Acta Cybernetica **19** (2010) 591–605.

An Ontology Segmentation Tool*

András Simonyi† and Miklós Szőts†

Abstract

Extracting a minimal relevant segment of an extensive domain ontology is an often recurring problem in ontology engineering. We present a software solution to this problem that is the combination of an ontology-independent user interface generator and a module implementing an ontology segmentation algorithm. We describe the algorithm and compare it with other ontology segmentation methods proposed in the literature.

Keywords: ontology segmentation, ontology module extraction, user interface generation, cross-domain engineering

1 Introduction

The goal of the ImportNET project has been to develop an 'intelligent modular open source platform for intercultural and cross-domain SME networks' for the field of mechatronic design. According to the high level architecture, the general knowledge of the field is represented in a comprehensive ontology, the *reference ontology* in ImportNET terminology. When a new project is started, the relevant knowledge, represented in a so called *collaboration ontology*, is generated from the reference ontology by a software tool named *Ontology Integration Tool* (henceforth OIT).[1] The generation process consists of two phases:

1. Domain experts mark the important relevant/irrelevant components of the reference ontology, and make some minor modifications (add individuals, concepts etc.).

2. OIT generates a minimal (as far as the algorithm allows) subontology of the reference ontology that contains those pieces of knowledge that are related to the marked items.

*This work has been supported by ImportNET, a research and development project co-funded by the European Commission within the Sixth Framework Programme under Contract 033610, in the area of ICT for Networked Businesses.

†Applied Logic Laboratory, E-mail: {simonyi,szots}@all.hu

[1]This approach was first outlined in [9].

During the development of OIT we have devised a general method for the selection of the relevant subontology — the present paper offers an overview of this approach. Although in the ImportNET platform the generated collaboration ontology has to be transformed into an object oriented data model, we do not discuss this phase here.

2 Problem Analysis

There is a well known taxonomy of ontologies according to their generality: [6] introduced the notions of upper, domain, and application ontologies. We quote the characterisation of upper and domain ontologies from [12]:

> An upper ontology [...] is a high-level, domain-independent ontology, providing a framework by which disparate systems may utilise a common knowledge base and from which more domain-specific ontologies may be derived. The concepts expressed in such an ontology are intended to be basic and universal concepts to ensure generality and expressivity for a wide area of domains. (p. 2-2)

There are some well known upper ontologies; we use DOLCE [8]. Upper ontologies are in marked contrast to domain ontologies:

> A domain ontology specifies concepts particular to a domain of interest and represents those concepts and their relationships from a domain specific perspective. While the same concept may exist in multiple domains, the representations may widely vary due to the differing domain contexts and assumptions. [12, p. 2-3]

Finally, an application ontology is used in a software system. It represents a part of the domain knowledge that is relevant to a special task concerning the domain in question, and may be tuned to the requirements of the application.

Note that the above characterisations cannot be regarded as precise definitions, since their meaning depends on what we consider a domain. Moreover, there may exist ontologies that formalise only a small piece of knowledge concerning a domain, but are not connected to any software system — such an ontology would not fit into the above classification.

Clearly, if we have a domain ontology and an application ontology for the same domain, then the application ontology (with certain modifications) is a subontology of the domain ontology. Accordingly, the reference ontology of the ImportNET project can be considered a domain ontology and the collaboration ontology an application ontology.

In more general terms our problem can be formulated in the following way: how to obtain an application ontology from a domain ontology? In order to obtain an application ontology, firstly the intended application itself has to be specified. The simplest way of doing so is to mark some ontology items as relevant or irrelevant to

the application in question. In this case an appropriate application ontology will be a subontology of the domain ontology that

- contains the items marked relevant, but does not contain those marked irrelevant,
- contains those ontology items and pieces of information that are connected to the relevant items and are not marked irrelevant.

Of course, an *optimal* application ontology has to satisfy a further condition: it has to be minimal among the appropriate application ontologies in the sense that it cannot have a (proper) subontology which is also appropriate. The proper construal of the notion 'ontology items and pieces of information that are connected to the relevant items' is itself part of the problem. There are two important factors that must be taken into account by any solution of the problem: the restrictions contained by the ontology and the criterion of connectedness in the user's mind. While the first factor can be calculated automatically, the second has to be specified by the user. Consequently, the following two problems have to be addressed:

1. The naive user, who does not know the structure of the ontology, has to be able to select the relevant concepts and to parametrise the criterion of connectedness to be used, via a graphical user interface.
2. The concepts marked relevant and the connected ontology items have to be integrated into a quasi-optimal subontology which is an appropriate application ontology and a good approximation of an optimal one.

