Stephanie Lacroix, Jean-Charles Marty, Christophe Roche
Laboratoire de Génie Informatique de Savoie, France

OK : A Model of Ontologies by Differentiation

Abstract : This paper introduces a new model for the representation of ontologies. We expose our own definition of the term ontology, and we define the concepts, objects, attributes and classes manipulated. Then, we introduce the notion of “specific difference”, which is central in our work. Each concept owns a specific difference, which is an attribute belonging to it, and which constitutes its main characteristic. While representing a particular ontology, it allows to specify which characteristic is the most representative of a concept, in order to differentiate it from its siblings and to specialise it from its parent. An ontology by differentiation (using the specific difference) is therefore constituted by a hierarchy of concepts, each concept being defined from a more abstract one by the addition of a specific difference. The construction of such an ontology also implies the use of a methodology for the acquisition of knowledge. This aspect of our work is therefore developed in this paper. The methodology is defined as a combination of diverse existing techniques, which we adapt to the acquisition of ontologies by differentiation.

1. Objectives

The description of large systems (for example, in the concurrent engineering domain) implies the modelization of different components needing to share and exchange knowledge. This approach leads to focus on the communication of information problems, and on the mutual understanding of the same piece of information, among different components.

The communication aspects are treated broadly in the multi-agent systems, where agents communicate knowledge to each other through a standard communication language, e.g. KQML (Finin et al., 1993). Very often, this knowledge is described through concepts represented by their names, such as in the STEP project. This approach works as long as the different agents associate the same meanings to the same names. Actually, this is not the case when two specialists of different domains communicate. In fact, the semantics of the different exchanged concepts is not described. The mutual understanding is therefore superficial, and leads very often to system failure when an ambiguous term is used. That is why, in our research, we are concerned with the meaning of the knowledge exchanged by the agents.

We believe that it is better to define a knowledge representation which allows the expression of knowledge semantics in respect to sound properties, in the logical sense. We therefore chose a model based on ontologies. Ontologies are often defined as “explicit specifications of conceptualisations” (Gruber, 1992). They can be seen as a reference that the different agents can consult to find the meaning of a term. In the concurrent engineering domain, ontologies have been used as the basis of many projects: for example SHADE/PACT (Kuokka et al., 1994), TOVE (Gruninger et Fox, 1994), or the Enterprise Ontology (Uschold et al., 1997). The goal of ontologies, in this context, is to provide a shared terminology to the different agents of an enterprise (Olsen et al., 1994). This shared terminology includes the objects of interest in different domains related to enterprise, and provides definitions for the meaning of the terms. In more general domains, let us also quote the Mikrokosmos ontology (Mahesh, 1996).
2. Definition of Ontologies

One can define three main levels of knowledge: domain knowledge, knowledge describing real-world situations, or knowledge that results from deductions or inferences. Each level corresponds to a particular logical formalisation. The domain level describes knowledge that constitutes the ground of the domain and never changes. This includes mainly the concepts and the permanent relations between them (for example, “a square is a kind of rectangle”, or “a knee is a part of a leg”). It relies on an ontological logic, for example a terminological logic. The real-world level expresses the relations between objects (for example “this cup is on this table”). These relations are applied to objects (instances), they are true at a given time and can therefore change. The real-world level relies on a propositional logic. Finally, the inference level is aimed at the expression of inference relations on concepts or objects (for example, “if x is a triangle, and if the sides of x have the same length, then x is equilateral”). The underlying logic is a reasoning logic, like a predicate logic.

We are interested in the first level, which is the domain level. In other words, our aim is to express knowledge about the concepts of a domain. This includes the definition of the semantics of the concepts, and fundamental relations between these concepts, such as kind-of or part-of links in the above examples.

