Semantic Commitment for Designing Ontologies: A Tool Proposal

Antoine Isaac[1,2], **Raphaël Troncy**[1,3]
[1] Institut National de l'Audiovisuel, Direction de la Recherche
4, Av. de l'Europe - 94366 Bry-sur-Marne
`{aisaac,rtroncy}@ina.fr` & `http://www.ina.fr/`
[2]LaLICC, Université de Paris-Sorbonne [3] Action EXMO, INRIA Rhône-Alpes

1 Methodological background

We propose here to implement a method for building ontologies that uses linguistic considerations to better structure the subsumption hierarchies. In fact, we think that existing methodologies, while being quite satisfying w.r.t. the life-cycle management, still lack a semantic commitment clear enough to make the primitives elicited easily understandable and, as a result, conveniently usable in knowledge-based systems.

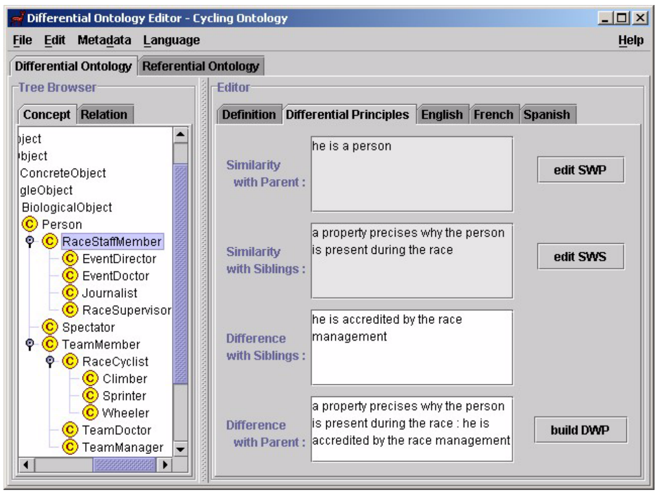
The methodological work has resulted in the definition of three steps: a semantic *normalization*, followed by the *formalization* and *operationalization* of knowledge. What makes this approach really complementary to other existing methodologies is the first step, which is based on *differential semantics*. It defines the meaning of linguistic units by comparing these units to one another by the means of natural language (NL). Practically, a primitive is defined by the differences and similarities it maintains with its close neighborhood (in a taxonomy, its parent and its siblings):

- The *similarity with parent* (**SWP**): explicits why the primitive inherits properties of the one that subsumes it;
- The *similarity with siblings* (**SWS**): gives a semantic axis, a property – assuming exclusive values – enabling to compare the primitive with its siblings;
- The *difference with siblings* (**DWS**): precises the property enabling to distinguish the primitive from its siblings;
- The *difference with parent* (**DWP**): explicits the difference enabling to distinguish the primitive from its parent.

Defining a primitive by the means of these structuring principles allows to fix the context (given by the ontology application) that will constrain the *interpretation* of this primitive. As a result, it helps hierarchies construction while clarifying their meaning.

2 The *DOE* Editor

DOE[1] (*Differential Ontology Editor*) is a simple prototype (developed with Java) that supports this methodology. It allows building a *differential ontology* and a *referential ontology*, corresponding to the first two steps of the methodology. These two activities are articulated in a way that enables an almost simultaneous construction while respecting the basic differences between them.


Differential Ontology Editor - Cycling Ontology
File Edit Metadata Language Help
Differential Ontology Referential Ontology
Tree Browser
Concept Relation
Editor
Definition Differential Principles English French Spanish
edit SWP
edit SWS
build DWP

bject
bject
ConcreteObject
gleObject
BiologicalObject
Person
RaceStaffMember
EventDirector
EventDoctor
Journalist
RaceSupervisor
Spectator
TeamMember
RaceCyclist
Climber
Sprinter
Wheeler
TeamDoctor
TeamManager

Similarity with Parent : he is a person

Similarity with Siblings : a property precises why the person is present during the race

Difference with Siblings : he is accredited by the race management

Difference with Parent : a property precises why the person is present during the race : he is accredited by the race management

Figure 1: The differential principles bound to the concept *Race Staff Member* in the *DOE* tool

In the first one, the user can build two trees (concepts and relations) of primitives defined by the means of the differential principles (Figure 1). The tool somewhat helps automating the typing of those NL definitions. In the second one, the tool allows to introduce in the two hierarchies primitives whose meaning is based on formal considerations. Multiples inheritance is managed, as well as the definition of the arity and domains of relations. Nevertheless, complete formal definition is not supported: we preferred not to implement what was quite well dealt with in existing tools. As a consequence, an export mechanism is implemented to translate the taxonomies into convenient exchange languages (for example, DAML+OIL). That mechanism can also be used to realize the third step by translating ontologies into knowledge-representation languages allowing computational operations.

[1]The tool is available for free at `http://opales.ina.fr/public/`.