Published in the Proceedings of OOPSLA'98 workshops, Vancouver, Canada, October 1998

Modeling and Metamodeling Requirements for Knowledge Management

Olivier Gerbé[1], Brigitte Kerhervé[2]

[1]DMR Consulting Group Inc.
1200 Mc Gill College
Montréal, Québec
Canada H3B 4G7
e-mail: Olivier.Gerbe@dmr.ca

[2]Université du Québec à Montréal
Département Informatique
CP 8888, succursale centre ville
Montréal, Québec, Canada H3C 3P8
e-mail: Kerherve.Brigitte@uqam.ca

1 Introduction

A corporate knowledge management system is an absolute necessity, regardless of whether this involves storing information, capturing it from senior staff or transferring it to juniors. Corporate knowledge is like a repository of a company's know-how, that is, its business processes, procedures, policies (mission, rules, standards) and data (sales, purchases, salaries, etc.).

Defining a data model and using a database management system, such as a RDBMS or OODBMS, already goes a long way towards improving knowledge management. Nevertheless, all of a company's know-how, objects of interest, processes, procedures or even policies vary too widely in nature to be supported by database management systems. Too much knowledge remains implicit, texts are not analyzed, constraints or rules are not represented.

1.1 Models

A knowledge management system must be developed around models for it to represent all forms of knowledge. For the static or structural part of corporate knowledge, structure models represent the company's objects of interest and their interrelationships. For the dynamic part, behavior models represent the processes or procedures of the company. And regardless of whether the structure or behavior level is involved, the constraints or rules that govern them must be represented. Constraint models do this job efficiently.

However, besides their main functions, the different types of models do have other uses.

Structure models, which store knowledge, can be also used as a teaching tool to explain the models, or the knowledge they represent can even be used

to convert a model from one formalism to another (for example, to change from an entity-relationship to an OO model).

Behavior models that represent and store processes can, in a learning context, be used to teach and explain those processes. They can also provide the information needed to simulate or perform a process (workflow manager). In the context of an electronic performance support system, structure models can provide the context-sensitive help needed to perform a task.

Constraint models that represent the constraints or rules governing knowledge can be used to validate knowledge. When used in a context of knowledge acquisition, these same models provide information for explaining the constraints, that is, they give reasons why some knowledge is unacceptable.

1.2 Levels of Model Use

We have just seen that the different types of models needed to represent knowledge can be used in different ways. More particularly, we can distinguish three different ways or rather three levels of use: model, meta and metameta.

The model level corresponds to the use of models based on their main function, that is, to store and validate knowledge, store and validate behavior, simulate or apply behavior and store constraints. In terms of a corporate memory, the model level provides the information employees need to perform their jobs.

The meta level corresponds to the use of models not to store knowledge but to provide information. This level is used to explain structures, behavior and constraints. The meta level supports the learning of tasks by an employee.

The metameta level corresponds to the use of models to provide information on the models. This information is used to convert models, compare and integrate models, behavior and constraints, or even validate the consistency of a set of constraints.

All these kinds and uses of knowledge rise problem...

This document has been organized as follows. Section 2 details how the corporate knowledge can be structured. Section 3 and Section 4 present specific requirements for modeling and metamodeling corporate knowledge. Then we conclude in Section 5.

2 Structuring Knowledge

Knowledge related to a company's know-how raises a problem of volume and complexity. One solution is to structure knowledge. This can be done along two axes: a horizontal axis that defines knowledge types and a vertical axis that defines modeling levels.

2.1 Knowledge Types

The horizontal axis involves three aspects: structure, which represents the static part of knowledge; behavior, which represents to behavioral part of knowledge,

and constraints, which represents the rules that must be obeyed.

2.1.1 Structure

The structure aspect defines the basic concepts and their interrelationships. In the case of a corporate memory, the structure aspect defines and describes objects of interest to a company, as well as the relationships linking them.

2.1.2 Behavior

The behavior aspect defines the concepts used to represent the behavior of knowledge objects. In the case of a corporate memory, this involves especially the representation and description of business processes.