The next two sections present our solutions to these problems.

3 A Graphical User Interface for Naive Users

The need for generating user interfaces for ontologies is not new; see e.g. [2]. Paper [1] presents an ontology based portal where the interface is generated from a domain ontology and a small ontology describing GUI elements. Our problem is slightly different from theirs, because we aim at developing a general tool, which is ontology independent. In our solution (see Figure 1 on the following page) the specification of the interface is represented in an XML file called *design template*. A design template describes for every screen (state) of the user interface

- the GUI objects shown by it,
- how the data presented by the GUI objects can be obtained from the ontology,
- how to edit the ontology according to the user's activity,
- how to change the screens (states).

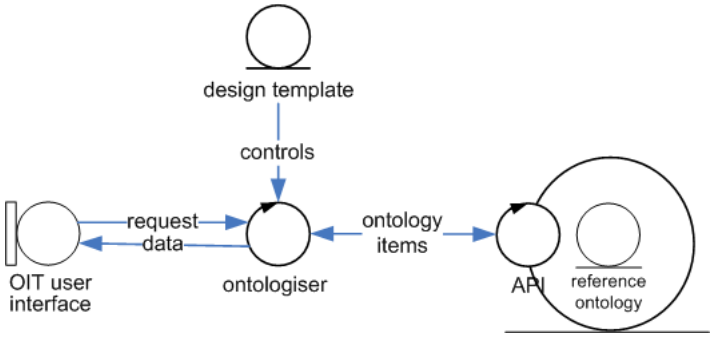

Figure 1: User interface architecture

The software module *Ontologiser* parses the design template, interprets it, and controls the operation of the whole program. The ontology, which is formulated in the OWL-DL ontology language, is stored in the memory of the Protégé editor [18] and accessed via Protégé's APIs. The user interface is totally thin: it only shows data received from the Ontologiser and receives data from the user, which is in turn transferred to the Ontologiser module.

A more detailed explanation of the structure of design templates and the operation of the Ontologiser module can be found in [17].

4 Integration Algorithm

A concept can be connected to a relevant concept in one or more of the following ways:

1. *Via a relevant ontology relation.* We discuss this type of connection, which we term 'relational connection', in the next subsection.

2. *By being a subconcept of the relevant concept and not marked irrelevant.* It can be assumed that if a subconcept of a relevant concept is irrelevant, then this fact is explicitly indicated by the user — in absence of such an indication, there is no reason to suppose that a subconcept can be left out from the collaboration ontology.

3. *By being a superconcept of the relevant concept and not marked irrelevant.* In contrast to the previous two connection types, superconcepts of a relevant concept are, in many cases, not specifically relevant to the collaboration. Nevertheless, they have to be included into the collaboration ontology in order to ensure that it forms a proper ontology with an appropriate upper ontology layer.

Starting from the concepts marked relevant by the user, the integration algorithm first has to find all concepts of the reference ontology that are relevant to the

collaboration by being connected to a relevant concept (either via a relational connection or the subconcept relationship). Secondly, the relevant fragment of the reference ontology has to be extended into a self-contained ontology by including the superconcepts of the relevant concepts.

4.1 Generating the Hull of a Concept

One of the key tasks of the sketched integration process is to find those concepts that are connected to a single relevant concept through relational connections. An ontology fragment containing just these concepts and the relevant concept in question is called the *hull* of the concept [9]. Before describing the integration algorithm, first we examine and make more precise the notions 'relational connection' and 'hull'.

4.1.1 Logical Analysis

There is a **relational connection** between two concepts A and B via relation R if and only if

(i) there may exist two individuals that are instances of A and B respectively and relation R holds between them, formally $\Diamond\exists x\exists y(A(x) \wedge B(y) \wedge R(x,y))$; and

(ii) the ordered pair $\langle A, B\rangle$ is minimal among those pairs of concepts that satisfy condition (i), that is, there are no concepts A' and B' such that A' is a subconcept of A, B' is a subconcept of B, $A' \neq A$ or $B' \neq B$, and $\Diamond\exists x\exists y(A'(x) \wedge B'(y) \wedge R(x,y))$.