Taking this approach, we need to represent concepts through domain knowledge. There are many knowledge representation languages allowing this: for example KL-ONE (Brachman et Schmolze, 1985), LOOM (Loom, 1995), or Ontolingua (Gruber, 1992). Nevertheless, these languages don't offer sufficient sound properties in terms of consistency: they do not avoid conflicts between concept definitions. For example, they allow multiple inheritance. For us, multiple inheritance is “dangerous”, because when a concept inherits from two (or more) different concepts, it inherits all properties of all its parents. In some cases, these properties can be contradictory (for example a same attribute can have different values according to the parent, or two attributes can be incompatible). If the inherited properties are not consistent, they generate conflicts in the concept definition. Sometimes, a resolution of such conflicts is prepared; but in most cases, it is not satisfying. Then, most of the times, such a conflict proves that there is a conception problem in the definition or in the organisation of the concepts.

We therefore propose another model, which we call OK for Ontological Knowledge. In OK, we modelise ontologies in a particular way, and we call these results “ontologies by differentiation”. As a matter of fact, a concept is defined by differentiation: it is defined from a more abstract concept by adding a specific difference. It is the specific difference that enables one to specialise this concept from its parent, and to distinguish it from its siblings. All concepts of each domain are therefore organised in a hierarchical structure.

In this context, our own definition of the term ontology is the following: for us, an ontology is a way to express the semantics of the terms of a domain, through a taxonomy of the concepts of this domain. The semantics of a concept are given by the characteristics of its parents in the taxonomy, and by a particular characteristic, representative of this concept, called its specific difference. Specific differences are expressed by attributes. As seen further, this definition implies a simple hierarchy; so there can be no inheritance conflict in a concept definition.

We now develop deeper our model. First, we need to define properly the central notions of our model: concept, attribute and object. Then, we explain in more detail the role of the specific difference. Lastly, in order to create sets of objects, we define classes referring to them.
3. Definition of Concepts, Objects, and Attributes

We are interested in descriptive knowledge, knowledge which can be described in terms of a finite set of attributes. Then, a concept is defined by a finite set of properties, which corresponds to its meaning. A concept is different from an object: the term concept is applied to abstract ideas, whereas the term object is applied to anything of the real world. As an example, we could say that “cat” is a concept, whereas “Garfield” is an object. Objects are instantiated from particular concepts (the closest to the real world). All concepts of a same domain are organised into a simple hierarchical model. In the object-oriented knowledge representation vocabulary, a concept is usually named a class, and an object is named an instance.

In our approach, there are different types of concepts, according to their level of abstraction: the root concept, the intermediate concepts, and the terminal concepts. The root concept is the most abstract, it has no parent concept, and is at the top of a hierarchy. There is a root concept for each domain. Intermediate concepts have a parent and sons; they are used in order to successively refine abstract ideas. Terminal concepts represent the nearest to reality concepts; they are at the bottom of the hierarchy, and have a parent but no sons. As these terminal concepts represent the last possible refinements of the domain, their definition is the most specialised (each time a concept is refined, there is at least one more attribute which appears in the new definition). The objects of the domain can be instantiated only from terminal concepts, because they are the more detailed and consequently the closest to reality. In fact, if an object is first instantiated from a concept \( C_1 \) which is not terminal, one has to choose which terminal concept defined in the sub-tree of \( C_1 \) is the most appropriate for this object. Of course, if none of the terminal concepts is representative of the object, one can also refine this sub-tree and add new intermediate and terminal concepts.

Attributes are used in order to express the definition of the concept. There are two main types of attributes: some attributes are essential to the definition of concepts, other ones are secondary. For example, if we modelise the domain of vehicles, an essential attribute of the concept car would be “motorised” ; a secondary attribute of this same concept would be the number of doors. The set of essential attributes of a concept defines its meaning (these are the essential attributes defined at the level of the concept, and the essential attributes inherited from its parents). Among the essential attributes, some are used in order to differentiate concepts from other close concepts, and to specialise them from their parent: we call them specific attributes or specific differences. In fact, when a concept is defined from a parent concept, at least one additional attribute appears in its definition, i.e. its specific difference (and eventually other attributes).

