Published in the Proceedings of the Fifth International Conference on Conceptual Structures (ICCS'97)

Conceptual Graphs for Corporate Knowledge Repositories

Olivier Gerbé

DMR Consulting Group Inc.
1200 McGill College, Montréal, Québec, Canada H3B 4G7
e-mail: Olivier.Gerbe@dmr.ca

Abstract

The challenge companies will have to meet when making the leap from the industrial era to the knowledge era is the memorization of corporate knowledge and its dissemination to employees throughout the organization. Developing a corporate memory is the means chosen by DMR Consulting Group to capitalize on and manage its expertise in information technology. This paper presents a study conducted to choose a formalism to represent the know-how and methodologies – processes, techniques and learning materials – in corporate memory. It compares modeling formalisms against specific requirements and demonstrates that conceptual graphs are well suited to implement corporate memories. More specifically, we show that conceptual graphs support: (i) classification and partial knowledge, (ii) category or instance in relationship and (iii) category or instance in metamodel.

1 Introduction

Nowadays there is consensus on the value of corporate knowledge. The knowledge is at the center; an employee who must know a work process; a manager who must anticipate market trends; a researcher who must know about the state of the art. Corporate knowledge is made up of strategies, visions, rules, procedures, policies, traditions and people. The knowledge assets and the learning capacity of an organization are seen as the main source of a competitive advantage [3], and the challenge the management of this corporate knowledge [12].

The challenge companies will have to meet is the memorization of knowledge, its storage and, its dissemination to employees throughout the organization. Knowledge may be capitalized on and managed in corporate memories in order to ensure standardization, consistency and coherence. Knowledge management requires the acquisition, storage, evolution and dissemination of knowledge acquired by the organization [15] and computer systems are certainly the only way to realize corporate memories [16] which meet these objectives.

DMR Consulting Group is one of the largest service providers in the information technology (IT) in the world. Mastering the evolution and management of IT is a challenge that requires methods, processes, software tools and training programs, as well as a systematic and consistent approach to implement them. Several products describing methods and processes, such as guides, tools, lecture materials, self-learning courses, reference texts, templates and videos, have been developed [11]. These products are stored in a corporate memory whose data structure and functionality allow for easy consultation, adaptation to the particular needs of an organization, and evolution at acceptable levels of cost [8]. This corporate memory, called the Method Repository, plays a fundamental role. It captures, stores [9], retrieves and disseminates [10]throughout the organization all the consulting and software engineering processes and the corresponding knowledge produced by the experts in the IT domain. During the early stage of the development, the choice of a knowledge representation formalism was identified as a key issue of the development of the Method Repository. That leaded us to define specific requirements for corporate memories, to identify suitable knowledge representation formalisms and to compare them in order to choose the most appropriate formalism.

This paper presents the study we conducted to choose a formalism to represent the know-how and methodologies – processes, techniques and learning materials – for the Method Repository. It compares five modeling formalisms, extended entity-relationship, object-oriented (UML), relational model, classic – a KL-One-like formalism and conceptual graphs, against our specific requirements. This study demonstrates that conceptual graphs are particularly well suited to implement corporate memories since they support: (i) classification and partial knowledge, (ii) category or instance in relationship and (iii) category or instance in metamodel.

The paper is organized as follows. Section 2 defines specific requirements for corporate memories. Section 3 compares both traditional formalisms used in data modeling and formalisms used in knowledge representation and demonstrates that conceptual graphs support our predefined requirements. Finally, Section 4 concludes and provides some technical information about the Method Repository we have developed.

2 Requirements

Modeling techniques aim at defining simplified, computerized models of real or hypothetical worlds, in order to gather and store information about them. Defining model starts with the identification and definition of categories of things and of relationships between these things. They help describe the application domain. Things are interrelated and organized into categories according to established similarity criteria. Models must reflect the structure and present these categories and their interrelationships.

In order to compare modeling formalisms, we looked at the three specific concerns, encountered when we began to study how to represent methods, pro-

cedures and techniques, and which 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 in this paper above questions may be translated into three requirements:

- Classification and partial knowledge (2.2);
- Category and/or instance in relationship (2.3); and
- Category or instance in metamodel (2.4).