The modality invoked in the definition is that of conceptual possibility or conceivability (from the user's point of view) — two ontology concepts are connected via a relation R if the user deems it conceptually possible that they have instances standing in relation R, and they also satisfy the minimality condition given by (ii).

The **hull** of a selected concept C is generated by forming the closure of the set $\{C\}$ under relational connections. The following is a simple illustration of the evolution of a concept's hull (see also Figure 2 on the next page):

Let A be a selected concept, let there be relational connections between A and $B_1, \ldots B_n$ via relations $R_1, \ldots, R_n$, respectively; and similarly relational connections between $E_1, \ldots, E_m$ and A via $Q_1, \ldots, Q_m$. At the first step concepts $B_1, \ldots, B_n$ and $E_1, \ldots, E_m$ make up the hull of A. The generation of the hull is iterative: in the same manner new concepts have to be added to $B_1, \ldots, B_n, E_1, \ldots, E_m$ and so on. Note that the inverses of the relations also have to be considered when relational connections are searched for. Clearly, relations $R_1, \ldots, R_n$ and $Q_1, \ldots, Q_m$ also belong to the hull.

Unfortunately, description logic does not allow expressing the notion of relational connection in the form we have defined above, and therefore we cannot determine precisely whether there is a relational connection between two concepts

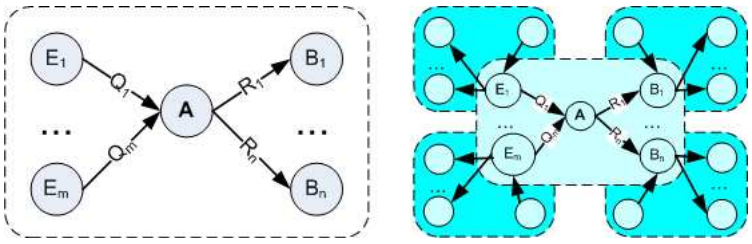

Figure 2: Generation of the hull of a concept

of an ontology that is formulated in a description logic language. Assuming that the class of concepts is closed under intersection, we can formulate a simple necessary condition: there may be a relational connection between two concepts A and B via relation R only if A is a subconcept of the domain of R, and B is a subconcept of the range of R. This condition is too permissive, since there may well be several subconcepts of the domain and range of a relation without a relational connection between them. This kind of situation often occurs when a relation is defined at a very high level. For instance, the PART_OF relation[2] often has the root concept of the ontology as its domain and range — e.g. in the DOLCE top ontology the domain and the range of PART_OF is the most general concept PARTICULAR. Nonetheless, certain restrictions provide clues that can help deciding whether there is a relational connection between two concepts.

Restrictions[3] on a concept A that *imply* the existence of some relational connections via R:

someValuesFrom	$\exists R.C$	There must be a relational connection between A and C via R.[4]
minCardinality	$\geq nR$	There must be a relational connection between A and one of the subconcepts of R's range via R.
hasValue	$R : i$	There must be a relational connection between A and the minimal concept(s) containing i via R.

Restrictions on a concept A that *exclude* the existence of some relational connections via R:

allValuesFrom	$\forall R.C$	C is the only subconcept of the range of R such that there is a relational connection between A and it via R.[5]
	$\forall R.\bot$	A has no relational connections via R.

[2] PART_OF(a, b) means that a is part of b.

[3] We consider only restrictions used in OWL.

[4] The implication holds only if the ontology is well axiomatised in the sense that C does not have a proper subconcept C' for which $\exists R.C'$ is true.

[5] Analogously to the case of minCardinality, the implication holds only if the ontology is reasonably axiomatised insofar as C does not have a proper subconcept C' for which $\forall R.C'$ is true (see previous footnote).

