Operationalizing domain ontologies: a method and a tool

Frédéric Fürst[1], **Michel Leclère**[2], **Francky Trichet**[3]

Abstract. The work presented in this paper deals with the integration of domain ontologies in Knowledge Based Systems (KBS). We claim that, for using an ontology in a KBS, it must be *operationalized*, that is transcribed in an operational knowledge representation language, according to a precise scenario of use. The use of an operational ontology of geometry led us to propose a formal method to operationalize an ontology with the Conceptual Graphs model, in particular by producing operational forms of the axioms, appropriated to the scenario of use. This method has been implemented into a tool called TooCoM (*a Tool to Operationalize an Ontology with the Conceptual Graph Model*) which allows users to edit an ontology and to automatically produce operational forms of it, according to different scenarii of use.

1 Introduction

Ontologies have been introduced in Knowledge Engineering for allowing the representation of knowledge at the conceptual level, independently from the different uses of the systems where the knowledge is integrated. This approach differs from the one that leads the design of rule based expert systems for which the form of the knowledge representation is determinated by the use, essentially problem solving. Drawing a dividing line between knowledge representation at the conceptual level and operational use of knowledge allows us to integrate knowledge in Knowledge-Based Systems (KBS) without taking into account the operational mechanisms that can be implemented to manipulate the knowledge.

But ontologies are intented for supporting or enabling query answer, classification, association between documents, communication between agents on the Web or reasoning in Web Services, as described for instance in the Web Ontology Language Requirements [14]. Because ontologies will be used as resources for reasoning, in particular on the Web, they must have operational forms appropriated to the different possible operational uses, while being built at the conceptual level. Thus, the use of a domain ontology (limited to a precise field of knowledge) requires its operationalization, *i.e.* its transcribing into an operational knowledge representation language (that is a language endowed with an operational semantics and likely to be runned) according to a precise operational goal.

Then, an ontology is a conceptual representation of a knowledge domain, and using it in a knowledge-based application requires its operationalization for each type of application. The problem is to define automatic operationalization mechanisms that transcribe an ontology in its different operational forms according to the corresponding operational goals. In this paper, we propose a method dedicated to the operationalization of domain ontologies. This method has been tested in the context of an experiment related to the building of an ontology of geometry [6] with the Conceptual Graphs (CGs) model [15]. This experiment led us to define operationalization rules and to develop a tool for editing and operationalizing ontology.

The rest of the paper is structured as follows. After introducing the notion of ontology and pointing at the importance of one of its main components, that is the axioms (cf. section 2), the necessity of an operationalization step for the use of ontology is claimed, and the different scenario of use are described (cf. section 3). The section 4 proposes a formal operationalization method within the context of the CGs model. The explanation of this method is illustrated by the presentation of TooCoM, a tool dedicated to the edition and the operationalization of ontology developed in the context of our work. Finally, the application of the operationalization mechanisms to ontology validation is introduced.

2 The components of a domain ontology

A domain ontology is defined in a consensual way as *a specification of a conceptualisation* [11]. An ontology contains a description of the domain knowledge, structured by a conceptual paradigm. Built on a particular domain of knowledge (*i.e.* a bounded and coherent knowledge with a consensual semantics), an ontology contains first the terminological primitives of the domain, that is the conceptual vocabulary. In the context of the Entity-Relationship paradigm [3] used in our work, the conceptual vocabulary is structured in a set of **concepts**, which represent the objects of the domain, and a set of **relations** between these concepts.

Secondly, a set of **axioms** must be included in the ontology, to express the semantics of the domain. Axioms represent the non terminological knowledge and describe the way the terminological primitives (concepts and relations) can be used [16]. Although the axioms are not usually integrated into the ontologies until now, they are essential to the specification of knowledge and distinguish between light-weigth ontologies and heavy-weight ontologies, which are only structured terminologies, whereas ontologies include the whole knowledge [10]. Axioms are required to use ontologies in an operational way because a KBS can use a concept only with the help of rules that are associated with it.

Axioms can represent common properties of concepts or relations, *i.e.* properties of a lot of conceptual primitives in many knowledge domains. These properties, that we propose to call **axiom schemata**, can be algebraic properties of a relation (symmetry, reflexivity, transitivity), the subsomption property between two concepts or two relations, the genericity of a concept[4], the signature and cardinality of a relation, the exclusivity and incompatibility between two primitives[5].

