Ontology Modularization for Knowledge Selection: Experiments and Evaluations *

Mathieu d'Aquin[1], Anne Schlicht[2], Heiner Stuckenschmidt[2], and Marta Sabou[1]

[1] Knowledge Media Institute (KMi), The Open University, Milton Keynes, UK
{m.daquin, r.m.sabou}@open.ac.uk
[2] University of Mannheim, Germany
{anne, heiner}@informatik.uni-mannheim.de

Abstract. Problems with large monolithical ontologies in terms of reusability, scalability and maintenance have led to an increasing interest in modularization techniques for ontologies. Currently, existing work suffers from the fact that the notion of modularization is not as well understood in the context of ontologies as it is in software engineering. In this paper, we experiment on applying state-of-the-art tools for ontology modularization in the context of a concrete application: the automatic selection of knowledge components to be used for Web page annotation and semantic browsing. We conclude that, in a broader context, an evaluation framework is required to guide the choice of a modularization tool, in accordance with the requirements of the considered application.
Keywords: Ontology modularization, partitioning, module extraction

1 Introduction

Modularization is a crucial task to allow ontology reuse and exploitation on the Semantic Web. The notion of modularization comes from Software Engineering where it refers to a way of designing software in a clear, well structured way that supports maintenance and reusability. 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. In reality, even if they implicitly relate several sub-domains, most of the ontologies are not structured in a modular way. Therefore, in order to facilitate the management and the exploitation of such ontologies, *ontology modularization techniques* are required to identify and extract significant modules in existing ontologies.

While there is a clear need for modularization, there are no well-defined and broadly accepted definitions of modularity for ontologies. Several approaches have been recently proposed to extract modules from ontologies, each of them implementing its own intuition about what a module should contain and what

* This work is partially funded by the Open Knowledge (IST-FF6-027253) and NeOn projects (IST-FF6-027595), and partially supported by the German Science Foundation under contract STU 266/1 as part of the Emmy-Noether Program.

should be its qualities, generally without making this intuition explicit. This lack of consensus and of clarity hinders the application of these techniques in concrete scenarios, leading to difficulties in choosing the appropriate one. Moreover, to our knowledge, no other study has focused on the evaluation and comparison of ontology modularization techniques.

Our hypothesis is that there is no universal way to modularize an ontology and that the choice of a particular technique should be guided by the requirements of the considered application. We believe that modularization criteria should be defined in terms of the applications for which the modules are catered. For this reason, we detail in this paper some experiments conducted with several ontology modularization tools on a particular application: the selection of relevant knowledge components from online available ontologies. The goal is to characterize the requirements of this particular application using criteria from the literature on ontology modularization, and thus, to analyze the results of existing ontology modularization techniques regarding these requirements. In this way, we aim at better understanding the fundamental assumptions underlying the current modularization techniques. This work can be seen as a first step towards a broader framework, guiding application developers in choosing the appropriate technique and the designers of techniques in further developments.

The paper is structured as follows. Section 2 briefly describes the concrete scenario in which we apply modularization techniques. Section 3 and Section 4 respectively overview ontology modularization techniques and evaluation criteria that have been proposed in the literature. In Section 5 we evaluate, using the considered criteria, the results of the application of modularization techniques on our case-study. We conclude in Section 6 on the need for a comprehensive evaluation framework for ontology modularizations.

2 A Case-Study for Modularization: The Knowledge Selection Scenario

Knowledge selection has been described in [1] as the process of selecting the relevant knowledge components from online available ontologies and has been in particular applied to the Magpie application. Magpie [2] is a Semantic Web browser, available as a browser plugin, in which instances of ontology classes are identified in the current Web page and highlighted with the color associated to each class. In our current work we are extending Magpie towards open semantic browsing in which the employed ontologies are automatically selected and combined from online ontologies. As such, the user is relieved from manually choosing a suitable ontology every time he wishes to browse new content. Such an extension relies on mechanisms that not only dynamically select appropriate ontologies from the Web, but also extract from these ontologies the relevant and useful parts to describe classes in the current Web page.