Relying on these conditions, we can define the maximal and minimal hull of a concept, which provide an 'upper bound' and a 'lower bound' of the concept's hull in the sense that the hull of a concept contains its minimal hull and is contained by its maximal hull. The **maximal hull** of a concept C is computed by forming the closure of $\{C\}$ under the relation which holds between two concepts A and B iff there is a relation R such that A is a subconcept of R's domain and B is a subconcept of R's range, while the **minimal hull** is calculated by forming the closure of $\{C\}$ under the relation which holds between A and B iff there is an R for which the ontology contains a restriction on A that *implies* that A and B is connected via R.

4.1.2 Heuristic Rules

Although both the minimal and maximal hull of a concept can be precisely determined in the case of a description logic based ontology, in practice we look for an ontology fragment which is in between the minimal and maximal hulls, as the maximal hull is frequently too large — sometimes including the whole reference ontology — while the minimal hull may leave out relevant concepts whose relevance is not declared explicitly by restrictions. To find an appropriate fragment, the knowledge of the ontology engineer or user has to be relied upon — knowledge which is often not expressible in the ontology language, but nevertheless can be utilised for checking the extension of hulls in the form of heuristic rules.

We have considered two types of such knowledge:

- the meaning of a relation makes it superfluous to consider its domain or range — this makes it possible to use relation filtering in OIT;
- the user may provide some information in the selection phase that may control the selection of relational connections.

The first case can be clarified using the example of the PART_OF relation. Knowledge concerning the parts of an object may surely be relevant if the object in question is relevant. Consequently, if the selected concept is included in the range of the PART_OF relation, then its domain has to be included into its hull. However, even if something is relevant for a collaboration, it cannot be said that everything that contains it as a part is also relevant. Accordingly, if the concept is in the domain of the PART_OF relation, the range may be irrelevant, and it may not get into the hull. Since the PART_OF relation is transitive, this means that only a segment of a chain of concepts is added to the hull, as Figure 3 shows.

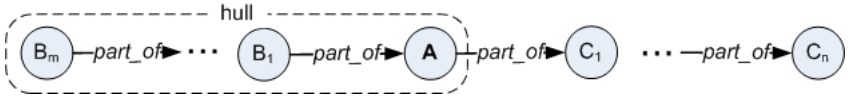

Figure 3: An example of directed relations

There can be other relations whose domain or range may be irrelevant — we will call relations belonging to this class **directed relations**. They have to be collected from the actual reference ontology, and it has to be decided in which direction they are irrelevant. In our implementation this information is described in a parameter file.

We have introduced two ways of manual control, both provided in the selection (customisation) phase: global and local manual control. In both cases some relations are marked as irrelevant for the hull generation process.

Global manual control influences the hull generation in a completely general manner: the user may select some relations that are not to be considered in the generation of the hull at all.

In the case of **local manual control** the user may mark certain relational connections of some concepts as irrelevant, and these connections are not considered when generating the hull. The user interface outlined in the previous section helps the user to select these relational connections.

In general, global and local manual control can help to generate a quasi-optimal hull. However, certain relations require special ways of handling. We have used the quality/quale structure of the DOLCE upper ontology [8] to model the properties of concepts. In this case subrelations of the relation HAS-QUALITY have to be considered only if the necessary conditions of relational connections hold. At the same time, when a quality concept is included, the corresponding quale concept has to be included as well.

4.2 The Algorithm

The collaboration ontology is generated in a complicated system of iterations. During this process we do not build a new ontology, but mark the concepts and relations to be included in the collaboration ontology with labels. The collaboration ontology as a new ontology is generated only when the user indicates that the process has been finished. This solution allows saving the labelled reference ontology, and using it again as a starting point when a new project is built upon the previous one.

Concepts can enter the collaboration ontology in a number of different ways:

- the user selects them,
- they are in the hull of an included concept,
- they are subconcepts of an included concept,
- they are superconcepts of an included concept.

In the last three cases the concepts are included by the integration module of the program. Although these cases are handled by three different procedures, they have to be organised into an iterative process, since the subconcepts of the concepts generated as elements of a hull also have to be generated, and the hulls of subconcepts have to be included as well. As we have already noted, the hulls of superconcepts

do not have to be generated; the superconcepts are needed only to form a complete ontology from the generated concepts. In order to control the iteration, the included concepts are collected into a list (called CONCEPT_TO_BE_PROCESSED), and it is marked whether the 'generation of hull' and 'generation of subconcepts' steps of the algorithm (see below) have already been executed on them.