[1] LINA, University of Nantes, France, *furst@lina.univ-nantes.fr*
[2] LIRMM, University of Montpellier II, France, *leclere@lirmm.fr*
[3] LINA, University of Nantes, France, *trichet@lina.univ-nantes.fr*

[4] a generic concept, or abstract concept, can not have instance.
[5] the incompatibility between two primitives P_1 and P_2 can be formalized by $\neg(P_1 \wedge P_2)$, the exclusivity by $\neg P_1 \Rightarrow P_2$.

For example, in the domain of geometry, the HILBERT's axiom 1.3.1 *« On every line there exist at least two distinct points »* corresponds to a cardinality property: the membership relation between a point and a line has a mimimum cardinality of 2 towards the line.

Some axiom schemata are integrated in knowledge representation formalisms used to describe ontologies. For example, the *is-a* relation appears both in languages based on the Entity-Relationship paradigm (like Conceptual Graphs) and in languages based on the Frame paradigm (like Ontolingua) [4].

Properties which are peculiar to the considered domain can also be included in the ontology. For example, the HILBERT's axiom 2.1.1 *« If point B is between points A and C, then A, B, C are distinct points on the same line »* is not a matter for one of the classical axiom schema. But it must be included in the ontology because it takes part in the definition of the semantics of the domain.

The axioms specify the semantics of the conceptual primitives, *i.e.* the way the primitives are used to express knowledge in the considered domain. But to use the axioms in a KBS, operational semantics must also be specified, *i.e.* the way the axioms are used in the context of the considered application. Because this semantics depends on the operational goal of the application, it can not be integrated in the ontology, which must be independent from such goals. The specification of this semantics leads, through an *operationalization* process, to an *operational ontology*.

3 The operationalization of a domain ontology

An ontology is only a conceptual representation of a domain, independently from the possible operational uses. To integrate an ontology into a KBS, it must be transcribed in a form appropriated to the use that the KBS is dedicated to. This operationalization consists, on the one hand, in choosing the operational representation language which offers manipulation mechanisms compatible with the considered operational goal and, on the other hand, in adapting the representation of the ontology to this goal by specifying the manipulation semantics of the axioms, semantics which is determinated by the considered application, and not by the considered domain. Thus, operationalizing an ontology consists in transcribing it in an operational knowledge representation language according to a scenario of use which describes the operational goal of the KBS.

The operationalization of an ontology can only be made for a well defined operational use, characterized by a precise *scenario of use* [12]. A scenario of use describes how the knowledge specified in the ontology will be used, *i.e.* essentially what the axioms will be used for. Because the representation of terminological knowledge of the domain does not depend on the many possible application contexts, the representation of a concept or a relation will be the same for a system dedicated to knowledge validation or a system dedicated knowledge inference. Then, only operational representations of axioms have to be adapted to the goal of the considered application.

An axiom can be used to *produce* new knowledge from a knowledge base, or to *check* the compliance of a knowledge base with the semantics of the considered domain. For example, the HILBERT's axiom 1.6 *« If two different points A and B belong both to a line m and a plane p, all points of m belong to p »* can be used to deduce the membership of points to a plane. But it can also be used to show that a statement is not in accordance with the semantics of geometry if two points belong to both a line and a plane and a third point of the line does not belong to the plane.

Moreover, an axiom can be used only at the KBS user request, or it can be applied automatically by the KBS. The first use is called *explicit*, the second *implicit*. The HILBERT's axiom 1.3.1 *« On every line there exist at least two distinct points »* can then be used automatically in an implicit way in order that the user does not have to apply the axiom to consider points on a line. But it can be also used in an explicit way, if we want that the user systematically applies the axiom before considering points on a line, *e.g.* for educational purposes.

So, the operationalization of an ontology requires the choice, for each axiom, of a **context of use**, that specifies for what will the axiom be used and how it will be used. The different contexts of use that we propose to consider are:

- The **inferential and explicit** context of use: the user applies the axiom by himself on a fact base to produce new facts;
- The **inferential and implicit** context of use: the axiom is applied by the system on a fact base to produce new facts;
- The **validation and explicit** context of use: the user applies the axiom by himself to check that a fact base is in accordance with the semantics of a domain;
- The **validation and implicit** context of use: the axiom is applied by the system to verify that a fact base is in accordance with the semantics of a domain.