This section is organized to first introduce briefly the basic notions and terminology used in modeling techniques, and then to detail each of our three requirements.

2.1 Basic Notions

The main elements used by modeling techniques are categories or types, instances and relationships. A *category* represents a set of things that share the same properties: attributes, relationships and behavior. An *instance* of a category is a thing that conforms to the definition of the category. A *relationship* is an association between things or categories.

Let Employee be a category that defines what an employee is. An employee may have attributes like employee number, name, etc. An employee also has a relationship with a company he or she is working for. Figure 1 illustrates this example. EMPLOYEE and ORGANIZATION are categories, and instances of these types should be John, Paul, IEEE, UNU (United Nations University), etc. 'Works for' is a relationship between the two categories.

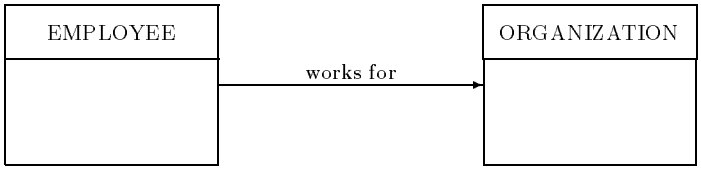


Figure 1: Model containing two categories and one relationship.

Inheritance and classification are two other important notions used in modeling techniques. *Inheritance* is a kind-of relationship between categories. A

category inherits the properties of a higher category. *Classification* is the hierarchy of categories built upon the inherited properties. For example, the mammal classification is a hierarchy that defines types of species.

2.2 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. 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?

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. Figure 2 illustrates the different hierarchies resulting from these three criteria.

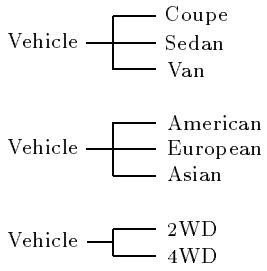


Figure 2: Different vehicle hierarchies according to three criteria.

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, as shown in Figure 3. We can see that the number and the volume of all the possibilities may rapidly become difficult to manage.

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?

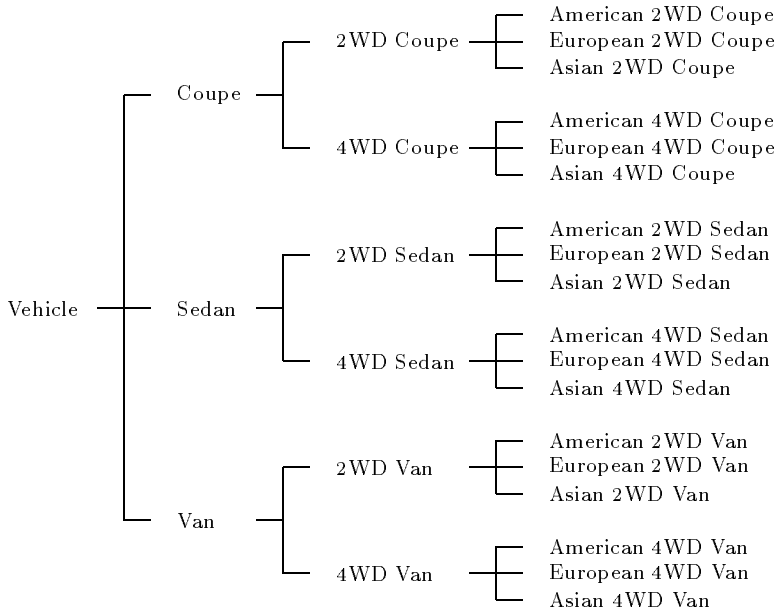


Figure 3: A unified vehicle hierarchy comprising all three criteria.

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.

2.3 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 (see Figure 4).

In most modeling techniques, it is very difficult to express the fact 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 as shown in Figure 4.