The process of generating the collaboration ontology consists of the following steps:

0 The process is prepared by collecting the highest selected concepts into the list CONCEPT_TO_BE_PROCESSED.

1 The superconcepts of concepts in the list CONCEPT_TO_BE_PROCESSED are generated.

2 A cycle is started and runs as long as there are unprocessed concepts in CONCEPT_TO_BE_PROCESSED as follows:

  2.1 The hulls of those concepts in CONCEPT_TO_BE_PROCESSED that have not yet been processed by the procedure GENERATION_OF_HULLS are generated. In parallel with the labelling of ontology items, the generated concepts are added to the list CONCEPT_TO_BE_PROCESSED. The processed concepts are marked as 'processed by the procedure GENERATION_OF_HULLS.'

  2.2 The subconcepts of those concepts in the list CONCEPT_TO_BE_PROCESSED that have not yet been processed by the procedure GENERATION_OF_SUBCONCEPTS are generated. In parallel with the labelling of ontology items, the generated concepts are added to the list CONCEPT_TO_BE_PROCESSED. The processed concepts are marked as 'processed by the procedure GENERATION_OF_SUBCONCEPTS.'

  2.3 If there are no further unprocessed concepts in the list CONCEPT_TO_BE_PROCESSED then the cycle is finished.

3 The instances of concepts in the list CONCEPT_TO_BE_PROCESSED are generated (GENERATION_OF_INSTANCES).

4 The highest concepts in the newly labelled ontology are generated and marked as processed.

5 The superconcepts of the highest concepts in the list CONCEPT_TO_BE_PROCESSED are generated by the procedure GENERATION_OF_SUPERCONCEPTS.

Note that only the highest selected concepts are considered in the starting step, since the subconcepts of every selected concept are generated automatically.

The main danger of automatic generation is that some very high level concepts get into the collaboration ontology, and the further iteration steps include almost the whole ontology. Step 1 is placed before the iteration in order to avoid this

problem. Nonetheless, the effectiveness of the algorithm is highly dependent on the structure of the source ontology — only a suitably axiomatised and coherent ontology can be processed with good result.

5 A Walk-Through Example

In this section we illustrate the operation of OIT with a simple example, in which the user selects only one electronic component. The purpose of the example is to show the relationship between components and their functions, to illustrate the 'undesired side effects' the integration algorithm might have, and to indicate how the resulting segment can be improved upon by changing some of the input parameters.

Let us suppose that the user selects the ontology concept MICROCONTROLLER as relevant (this makes it likely that she considers the siblings of MICROCONTROLLER to be irrelevant, since otherwise she would have selected some of the siblings as well). Figure 4 on the next page shows a part of the segmentation's result, that is, the components that are included into the resulting segment.

As can be seen, the subconcepts and superconcepts of the selected concept are included; however, the inclusion of the concept CONNECTOR may seem at first sight puzzling. For an explanation let us consider a small fragment of the reference ontology (Figure 5 on page 602). In the first step of making the hull concepts CONTROL, DETECT, and CHANNEL are included. Note that the 'forAll' restriction restricts the concepts included from the range of relation TYPICALLY-HAS-FUNCTION. However, in the second phase, when the hull of concept CHANNEL is generated, the concept CONNECTOR is included because of the relation TYPICALLY-FUNCTION-OF.

Note that if the inverse of the relation TYPICALLY-HAS-FUNCTION was unnamed, CONNECTOR would still be included, since the generation of the hull considers the inverses of relations as well (see Figure 2 on page 596). If the user looks at the result and decides that CONNECTOR is irrelevant and should not be included, then she may unselect it, and initiate another execution of the segmentation algorithm. In this new session the inclusion of concept CONNECTOR and its subconcepts will be prevented.

6 Related Work

The problem of ontology segmentation and modularisation had received relatively little attention until the last few years, when comprehensive studies of ontology segmentation and modularisation were conducted within important semantic web projects such as WonderWeb [15] and Knowledge Web [13], and a number of different approaches to automatic and semi-automatic ontology segmentation have emerged in the literature.

