A Metamodel for Knowledge Management

DINH Thi Lan Anh[1], Olivier Gerbé[2]

Abstract— **Knowledge management is a key issue for many public and private organisations. We propose in this paper a metamodel for corporate knowledge representation and management.**

Index Terms— **Metamodel, Model, Knowledge Management.**

I. Introduction

Knowledge management is a very significant problem for many enterprises. According to research of Carnal Havens [8], the professionals used approximately 60% of their time to gather and verify information, 18% in effective work and 22% in reunion, etc. Thus the reduction of time used for seeking and validating information is a major question for many enterprises. Another problem is the loss of competences that is related to the leaving of experienced employees and due to a lack of means to capitalize knowledge of these employees. It is also necessary to optimize the training of new employees in order to shorten the training time and to provide the support necessary to theirs tasks.

Enterprise memory (EM) memorizes the corporative knowledge (CK), which is the know-how of the enterprise such as its businesses processes, its procedures, its policies (mission, payments, standards) and its data (sales, purchases, employee information, etc.). The management of EM raises serious problems of quantity, complexity and diversity. It implies a challenge for the representation and the modelling of this memory.

The Entity-Relationship [4][9] or oriented-object [6][11][14][15] formalisms used for information systems modelling, although very powerful, are badly adapted to knowledge modelling. These formalisms manage in a completely independent way the type level (class) and the instance level. However, knowledge is often shared on these two levels. For example, in the description of a business process, one speaks about an activity 'to extend a loan' and of its tasks 'to fullfill the form and to sign the form' with a description of the activity and its tasks. A question may be raised: What is the difference between activity and task? The entreprise memory must preserve the information and restore it. A more detailed analysis for these formalisms in a EM management context can be found in [7].

Conceptual Graphs introduced by John Sowa in 1984 [18] are a formalism whereby the universe of discourse is modeled by concepts and conceptual relations. Conceptual graphs are a very powerful formalism but its implementation is not easy and still at a prototype step.

We propose in this paper a new metamodel based on the Entity-Relationship formalism but extended with functions of the conceptual graphs. We present the specification of a metamodel for the representation of knowledge and we have developed a prototype implemeting this metamodel. This metamodel implements extensions to the Entity-Relationship model : inheritance on attributes, inheritance on associations and knowledge contextualization.

This paper is organized as follows. Section 2 describes our framework. Section 3 introduces the proposed metamodel and Section 4 presents its implementation. Finally Section 5 summarizes and discusses the results.

II. Architecture

A. Modeling Levels

Lot of work about modelling and metamodelling has been carried out by various groups of standardization Object Management Group[12][13][14][15], ANSI [1] and also by groups interested in models exchange CASE Data Interchange Format (CDIF) [3].

There is nowadays a consensus on an architecture based on four levels and adopted by OMG and CDIF: data, model, metamodel and meta-metamodel. Each of the four levels is briefly described bellow :

1) M3 (meta-metamodel) is the most abstract level in this architecture and describes the basic concepts used for the representation of the lower levels but also for itself.
2) M2 (metamodel) level defines all the vocabulary and also the way used to build models by applying the grammar represented in level M3.
3) M1 is the model level. By respecting the grammar specified in level M2, it defines types and instances that accord with the particular environment and represent the real world.
4) M0 is the real world described at level M1.

It should be noted that in this architecture, only the first three levels (M3, M2, M1) belong to the modelling levels while

[1]DINH Thi Lan Anh, *PhD student of Montreal University (*phone: (514) 340-6893; e-mail: lan-anh.dinh-thi@hec.ca).
[2]Olivier Gerbé, *Prof. of HEC Montréal* (phone: (514) 340-6855; fax: (514) 340-6132; e-mail: olivier.gerbe@hec.ca).