A *scenario of use* consists in a set of contexts of use choosen for each axiom of the ontology. For the axiom schemata, the same context of use can be specified for all the axioms that correspond to a given schema. For example, the user can choose an inferential and implicit context of use for all the axioms that express a symmetry relationship, in order to automatically produce symmetric relations in the knowledge base.

Generally speaking, the operational form of an ontology includes inferential mechanisms and validation mechanisms. For instance, a scenario dedicated to a computer-aided teaching application allows the user to apply knowledge to deduce new facts or to check his work. Such a scenario comprises automatic inferences and validation processes, in accordance with the level of the user. Two particular cases of scenario of use can be distinguished: the pure validation scenario, where the axioms are only used to validate a knowledge base according to the semantics of the domain, and the inferential and implicit scenario where the axioms are automatically used to produce new facts, without user intervention. In the last case, which is those of expert systems, the automatic inferences are supposed to produce knowledge in accordance with the semantics of the domain and no validation step is required.

Figure 1 presents the general reasoning cycle through which the axioms are applied in a KBS. First the user can add facts to the fact base (1), then he can apply an axiom choosen between the inferential and explicit ones (2). Next, the system applies all the inferential implicit axioms in order to sature the fact base with implicit knowledge (3). Finally, a validation step, which can be partially leaded by the user, permits to detect « semantical inconsistencies » in the fact base (4).

4 The operationalization of the axioms with the Conceptual Graphs model

In this section, we propose an operationalization method based on the principles stated in section 3, but in the specific case of the CGs model. This method has been defined from an operationalization experiment of an ontology of geometry with an extension of the CGs model, the SG-family [1]. The SG-family is an operational knowledge representation formalism which allows us, on the one hand, to

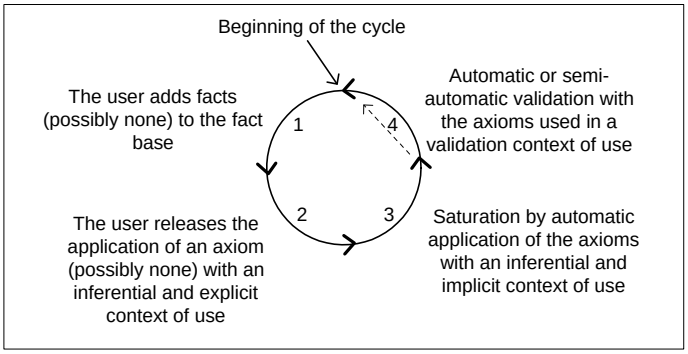

Figure 1. The reasoning cycle dedicated to the use of an operational ontology.

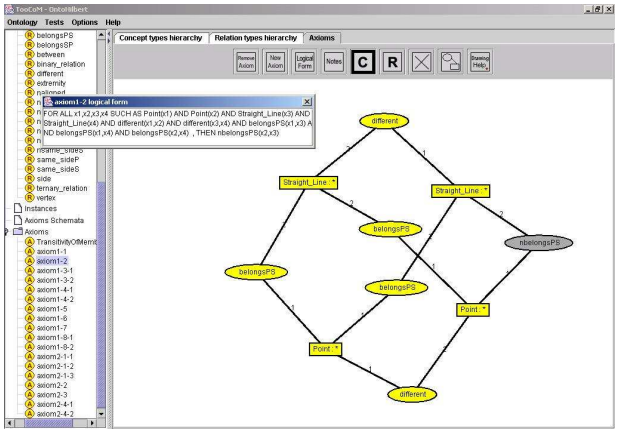

Figure 2. Specifi cation of an axiom in TooCoM. The bright concepts and relations represent the hypothesis part of the axiom and the dark concepts and relations represent the conclusion part. Semantics of this axiom is as follows: *given two different points and two different lines, if one of these points belongs to the two lines, and if the other belongs to one of the lines, it does not belong to the second line.*

represent conceptual primitives (concept and relations) and, on the other hand, to represent axioms that express the manipulation semantics of these primitives with CG rules and CG constraints. This method has been implemented in a tool dedicated to the edition and operationalization of ontology, called TooCoM [5] (*a Tool to Operationalize an Ontology with the Conceptual Graph Model*). This tool allows the user to graphically edit an ontology with the Entity/Relationship paradigm and to operationalize the ontology with the CGs model after specifying the choosen scenario of use. An inference engine based on the CGs manipulation library CoGITaNT [7] implements the reasoning cycle shown in figure 1.