The proposed solutions can be classified into three broad categories. There are graph-theoretic approaches (e.g. [16, 4]), which transform the input ontology into a directed graph and utilise general network-theoretic algorithms to find minimal

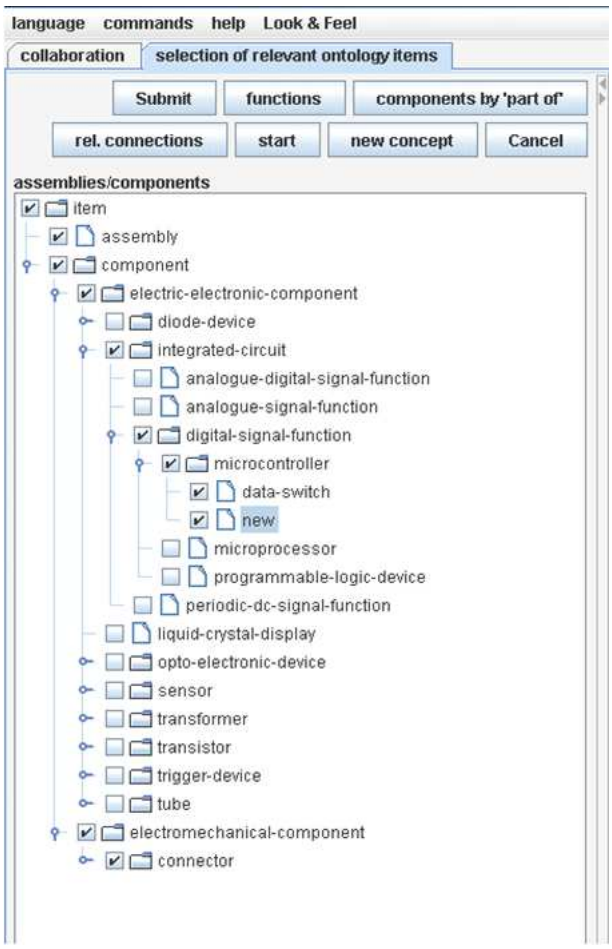

Figure 4: A segmentation result (concept NEW was added by the user)

subgraphs that contain the input items and meet certain abstract conditions of connectedness; model-theoretic proposals, which construe the criterion of relevance in terms of semantic consequence (e.g. [5, 7]), and, finally, heuristic solutions seeking to find the relevant subontology on the basis of certain special relationships between concepts (e.g. different types of relational connections, role in a quality-quale structure, mereological relationships etc.) that are treated individually, according to their semantics (e.g. [11, 3]). Our approach belongs to the third category.

A comparison of different ontology segmentation methods has to keep in view the purpose of the segmentation to be performed. In our case the generated ontology fragment has to serve as the knowledge base of a software system — accordingly, both the criteria of relevance and the requirements towards the generated ontol-

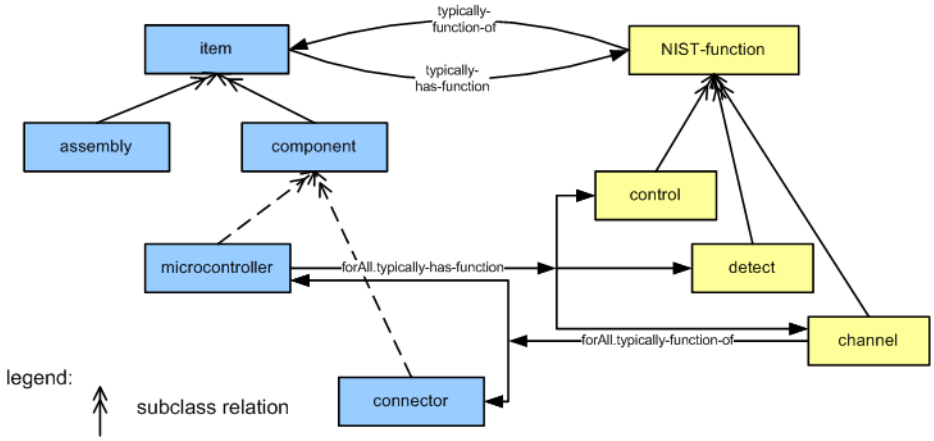

Figure 5: A fragment of the reference ontology

ogy differ from those arising in the case of other applications, e.g. in an ontology search system. Firstly, there are determinate — although often not explicit — requirements and criteria of correctness. It follows that abstract, e.g. graph-theoretic methods cannot be applied without extensive modifications and 'customization', because the segmentation process has to take into account the semantics of relations e.g. that of HAS-QUALITY or HAS-QUALE. Moreover, the input ontology cannot be treated simply as a directed graph since inverse relations have to be considered even if they are anonymous. Our case is also complicated by the fact that the task is not simple concept hull generation, because the input contains several concepts.