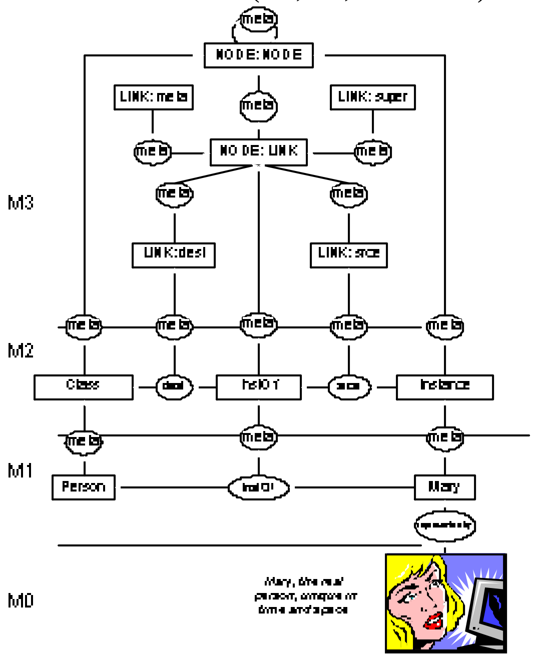
the level M0 does not. This means that types and instances, contrary to what is often perceived, are on the same M1 level.

The achitecture presented above is based on a consensus but important questions about the levels of modeling still remain to be discussed such as: the number of levels really necessary for modelling; need or not of an architecture of more than four levels; the fundamental difference between a model and a metamodel and betwween a metamodel and a meta-metamodel; the possibility for a model to specify another model and also the possibility for a metamodel to specify another metamodel; etc.

We present in next section our proposed meta-metamodel and architecture for knowledge modeling which, we hope, tries to give an answer to the questions listed above.

B. *Proposed Architecture*

Our architecture (See Figure 1) is in conformity with the consensus and is on four levels (M3, M2, M1 and M0).



This architecture gives an answer to another question largely discussed: the double instanciation. In Figure 1, Mary is an instance of Instance in the global context (that is specified by the relation meta on vertical axis), and is an instance of Person in the local context (what is specified by the InstOf relation on horizontal axis).

C. *Meta-metamodel (M3)*

The meta-metamodel provides the language and grammar to describe modelling formalisms. Figure 2 presents the meta-metamodel elements.

The elements of the level M3 are: NODE, LINK, 'super', 'meta', 'srce' and 'dest'. NODE and LINK are NODE and 'super', 'meta', 'srce' and 'dest' are LINK. The NODE and the LINK make it possible to represent the objects of the universe of discourse. A NODE represents an entity, a LINK represents an association. A LINK is defined by a NODE source, and a NODE destination. The relation 'super' makes it possible to classify the NODE and implements the inheritance relation. The relation 'meta' is an instanciation relation and allows indicating the nature of a represented object. Finally the relations 'srce' and 'dest' make it possible to specify the sources and destinations of the LINK.

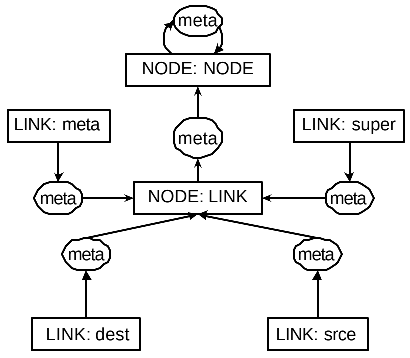


Fig. 2. The meta-metamodel M3.

In order to better understand and better distinguish the modeling levels in our architecture (Fig. 1), the following rules are to be noted:

1) The instanciation relation 'meta' links concepts defined either all on the level M3, or on two adjacent levels in the modeling architecture. In the second case, 'meta' indicates the transition between levels.
2) Each concept is defined in only one level, and once it is defined, it is attached by the relation 'meta' to one and only one another existing concept. Thus the relation 'meta' is not transitive.
3) For the relation 'super', if a concept A is a super-type of B, it states that A is related to B by a link 'super', then B inherits from A all possible attributes and associations.
4) As 'super' is a transitive relation, so if A is super-type of B and B is super-type of C, then A is also super-type C.
5) In the level M2, all instances of NODE, which are thus connected to NODE by link 'meta', represent concepts called non-relational concepts. All instances of LINK, which are connected to LINK by link 'meta', represent concepts called relational whose instances link instances of NODE and/or LINK
6) Each concept defined in the level M2 must be linked by the relation 'meta' with an element of M3.
7) Each association model defined in a given level must comply with the syntactic and semantic rules specified at