Our previous work and experiences in ontology selection [3] made it clear that modularization may play a crucial role in complementing the current selection techniques. Indeed, selection often returns large ontologies that are virtually

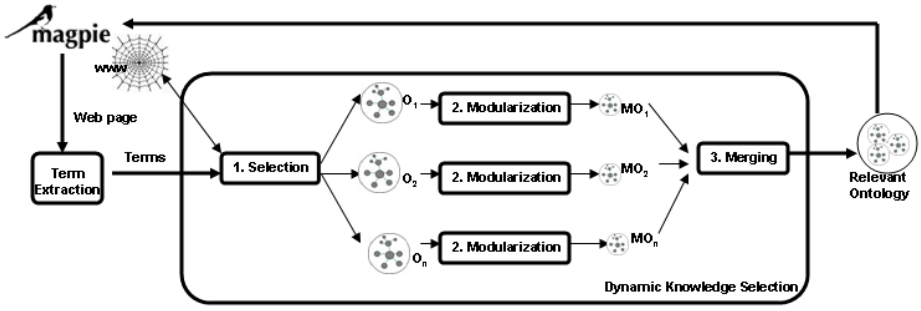

Fig. 1. The knowledge selection process and its use for semantic browsing with Magpie.

useless for a tool such as Magpie which only visualises a relatively small number of classes at a time. What is needed instead is that the selection process returns a part (module) of the ontology that defines the relevant set of terms. These considerations justify the need to extend selection techniques with modularization capabilities. In Figure 1 we depict the three major generic steps of the *knowledge selection* process that integrates ontology selection, modularization and merging. We focus in this paper on applying existing techniques for the second step of this process: ontology modularization.

3 Modularization Techniques

We consider an ontology O as a set of axioms (subclass, equivalence, instantiation, etc.) and the signature $Sig(O)$ of an ontology O as the set of entity names occurring in the axioms of O, i.e. its vocabulary.

In the following, we deal with several approaches for ontology modularization, having different assumptions about the definition of an ontology module. Therefore, we define an ontology module in a very general way as a part of an ontology: a module $M_i(O)$ of an ontology O is a set of axioms, such that $Sig(M_i(O)) \subseteq Sig(O)$.

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.

3.1 Ontology Partitioning Approaches

The task of partitioning an ontology is the process of splitting up the set of axioms into a set of modules $\{M_1, \cdots, M_k\}$ such that each M_i is an ontology and the union of all modules is semantically equivalent to the original ontology O. There are several approaches for ontology partitioning that have been developed for different purposes. We have chosen to consider only available techniques that are sufficiently stable:

PATO refers to a standalone application described in [4]. The goal of this approach is to support maintenance and use of very large ontologies by providing the possibility to individually inspect smaller parts of the ontology. The algorithm operates with a number of parameters that can be used to tune the result to the requirements of a given application.

SWOOP refers to the partitioning functionality included in the SWOOP ontology editor and described in [5]. This tool partitions an ontology into a set of modules connected by ε-connections. It aims at preserving the completeness of local reasoning within all created modules. This requirement is supposed to make the approach suitable for supporting selective use and reuse since every module can be exploited independently of the others.

3.2 Module Extraction Approaches

The task of module extraction consists in reducing an ontology to the sub-part, the module, that covers a particular sub-vocabulary. This task has been called segmentation in [6] and traversal view extraction in [7]. More precisely, given an ontology O and a set $SV \subseteq Sig(O)$ of terms from the ontology, a module extraction mechanism returns a module M_{SV}, supposed to be the relevant part of O that covers the sub-vocabulary SV ($Sig(M_{SV}) \supseteq SV$). Techniques for module extraction often rely on the so-called *traversal approach*: starting from the elements of the input sub-vocabulary, relations in the ontology are recursively "traversed" to gather related elements to be included in the module.

