Enriching Ontology Languages Adequacy for eBusiness Domain

Michele Missikoff, Federica Schiappelli

LEKS, IASI-CNR
{missikoff,schiappelli}@iasi.cnr.it

Abstract. The definition of a domain ontology is a complex activity that requires two kinds of expertise: a deep knowledge of the domain to be modeled and a good level of familiarity with the ontology representation language to be used in order to formalize the built model. Existing ontology languages (such as SHOE, DAML+OIL, OWL) are inherently general purpose, allowing different domains to be tackled and modeled. We believe it is possible to enrich an ontology language with domain specific features to ease the ontology building process. Such a goal can be achieved along two lines. One is to provide a core domain ontology, containing the most general concepts in the domain of interest; another is to introduce specific modeling constructs in the ontology language. The latter option, referred to as domain adequacy, is addressed in this paper. We will show that adding a few domain specific concept categories in the language allows better ontologies to be built.

Keywords. ontology language, domain ontology, concept representation, ontology building, language adequacy

1. Introduction

Domain ontology building is one of the most critical activities required in ontology-based applications. The task is generally performed by domain experts, who do not (generally) have the background nor the experience of a knowledge engineer. To ease this task, ontology management systems (such as Protégé[6], OntoEdit[8] or SymOntoX[5]) are characterized by user friendly interfaces. However the problem is mainly of cognitive nature and a GUI can only solve part of it. The other part of the problem is represented by the inherent complexity of the conceptualization activity and the ontology languages, difficult to be used by domain experts (with a limited knowledge engineering expertise).

Existing ontology languages (such as SHOE[3], DAML+OIL[1], or OWL[2]) are of a general purpose nature and therefore give to the user great freedom and, conversely, low domain specific guidance. Enhancing domain specificity of ontology building tools will support domain experts in their challenging tasks. Domain specificity can be achieved with two different approaches. One is to provide a core domain ontology, containing the most general concepts that characterize a given domain. Then domain experts can start building the ontology in a top-down fashion, by refining such concepts. Another approach is to enrich the constructs of the ontology language with primitives that provide a guidance for the user when representing the domain concepts. The two approaches are not mutually exclusive.

In this paper we will focus on the latter, illustrating a proposal for augmenting ontology languages, to be used in the business domain. In this perspective, we introduce the notion of "domain adequacy" that represents, for an ontology language, the degree of domain specificity.

In this paper we address the business domain, without reference to a specific production sector (such as automotive, tourism or banking), to introduce a number of categories aimed at supporting the construction of business ontologies. The main proposed categories (referred to as *concept kinds*) are: Business Actor, Business Process, Business Object, Business Event, Business Message, Business Goal (we will drop the term "business" in the rest of the paper).

In the paper, we present an ontology model, OPAL (Object, Process, Actor modelling Language) where the mentioned concept kinds are used for the definition (and maintenance) of a consistent ontology.

2. The benefits of domain adequacy

The main goal of the proposed solution is to provide techniques able to ease the work of domain experts when building and maintaining an ontology. In this direction, such techniques should:

– provide guidelines for domain experts in analyzing the problem domain and organizing its conceptualization
– reduce cost and time required for domain ontology building
– enhance the quality of the built ontology, with a support for consistency checking

This goal is pursued by introducing elements (concepts) of the application domain in the ontology building environment. In particular, the ontology language is augmented with new constructs, that represent a bias imposed by the domain (meta) knowledge to the language.

Domain adequacy is not aimed at improving the expressive power; conversely, a language that exhibit a higher adequacy in a specific domain is less usable in a different domain: domain adequacy reduces the generality (i.e., applicability) of an ontology language, but allows more synthetic, concise, precise sentences to be formulated in the specific domain (intuitively, we may say that the same content can be represented by a shorter string, assuming an ontology representation in a linear form).

3. Business Meta-Concepts[1]: a first proposal

Here we introduce a first set of meta-concepts that represent a few categories in the business domain. Referring to the traditional Frame-Slot-Facet modeling paradigm[4], we may consider these categories as "templates" to be used by business people to model their concepts.

All templates are characterized by three sections:

(i) *identification section*, that contains the concept label, description, XML tag, and other meta-data (such as creation date, author) with an administrative flavor;

(ii) *structural section*, that contains the slots corresponding to the attributes (simple or complex) that will be instantiated in the corresponding object;

(iii) *specific section*, that contains information and references to other entities that play a specific role (predefined) in the correct definition of the concept.

The first two sections are the same for all the kinds, while the third is differently tailored for each category. Below we briefly describe the specific section for each kind.