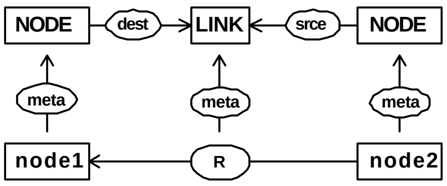


Fig. 3. Semantic Rule.

the higher and adjacent level (fig. 3)

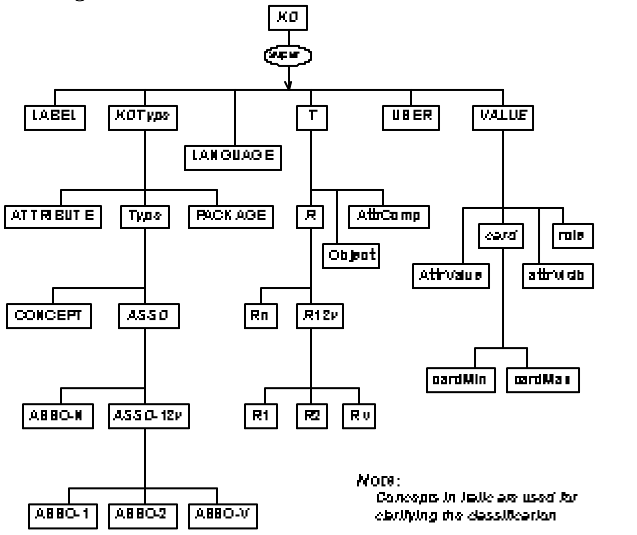
D. *Metamodel*

The metamodel contains all the concepts as well as all the relations existing between them. It can be seen as the vocabulary used to describe the application level. We defined two kinds of concepts for knowledge representation:

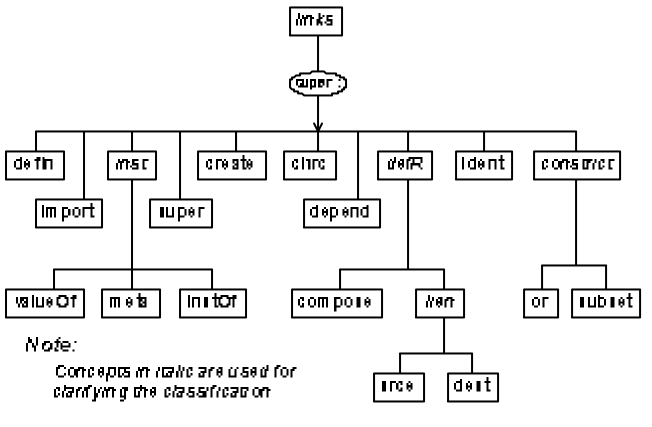
1) The first kind includes concepts called *types*, all being 'instances' of NODE.
2) The second includes concepts called *links* to specify of interelations between types. These concepts are all 'instances' of LINK.

Types and links are organized into two separate hierarchies (see Figure 4 and 5). Figure 6 presents the associations between types and links. In the type hierarchy *KO* (Knowledge Object) is at the top, KO is the root. We sub-classified the types into six categories:

1) *LABEL* that specifies labels used to name concepts by using the link *ident*.



2) *LANGUAGE* presents languages in which a value can be interpreted, e.g, a man named toto will be called 'Monsieur toto' in French but 'Mister toto' in English, etc. It is indicated by link depend.



3) *USER* is the representation of persons such as administrators and users who create and insert data in the data base. It is linked to KO by the link *create*.