However, what we want to express is: "Each instance of UNU Employee has

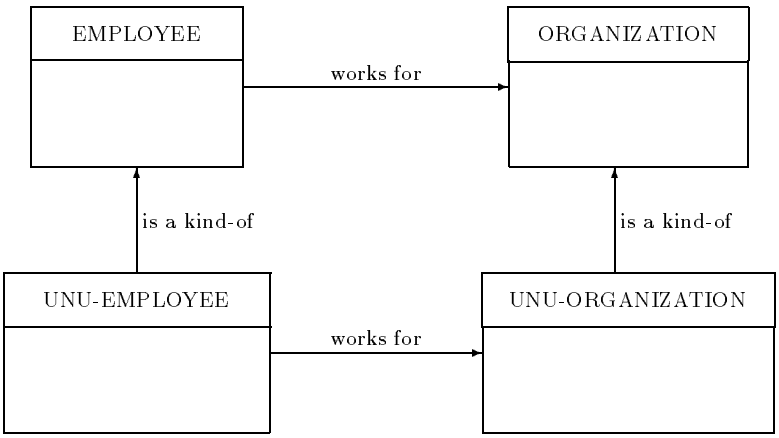


Figure 4: Complete Model.

a relationship 'works for' with UNU, instance of the type Organization." This is stated by the following predicate:

$$\forall x, \text{UNU-EMPLOYEE}(x) \wedge \text{ORGANIZATION}(UNU) \wedge \text{works-for}(x, UNU)$$

In most modeling techniques, this is stated by the following two expressions where the first expression states that a UNU employee works for a UNU organization and the second expression states that any UNU organization is the UNU:

$$\forall x, \exists y, \text{UNU-EMPLOYEE}(x) \Rightarrow \text{UNU-ORGANIZATION}(y) \wedge \text{works-for}(x, y)$$

$$\forall z, UNU - ORGANIZATION(z) \Rightarrow z = UNU$$

These two formulations are equivalent. However, the first formulation is simpler and therefore preferable.

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.

2.4 Category or instance in Metamodel

We seek to develop models that are formal descriptions of objects or notions in order to make sound and complete inferences from this model. 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. But saying that FEDERAL-ORGANIZATION is a category implies that FEDERAL-ORGANIZATION is an instance of a model component CATEGORY. Figure 5 illustrates this situation where FEDERAL-ORGANIZATION may be seen as a category that carries its associated semantics, or as an instance of the category CATEGORY, depending on the perspective.

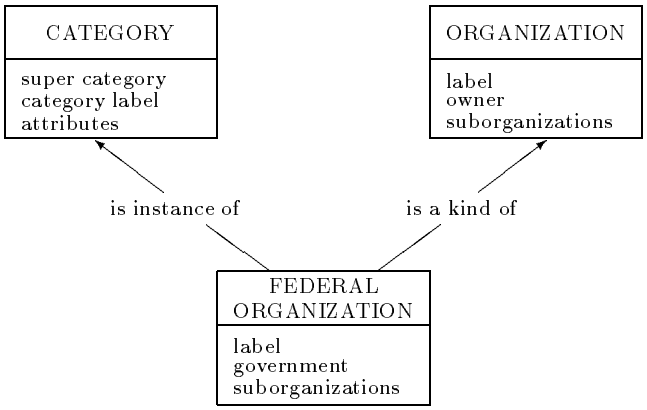


Figure 5: Category or instance.

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.

3 Considered Formalisms

This section examines five formalisms we believe the most promising: three well-known formalisms frequently used in the domain of information technology, Extended Entity-Relationship [4], Object-Oriented [1], and Relational Model [5] formalisms; and two formalisms frequently used in the domain of knowledge representation, Classic: a KL-One-like language [2] and Conceptual Graphs [14]. For each of them, we introduce pertinent notions and discuss to which extent they fulfill our requirements.

3.1 Extended Entity-Relationship Formalism

The Extended Entity-Relationship formalism was originally developed by Peter Chen [4] in 1976 and was later extended [6, 7].

The Extended Entity-Relationship formalism supports categories through entities and relationships; there is no explicit component in the formalism to represent instances. All the knowledge is specified at the entity level. Entities are represented by boxes with the entity's name at the top, and relationships by ovals with two lines that link the entities concerned. Figure 6 presents the Extended E-R diagram corresponding to "employee works for organization". This means that all employees will be said to work for some organization.