4. The Specific Difference

A specific attribute is an essential attribute which represents what is typical of a concept, when comparing it to its parent or its siblings. The definition of a concept is therefore given by the union of the definition of its parent and its specific difference. Such an approach is more consistent to us than the existing ones, because it implies a simple hierarchy. In fact, if a concept is defined from its parent with a specific attribute, it means that its siblings cannot have this specific attribute. Then, as a concept or an object always inherits all its parents attributes, multiple inheritance would necessarily imply a contradiction on at least one attribute.

For example, the concept “book” could be defined from the concept “document” with the specific attribute “has an editor”. That means that this attribute cannot be applied to all of the other sons of the concept “document”, in other words all documents which are not books
don't have an editor.

Our model can be summarised by the following picture:

![Diagram of a hierarchy of concepts defining an ontology by differentiation](image)

Finally, a concept is defined by the following parameters: its name, a comment (a short text), the name of its parent, its type (root, intermediate or terminal), its specific attribute and the list of its other attributes. If the concept is terminal, a referent is also asked (it means an example of possible instances of that concept). This referent can be a plan, a photo, a text, ... Then, the syntax of the definition of a concept is the following:

```plaintext
(defConcept ConceptName
  (comment "text")
  (type {root / intermediate / terminal })
  (is-a NameOfParent)
  (with specificAttribute)
  (attributes listOfAttributes)
  [(referent pointerOnReferent)]
)
```

### 5. Example

Our model of ontologies by differentiation is currently used as the basis for the development of ontologies, for the Eureka project PVS98, EU 1439. This project is dedicated to the product management domain. For this project, we also need to build different ontologies in this domain. Here is an example of such an ontology, concerning the classification of articles. Of course, the whole ontology of articles is not developed in this paper, because it involves the development of approximately eighty concepts. Only the most abstract concepts are therefore defined here.

The specific differences are annotated on the hierarchical links. As specified before, if a concept is defined from its parent with a specific difference, it means that its sibling does not have this specific attribute. For example, the concept physical-article is defined from the concept article by addition of the specific attribute "concrete". Then, the sibling concept, abstract-article, cannot have this attribute concrete: more, its specific difference is constituted by the attribute "not concrete". Let us also notice that the concepts have other attributes (secondary attributes), which are not developed here.
6. Definition of Classes

The representation of concepts in our model of ontologies by differentiation enables us to organise them in accordance to their essence. Their essence is defined by their structure, and does not depend on the possible values of their attributes. That means that concepts, and objects defined from terminal concepts, are organised depending on their sense. Now, we can imagine that we need to group objects differently, according to attributes which are not used as specific differences. The couples composed of the attributes and their values are linked to the state of the objects (e.g., the set of objects whose colour is red). This implies a new notion in our model: the classes. These classes are the way to create sets of objects, by expressing logical properties on them. Classes can refer to groups of objects which are not necessarily "close" in the hierarchy, for example which do not depend on the same terminal concept.

Concepts and classes are often used to design the same thing in the object-oriented literature; but in fact it is not the same. Of course, both can be seen as sets of objects, but the way they are built and their aim are completely different. Concepts are aimed at the expression of the semantics of the terms in a domain, that is to say the real essence of the vocabulary used in the domain. On the other side, classes enable us to create groups of objects verifying a given property; but this property is not necessarily linked to the semantics of the objects. Properties are expressed in a logical manner.

For example, if we create an ontology of the domain of fruits, the concepts related to different fruits will be organised in a hierarchy, depending on several specific differences contributing to express their semantics. But the colour of a fruit is an attribute which is not a differentiating attribute; it is only a secondary attribute. It is applied to concepts and therefore to their instantiated objects, but it does not participate to their definition. So, if we want to obtain the set of all yellow fruits (in fact the fruits whose colour is yellow), we have to build a class. This class is the set of all objects defined in the domain of fruits, which have the attribute colour and for which this attribute has for value yellow.