4.1 Defining the conceptual vocabulary and the axioms with TooCoM

The definition of conceptual vocabulary must precede the specification of axioms. The conceptual vocabulary consists of a set of *concept types* and a set of *relation types* (the terms concepts and relations designate here instances of concept types and relation types respectively); these sets are structured by the *is-a* relationship. Only « ontological » instances of concept types can be defined in an ontology, that is these instances must take part in the definition of the semantics of the domain. For example, π is an ontological instance of the number concept type in an ontology of trigonometry.

In the Entity-Relationship paradigm, concepts and relations can be use to build graphs that express facts where concepts are nodes of the graphs and relations are edges. A couple of graphs are used to represent an axiom at the ontological level, with one of the graphs as hypothesis and the other as conclusion, with links between concepts of each graph. The subsomption properties between primitives and the signatures of relation types are embeded in the model. All the other properties have to be expressed by a couple of graphs. The figure 2 shows an axiom of the geometry domain. Note that this axiom is specified at the ontological level because its form does not constrain the future operational use of the axiom.

Classical axiom schemata can be specified by simply indicated the property of the relation types in the dedicated tool box. If such a property of relation type (symmetry, transitivity or reflexivity) is specified, the corresponding axiom is automatically created and added to the ontology.

4.2 Operationalizing the axioms with TooCoM

The operational representation of the axioms is based on three types of reasoning primitives allowed by the SG-family: (1) the **positive constraints**, with an hypothesis graph and a conclusion graph of which the semantics is *« if the hypothesis part is present in a graph G, then the conclusion part must be present in G » (otherwise the constraint is broken by G)*; (2) the **negative constraints**, with an hypothesis graph and a conclusion graph of which the semantics is *« if the hypothesis part is present in a graph G, then the conclusion part must be absent in G » (otherwise the constraint is broken by G)*; (3) the CG **rules**, with an hypothesis graph and a conclusion graph of which the semantics is *« if the hypothesis part is present in a graph G, then the conclusion part can be added to G »*.

A rule can be implicitly used by the system (*i.e.* applied everywhere the hypothesis of the rule is present) or explicitly applied by the user (on a given fact in the knowledge base). A negative or positive constraint can be automatically used by the system (*i.e.* checked everywhere in the knowledge base) or explicitly applied by the user.

Once the ontology is built by specifying the conceptual vocabulary and the axioms, the operationalization allows the user to generate a set of implicit or explicit rules and constraints that can be used in the inference engine.

The production of rules and constraints is guided by the choosen scenario of use, composed by the contexts of use of each axiom of the ontology. But, for each context of use, the rules and constraints that correspond to the operational form of an axiom also depend on the ontological form of the axiom, that is the form in which the axiom is specified in the ontology. The ontological forms of the axioms that are taken into account in our work are those which have been encountered during the operationalization of the ontology of geometry. Generally speaking, these axioms, including those that come within one of the axiom schemata described in section 2, have the following ontological form : *« if antecedent then consequent »* and, more precisely : *« if there are some instances* $\{x_i\}_{i=1..p}$ *such as* $\{T_i(x_i)\}_{i=1..p}$ *and* $\{r_j(x_{n_j}, x_{m_j})\}_{j=1..q}$*, with* $\{T_i\}_{i=1..p}$ *concept types and* $\{r_j\}_{j=1..q}$ *relation types and* n_j *et* m_j *in* $[1..p]$*, then, there are instances* $\{y_k\}_{k=1..r}$*, such as* $\{T_k(y_k)\}_{k=1..r}$ *and* $\{r_l(z_{u_l}, z_{v_l})\}_{l=1..s}$*, with* $\{T_k\}_{k=1..r}$ *concept types and* $\{r_l\}_{l=1..s}$ *relation types and* z_{u_l} *and* z_{v_l} *in* $\{x_i\}_{i=1..p} \cup \{y_k\}_{k=1..r}$ *».* The instances x_i and y_k can be undefined instances[6] or ontological in-

[6] An undefi ned instance is an instance of a concept type with any known identity. In an axiom, such an instance can represent any instance of a given

stances.

The operational representations of the axioms are different according to the different form derived from this general form of the axioms. In each case, and for each context of use, the operationalization conducts to generate a set of CG rules and CG constraints that corresponds to the operational semantics of the axiom in this context of use.