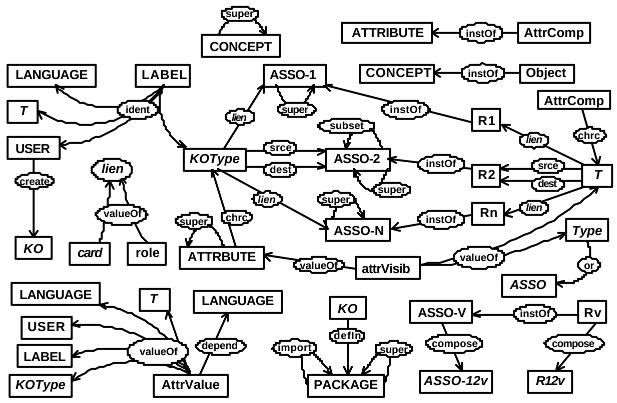


Fig. 6. Diagram of associations between types in level M2

4) *KOType* (Knowledge Object Type) classifies types having same properties. It is divided into three groups: PACKAGE, ATTRIBUTE and TYPE.
5) PACKAGE (packages) is used to organize data using the link *defIn*. Packages can be imported by (link 'import'), included in (link 'defIn'), and be super-type of (link 'super') another packages.
6) ATTRIBUTE (attribute type) is used to represent attributes of types (TYPE). An attribute is linked to its type by a *chrc* link. An attribute can be composed of another attributes.
7) CONCEPT (non-relational concept types) is mainly used to model the application level.
8) Relational concept types: ASSO-1 (1-adic associations), ASSO-2 (binary associations), ASSO-V (virtual associations – See an example in Figure 7), and ASSO-N

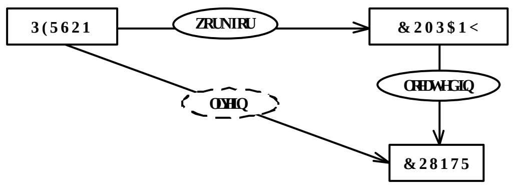
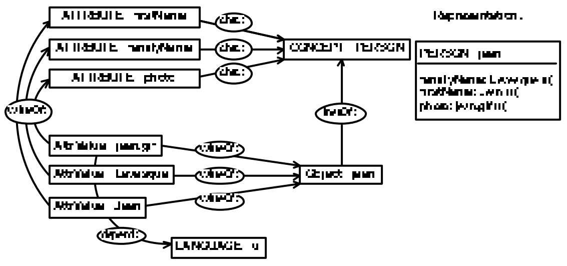


Fig. 7. Example of virtual associations - 'live in'

(n-adic associations). Each relational type is specified by the link *srce* and/or the link *dest*. A virtual association is composed of a sequence of associations, this is specified by the link *compose*.

9) *T* groups all the concepts, which are defined at the model level and represent objects of the real world. It aimed at the representation of the real instance according to the class. Here, composed attributes (AttrComp), objets (Object), and relations (R1, R2, Rv, Rn) are respectively instances of composed attribute types (ATTRIBUTE), non-relational concept types (CONCEPT) and relational concept types (ASSO-1, ASSO-2, ASSO-V, ASSO-N) by using the link *instOf* (See Figure 8).
10) *VALUE* represents the value attached to object attribute

values. A value is different from an object attached to concepts. An object can be changed or transformed into other over the time, but a value always remains itself. For example, let NAME an attribute of the concept PERSONNE and toto (NAME=Papin) and tata (NAME=Papin) which are two instances of PERSONNE. If the values of NAME were seen as objects, then Papin would be an object attached to the two objects 'toto', 'tata'. Therefore if 'Papin' was changed to 'Levesque', then NAME value of 'toto' and 'tata' would become automatically 'Levesque'. This effect could not be what we want. But if NAME values are seen as values, even if the NAME value of toto is changed to 'Levesque', then the NAME value of tata remains the same one: Papin. It is why we distinguish value from object, and as showed in Figure 6, the link from a value to an element of an object is 'valueOf'. VALUE is a



super type of 'AttrValue' (attribute values), of 'role' (values allowing the explicit distinction of axis linking to entities in an n-adic association), of 'cardMin' and of 'cardMax' (cardinality constraints), of 'attrVisib' (visibility of concepts in a package: public, protected, or private). A VALUE is linked to an object by a 'valueOf' link (See Figure 8).

More details on the interaction between concepts can be found in [5].

E. Metamodel evaluation

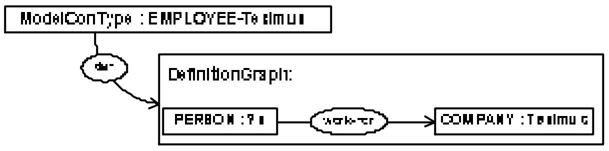
We evaluate our metamodel in face of knowledge representation issues.