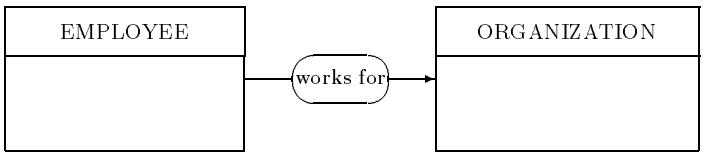


Figure 6: Extended E-R: Employee works for organization.

Classification and Partial Knowledge. There is no means to represent knowledge about instances. In the Extended E-R formalism, an instance is implicitly an instance of one and only one entity and there is no support for migration of instances and partial knowledge.

Category and/or instance in relationship. As instances are not represented in the formalism, there is no possibility to specify constraints at the instance level. The commonly used solution in this case is to define an entity that has only one instance, to express the relationship at the entity level (Figure 7), and to add an explanatory note.

Category or instance in Metamodel. The Extended E-R formalism is applicable at one and only one level. The only way is to define two diagrams, one for each level, and to link them by an explanatory note (Figure 8).

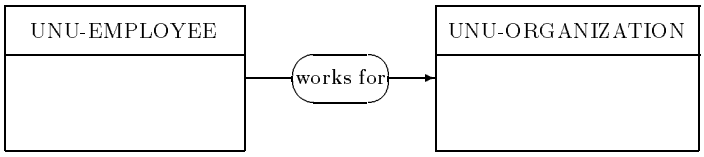


Figure 7: Extended E-R: Category and/or instance in relationship.

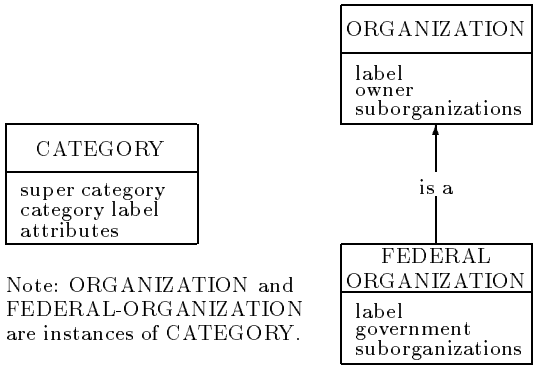


Figure 8: Extended E-R: Category or instance in Metamodel.

3.2 Object Oriented Formalism

As representative of object oriented formalisms, we chose the Unified Modeling Language [1], a language for specifying, visualizing, and constructing the artifacts of software systems, as well as for business modeling, that was developed by Grady Booch, Jim Rumbaugh and Ivar Jacobson from the unification of the Booch, OMT and OOSE methods.

The Unified Modeling Language defines instances called objects and categories called classes, and supports classification. In Unified Modeling Language notation, classes are represented by rectangular boxes, objects are represented as classes with an underlined label and relationships are represented by lines. "Employees work for organization" is represented as in Figure 9.

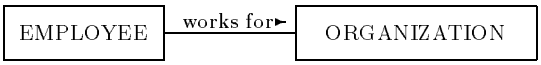
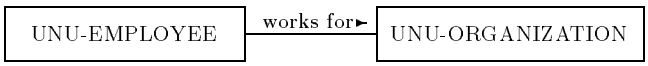


Figure 9: UML: Employees works for Organization.

Classification and Partial Knowledge. Although the ordinary UML semantics assume multiple inheritance, no multiple classification, and no dynamic classification, different semantics can be permitted by identifying semantic vari-

ation points that users and tools could understand.

Category and/or instance in relationship. Relationships are defined at the class level. Similar to the Extended E-R formalism, the solution is to define a class with only one instance as shown in Figure 10 to represent singleton classes.



Note: UNU-ORGANIZATION has only one instance UNU

Figure 10: UML: Category and/or instance in relationship.

Category or instance in Metamodel. Formalisms are different to represent classes and objects and there is no means to state that an instance may also be seen as a class. The solution is to define a class and an object with the same label as shown in Figure 11.