Two module extraction tools are considered here:

KMi refers to a standalone application developed at the Knowledge Media Institute (KMi) of the Open University, for the purpose of the knowledge selection scenario, as described in [1]. The input sub-vocabulary can contain either classes, properties, or individuals. The mechanism is fully automatized, is designed to work with different kinds of ontologies (from simple taxonomies to rich and complex OWL ontologies) and relies on inferences during the modularization process.

Prompt refers to the module extraction feature of the Prompt toolkit, integrated as a plugin of the Protégé ontology editor, as described in [7]. This approach recursively follows the properties around a selected class of the ontology, until a given distance is reached. The user can exclude certain properties in order to adapt the result to the needs of the application.

It is worth mentioning that the technique described in [6] is also freely available, but can only be used on the Galen ontology in its current state.

4 Evaluation Criteria for Modularization

In the previous section, we have briefly presented a number of different approaches for ontology partitioning and module extraction. In this section, we take a closer look at different criteria for evaluating either the modules resulting from the application of a modularization technique, or the system implementing this technique.

4.1 Evaluating the Result of Modularization

In [8], the authors describe a set of criteria based on the structure of the modularized ontology and that have been designed to trade-off maintainability as well as efficiency of reasoning in a distributed system, using distributed modules.

Size. Despite its evident simplicity, the relative size of a module (number of classes and properties) is among the most important indicators of the efficiency of a modularization technique. Indeed, the size of a module has a strong influence on its maintainability and on the robustness of the applications relying on it.

Redundancy. Allowing the modules of a partition to overlap is a common way of improving efficiency and robustness. On the other hand, having to deal with redundant information increases the maintenance effort.

Connectedness. The independence of a set of modules resulting from a partitioning technique can be estimated by looking at the degree of interconnectedness of the generated modules. A modularized ontology can be depicted as a graph, where the axioms are nodes and edges connect every two axioms that share a symbol. The connectedness of a module is then evaluated on the basis of the number of edges it shares with other modules.

Distance. It is worth to measure how the terms described in a module *move closer to each other* compared to the original ontology, as an indication of the simplification of the structure of the module. This *intra-module distance* is computed by counting the number of relations in the shortest path from one entity to the other. An *inter-module distance*, counting the number of modules that have to be considered to relate two entities, can also be envisaged, as a way to to characterize the communication effort caused by the partition of an ontology.

Several authors also defined criteria for evaluating ontology modules, in general focusing on the logical and formal aspects of modularizations (see e.g., [9]). Logical criteria are of particular importance when the modules resulting of the modularization techniques are intended to be used in reasoning mechanisms, but should not be emphasized in our case-study, which focuses on a human interpretation of the module.

4.2 Evaluating the Modularization Tool

In [1] the authors focus on the use of modularization for a particular application. This leads to the definition of several criteria, most of them characterizing the adequacy of the design of a modularization tool with respect to constraints introduced by the application.

Assumptions on the ontology. Most of the existing approaches rely on some assumptions. For example, those described in [5] and [6] are explicitly made to work on OWL ontologies, whereas [4] can be used either on RDF or OWL but only exploits RDF features.

Level of user interaction. In many systems the required user entries are limited to the inputs of the algorithm. In certain cases, some numerical parameters can be required [4] or some additional procedures can be manually (de)activated [6]. The technique in [7] has been integrated in the Protégé ontology editor to support knowledge reuse during the building of a new ontology. In this case, modularization is an interactive process where the user has the possibility to extend the current module by choosing a new starting point for the traversal algorithm among the *boundary classes* of the module.

Performance. Most of the papers concerning modularization techniques do not give any indication about the performance of the employed method (with the noticeable exception of [6]). Performance is a particularly important element to be considered when using a modularization technique for the purpose of an application. Different applications may have different requirements, depending on whether the modularization is intended to be used dynamically, at run-time, or as a "batch" process.