1) Classification and partial knowledge: It means that a object can be defined as instance of more than one category, or it can be migrated dynamically from a category into another by the system. Links 'instOf' and 'super' permit respectively to represent the multi-instantiation and the multi-inheritance that are both seen as multi-classification. However, this version of the metamodel does not yet define the rule for dynamic migration as the one in the Conceptual Graphs formalism ([10][18][2]) as for the case like this once: 'if a person (instance of PERSON) works for (work-for) the company Teximus (instance of COMPANY), then this person becomes an employee (instance of EMPLOYEE-Teximus) of this company' or well 'all people working for the company Teximus are its employees' (fig. 9, fig. 10).

2) Relation between categories and/or instances: It means

Type EMPLOYEE-Teximus(x) is
[PERSON : *x] ? (work-for) ? [COMPANY-Teximus]

Fig. 9: GCs - example of type definition



that the formalism allows a category to be related to another category or to an instance.

3) Category or instance: It means that an element can be seen as a category or as an instance depending on the point of view. The metamodel allows the representation of an element as a class or as an instance. Figure 8 shows the concepts PERSON which is viewed as an instance of CONCEPT and also as a classe according to the instance level.

4) Constraint representation: The metamodel permits the reprsentetaion all of the kinds of cardinality constraints on associations as shown in Figure 11. The default values of 'cardMin' and 'cartMax' are respectively '0' and 'unlimited' (value 'unlimited' is represented by 'N', or by '*' as in the UML formalism[16][17])

The links 'subset' and 'or' permit respectively to represent

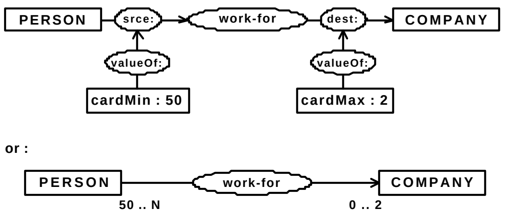


Fig. 11. Examples of Relations and cardinalities

constraints of subset and exclusiveness (See Figure 12 and 13). Figure 12 states that a person manages only an organization for which he works for; and Figure 13 states that a person cannot works for an ORG-l'ETAT organization and manages a ORG-Ltd organization in the same time.

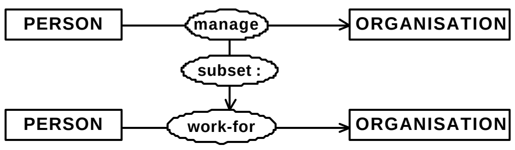


Fig. 12. Examples of subset constraints

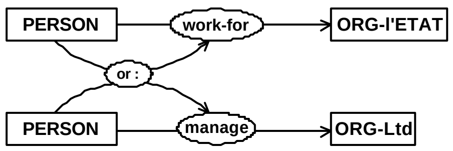


Fig. 13. Examples of 'or' constraints

Constraints on simultaneous existence of a non n-adic relations sequence are represented by introducing virtual association. Figure 7 states that if a person is living in a country, then she works for a company that is located in the country where she is living; Figure 14 illustrates an another example: each professor who is in charge of a course prepares and gives himself this course.

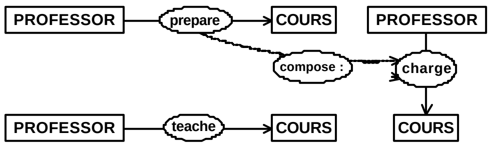


Fig. 14. Examples of simultaneous existence of relations

Our metamodel is based on the conceptual graphs formalism and implements some extensions to the Entity-Relationship formalism Our metamodel allows the characterization of types and objects by simple or structured attributes, the representation of constraints on cardinalities of relational types. Our metamodel supports the simple and multiple inheritances of attributes and of associations. For the contextualization problem, our metamodel supports data packaging similar as the one in UML. With our metamodel we can represent contextual assertions as 'Jean thinks that Marie is pretty'. But this version misses the representation of rule 'if … then …', which could help to implement implication constraints or to migrate elements dynamically. We need also to add the operator 'not' to define negation of model. This operator will help to represent constrained situations like 'each professor cannot follow the course which he gives', or 'there is not any conference room on the 5th floor'. A lot of work remains to be done to improve our metamodel.

