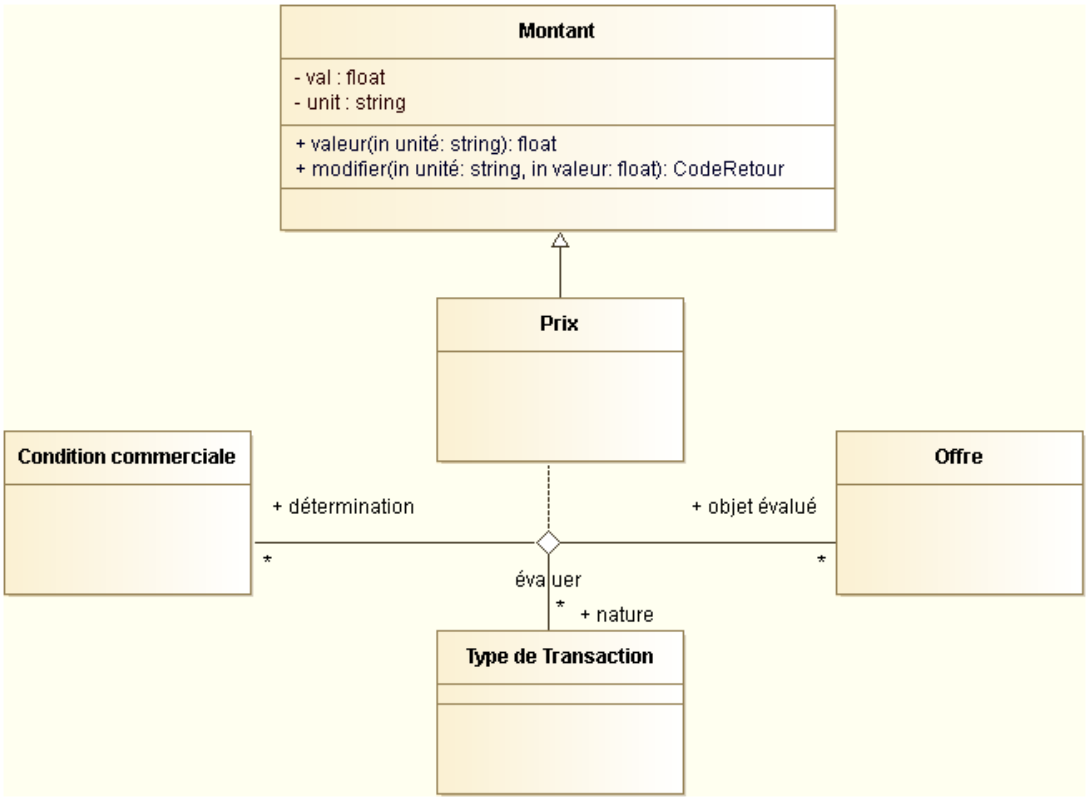
[41] See the terminological procedures, PxPCD-14.

[42] UML tooling can save this information as tagged values, as defined in the standard.

[43] See PxPCD-25.

The modeler seeks the expressivity of the model, but not at the cost of violating the principles. It has to be noted that the principles and precepts of semantic modeling take the modeler in an opposite direction. Thus, the more the model becomes generic, and so compact, the less easy it is to communicate. The precept of the uniqueness of terms enjoins us to only use each term once. As a term can only have one meaning in the model, it can only have one unique position. Let us take the example of price. A data model can have several "price" attributes, sometimes several dozen, to express notions from the basic price to a promotional offer price via the supplier's purchase price. The solution, which consists of creating an attribute for each of the price variants, has the disadvantage of diluting the determination of these variants. Yet, unearthing these determinations to clarify the concepts is precisely what we expect from the semantic model. The solution will be sought more in the structuring of the model: an associative class, most probably linked to an n-ary association, will carry the price attribute. This is the only way to express a price for the whole model. Such a solution does not hurt the expressivity: after all, it enables the notion of price to be found more easily and enables us to think about its semantics. However, there is no doubt that it reduces the ability to communicate the model. This point will be discussed in the following chapter.

Figure PxPRD-20_12. Modeling price in an input/output catalog



In this example (see the above figure), the price is set not only by the sales conditions, but also by its type, linked to the type of transaction considered: purchase from a supplier, sale to a wholesaler, sale to a retailer… A price is only meaningful for a triplet of offer, condition and transaction type. The only way it can be represented therefore is by the associative class linked to the ternary association. In this way, the position of the concept of price is perfectly fixed. Its content is not different to the more general notion of amount, which encapsulates the monetary unit. Thus price inherits from the Amount class.