For example, some axioms have a consequent part which contains only relations[7]. Their general form is *if there are some instances* $\{x_i\}_{i=1..p}$ *such as* $\{T_i(x_i)\}_{i=1..p}$ *and* $\{r_j(x_{n_j}, x_{m_j})\}_{j=1..q}$*, with* $\{T_i\}_{i=1..p}$ *concept types and* $\{r_j\}_{j=1..q}$ *relation types and* n_j *and* m_j *in* $[1..p]$*, then* $\{r_l(z_{u_l}, z_{v_l})\}_{l=1..s}$*, with* $\{r_l\}_{l=1..s}$ *relation types and* z_{u_l} *and* z_{v_l} *in* $\{x_i\}_{i=1..p}$. For example, the axiom 2.1.2 of the HILBERT's axiomatics of geometry, *« If point B is between points A and C, then B is between C and A »*, has the following ontological representation: *if there are instances* x_1 *and* x_2 *and* x_3 *such as Point*(x_1) *and Point*(x_2) *and Point*(x_3) *and between*(x_2, x_1, x_3)*, then between*(x_2, x_3, x_1).

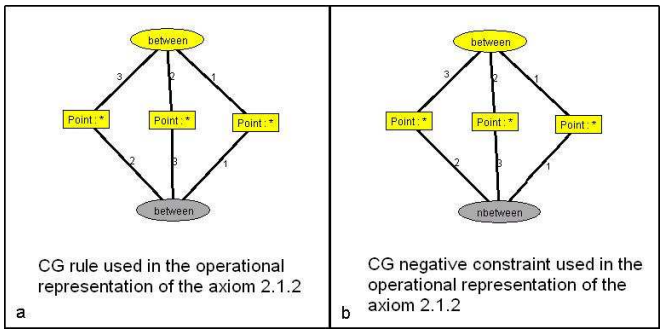

Figure 3. Operational representation of the axiom 2.1.2. The rule (a) permits, if the hypothesis is present, to produce a membership relation. The constraint (b) prevents, if the hypothesis is present, the existence of a non membership relation (the hypothesis parts of the rule and the constraint are in bright color, the conclusion parts in dark color).

In the case of an **inferential and implicit context of use**, such an axiom is operationalized by one implicit CG rule which permits to produce the relations that appear in the consequent if the antecedent of the axiom is encountered. In this context, the axiom 2.1.2 is operationalized by the rule shown figure 3a. In the case of an **inferential and explicit context of use**, the user must be able to use or not use the axiom to produce new facts, but the system must control that he does not add to the knowledge base some facts that contradict the axiom. So, the operational form of the axiom consists in one explicit CG rule (the same as the previous one) and s negative constraints which prevent that, for each relation $\{r_k\}_{k=1..s}$ of the consequent part of the axiom, if the antecedent part of the axiom is present, the relation exclusive with r_k is present. For each r_k of the consequent part, if any relation exclusive with r_k exists in the ontology, and if p relations incompatibles with r_k exist, the k-th constraint is represented by p negative constraints which prevent that, if the antecedent part of the axiom is present, one of the relation incompatible with r_k is present. For example, the axiom 2.1.2 is operationalized in such a context by the CG rule of the figure 3a and the constraint of the figure 3b because, in the ontology of geometry, the relation type *nbetween*, which is exclusive with the relation type *between*, is defined. In the case of a **validation context of use**, only the constraints are taken into account in the operational form of the axiom. So, in a validation context, the axiom 2.1.2 is operationalized by the constraint of the figure 3b.

5 Operationalization and validation of ontology

Operationalizing an ontology allows the user to produce an operational form of an ontology, appropriate to a given type of application. This mechanism is then very useful to perform validation operations: the validation[8] of an ontology consists in checking its accuracy in relation to the semantics of the considered domain [9]. Validation requires to answer to the following questions:

1. are the whole knowledge of the domain represented in the ontology (completeness of the ontology towards the domain)?
2. is the semantics expressed in the ontology in accordance with the one of the domain (coherence of the ontology)? This coherence is relative to the domain because we suppose that the knowledge domain is coherent. So, a coherence fault in the ontology reveals a modelisation that is not in accordance with the domain.
3. is the ontology minimal?

