An Approach for Ontology Modularization

Camila Bezerra [1], Fred Freitas[1], Jérôme Euzenat[2], Antoine Zimmermann [3]

[1]Centro de Informática - Universidade Federal de Pernambuco
Av. Prof. Luis Freire, s/n, Cidade Universitária - 50740-540
Recife-PE - Brasil

[2]INRIA Grenoble - Rhône-Alpes Inovallée,
655 avenue de l'Europe, Montbonnot 38 334
Saint Ismier, Cedex, France

[3]Digital Enterprise Research Institute, National University of
Ireland, Galway IDA Business Park, Lower Dangan, Galway, Ireland

{cbs,fred.freitas}@cin.ufpe.br

{Jerome.Euzenat, antoine.zimmermann}@inrialpes.fr

antoine.zimmermann@deri.org

Abstract. *Ontology modularization could help overcome the problem of defining a fragment of an existing ontology to be reused, in order to enable ontology developers to include only those concepts and relations that are relevant for the application they are modeling an ontology for. This paper presents a concrete tool that incorporates an approach to ontology modularization that inherits some of the main principles from object-oriented software engineering, which are encapsulation and information hiding. What motivated us to track that direction is the fact that most ontology approaches to the problem focus on linking ontologies rather than building modules that can encapsulate foreign parts of ontologies(or other modules) that can be managed more easily.*

1. Introduction

Although the field of ontology engineering would take advantage of reuse, a culture of building blocks for ontologies – such as the component culture that took over in the realm of object-oriented software engineering – has not shown up yet. This constitutes an absence that should not be neglected, once the realization of the Semantic Web depends on the ability to reuse ontologies [Stuckenschmidt and Klein 2004].

Currently, there are two major ways of reusing an ontology. Ontology editors such as Protégé[1] allow the reuse of another ontology by including it in the model that is being designed. The Web ontology language [McGuinness and van Harmelen 2004] offers the possibility to import an OWL ontology by means of the owl:imports statement. In both cases, the whole ontology has to be included. Ontology modularization could help overcome the problem of defining a fragment of an existing ontology to be reused, in order to enable ontology developers to include only those concepts and relations that are relevant for the application they are modeling an ontology for.

[1]http://protege.stanford.edu/

This paper presents a concrete tool that incorporates an approach to ontology modularization that inherits principles of object-oriented software engineering. What motivated us to track that direction is the fact that most ontology approaches to the problem focus on linking ontologies (or modules) rather than building modules that can encapsulate foreign parts of ontologies (or other modules) that can be managed more easily.

2. Ontology Modularization

The notion of modularization comes from Software Engineering where it refers to develop software in structured way that supports the combination of self-contained components that are easier to build, reuse and maintain.

From an ontology engineering perspective, modularization should be considered as a way to structure ontologies, meaning that the construction of a large ontology should be based on the combination of self-contained, independent and reusable knowledge components[d'Aquin et al. 2007]. A possible definition for an ontology module is the following: "An ontology module is a reusable component of a larger or more complex ontology, which is self-contained but bears a definite relationship to other ontology modules" [P.Doran 2006]. Alan Rector adds to that definition by specifying three requirements that should hold for ontology modules [Blomqvist 2004]: loose coupling, which means the modules might have very little in common, and therefore as little interaction as possible should be required between the modules; self-containment, which means that every module should be able to exist and function without any other module; integrity, which means that there should be ways to check for, and adapt to, changes in order to ensure correctness. In the next section, we describe our approach to modularization of ontologies and our modularization tool, ModOnto.

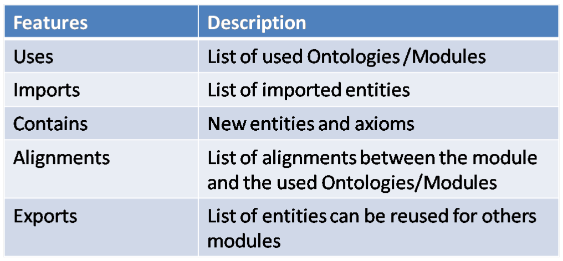
3. The Proposed Ontology Modularization Approach

The goal of our ongoing research is to define and implement an ontology modularization approach that provides the definition and composition of modules(or ontologies). The approach was inspired by the object-oriented components market, for it is indeed the most successful reuse approach in the computer science mainstream. The properties typically expected from a module system are: separate development, separate execution, separate compilation and reusability.

Bearing this in mind, we defined a module language which comprehends some aspects related to an organized way of implementing module composition with the above principles. On the syntactic side, a module language must be able to encapsulate ontology fragments, to refer to other modules and to define the interface between these modules. The module tags that compose the language syntax to define modules are displayed in Figure 1 below.