---

[44] Some regret that the majority of modeling tools do not allow two names to be given to an association, each one corresponding to a direction it can be read in, as was planned by the standard. The same people regret that these tools do not provide, either, the small symbols that would enable us to indicate the direction it should be read in. These shortcomings do not seem to trouble those who work in IT in their use of the notation, but their lack is felt in semantic modeling.

At first glance, such a model may disconcert the business representative, more used to associating a price with a form, in a precise context. Here, all the possible contexts have been regrouped, which requires the determinations of price to be clarified. The model becomes more compact, which in itself is not a bad thing, but which hampers communication.

4. Use of semantic models

4.1 Analysis and design of the semantic aspect

When approaching the semantic aspect, the modeler can adopt either one of the positions of analysis and design. Analysis consists in expressing the business knowledge as revealed in practice. Design leads one to re-examine the business concepts. In fact, applying the semantic modeling principles quickly leads from analysis to design, almost against our will. The representation technique based on the object-oriented approach pulls us towards genericity and quickly leads us to go beyond current knowledge. For project management, this can be a risk, but it is important to view it, above all, as a tremendous source of innovation.

The example given below shows how semantic modeling can help clarify general notions. We have to clarify what we mean by the expression "customer focus". There are two opposing meanings. They are first expressed intuitively, then by draft UML diagrams. This use of the modeling technique helps us to think about the fundamental business notions. They can lead to radical innovations.

Figure PxPRD-20_13.Classic interpretation of customer focus

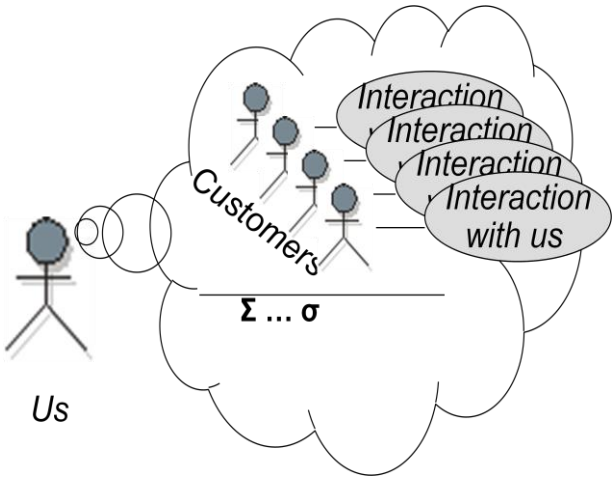
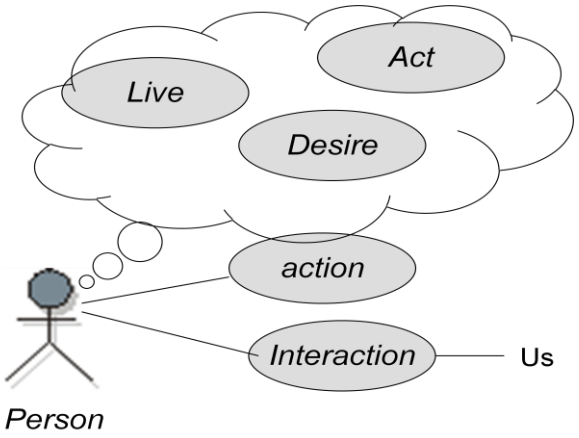


Figure PxPRD-20_14.Radical interpretation of customer focus



1. The first interpretation, classic, consists in placing the customer in the center of our view and exploiting to maximum potential the information that the enterprise has on him/her.
2. The second interpretation, radical, consists in adopting the customer viewpoint, rather than that of the enterprise. It leads to significant changes, beginning by naming the central actor him-/herself: indeed, the "customer" does not see him-/herself as a customer, but as a person.

To talk about the "customer" means that we position the person uniquely or mainly regarding his/her relationship with the enterprise that supplies him/her. It therefore overlooks part of the reality:

- that of the person in his/her aspirations or relations with other suppliers;
- that of those people who are not customers, who interact or could interact with the enterprise.

We raise these cultural approaches here, as their influence will be read directly in the semantic model. The approach chosen will color the model: it is not only the choice of terms that is at stake, but especially the structuring of the model. At stake are the draft models that correspond to both interpretations.