Answering to these questions can not be done exhaustively, but only partially through the operational use of the ontology. The validation of an ontology can only be done through the operational use of the ontology, because testing in a formal way its semantics (*i.e.* by testing the form of the ontology) requires a pre-existing formal model of the knowledge of the domain. As the goal of the building of an ontology is to create such a formal model of knowledge, validating an ontology can not be done by testing its form, but only through its operational use.

GRUNINGER proposes in [12] to use a set of *competency questions* to which the ontology must permit to answer. A competency question is composed by an initial fact from which another given fact must be deduced by using knowledge represented in the ontology. The impossibility to answer to such a question implies that some knowledge of the domain are missing in the ontology. Thus, validation of ontology is based on external specifications (competency questions) and internal specifications (axioms of the ontology which describe the semantics of the domain).

Testing the completeness of the ontology in relation to the domain with competency questions can be done automatically by using the operationalization mechanisms that we propose: after the competency questions have been formalized in the choosen operational language, the ontology is operationalized in an inferential scenario of use, then the inference engine is used for each question to produce the conclusion fact from the initial fact of the question. The impossibility to produce one of the conclusions reveals the incompleteness of the ontology.

For example, in the context of geometry, the basic theorems of the domain compose a set of competency questions: to performing the proofs of these theorems by using the axioms of the plane geometry ensures that the ontology is complete. In the context of our work, we have tested the ontology of geometry to automatically prove these theorems [6]. The impossibility to prove some theorems led us to

concept type.

[7] Note that in this paper, we only treat one example of operationalization for one of the particular forms of the axiom, in order to give an intuitive explanation of our method. See [6] for more details about the other operationalization rules for the other axiom forms.

[8] Another level of checking of an ontology is the verification, which consists in testing if the ontology is properly built in relation to the model of knowledge representation. Verification deals with building the ontology right, validation deals with building the right ontology.

identify knowledge used in the domain but not expressed explicitly in the corpus which correspond to the book of HILBERT (for instance, the symmetry of the relation *between(Point,Point,Point)*) [13].

The operationalization of an ontology allows us to test its coherence. Operationalizing an axiom in a validation context of use permits to produce the resulting constraints. These constraints can then be used to test if, for each of the other axioms of the ontology, the antecedent or the consequent of the axiom break one of the constraints. In this case, it means that one of the two axioms (those which is operationalized and those which is tested) does not correspond to the semantics of the domain. Moreover, specifying the axioms with the same ontological form, independent of the operational uses, makes easier to test the minimality of ontology. The unity of forms of the axioms allows us to easily compare them.

6 Related work

Most of existing tools dedicated to the edition of ontology allows the user to specify axioms. But this specification is done at the operational level, not at the conceptual level. For instance, in OntoEdit, the specification of a non-predefined type of axiom requires the use of the F-Logic syntax and the operational semantics of the axioms is fixed in the ontology [17]. All the axioms are used in the inference engine as rules. In Protégé, the PAL language allows the user to define constraints, in rule form, that are used in the inference engine to test the coherence of the ontology [8]. So these tools do not provide a real operationalization step before using the ontology, because the specification of axioms is already done at the operational level for a specific task, often the test of the ontology.

The main innovative aspect of our tool is to provide the definition of axioms at the conceptual level, in a graphical syntax that does not constrain the operational use of the axioms. Then, the reusability of ontology is reinforced, because the expression of the axioms at the ontological level does not depend on any operational goal. Then, the same ontology can be used in several operational systems after operationalization, and comparison of ontologies for merging or mapping purpose is more easier. Moreover, the graphical syntax facilitates the definition of axioms, compared to the textual languages used in other tools. Besides, operationalization mechanisms allows the user to automatically produce a large set of operational ontologies, derived from the conceptual one and each appropriate for a specific operational task. These mechanisms allows the user to produce an operational ontology for testing its coherence and completeness, but also to produce different operational forms of the ontology for using it in reasoners or classifiers, for example in Web agents.

7 Conclusion

In this paper, we claim that the use of an ontology in a KBS requires its operationalization, *i.e.* its transcribing in an operational knowledge representation language, especially the transcribing of the axioms. This transcribing is only conceivable for a precise scenario of use, which specifies the context of use of each axiom of the ontology, that depends on the way the axiom is used in the KBS. Each scenario of use lead to a different operational form of the ontology. Based on a scenario of use, the operationalization of an ontology can be automatized for a given operational language by using a formal method.