5 Experiments

In the scenario described in Section 2, modularization is integrated in a fully automatic process, manipulating automatically selected online ontologies for the purpose of annotation in Magpie. In this section, we simulate the process of knowledge selection on two examples, using four different techniques, in order to evaluate and compare their results[3]. The purpose is to characterize the requirements of this particular scenario using the criteria defined in Section 4, and to show how modularization techniques respond to the selected experiments regarding these requirements.

As already described in [1], it is quite obvious that module extraction techniques fit better in the considered scenario than partitioning tools. Indeed, we want to obtain *one* module covering the set of keywords used for the selection of the ontology and constituting a *sub-vocabulary* of this ontology. However, the result of partitioning techniques can also be used by selecting the set of generated modules that cover the considered terms. The criteria are then evaluated on this set of modules as grouped together by union. Furthermore, we primarily focus on the criteria that appear to be relevant in our scenario: application related criteria (Section 4.2), the size, and the intra-module distance (Section 4.1).

5.1 Considered Ontologies

We consider two examples, originally described in the context of ontology selection in [3], where the goal is to obtain an ontology module for the annotation of news stories. We simulate the scenario described in Section 2 by manually

[3] Actual results are available at `http://webrum.uni-mannheim.de/math/lski/Modularization`

extracting relevant keywords in these stories, using ontology selection tools[4] to retrieve ontologies covering these terms, and then applying modularization techniques on these ontologies (steps 1 and 2 in figure 1).

In the first example, we consider the case where we want to annotate the news stories available on the KMi website[5]. We used the keywords *Student*, *Researcher*, and *University* to select ontologies to be modularized, and obtain three ontologies covering these terms:

ISWC: `http://annotation.semanticweb.org/iswc/iswc.owl`
KA: `http://protege.stanford.edu/plugins/owl/owl-library/ka.owl`
Portal: `http://www.aktors.org/ontology/portal`

It is worth mentioning that this example is designed to be simple: we have chosen a well covered domain and obtained three well defined OWL ontologies of small sizes (33 to 169 classes).

The second example was used in [3] to illustrate the difficulties encountered by ontology selection algorithms. Consequently, it also introduces more difficulties for the modularization techniques, in particular because of the variety of the retrieved ontologies in terms of size and quality. It is based on the following news snippet:

"The Queen will be 80 on 21 April and she is celebrating her birthday with a family dinner hosted by Prince Charles at Windsor Castle"[6]

Using the keywords *Queen*, *Birthday* and *Dinner*, we obtained the following ontologies, covering (sometimes only partially) this set of terms:

OntoSem: `http://morpheus.cs.umbc.edu/aks1/ontosem.owl`
TAP: `http://athena.ics.forth.gr:9090/RDF/VRP/Examples/tap.rdf`
Mid-Level: `http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl`, covering only the terms *Queen* and *Birthday*

Compared to Example 1, the ontologies used in Example 2 are bigger (from 1835 classes in **Mid-Level** to 7596 in **OntoSem**). Moreover, they contain different levels of descriptions. For example, **OntoSem** is a big, complex OWL ontology containing a lot of properties (about 600), whereas TAP is simple RDFS taxonomy without any properties. In that sense, we use Example 1 to assess basic characteristics of the modularization techniques and then, rely on Example 2 to show how these characteristics are influenced by the properties of the ontologies.

5.2 Results for Example 1

Running the four modularization techniques on the three ontologies of the first example allowed us to test how they behave on simple, but yet practical real word examples.