III. PROTOTYPE

We developed a prototype using JScript language. The prototype is running in Microsoft IIS environment. The prototype is interfaced with the relational database SQL Server. The knowledge input format is XML (Extensible .Markup Language).

A. Data storage model

We created two tables within SQL Server in order to store knowledge representation models. This implementation choice can be discussed. We chose the most simple solution even if not optimal to make a proof of concept.

Data is organized in the two following categories :

1) Concepts (objects): Relational concepts (each one links concepts called entities); Non-relational concepts (each one does links any concept);
2) Values

As the behaviour of values is quite different from one of concepts, all concepts are stored in a table and all value in another one.

Each concept is identified by one and only one label, defined in one and only one package and in relation 'meta' with only one other concept; moreover one concept of relation either 1-adic or n-adic can be coded by binary relation form, and the two relations 'srce' and 'dest' are used to define binary relations, so all links 'ident', 'defIn', 'meta' are directly coded into table fields. In order to simplify the implementation of the prototype, 'instOf' is directly coded to table field as 'meta'. Figure 15 presents an example of concepts and values.

We can also represent all concepts defined in the metamodel in a XML file to charge in database.

B. Knowledge validation

Information about each concept are the following: the package in which the concept is declared, the label that names it (koCode), the type (koType) of the concept, the linked entities (only for relational concept), the attributes (including the inherited attributes) , and information about its relationships (including all inherited relationships).

The knowledge validation is done at loading time. Following is the process to validate a concept (if a step fails, the followings are cancelled):

1) Verify the existence of the package and check if the concept type is accessible from the current package;
2) If this concept is a relational one, read all its entities in order to check their accessibility from the current package;
3) Verify if not already stored in the current package, then decide insert or not it into the database;
4) Validate information about the attributes and associations.

The validation process of a relational concept is made of three steps.

1) Read all information of the entities in order to check their accessibility from the current package;
2) Check whether it is really instantiated by its type and is a relation of that concept;
3) If it is not yet existent in the database, check whether the realtional concept satisfies the constraints on cardinalities, if it does then insert it into the database and continue to validate information about the attributes.

The validation process of a value is quite simple and done in two steps :

1) Check whether this attribute is really linked to this concept according to the definition model;
2) Insert it into the database if it is not already in the database.

C. Data visualisation

All information about concepts are visualized by the package in which the concepts are defined (See Figure 15).

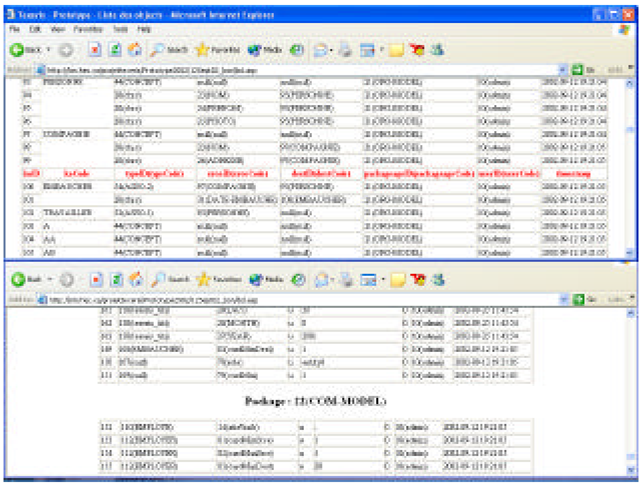


Fig. 15. Example of concept

Concepts can be divided into two groups: one for concepts that are seen as classes and can be instantiated using relation *meta* or *instOf*, and the other one for concepts that are seen as objects and cannot be instantiated. A class can also be observed like either a class or like an object which is instance of its type. Our metamodel allows the representation of relations between concepts, relational or not, then a relational concept can be observed by context like a relation between entities or an entity with its relations.

D. *Prototype evaluation*

The prototype demonstrates that knowledge represented in our model can 1) be stored within a relational database and 2) can be visualized as the object-oriented model with the support of the inheritance on attributes and associations, and with knowledge segmentation.

It ensures the single reference for each concept stored in the database. However, the prototype does not verify constraints concerning the impact of relations 'subset' and 'or', and minima for cardinalities at the instance level. These constraints should be checked after the completion data loading.