From this syntax, every feature represents a block of information of the module.

- The Uses Feature is the list of external modules or ontologies being used to build the new one.
- Imports are the list of entities (classes, properties and instances) from other ontologies or modules that are needed to create the module.
- Contains represents the new entities and axioms created using or not imported entities.



Features	Description
Uses	List of used Ontologies /Modules
Imports	List of imported entities
Contains	New entities and axioms
Alignments	List of alignments between the module and the used Ontologies/Modules
Exports	List of entities can be reused for others modules

Figure 1. Module Language tags and descriptions.

- The alignments feature is needed when working with different modules and mappings are needed involving exported entities.
- The exports are entities from the content or from the list of the imported entities that will be available when one reuses the module being defined, i.e. , the classes, properties, axioms and individuals ready for reuse.

On the semantic side, it is necessary to define unambiguously what is the meaning of "encapsulating" is, i.e., what is hidden and what is exposed in a module. This definition leads to a way to determine which assertions are logical consequences of the defined modules. For that purpose we have provided our ontology modularization approach with very general definitions for the semantics, and then show that different actual semantics can be applied to our module language, each having advantages and drawbacks. The disadvantage of most distributed semantics is that knowledge does not fully propagate from imported modules to importing modules. It can be an advantage too, with regard to robustness towards heterogeneity. Besides this flexibility, our semantics' definitions possess another interesting feature in the context of ontology modules: It does not depend on a particular ontology semantics, but only needs to be able to determine what are the logical consequences of an ontology. This is very useful when using modules in which the ontology languages can be different and also enables connected ontologies or modules to be replaced by similar ones.

Beyond the design of the language with its syntax and semantics, a software tool is needed for modularization to be accepted at a larger scale, thus avoiding burden over ontology developers. A Tool, called ModOnto (see figure2), has been developed as a standalone tool, providing an intuitive graphical interface for modularization that facilitates the selection of ontology entities for building modules.

Module API: based on the implementation of the module specifications as described in the syntax. Provide the basic data structure and accessors as well as a parser and serializer on top of the structure.

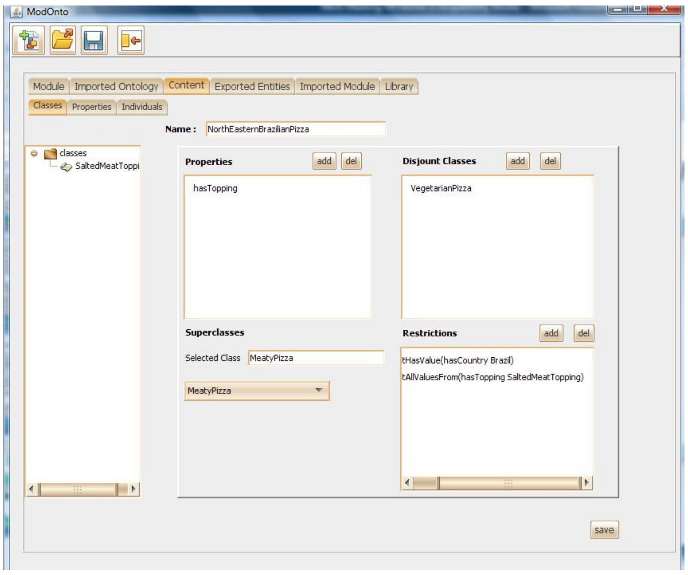
Module extractor: consist on the part of the tool that provide a graphical interface to extract elements from the imported ontology. A user of this tool should be able to make a 'select' over an ontology in a number ways, such as selecting/ruling out entities (classes, roles, individuals). It is possible to import items from more than one ontology.

Module checker: the checker should verify if all of the definitions needed by one module are present on its imported modules or in their own imported modules.

Module library: building on the Module API, the Module library is a repository for "off-the-shel" modules that will help developers to choose its proper modules.

Module linker: is possible to import entities from other other modules. Could be to interact with the module library to pick up imported modules to be included,

Reasoner interface for modules: modular ontologies are ontologies. So they should be usable just as yet another ontology: being able to query if an assertion is entailed by a particular module.



Name : NorthEasternBrazilianPizza

classes
SaltedMeatToppi

hasTopping

VegetarianPizza

Selected Class MeatyPizza

MeatyPizza

tHasValue(hasCountry Brazil)
tAllValuesFrom(hasTopping SaltedMeatTopping)

Figure 2. Create a new class in ModOnto

The Java programming language, the OWL API from Manchester, which contains an implementation for the W3C Web Ontology Language OWL, were used for the tool, as well as the alignment API[Euzenat 2004] and a reasoner based on IDDL[Zimmermann and Duc 2008] developed in INRIA Rhone-Alpes.