[4] in particular Watson (`http://watson.kmi.open.ac.uk`).
[5] `http://news.kmi.open.ac.uk/`
[6] `http://news.billinge.com/1/hi/entertainment/4820796.stm`

Concerning the *level of user interaction*, **SWOOP** is fully automatic and does not need any parameters besides the input ontology. As a module extraction tool, **KMi** requires, in addition to the source ontology, a set of terms from the signature of the ontology, defining the sub-vocabulary to be covered by the module. This sub-vocabulary corresponds to the initial terms used for selecting the ontology: *Researcher*, *Student* and *University*. **Pato** has to be fine tuned with several parameters, depending on the ontology and on the requirements of the application. Here, it has been configured in such a way that modularizations in which the considered terms are in the same module are preferred. **Prompt** is an interactive mechanism, in which the user is involved in each step of the process. In particular, the class to be covered and the property to traverse have to be manually selected, requiring that the user has a good insight of the content of the ontology, can easily navigate in it, and that he understands the modularization mechanism. When using **Prompt**, we manually included the input terms and tried to obtain an (intuitively) good module, without going too deep in the configuration. Note that, since the system crashed at the early stage of the process, we did not manage to obtain results for the **KA** ontology with **Prompt**.

Concerning *performance*, apart from **Prompt** for which this criteria is irrelevant, each tool has only taken a few seconds or less on these *small ontologies*. Experiences in Example 2 should give us a better insight on this criteria and on the way techniques behave on different and larger ontologies.

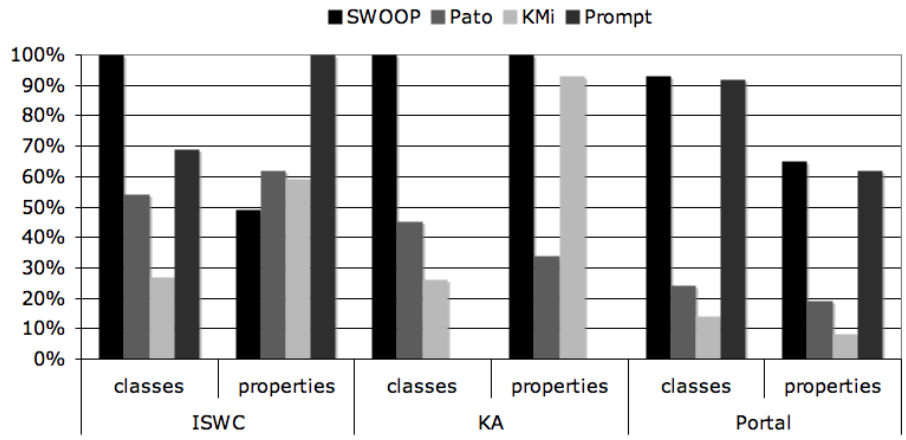

Fig. 2. Relative size of the resulting modules for the first example.

Figure 2 shows the *size* of the resulting modules for each system in terms of number of classes and properties. It can be easily remarked that **SWOOP** generally generates very large modules, containing 100% of the classes for two of the three ontologies, and an important proportion of the properties: in most of the cases, **SWOOP** generates one module with almost the same content as the

original ontology. The tool developed in **KMi** is focused on generating modules with a small number of classes (the smallest), so that the ontology hierarchy would be easy to visualize. It nevertheless includes a large proportion of the properties, in order to keep the definition of the included classes intact. **Pato** is optimized to give an appropriate size. It generally operates an important reduction of the size of the ontology.

The **KMi** tool relies on mechanisms that "take shortcuts" in the class hierarchy[7] for reducing the size of the module. It is thus the only one that potentially reduces the *intra-module distance* between the considered terms. For example, in the **Portal** ontology, by eliminating an intermediary class between *Researcher* and *Person*, **KMi** has reduced the distance between *Researcher* and *Student*, while keeping a well formed structure for the module.

5.3 Results for Example 2

The second example concerns larger ontologies, with more heterogeneous levels of description. For example, **TAP** contains around 5500 classes, but no property or individual, whereas **Mid-Level** relies on almost 200 properties and is populated with more than 650 individuals for less than 2000 classes. These elements obviously have an important impact on the *performance* of the modularization techniques: in the worst cases (**Pato** and **KMi** on **TAP**), it takes several minutes to get a modularization and none of the tested techniques can be used at run-time for such ontologies.