Concerning contextualisation problem, the metamodel can represent assertions such as 'Jean thinks that Marie is pretty' but the prototype does not permit to distinguish if Marie is pretty only in thoughts of Jean or in the universal context.

IV. CONCLUSION AND FUTURE WORK

Several models were proposed for knowledge representation but each one has strong points and weak points. For example, conceptual graphs are known for their simplicity and flexibility and are very close to the natural language, but the implementation is very complex. It is why our research focused on the development of a new metamodel to knowledge management system. This metamodel is based on Entity-Relationship metamodel but is extended with somme functionalities of conceptual graphs. The proposed metamodel authorizes the representation of static knowledge . It allows to characterize types/objets by simple or composed attributes, to represent constraints on cardinalities for relational types; it supports simple and multiple inheritance on types, on attributes and on associations. It also supports the knowledge segmentation.

We developed a prototype that has validated the metamodel with a simple implementation. It accepts in input data represented in XML document by checking part of the constraints attached to the data while loading. Data are stored within a relational database.

The work presented here is only a stage in the specification and the realization of a metamodel for knowledge management. It remains several points to study like adding other elements to the metamodel in order to represent dynamic knowledge and rules.

REFERENCES

[1] American National Standard, " Conceptual Graph Standard ", 1999. http://www.bestweb.net/~sowa/cg/cgdpans.htm.

[2] J. Bézivin, O. Gerbé, "Towards a Precise Definition of the OMG/MDA framework", in *Proceedings of the 16th International Conference on Automated Software Engineering*, 2001.

[3] CDIF Technical Committee, Electronic Industries Association. *CDIF - CASE Data Interchange Format : Overview*. January 1994.

[4] P. Chen. The Entity-Relationship Model -- Towards a Unified View of Data. *ACM Transactions on Database systems*, 1(1) : 9—36, 1976

[5] T.L.A. Dinh, " Spécification d'un métamodèle pour la représentation d'une mémoire d'entreprise ", Mémoire de maîtrise-Institut de la Francophonie pour l'Informatique, 2001.

[6] Gregor Engels, Reiko Heckel, Stefan Sauer, " UML - A Universal Modeling Language ", University of Paderborn, 2000. engels|reiko|sauer@upb.de.

[7] Olivier Gerbé, " Un modèle uniforme pour la modélisation et la métamodélisation d'une mémoire d'entreprise", Université de Montréal (UdM). Janvier 2000.

[8] Charnel Havens, " Enter, the chief knowledge officer CIO Canada, 4(10), 1996, pages: :36-42.

[9] Peter Loos, " Capture More Data Semantic Through The Expanded Entity-Relationship Model (PERM) ", University of Münster, Institute of Business Informatics, Grevener Str. 91, D-48159 Münster, Germany –

[10] Régis Monte, " Gestion d'un Système de Mémoire d'Entreprise par Graphes Conceptuels", Mémoire de maîtrise, Université Joseph Fourrier, Grenoble. Mai-Août 2000.

[11] Pierre-Alain Muller, " Instant UML ", Wrox Press Ltd, 1997

[12] Object Management Group. Meta Object Facility (MOF), version 1.4. Formal/02-04-03. April 2002. http://www.omg.org/docs/formal/02-04-03.pdf

[13] Object Management Group. Meta Object Facility (MOF) 2.0 Core Proposal - Revised Submission to OMG RFP ad/2003-04-07. Apr 2003. Object Management Group. OMGUnifiedModelingLanguage Specification. Version 1.3, June1999. http://citeseer.nj.nec.com/307472.html

[14] Object Management Group. *Unified Modeling Language: Infrastructure*. Version 2.0. March 2003.

[15] Object Management Group. *Unified Modeling Language: Superstructure*. Version 2.0. OMG Adopted Specification, ptc/03-08-02. August 2003.

[16] James Rumbaugh, Ivar JACOBSON, Grady Booch , " The Unified Modeling – User Guide ", Addison wesley, 1999.

[17] James Rumbaugh, Ivar JACOBSON, Grady Booch, “ The Unified Modeling – Language Reference Manual ”, Addison Wesley, 1998
[18] J.F. Sowa, “ Conceptual Structures – Information processin g in mind and machine ”, Addison wesley 14472, 1984