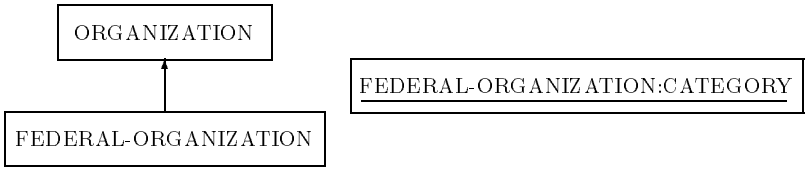


Figure 11: UML: Category or instance in Metamodel.

3.3 Relational Model Formalism

The relational model [5] is probably the simplest and most flexible formalism of the compared formalisms, but unfortunately its current implementations in database management systems reduce its power.

The relational model defines tuples (instances) and tables (categories). Most relational database management systems use the same formalism to describe tuples and tables, and the table descriptions are stored in a set of tables generally identified as system tables or metabase. Relationships are expressed by using join attributes that link tables. "Employees work for organizations" is represented in Figure 12 by two tables, and in the Employee table the attribute OrgId is a foreign key that establishes the relationship between an employee and the organization he or she works for provided that no null values are allowed to ensure referential integrity.

Classification and Partial Knowledge. A tuple is a row of one and only one table and there is no specific means to implement partial knowledge, but the relational view mechanism and the use of null values may be used to implement partial knowledge management though they introduce other problems.

Organization	OrgId	Name	...

Employee	EmpId	Name	OrgId

Figure 12: Relational Model: Tables and Join Attributes.

Category and/or instance in relationship. Relationships are defined at the instance level and implemented using join attributes. For the UNU example, the solution is to define a UNU Employee table with a join attribute, OrgId, always equal to the organization identifier of UNU as shown in Figure 13.

Organization	OrgId	Name	...
	123	UNU	

UNU Employee			OrgId
			123
			123

Figure 13: Relational Model: Category and/or instance in relationship.

Category or instance in Metamodel. The descriptions of tables created in a relational database management system are normalized and stored in system tables (Figure 14). Like any other tables, system tables may be manipulated using SQL statements, and using a SELECT clause in a FROM clause like in SELECT Government FROM (SELECT TableName FROM Attribute WHERE AttributeName='Government') would be a solution, but current implementations do not authorize such SQL statement.

Table	Name	...	...
	Organization		
	Federal Organization		

Attribute	TableName	AttributeName	AttributeType
	Federal Organization	Name	String
	Federal Organization	Government	String
	...	...	...

Federal Organization	Name	Government	...

Figure 14: Relational Model: Category or instance in Metamodel.

3.4 Classic: A KL-One-like Language

Classic [2] is a KL-One-like system; it is a frame-based knowledge representation system. Knowledge is represented by describing objects using frames, as opposed to asserting arbitrary logical sentences. Classic defines instances called individuals and categories called concepts. Relationships between individuals are implemented using attributes, called roles in Classic. Figure 15 shows the definition of the concept EMPLOYEE that represents "employee works for an organization". EMPLOYEE is defined as a subtype of PERSON that has an attribute works-for which represents the relationship with the concept ORGANIZATION.

```
EMPLOYEE ⇔ (AND PERSON(ALL works-for ORGANIZATION))
```

Figure 15: Classic: Concepts and relationships.

Classification and Partial Knowledge. Classic supports multiple classification. An individual can satisfy more than one concept. Classic also supports partial knowledge through dynamic classification. When a new individual is introduced into the system, classification is invoked to find all the concepts that are satisfied by the individual.

Category and/or instance in relationship. Relationships between individuals are implemented using roles. Operators have been defined to express restrictions on roles. One of these operators, FILLS, specifies that a role is filled by some specified individual. Figure 16 shows the definition of the concept EMPLOYEE as a person who works for a company, and the definition of UNU-EMPLOYEE as an employee that works for UNU.

```
EMPLOYEE ⇔ (AND PERSON(ALL works-for ORGANIZATION))
EMPLOYEE-UNU ⇔ (AND EMPLOYEE(FILLS works-for UNU))
```

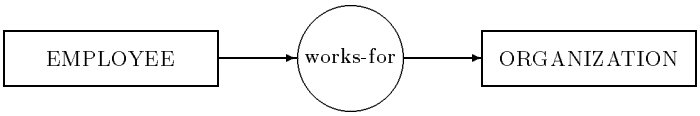
Figure 16: Classic: Category and/or instance in relationship.