Moreover, some of the techniques are not designed to take into account such big and heterogeneous ontologies. It is particularly hard for the user to handle the process of module extraction in **Prompt** when having to deal with several thousands of classes and hundreds of properties. We also did not manage to partition the **OntoSem** ontology using **Pato**.

Finally, concerning the *size* of the resulting modules, the difference between **SWOOP** and other techniques is even more significant in this example. Indeed, because of the poor structure of the considered ontologies (restricted uses of OWL constructs, few or insufficiently defined properties), **KMi** and **Pato** result in particularly small modules (less than 10 classes), whereas **SWOOP** still includes most of the content of the ontology in a single module. Therefore, regarding the requirement about the *assumption on the ontology*, this shows that techniques are highly influenced by the inherent properties of the ontology to be modularized and that, in general, they *assume* a high level of description.

6 Conclusion and Discussion: Towards a Benchmark for Modularization Techniques

There is currently an important growth in interest concerning modularization techniques for ontologies, as more ontology designers and users become aware

[7] Instead of including all the super-classes of the included classes, it only considers classes that relate these entities: their common super-classes.

of the difficulty of reusing, exploiting and maintaining big, monolithic ontologies. The considered notion of modularity comes from software engineering, but, unfortunately, it is not yet as well understood and used in the context of ontology design as it is for software development. Different techniques implicitly rely on different assumptions about modularity in ontologies and these different *intuitions* require to be made explicit.

This paper reports on preliminary steps towards the characterization of ontology modularization techniques. We reviewed existing modularization tools as well as criteria for evaluating different aspects of a modularization, and used them on a particular scenario: the automatic selection of knowledge components for the annotation of Web pages. The main conclusion of these experiments is that the evaluation of a modularization (technique) is a difficult and subjective task that requires a formal, well described framework – a *benchmark* – taking into account the requirements of applications. Such a framework would be useful in two ways: first for application developers, it would provide a guide for choosing the appropriate modularization technique, and second, for the developers of modularization techniques, it would give directions in which techniques can be improved with respect to particular scenarios. The definition of this evaluation framework requires to build an adequate, well understood dataset for benchmarking and to improve the definition of the criteria for evaluation, in particular to allow the expression of requirements concerning subjective notions like the *quality of the module*.

References

1. d'Aquin, M., Sabou, M., Motta, E.: Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)
2. Dzbor, M., Domingue, J., Motta, E.: Magpie - towards a semantic web browser. In: Proc. of the Second International Semantic Web Conference (ISWC). (2003)
3. M. Sabou, V.L., Motta, E.: Ontology Selection on the Real Semantic Web: How to Cover the Queens Birthday Dinner? In: Proc. of the European Knowledge Acquisition Workshop (EKAW), Podebrady, Czech Republic (2006)
4. Stuckenschmidt, J., Klein, M.: Structure-Based Partitioning of Large Concept Hierarchies. In: Proc. of the International Semantic Web Conference (ISWC). (2004)
5. Cuenca Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Automatic Partitioning of OWL Ontologies Using E-Connections. In: Proc. of Description Logic Workshop (DL). (2005)
6. Seidenberg, J., Rector, A.: Web Ontology Segmentation: Analysis, Classification and Use. In: Proc. of the World Wide Web Conference (WWW). (2006)
7. Noy, N., Musen, M.: Specifying Ontology Views by Traversal. In: Proc. of the International Semantic Web Conference (ISWC). (2004)
8. Schlicht, A., Stuckenschmidt, H.: Towards Structural Criteria for Ontology Modularization. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)
9. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: A Logical Framework for Modularity of Ontologies. In: Proc. of the International Joint Conference on Artificial Intelligence, IJCAI. (2007)