4. Related Work

During the last three years, there was an increasing interest and active research about ontology modularization. Some approaches propose translation of so called modular languages (DDL,e-connection, etc.) into standard DL and describe new reasoning services to ensure modularity, without introducing new formal languages. This can be satisfactory from a logical point of view but, is does not take into account engineering aspects like encapsulation, separation of ontologies from mappings, etc. For example, P-DL([Bao et al. 2006]) was designed in order to improve modularization of web ontologies. In P-DL, an ontology is divided into several packages (i.e., a module in their sense), which are interpreted separately, but the interpretations have to coincide on imported terms.

There are two different approaches have been considered for the modularization of existing ontologies: ontology partitioning, which divides an ontology into a set of modules, and module extraction, which reduces an ontology to a module focusing on a given set of elements. For example, Doran([P.Doran 2006]) proposed an approach which has been implemented as a standalone application, called ModTool which has an intuitive graphical interface. ModTool recursively applies a set of rules to generate an ontology module and define an abstract graph model and an algorithm for module extraction.

Regarding tool support for ontology modularization, for the goals of creating and encapsulating ontology modules (among other operations), there is no framework available yet, what could assure the originality of our work. However, at least one implementation for an ontology linkage approach deserves attention. The Manchester's OWL-API([Bechhofer et al. 2003]), the SWOOP Web Ontology browser and editor and the Pellet OWL reasoner have been extended to comply with E-connections ontologies. Some good features have been added to SWOOP with this extension, like incorrect editing prevention and foreign ontology visualization with graph layouts.

5. Conclusions and Future Work

Along the OntoCompo project, we have designed an ontology modularization approach inspired by one of the most successful techniques of computer science, object orientation. We developed a module language with syntaxes and a formally well-defined semantics and we present a concrete implementation of the language, the ModOnto tool that assists users in the construction of new modules starting from ontologies or other modules. This project anticipates an ontology module market that can possibly come out as a consequence of the Semantic Web, in the same flavor as the Java components market in the Web, when module repositories become available.

As we just produced the first prototype of the modularization tool, it naturally will be extended and improved in many ways. The suite must be integrated with popular ontology frameworks like Protégé [2], KAON2[3] and the ones from NeOn project[4] too.

6. Acknowledgements

The authors wish to thank CNPQ (National Council of Technological and Scientific Development) and INRIA (The French National Institute for Research in Computer Science and Control) for its partial sponsorship through the research project Composition and Modules for Ontology Engineering.

References

Bao, J., Caragea, D., and Honavar, V. (2006). Towards collaborative environments for ontology construction and sharing. In *CTS '06: Proceedings of the International Symposium on Collaborative Technologies and Systems*, pages 99–108, Washington, DC, USA. IEEE Computer Society.

Bechhofer, S., Volz, R., and Lord, P. (2003). Cooking the semantic web with the owl api. pages 659–675. Springer.

Blomqvist (2004). State of the art: Patterns in ontology engineering. Technical report, Jönköping University.

d'Aquin, M., Schlicht, A., Stuckenschmidt, H., and Sabou, M. (2007). Ontology modularization for knowledge selection: Experiments and evaluations. In *DEXA*, pages 874–883.

[2]http://protege.stanford.edu/
[3]http://kaon2.semanticweb.org/
[4]http://www.neon-project.org/

Euzenat, J. (2004). An api for ontology alignment. In *Proc. 3rd conference on international semantic web conference (ISWC), Hiroshima (JP)*, pages 698–712.

McGuinness, D. L. and van Harmelen, F. (2004). Owl. http://www.w3.org/TR/owl-guide/m.

P.Doran (2006). Ontology reuse via ontology modularisation. In *Proceedings of KnowledgeWeb PhD, Budva, Montenegro*.

Stuckenschmidt, H. and Klein, M. C. A. (2004). Structure-based partitioning of large concept hierarchies. In *International Semantic Web Conference*.

Zimmermann, A. and Duc, C. L. (2008). Reasoning on a Network of Aligned Ontologies. In Calvanese, D. and Lausen, G., editors, *Web Reasoning and Rule Systems, Second International Conference, RR 2008, Karlsruhe, Germany, October/November 2008, Proceedings*, volume 5341 of *Lecture Notes in Computer Science*, pages 43–57. Springer.