Category or instance in Metamodel. Classic distinguishes individuals from concepts and does not support the notion of metaconcept. Therefore, the system is not suitable in situations where some individual may be viewed as a class with instances.

3.5 Conceptual Graphs

Conceptual graphs are a formalism whereby the universe of discourse can be modeled by concepts and conceptual relations. A concept represents an object of interest or knowledge. A conceptual relation makes it possible to associate

these concepts. Conceptual graphs were developed by John Sowa in the early 80s [14]. They are a system of logic based on the existential graphs of C.S. Peirce [13] and semantic networks. Conceptual graphs define knowledge both at the type and instance levels. Concepts are represented by boxes and relationships by circles with arrows that link the concepts associated. Figure 17 represents the sentence "There exists an employee that works for an organization"




```
[EMPLOYEE]->(WORKS-FOR)->[ORGANIZATION]
```

Figure 17: Conceptual Graphs: Concepts and Relationships.

Concepts may be categorized based on the type of conceptual relations they have with other concepts. Concept types define these categories. A concept type is defined by a definition graph to which any instance of that concept type must comply with. Figure 18 presents the definition graph of EMPLOYEE that means that all employees are persons that work for some organization.

```
Type EMPLOYEE(x) is
  [PERSON:*x]->(WORKS-FOR)->[ORGANIZATION].
```

Figure 18: Conceptual Graphs: Type Definition.

Classification and Partial Knowledge. Conceptual graph theory defines one type hierarchy; this hierarchy is a lattice with the universal type $\top$ at the top and the absurd type $\bot$ at the bottom. Multiple inheritance and multiple classification are supported by conceptual graph theory. Multiple inheritance is often difficult to use in a practical way, especially when dealing with hundreds of concepts. Multiple classification allows multiple perspectives, each perspective with its own vocabulary.

If we go back to the car dealer example, we do not need multiple inheritance. This situation corresponds to multiple perspectives: number of seats, propulsion and origin, and the use of multi-classification is certainly better. For example, if car #123 is an American 4WD Sedan, then we will create three concepts `[SEDAN:#123]`, `[4WD:#123]`, `[AMERICAN:#123]` each of them corresponding to the perspectives, number of seats, propulsion and origin. For partial knowledge, known information is stated according to its relevant perspective independently of other information.

Category and/or instance in relationship. Instances must conform the definition of the concept type to which they are associated. For instance, in the case of UNU employees, the definition graph of UNU-EMPLOYEE is:

```
Type UNU-EMPLOYEE(x) is
  [EMPLOYEE:*x]->(WORKS-FOR)->[ORGANIZATION:UNU].
```

Figure 19: Conceptual Graphs: Category and/or instance in relationship.

Category or instance in Metamodel. A concept is the association of two markers: one for type and one for instance. Changing the position of an instance marker from left to right promotes it to a type marker. Figure 20 illustrates how an individual may be seen either as a type or as an instance.

When the marker is on the left side [CONCEPT-TYPE:FEDERAL-ORGANIZATION], it represents an instance of the right side type, and when the marker is on the right side, it represents a category as in [FEDERAL-ORGANIZATION:ENV-AGENCY].

```
[CONCEPT-TYPE:FEDERAL-ORGANIZATION]-
  (SUBTYPE)->[CONCEPT-TYPE:ORGANIZATION]
  (SYMB)<-[CATEGORY-LABEL:'federal-organization']
  (DEFINED-BY)<-[GRAPH: Type FEDERAL-ORGANIZATION(x) is
                          [ORGANIZATION:*X]-
                            (SYMB)<-[LABEL:*]
                            (ATTR)<-[GOVERNMENT:*]
                            (MEMBER)<-[FEDERAL-ORGANIZATION:*] ].

[FEDERAL-ORGANIZATION:ENV-AGENCY]-
  (SYMB)<-[LABEL:'environment agency']
  (ATTR)<-[GOVERNMENT:'Canada'
  (MEMBER)<-[FEDERAL-ORGANIZATION:AIR-AGENCY].
```

Figure 20: CG Formalism: Category or instance in Metamodel.