Figure PxPRD-20_15. Classic interpretation of customer focus diagram

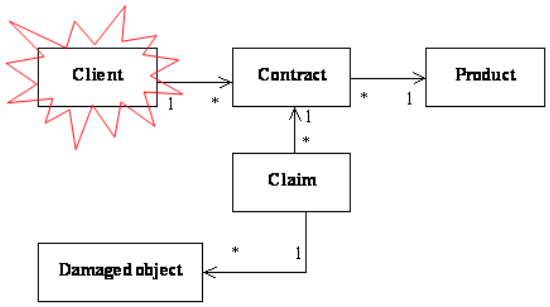
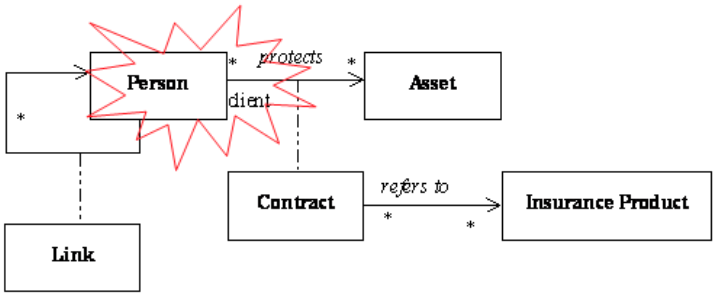


Figure PxPRD-20_16. Radical interpretation of customer focus diagram



In our example, adopting the radical interpretation requires a true decentering, typical of the major design revolutions. It leads to an upheaval in our perception and a new model around the person and the product. Its shockwave also reaches the pragmatic aspect and, as a consequence, all the aspects of the Enterprise System.

4.2 Using the semantic model to communicate

The semantic aspect belongs to what is common practice to call the "business view" of the enterprise. Its content concerns the actors of the enterprise. Its model should therefore be a natural tool to speak about the business, train new contributors and invent new approaches. However, as mentioned several times in the previous pages, the semantic model is not as obvious a communication tool. This is due to the requirement for formalism, as well as the destabilizing effect of abstraction.

We really must not give up on this requirement as it may risk upsetting the transformation chain.

First of all, the readability and exploitability of the model can be greatly enhanced by some simple measures:

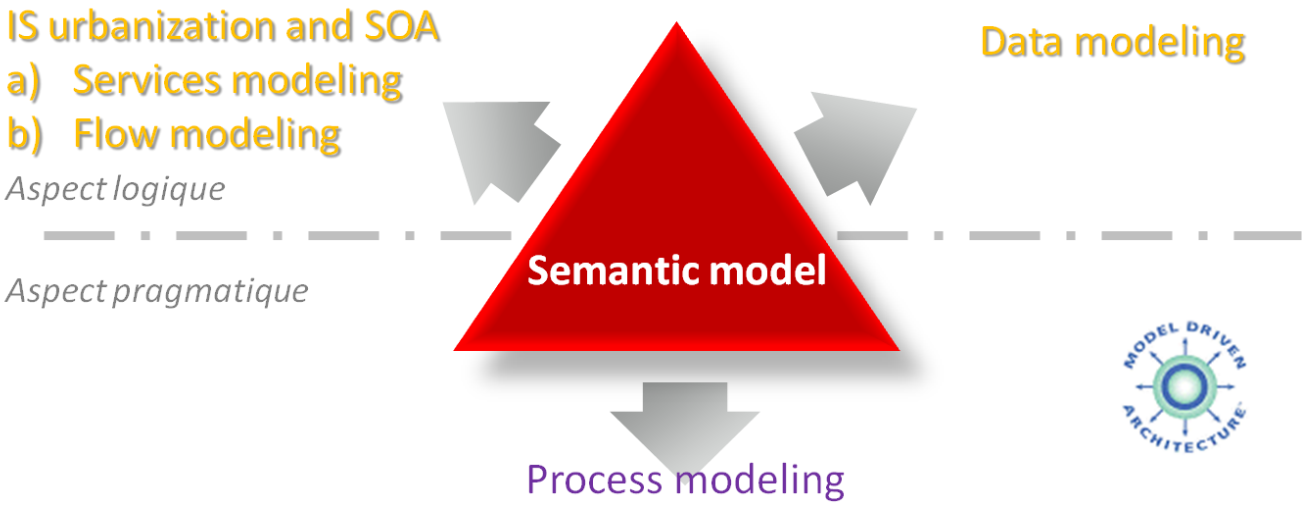
- The model is not restricted to its formal elements, nor to an amount of diagrams; it is widely commented.
- All comments include the descriptions, as well as the modeling decisions and their justifications.
- All diagrams are headed by a title explaining their communication purpose. This is the first thing the modeler must think about before building the diagram: what question does it answer? What story does it tell?
- The diagrams come with a comment that explains them.
- The sophistications of the model (n-ary associations, reflexive, etc.) call for additional care: the modeler illustrates them with object diagrams that unfold the model.
- The modeler is careful not to overload the diagrams. It is better to make several diagrams with few elements, each time it is necessary to explain a specific point.
- We must not hesitate to restate in text form that which the diagrams explain graphically.