Actor

A *business Actor (Ar)* is an operating element of a business domain. The domain expert, in analyzing the reality, is asked to identify relevant actors that operate producing, updating or consuming information. An actor can be an enterprise, a computer system, an employee, a business unit. The corresponding concept is modeled in the ontology, it is categorized as an Actor, when having the following specific section:

- *Goals:* objectives that must be accomplished by the Actor
- *Skills:* indicating the actions it is able to perform
- *Responsibilities:* the actions performed or monitored in achieving Goals
- *Managed Objects:* the information objects required, generated, updated or deleted by the Actor in achieving the Goals.

Object

A *business Object (O)* is a category that gathers passive entities involved in one or more business processes (subject to the so called CRUD lifecycle: create, read, update, delete). It can be either a material or an immaterial entity. An Object has a specific section that describes:

- the Processes that *create*, *manipulate*, *delete* it
- the Actors that *insist* on it
- *States*, defined as boolean expressions over its attributes or those of related concepts.

Process and Action

A *business Process (P)* is a category that gathers business activities. A Process is aims at accomplish one or more business goals, modifying a set of objects. It can be

[1] We experimented already some of them in EU Projects: Fetish and Harmonise, where an interoperability platform, based on a tourism ontology, has been built. The proposed approach has been further elaborated within IDEAS and is currently at the base of the ontology research in the INTEROP Noe and the Athena IP.

rather simple, with a limited duration in time, or complex, with parallel branches and phases that last for a long time span.

Processes, as Objects and Actors, can be organized according to a specialisation and decomposition hierarchy, in order to obtain a more structured view of the enterprise activities. Process are decomposed into *Actions (An),* that can be recursively decomposed till we get to the *Elementary Actions (EA),* that are not further decomposable.

Complex Attribute and Atomic Attribute

In modeling the information of an entity or a process, we distinguish between structured information, such as “address”, and elementary information, such as “street name”. Essentially, a (structured) *Complex Attribute (CA)* is defined as an aggregation of lower level *CA* and/or *Simple Attributes (AA)*.

In OPAL we identified other conceptual categories, in particular: *Business State, Event, Message, Goal, Rule* (restrictive or prescriptive) and *Decision*. Their formal definition is currently in a preliminary stage therefore, also for sake of conciseness, we are not going to further elaborate on them, but we believe that these additional OPAL kinds will contribute in enhancing the domain adequacy of the ontology languages.

4. Building a consistent ontology

As anticipated in the Section 2, the introduction of concepts kinds, besides representing a guideline for the ontology developer, has the purpose of improving the verifiable quality of an ontology. In fact, OPAL defines a rich set of constraints inherent to the ontology representation model. Consistency, a key formal property automatically verifiable, is valuable if a sufficient number of constraints is introduced in the ontology. We distinguish three sorts of constraints:

Inherent constraints – imposed by the modeling approach and, therefore, transparent to the user. E.g., the *acyclicity* of a specialisation hierarchy.

Built-in constraints – specific language constructs, having a precise semantics. E.g., *disjoint(A,B)* asserts that the concepts A and B do not have common instances.

Explicit constraints –defined by the user by means of a constraint language (such as OCL, the UML constraint language). The semantics of an explicit constraint is computed by the system on the bases of the supplied *constraint expression*.

In OPAL we aim at enriching the inherent constraints, reducing the amount of code (and therefore, of effort, time, and errors) that an ontology engineer has to write when building a domain ontology.

In the next section we present a first set of constraints inherent to the OPAL model. Such constraints are assumed when an OPAL ontology is constructed and automatically enforced by SymOntoX, the ontology management system developed at the LEKS, IASI-CNR.

4.1 The inherent constraints of the OPAL ontology model

Besides the concept categories presented in Section 3, the OPAL model includes meta-relations defined among categories. The OPAL meta-relations represent well known modeling notions common to (the knowledge model of) the most Knowledge Representation Languages:

(i) inheritance hierarchies among concepts
(ii) concept structure, defined by its attributes
(iii) domain specific relationships with other concepts.

In the following they are described and, for each of them, the inherent constraints of the OPAL model are presented. Examples illustrating the constraints are taken from the business domain.

Specialisation (ISA)

Specialisation (inverse: *Generalisation*) expresses the relation among a broader concept and a narrower concept. For example the concept *Person* can be specialized into *Manager*. This relation must be defined among concepts belonging to the same conceptual category (e.g *Objects, Actors, Processes*). Formally, let c_i (i=1,…,n) be concepts, let $kind(c_i)$ be the OPAL category of the concept c_i. The following constraint must be verified:

1) $(c_1,c_2) \in ISA \Rightarrow kind(c_1) = kind(c_2)$

Furthermore the Specialisation relation is transitive and must be acyclic.