3.6 Summary

Table 1 presents a summary of the formalisms compared.

	Classification Partial Knowledge	Category or instance in Relationship	Category or instance in Metamodel
E-R	No	No	No
OO	Yes	Using Composite	Using Same Label
Relational	Using View	Yes	No
Classic	Yes	Yes	No
CG	Yes	Yes	Yes

Table 1: Summary.

4 Conclusion

This paper has compared five knowledge representation formalisms. Our aim is to develop corporate knowledge repositories. This comparison has shown how conceptual graphs are a response to the specific requirements involved in the development of corporate knowledge repositories.

Using this formalism, a corporate knowledge repository [8] is developed and implemented at the Research & Development Department of DMR Consulting Group Inc. in order to memorize the methods, know-how and expertise of its consultants. This corporate knowledge repository, called Method Repository, is a complete authoring environment used to edit, store and display the methods used by the consultants of DMR. The Method Repository has three components: a Method Knowledge Acquisition facility, a Conceptual Graph Knowledge Base and a Knowledge Dissemination engine. Method Knowledge Acquisition is an ad hoc module based on the method metamodel that allows method developers to create, maintain and adapt methods. The CG Knowledge Base is the core of the environment; it is a knowledge engineering system based on conceptual graphs. The Knowledge Dissemination facility provides view mechanism [10] using available technologies such as HTML files, SGML files and RTF files, among others.

In June 1996, five methods were commercially delivered: Information Systems Development, Architecture, Benefits, Technical Infrastructure and Estimating, in hypertext format generated from conceptual graphs. From about 80,000 conceptual graphs, we generated more than 100,000 HTML pages that can be browsed using commercial Web browsers. The power of conceptual graphs in terms of expressiveness and flexibility for corporate knowledge modeling has been demonstrated through the development of the Method Repository now being used in the field by thousands of consultants at DMR.

References

[1] G. Booch, J. Rumbaugh, and Jacobson I. *Unified Modeling Language, Version 1.0.* Rational Software Corporation, 1997.

[2] R. J. Brachman and al. Living with classic: When and how to use a kl-one-like language. In John Sowa, editor, *Principles of Semantic Networks: Exploration in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann, 1991.

[3] Prahalad C. and Hamel G. The core competence of the organization. *Harvard Business Review*, pages 79–91, 1990.

[4] P. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[5] E. F. Codd. A relational model of data for large shared data banks. *Communications of ACM*, 13(6):377–387, 1970.

[6] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. The Benjamin/Cummings Publishing Company Inc., Redwood City, California, 1989.

[7] G. Engels, M. Gogolla, U. Hohenstein, Hülsmann K., Löhr-Richter P., Saake G., and Ehrich H.-D. Conceptual modelling of database applications using an extended er model. *Data & Knowledge Engineering*, 9(2):157–204, 1992.

[8] O. Gerbé et al. *Macroscope Architecture: Architecture of DMR Repository*. DMR Group Inc., Montréal, Québec, 1994.

[9] O. Gerbé, B. Guay, and M. Perron. Using conceptual graphs for methods modeling. In *Proceedings of the 4th International Conference on Concpetual Structures*, Sydney, 1996.

[10] O. Gerbé and M. Perron. Presentation definition language using conceptual graphs. In *Peirce Workshop Proceedings*, Santa Cruz, California, 1995.

[11] DMR Consulting Group. *DMR Macroscope*. DMR Consulting Group Inc., 1996.

[12] C. Havens. Enter, the chief knowledge officer. *CIO Canada*, 4(10):36–42, 1996.

[13] C. S. Peirce. *Collected Papers of C.S. Peirce*. Harvard University Press.

[14] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[15] E. W. Stein. Organizational memory: Review of concepts and recommendations for management. *International Journal of Information Management*, 15(1):17–32, 1995.

[16] G. van Heijst, R. van der Spek, and E. Kruizinga. Organizing corporate memories. In *KAW 96 Proceedings*, Banff, 1996.