Such a method is given in the context of the Conceptual Graphs model. This method, based on a practical experiment of operationalization of an ontology of geometry, is implemented in a tool, called TooCoM, dedicated to the edition and the operationalization of ontology. This tool allows the user to build an ontology structured by the Entity-Relationship paradigm, and to produce the operational forms of the axioms according to a scenario of use. An embedded inference engine allows the user to use the produced operational ontology, for example to test the coherence or the completeness of the ontology in relation to the domain.

This work is currently in progress towards the application of our operationalization method on the Semantic Web languages that, as argued by T. BERNERS-LEE, must enable the representation of operational ontologies to perform the original goal of the Semantic Web: *« For the Semantic Web to function, computers must have access to [..] sets of inference rules that they can use to conduct automated reasoning »* [2].

REFERENCES

[1] J.F. Baget and M.L. Mugnier, 'Extensions of simple conceptual graphs: the complexity of rules and constraints', *Journal of Artificial Intelligence Research*, **16**, 425–465, (2002).

[2] T. Berners-Lee, J. Hendler, and O. Lassila, 'The semantic web', *Scientific American*, **248**(5), 35–43, (2001).

[3] Peter P. Chen, 'The entity-relationship model - toward a unifi ed view of data', *ACM Trans. Database Syst.*, **1**(1), 9–36, (1976).

[4] A. Farquhar, R. Fikes, and J. Rice, 'Ontolingua server: a tool for collaborative ontology construction', *International journal of Human-Computer studies*, **46**, 707–727, (2000).

[5] F. Fürst, 'TooCoM: a Tool to Operationalize an Ontology with the Conceptual Graph Model', in *Proceedings of the second Workshop on Evaluation of Ontology-Based Tools (EON'2003) at the International Semantic Web Conference (ISWC'2003)*, pp. 57–70, (2003).

[6] F. Fürst, M. Leclère, and F. Trichet, 'Ontology engineering and mathematical knowledge management: a formalization of projective geometry', *Annals of Mathematics and Artificial Intelligence, Kluwer Academic Publishers ISSN:1012-2443*, **38**(1), 65–89, (2003).

[7] D. Genest and E. Salvat, 'A platform allowing typed nested graphs : how CoGITo became CoGITaNT', in *Proceedings of the International Conference on Conceptual Structures (ICCS'98)*, volume 1453, pp. 154–161. Springer-Verlag LNAI, (1998).

[8] J.H. Gennari, M.A. Musen, R.W. Fergerson, W.E. Grosso, M. Crubezy, H. Eriksson, N.F. Noy, and S.W. Tu, 'The evolution of protégé: an environment for knowledge-based systems development', *International Journal of Human-Computer Studies*, **58**, 89–123, (2003).

[9] A. Gomez-Perez, 'Evaluation of taxonomic knowledge in ontologies and knowledge bases', in *Proceedings of the twelfth Workhop on Knowledge Acquisition, Modeling and Management, KAW'99*, (1999).

[10] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering*, Springer, Advanced Information and Knowledge Processing, 2003.

[11] T.R. Gruber, 'A translation approach to portable ontology specifi cations', *Knowledge Acquisition*, **5**(2), 199–220, (1993).

[12] M. Gruninger and M. S. Fox, 'Methodology for the design and evaluation of ontologies', in *Proceedings of the Workshop on Basic Ontological Issues on Knowledge Sharing, IJCAI'95*, (1995).

[13] M. Leclère, F. Trichet, and F. Fürst, 'Operationalising domain ontologies: towards an ontological level for the SG family', in *contributions to the International Conference on Conceptual Structures (ICCS'02)*. Bulgarian Academy of Sciences, (2002).

[14] OWL, 'Ontology web language guide', in *http://www.w3.org/TR/2002/WD-owl-guide-20021104/*, (2002).

[15] J. Sowa, *Conceptual Structures : information processing in mind and machine*, Addison-Wesley, 1984.

[16] S. Staab and A. Maedche, 'Axioms are objects too: Ontology engineering beyong the modeling of concepts and relations', Research report 399, Institute AIFB, Karlsruhe, (2000).

[17] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke, 'Ontoedit: colllaborative ontology development for the semantic web', in *Proceedings of the International Semantic Web Conference*, volume 2342, pp. 221–235. Springer-Verlag LNCS, (2002).