2) $(c_i,c_j), (c_j,c_k) \in ISA \Rightarrow (c_i,c_k) \in ISA$ (*transitivity*)

3) there are no sequences $c_1..c_n$ such that $(c_1,c_2), ...(c_{n-1},c_n) \in ISA$ and $c_1=c_n$ (*acyclicity*)

Predication (Pr)

Predication expresses the relation among a concept and its attributes. For example a concept *Person* can have as attributes the concepts *Name* and *Address*.

Since this relation typically connects a primary concept with *Complex Attributes* and *Simple Attributes*, it must be defined between concepts whose kinds are defined as follows:

4) $(c_i,c_j) \in Pr \Rightarrow kind(c_i) = O \mid Ar, kind(c_j) = CA \mid AA)$

Part of (Po)

PartOf expresses the relation among a concept and a concept representing one of its components. For example an *Enterprise* can be decomposed in *FunctionalUnits* or an *Address* can be decomposed in *City*, *Street*, *Number* and *ZipCode*.

The involved concepts must satisfy one of the following constraints, depending on their kind:

5) $(c_i,c_j) \in Po$ and $kind(c_i) = Ar \mid O \Rightarrow kind(c_i) = kind(c_j)$

6) $(c_i,c_j) \in Po$ and $kind(c_i) = CA \Rightarrow kind(c_j) = AA \mid CA$

7) $(c_i,c_j) \in Po$ and $kind(c_i) = P \Rightarrow kind(c_j) = P \mid An \mid EA$

8) $(c_i,c_j) \in Po$ and $kind(c_i) = An \Rightarrow kind(c_j) = An \mid E$

Relatedness (R)

Relatedness expresses the fact that, between two concepts, a domain relation exists. Such domain relation can be labelled. For example an *Enterprise* can be related to a *Manager* with a relation named *direction*. Relatedness is symmetric.

9) $(c_i,c_j) \in R \Rightarrow (c_j,c_i) \in R$ (*symmetry*)

Furthermore the related concepts may be of different kinds; one of the following conditions holds:

10) $(c_i,c_j) \in R$ and $kind(c_i) = Ar \mid O \mid P \Rightarrow kind(c_j) = Ar \mid O \mid P$

11) $(c_i,c_j) \in R$ and $kind(c_i) = CA \mid AA \mid An \mid EA \Rightarrow kind(c_i) = kind(c_j)$

Conclusions

The *Domain Adequacy* of an ontology language indicates the domain specificity of the language. It aims at providing a better guidance to domain experts, supplying a predefined categorization of the concepts introduced in the ontology. Such predefined characterization enhances the quality of the ontology, supplying a number of built-in constraints that can be automatically verified by the OMS supporting the language.

In this paper we briefly presented an ontology modeling language, OPAL, aimed at modeling the business domains. The OPAL concept kinds, and the meta-relations defined among them, have been described. Furthermore, for each meta-relation, inherent constraints of the OPAL model have been identified and their impact on the building (and maintenance) of a consistent ontology has been shown.

The inherent constraints of the OPAL model are automatically enforced by SymOntoX, the ontology management system developed at the LEKS, IASI-CNR.

Bibliography

[1] D.L.McGuinness, R.Fikes, J.Hendler, L.A.Stein. *DAML+OIL: An Ontology Language for the Semantic Web* . In IEEE Intelligent Systems, Vol. 17, No. 5, pages 72-80, Sept.2002

[2] F.van Harmelen, I.Horrocks, P.F.Patel-Schneider, *OWL Web Ontology Language Semantics and Abstract Syntax*, W3C Candidate Recommendation 18 August 2003

[3] J.Heflin, J.Hendler, S.Luke; *SHOE: A Knowledge Representation Language for Internet Applications*, Technical Report CS-TR-4078 (UMIACS TR-99-71). 1999.

[4] M. Minsky, *Frame-system: Theory in Thinking*, University Press, London, 1977.

[5] M.Missikoff, F.Taglino; *Symontox: A Web-Ontology Tool for eBusiness Domains*; Proc. of WISE2003, Roma-Italy, Dec 2003.

[6] Noy, N.F., Fergerson, R.W., and Musen, M.A. *The knowledge model of Protege-2000: Combining interoperability and flexibility*. Proc. EKAW 2000.

[7] R.Motschnig-Pitrick, J.Mylopoulos; *Classes and Instances*; International Journal of Intelligent and Cooperative Information Systems, 1(1), 61-92 (1992).

[8] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. *OntoEdit: Collaborative ontology engineering for the semantic web*. In International Semantic Web Conference 2002 (ISWC 2002), Sardinia, June 2002.