We compare our approach with the heuristic segmentation solution of [11], which is the most similar to our proposal. The method described by [11] generates the required ontology segment by computing the closure of the selected concepts under relational connections and the subconcept/superconcept relationship,[6] and tries to limit the size of the resulting segment both by filtering out certain relational connections on semantic grounds, and by limiting the distance of the included concepts from the initially selected ones.

In our case the fulfilment of the relevance criteria is ensured by the cooperation of automatic processes and the iterated manual intervention of the user. The important role played by the users' interaction with the system is due to the fact that in the ImportNET use case there is no fault tolerance: the resulting collaboration ontology has to contain each and every relevant ontology item, since it has to be transformed into an object oriented data model. As a consequence, the distance-based pruning method described in [11] has been unusable in ImportNET, in spite of the fact that it occurs in early descriptions of the platform (e.g. in [9]).

Another important difference lies in the special role of superconcepts. While

[6]Closure under the subconcept relation is partial, because only the initally selected concepts' subconcepts are included into the segment.

the elements of the hull and the subconcepts participate in the later iterations in the sense that further subconcepts and hulls have to be generated from them, the same does not apply to superconcepts. This feature of the algorithm solves the problem which motivated the introduction of distance limits in [11]. Our approach generates the hull of a concept on the basis of the notion of relational connection. In contrast, [11] treats restrictions as superconcepts, which are independent from the 'property filtering' function of the system i.e. from the domains and ranges of relations.

Finally, in our approach the user has an important role to play in the generation of concept hulls, therefore she can specify the relevance of individual relational connections depending on the concrete task at hand.

7 Conclusion and Perspectives

This paper has presented an ontology segmentation tool. The objective is to select a subontology of a domain ontology that is relevant to a given problem. The criterion of relevance is supplied by the end users by selecting relevant/irrelevant ontology elements (mainly concepts, but relations and relational connections can also be selected). The process of segmentation consists of a high level iteration: selection of relevant items by the users and automatic segmentation follow each other.

The end users' interaction with the system raises an important problem: the ontology has to be handled by naive users, who do not know anything about ontologies. Accordingly, a separate module of the system bridges the gap between ontology structure and the end users' business logic. The module is based on a general method for providing ontology editing tools for naive users, where the specification of the user interface is given in an XML file.

The ontology segmentation process consists of three activities: generating the hull of a concept, generating subconcepts, and generating superconcepts. The steps of generating the hulls and subconcepts of concepts are organised into an iteration. Superconcepts are included into the generated segment only to make it a coherent ontology; neither their hulls, nor their subclasses are generated.

The most important differences between our approach to the problem of ontology segmentation and other proposed solutions are due to the fact that in our use case the generated ontology segment has to be transformed into an object oriented data model, and, consequently, we have no fault tolerance. This is why the users have an exceptionally important role in the segmentation process.

The tool presented in the paper is in a prototype version, and has been used with a good result in the ImportNET project. However, further experiments are needed with different kinds of ontologies to develop it into a generally usable tool.

OIT consists of two well separated parts: a module making it possible for naive users to edit ontologies in a restricted way, and the segmentation module. These parts can be used in different scenarios and for different goals. At the end of the ImportNET project a new scenario has been outlined (see [10]), which connects functional design [14] with the generation of a collaboration ontology. This may help

the development of an up-to-date design technology in mechatronic engineering.

Acknowledgement. The authors are grateful to the ImportNET community for the inspiration and support they provided, and to two anonymous reviewers for constructive comments.

References

[1] Blaszczynski, J., Kosiedowski, M., Mazurek, C., and Wilk, S. Ontologies for Knowledge Modeling and Creating User Interface in the Framework of Telemedical Portal. In Stormer, H., Meier, A., and Schumacher, M., editors, *Proceedings of the ECEH'06, Fribourg, Switzerland, October 12-13*, pages 275–286, Bonn, 2006. Gesellschaft für Informatik.