Despite these precautions, it is possible that we come up against some obstacles. In this case, the semantic model must remain the source of truth, the canonic expression of the business knowledge. It is then necessary to find intermediate resources for the exchanges between business experts, who hold the knowledge, and the modelers, who master the form. The natural language remains the communication tool par excellence. Having the model available helps ensure that the right questions are asked in more detail.

4.3 Derivation of the semantic model

Derivation is a key notion in the transformation approach. If the semantic model is useful in itself, it is also the starting point for "derivation channels" that enhance it through the output of other aspects. Praxeme has identified four derivation channels from the semantic aspect.

Figure PxPRD-20_17. The derivation channels



The semantic model will feed into the following areas:

- It contributes to reforming business processes and to organization design, so long as we are prepared to adopt a new approach to processes[45]. This approach acts as leverage for simplification.
- It provides the starting point for many of the logical components, mainly the logical services of the internal layer[46].
- The logical data model is easily derived from it.
- Again towards the logical aspect, other derivation rules, applied to the semantic model, enable the design of structures for the flows that supply the system.

4.4 Changes to the semantic model

Without encroaching upon the Process dimension[47], this last section introduces several questions linked to the administration of the semantic aspect.

The key point is mastering the aspect in its entirety. The semantic aspect of a system can rarely be modeled in one go. We cannot be satisfied with a stack of successive models, produced on an as-need basis. The logic is one of the overall dynamics of transformation:

1. Structuring the aspect, that is to say defining the object domains and indicating how the main objects are distributed within (at least to the extent of 80%).
2. Create, on this base, the Repository of enterprise description.
3. Impose this frame on projects (potentially by consolidating it when shortcomings appear).
4. Enrich this repository by adding detailed models to it, after their validation and verification that they conform to the principles and decisions about the semantic aspect.

This logic presupposes organizational measures and tooling as mentioned elsewhere.

Regarding changes to the semantic model, we have to distinguish two phenomena:

- enrichment,
- change.

[45] This new approach to the processes, in contrast to the functional approach, is presented in the procedure data sheet "Innovating with process" (reference PxPCD-33).

[46] Cf. the guide "Logical aspect approach" (ref. PxPRD-50), completed by the logical aspect procedures (ref. PxPCD-5#).

[47] See in particular "The global dynamics of transformation" (ref. PxPCS-02).

With enrichment, the model is completed by a detail that was lacking. For example, new classes or new properties for existing classes can be added. These additions have an impact on the derived elements of the model, but it is controllable.

The case of change is more painful. It can be, for example, a change of class name or a different division of properties (and so a change of structure). A likely case is when we realize two concepts located in different activity domains are close enough to be merged. In these cases, the impact is far greater as it means we must rethink everything that is even remotely connected with the part of the semantics that change: processes, flow, database, logical components… If we do not agree to these changes, we will be confined to maintaining pragmatic or logistical elements that refer to different semantic versions. It is the beginning of a nightmare. A specific "governance" then comes into play. System maps have to help identify the zones which are dependent on the different semantic versions. Crossing the limits between these zones requires the information to be converted. These procedures have a cost. Above all, they risk perpetuating themselves and contributing to the weight that the enterprise will no longer be able to rid itself of.

All the more reason why it is necessary to spend time on the semantic modeling, stabilizing the model and even extending it a little further than what seems useful at first. It is also a good reason to seek the genericity, even the universality of the model, against the prevailing trends. When the model concerns the universal, we know that its reasons for change are minimized. If the enterprise adopts, from the outset, an address and geographic description that can be applied to all countries, it has eliminated one likely cause of future change. Almost all notions are open to such an effort of genericity. Gradually, the turn to tried-and-tested generic models will help reduce these risks and pool efforts.

Table of figures

Index