2.1.3 Constraints

The constraints aspect defines the concepts used to specify data rules. Constraints apply to both structure and behavior. Regarding structure, the constraints govern mainly potential object relationships. And regarding behavior, they define mainly the conditions under which processes are performed or not performed.

2.2 Modeling Levels

The bulk of the work involved in modeling and metamodeling was done by teams from various standards organizations. The need for a common language and for exchanging information between modeling tools led to thinking about the modeling objects, that is, metamodels. Today a consensus has been reached (UML, OMG, CDIF, etc.) on a four-level architecture: - metametamodel; - metamodel; - model; and data.

METAMETAMODEL A metametamodel is the most abstract level. It is the metamodel definition language. It defines the concepts underlying the representation of all the other levels as well as itself. Examples are MetaClass, MetaAttribute and MetaOperation in the case of UML, MetaEntity with (meta)attribute and MetaRelationship with (meta)attribute for CDIF, and concept, conceptual relation and graph for conceptual graphs.

METAMODEL A metamodel is an instance of the metametamodel. It defines the model representation language or formalisms. Examples are class, attribute, operation for UML; entity, attribute, relationship for the entity-relationship formalism; concept type and relation type for conceptual graphs.

MODEL A model is an instance of metamodel. It defines the representation language of the domain under consideration. Examples are employee, organization, client and mission.

DATA Data are instances of the model. They correspond to real-world objects that are being described. Examples are Paul Martin, Hydro-Québec, supply electricity.

3 Structure Modeling Requirements

Three specific concerns were encountered when we began to study how to represent methods, procedures and techniques, were not obvious.

- Before gathering and storing information about things, do we need to define all the possible categories of things that exist in the application domain? - Is it possible to represent associations between categories and things? - If we define categories of categories, is it possible to integrate this higher-order information in the same knowledge base?

As we shall see later above questions may be translated into three requirements: - Classification and partial knowledge; - Category and/or instance in relationship; and - Category or instance in metamodel.

3.1 Classification and Partial Knowledge

Categories are defined based on common properties of their instances. Therefore, defining categories corresponds to defining partitioning criteria that help to distinguish one instance from another. If the number of partitioning criteria increases, the number of categories may increase dramatically and rapidly become unmanageable.

In most modeling formalisms, a thing is an instance of one and only one category. This limitation forces the definition of all the possible or conceivable categories where a thing may be potentially classified. For example, a car dealer wants to classify cars to be sold. Considering the number of seats, three categories can be defined: coupe, sedan and van. Considering the number of wheels that provide propulsion, there are two categories: two-wheel drive (2WD) and four-wheel drive (4WD). Considering the origin of cars, three categories are possible: American, European and Asian.

If the car dealer wants to classify all the cars according to these three criteria, all possible combinations must be considered. In this example there are 18 possibilities.

Another aspect of classification but rarely connected with classification is partial knowledge. How can we classify a thing if all its properties are partially known? For instance, how is the person Brown classified if we have only two categories Male and Female and if Brown's sex is unknown? Let us assume that our car dealer receives two new cars. The first one is a 2WD Sedan. However the dealer does not know in which category to classify it because the car has been assembled in Europe from Asian parts. The car may be included in the 2WD Sedan category. But if the dealer can later describe the car's origin, what will happen? The second car he receives is an American one. In which category may it be included?

*To fulfill this classification and partial knowledge requirement, the formalism should support multi-classification; i.e. an object may be an instance of more than one category, or the system should dynamically migrate instances from one category to another more specialized category.

3.2 Category and/or instance in relationship

In most modeling techniques, relationships are established between categories and are applicable to their instances; however, this is often not sufficient.

In the previous example concerning employees and organizations, we want to distinguish UNU employees that work for the United Nations University from other employees. Let UNU-EMPLOYEE be a category that specializes UNU employees. Since employees of UNU have the same properties, same attributes and same kinds of relationships as employees, UNU-EMPLOYEE is defined as a sub-category of the category EMPLOYEE.