The logical properties of a class are first-order predicates. They are built using the
following logical operators: "and", "or", "not". Let us notice also that the name of a concept can be associated to the definition of a class. This concept can be indifferently root, intermediate or terminal. In this case, the syntax for the definition of a class is the following:

```
(defineClass Class(x) (ConceptName(x) logicalProperty))
```

For example, if we want to define teenagers, we will build the set of human persons whose age is between twelve and eighteen. We cannot define a sub-concept of the concept Human, because the age is not a structuring knowledge. So we have to define a class for teenagers. These teenagers are represented by all objects instantiated from the concept Human, which verify a logical property concerning the value of the attribute age:

```
(defineClass Teenager(x) (and (Hum(a)(x)(age x y) (<= y 18) (>= y 12))
```

As classes are sets of objects, we also use the usual operators referring to sets (intersection, union). In fact, a class can be defined as the intersection or the union of several existing classes, with a logical property:

```
(defineClass Class1(x) (ClassName1(x) \cup ClassName2(x) logicalProperty))
```

Taking the example of teenagers, we can build the set of people whose age is less than eighteen, and who don’t go to school. The activity of people could be represented by an attribute "activity", applied to the concept Human, and whose values belong to the set \{school, work, retirement, unemployment\}. The definition of the classes is in this case:

```
(defineClass Teenager(x) (and (Human(x)) (age x y) (<= y 18) (>= y 12)))
(defineClass NotScholar(x) (and (Human(x)) not (activity x "school")))
(defineClass NotScholarizedTeenager(x) (Teenager(x) \cap NotScholar(x)))
```

7. Other Works Around this Subject

This model is currently implemented in the Smalltalk language. This includes the implementation of ontologies by differentiation with concepts, attributes, objects and classes, and the integration of new knowledge into an existing hierarchy. Two aspects of the integration of knowledge are developed. The first one concerns the methods of acquisition of new knowledge from documents and experts. These methods are dedicated to our model of ontologies by differentiation. The second aspect is related to the evolution of an existing ontology when new knowledge is introduced. In both cases, the importance of separating the essential attributes from the other ones is fundamental.

In fact, the acquisition of the needed knowledge is an essential step (and one of the main difficulties met) in the construction of such systems. Knowledge Acquisition is a generic term which covers several aspects: the extraction, representation, organisation and assimilation of knowledge. Referring to the usual knowledge classification (Wielinga et Breuker 86), knowledge itself can be divided into different abstraction levels: the domain level (which concerns basic elements, concepts, relations and the model of the domain), the inference level (hierarchy of concepts), the task level (organisation of knowledge sources and goals), and finally the strategic level (methods for problem solving, heuristics).

To represent domain ontologies, we focused only on the domain level, and more precisely on the organisation of the conceptual knowledge. A combination of acquisition methods was in fact necessary to build a model of ontologies by differentiation, starting from a domain which is not at all modelised. We also had to define a methodology in order to re-organise a hierarchy of concepts when a new concept appears.

In order to acquire knowledge as part of the representation model, we adapted the existing knowledge acquisition techniques to our type of knowledge representation. Therefore, we first studied existing techniques, retained some of them and adapted them to our model. Then, we defined a methodology including different steps, each step corresponding to a
specific goal in the knowledge acquisition process and needing to use one or more techniques. Among the studied techniques, several were of interest regarding our definition of ontology by differentiation: the lexical analysis (in order to extract the main terms used in the domain), the interview techniques (in order to sort extracted concepts and attributes), the repertory grids (which enable the classification). We defined a methodology in order to use these techniques in turn to construct our ontologies. We applied this approach to two examples. For both examples, our method seemed appropriate: it allowed us to represent knowledge in accordance with our model of ontologies by differentiation, and to add knowledge to an existing classification (the re-ordering of the classification is specially studied in case of integration of new elicited piece of knowledge).

References