[2] Bojars, U. Captsolo Weblog: Ontologies => User Interface, May 17, 2004. WWW document. `http://captsolo.net/info/blog_a.php/2004/05/17/p520` (accessed May 18, 2010).

[3] d'Aquin, D., Sabou, D., and Motta, P. Modularization: A Key for The Dynamic Selection of Relevant Knowledge Components. In *WoMO'06 Workshop on Modular Ontologies*, 2006. `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-232/paper2.pdf` (accessed May 18, 2010).

[4] Doran, P., Tamma, V., and Iannone, L. Ontology Module Extraction for Ontology Reuse: An Ontology Engineering Perspective. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 61–70, New York, 2007. ACM Press.

[5] Grau, B.C., Horrocks, I., Kazakov, Y., and Sattler, U. Just the Right Amount: Extracting Modules from Ontologies. In *Proceedings of the 16th International Conference on World Wide Web*, pages 717–726, New York, 2007. ACM Press.

[6] Guarino, N. Formal Ontology and Information Systems. In Guarino, N., editor, *Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98), June 6-8, Trento, Italy*, pages 3–15, Amsterdam, 1998. IOS Press.

[7] Konev, B., Lutz, C., Walther, D., and Wolter, F. Semantic Modularity and Module Extraction in Description Logics. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., and Avouris, N., editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 55–59, Amsterdam, 2008. IOS Press.

[8] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. WonderWeb Deliverable D18: Ontology Library (final), 2003. `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf` (accessed May 18, 2010).

[9] Ovtcharova, J., Mahl, A., and Krikler, R. Approach for a Rule Based System for Capturing and Usage of Knowledge in the Manufacturing Industry. In Wang, K., Kovacs, G., Wozny, M., and Fang, M., editors, *Knowledge Enterprise: Intelligent Strategies In Product Design, Manufacturing, and Management*, pages 134–143, Boston, 2006. Springer.

[10] Ovtcharova, J., Marinov, M., Szőts, M., Simonyi, A., and Schubert, P. Ontology-Based, Function Oriented Support for Cross-Domain Engineering. In preparation.

[11] Seidenberg, J. and Rector, A. Web Ontology Segmentation: Analysis, Classification and Use. In *Proceedings of the 15th International Conference on World Wide Web*, pages 13–22, New York, 2006. ACM Press.

[12] Semy, S. K., Pulvermacher, M. K., and Obrst, L. J. Toward the Use of an Upper Ontology for U.S. Government and U.S. Military Domains: An Evaluation. Technical report, MITRE, 2004. `http://www.mitre.org/work/tech_papers/tech_papers_05/04_1175/04_1175.pdf` (accessed May 18, 2010).

[13] Spaccapietra, S., editor. Knowledge Web Deliverable 2.1.3.1: Report on Modularization of Ontologies, 2003. `http://wasp.cs.vu.nl/knowledgeweb/Deliverables/D2.1.3.1/D2.1.3.1-Modularization.pdf` (accessed May 18, 2010).

[14] Stone, R. B. and Wood, K. L. Development of a Functional Basis for Design. *Journal of Mechanical Design*, 122:359–370, 2000.

[15] Stuckenschmidt, H. and Klein, M. Modularization of Ontologies: WonderWeb Deliverable D21, 2003. `http://wonderweb.semanticweb.org/deliverables/documents/D21.pdf` (accessed May 18, 2010).

[16] Stuckenschmidt, H. and Klein, M. Structure-based Partitioning of Large Concept Hierarchies. In *The Semantic Web — ISWC 2004*, pages 289–303, Berlin, 2004. Springer.

[17] Szőts, M. and Schneider, F. Generating Ontology User Interface for Naïve Users. In *International Conference on Collaborative Mechatronic Engineering (ICCME'09), Salzburg, Austria*, 2009. `http://importnet.salzburgresearch.at/images/ImportNET_Bilder/Paper/session_ii_szots_schneider.pdf` (accessed May 18, 2010).

[18] The Protégé Ontology Editor and Knowledge Acquisition System. WWW document. `http://protege.stanford.edu` (accessed May 18, 2010).