In most modeling techniques, it is very difficult to express that any UNU employee has a relationship with the UNU organization. Modeling techniques formulate knowledge at the category level. Relationships link categories and are applicable at the instance level. In our example, a category UNU-ORGANIZATION, that has only one instance (UNU), has to be defined to be able to link UNU-EMPLOYEE and UNU-ORGANIZATION.

However, what we want to express is: "Each instance of UNU Employee has a relationship 'works for' with UNU, instance of the type Organization."

*To fulfill this requirement, the formalism should support category and/or instance in relationship; i.e. a category may be linked to an instance, or relationships may be established at the instance level.

3.3 Category or instance in Metamodel

Models are formal descriptions of objects or notions. This formal description uses different kinds of components and different kinds of relationships. A metamodel describes these components and their relationships. Metamodels deal with categories and categories of categories. In most cases, it is easy to make a distinction between categories and instances. Categories give information about instances, but categories may be seen as instances in a metamodel that gives information about the categories themselves.

In the previous example, let FEDERAL-ORGANIZATION be a kind-of ORGANIZATION. This means that FEDERAL-ORGANIZATION is a category that is a subcategory of ORGANIZATION. FEDERAL-ORGANIZATION as a category may be seen as an instance of a model component CATEGORY.

From the category perspective, FEDERAL-ORGANIZATION is a subtype of the category ORGANIZATION and it inherits its attributes. The attributes of ORGANIZATION are: label that names the organization, owner and sub-organizations. The inherited attributes of FEDERAL-ORGANIZATION are: label, owner that specializes in government, and sub-organizations.

From the instance perspective, FEDERAL-ORGANIZATION is an instance of CATEGORY. The attributes of CATEGORY are: super category that establishes the position of the category in the classification, category label that names the category, and attributes that list the attributes of the category. These attributes have the following values for FEDERAL-ORGANIZATION: - super category : ORGANIZATION - category label : FEDERAL-ORGANIZATION

- attributes : label, government, sub-organizations.

*To fulfill this requirement, the formalism should support category or instance in metamodel; i.e. an element should allow to be viewed as a category or an instance.

4 Process Modeling Requirements

This section presents the two main requirementswe encounterd when modeling business processes: representation of processes sharing activities and management of instances that are involved in a process.

4.1 Sharing Activities

Let us consider the case of two processes that share a same activity. The example deals with the fabrication of a product which is made out of two components: a software component and a hardware component, as in a cellular phone, a microwave oven or a computer. The first process describes the development of the software component. It is composed of Design Software, Validate Specifications, and Write Software Code. The second process describes the development of the hardware component. It is composed of activities Design Hardware, Validate Specifications, and Build Hardware. The activity Validate Specifications is an activity of synchronization and is shared by the two processes.

The problem in this example is the representation and identification of the two processes. Each process is composed of three activities, with one of them being in common. Therefore the formalism must offer reuse of parts of process definitions or support some kind of shared variable mechanism.

*To support the representation of business processes in corporate memory, a formalism must offer features to represent processes sharing the same activities.

4.2 Instance Management

To illustrate the problem of instance management, let us assume the example of a window manufacturer who has a special department for building non standard size windows. A fabrication order is established from a client order. A fabrication order defines the size and material of the frame and the size and thickness of the glasses to insert into the frame. The fabrication order is sent to the frame builder and glass cutter teams which execute the order. Then the frame and glasses are transmitted to the window assembly team which insert the glasses into the frame. The problem of this team is to insert the right glasses (size and thickness) into the right frames (size and material). Some frames take more time to build than others, so the frames may be finished in a different order than the glasses are. This problem can be solved by the assembly team by assembling the frame and glasses in conformity with the fabrication order. At the notational level, this requires the possibility of specifying instances of input and output participants.

*To support representation of business processes in corporate memory, the formalism must offer features to represent and manage the related instances needed by different processes.

5 Conclusion

In order to fulffill these requirements we chose conceptual graph formalism. Using this formalism, a corporate memory has been developed at the Research & Development Department of DMR Consulting Group where are memorized methods, know-how and expertise of DMR consultants.

About two hundred business processes have been modeled and from about 80,000 conceptual graphs, we generated more than 20,000 HTML pages in both English and French that can be browsed using commercial Web browsers.