Structure-Based Partitioning of Large Concept Hierarchies

Heiner Stuckenschmidt and Michel Klein

Vrije Universiteit Amsterdam
de Boelelaan 1081a, 1081HV Amsterdam
{heiner, mcaklein}@cs.vu.nl

Abstract. The increasing awareness of the benefits of ontologies for information processing has lead to the creation of a number of large ontologies about real-world domains. The size of these ontologies and their monolithic character cause serious problems in handling them. In other areas, e.g. software engineering, these problems are tackled by partitioning monolithic entities into sets of meaningful and mostly self-contained modules. In this paper, we suggest a similar approach for ontologies. We propose a method for automatically partitioning large ontologies into smaller modules based on the structure of the class hierarchy. We show that the structure-based method performs surprisingly well on real-world ontologies. We support this claim by experiments carried out on real-world ontologies including SUMO and the NCI cancer ontology. The results of these experiments are available online at http://swserver.cs.vu.nl/partitioning/.

1 Motivation

The increasing awareness of the benefits of ontologies for information processing in open and weakly structured environments has lead to the creation of a number of such ontologies for real-world domains. In complex domains such as medicine these ontologies can contain thousands of concepts. Examples of such large ontologies are the NCI cancer ontology [5] with about 27.500 and the Gene ontology [8] with about 22.000 concepts. Other examples can be found in the area of e-commerce where product classification such as the UNSPSC or the NAICS contain thousands of product categories. While being useful for many applications, the size and the monolithic nature of these ontologies cause new problems that affect different steps of the ontology life cycle.

Maintenance: Ontologies that contain thousands of concepts cannot be created and maintained by a single person. The broad coverage of such large ontologies normally requires a team of experts. In many cases these experts will be located in different organizations and will work on the same ontology in parallel. An example for such a situation is the gene ontology that is maintained by a consortium of experts.

Publication: Large ontologies are mostly created to provide a standard model of a domain to be used by developers of individual solutions within that domain. While existing large ontologies often try cover a complete domain, the providers of individual solutions

S.A. McIlraith et al. (Eds.): ISWC 2004, LNCS 3298, pp. 289–303, 2004.

are often only interested in a specific part of the overall domain. The UNSPSC classification, for example, contains categories for all kinds of products and services while the developers of an online computer shop will only be interested in those categories related to computer hardware and software.

Validation: The nature of ontologies as reference models for a domain requires a high degree of quality of the respective model. Representing a consensus model, it is also important to have proposed models validated by different experts. In the case of large ontologies it is often difficult—if not impossible—to understand the model as a whole due to cognitive limits. What is missing is an abstracted view on the overall model and its structure as well as the possibility to focus the inspection on a specific aspect.

Processing: On a technical level, very large ontologies cause serious scalability problems. The complexity of reasoning about ontologies is well known to be critical even for smaller ontologies. In the presence of ontologies like the NCI cancer ontology, not only reasoning engines but also modelling and visualization tools reach their limits. Currently, there is no OWL-based modelling tool that can provide convenient modelling support for ontologies of the size of the NCI ontology.

All these problems are a result of the fact that the ontology as a whole is too large to handle. Most problems would disappear if the overall model consists of a set of coherent modules about a certain subtopic that can be used independently of the other modules, while still containing information about its relation to these other modules.

- In distributed development, experts could be responsible for an single module and maintain it independently of other modules thus reducing revision problems.
- Users of an ontology could use a subset of the overall ontology by selecting a set of relevant modules. While only having to deal with this relevant part, the relations to other parts of the model are still available through the global structure.
- Validation of a large ontologies could be done based on single modules that are easier to understand. Being related to a certain subtopic, it will be easier to judge the completeness and consistency of the model. Validated modules could be published early while other parts of the ontology are still under development.
- The existence of modules will enable the use of software tools that are not able to handle the complete ontology. In the case of modelling and visualization tools, the different modules could be loaded one by one and processed individually. For reasoning tasks we could make use of parallel architectures where reasoners work on single modules and exchange partial results.

Recently, some proposals concerning the representation of modules and their connections have been made [9,3,7]. These papers propose languages and discuss issues like the organization of modules and dependencies between them. A problem that has not been addressed yet concerns the creation of modules from existing ontologies. This problem which we call *partitioning* or *modularization* is discussed in the remainder of this paper.

2 Structure-Based Partitioning

The key question connected to modularization is about suitable criteria for determining the assignment of concepts to modules. This requires a good intuition about the nature of a module as well as of the purpose of the modularization. As we have seen above, there are many different use cases that come with different requirements for criteria for determining modules. Further distinctions have to be made with respect to the nature of the ontologies addressed by the method. On the semantic web, we find all kinds of ontologies starting from simple class hierarchies to complex description logic models encoded in OWL. As an attempt to develop a method that works equally well for all kinds of ontologies is likely to fail, we focus on a particular type of ontology and a certain use of the resulting partitioning.

> **Contribution:** We propose a method for partitioning large light-weight ontologies that mainly consists of a class hierarchy into disjoint and covering sets of concepts for the purpose of easier browsing and exploring the hierarchy.

Note that the method we propose can be applied to any kind of ontology that contains some kind of hierarchy. In cases where concepts are further described by their features or logical axioms, however, our method is likely to produce sub-optimal results as the information in the definitions is not taken into account. Our method is sensitive to the intended use of the resulting model, as other purposes can require partially overlapping modules or the partitioning into sets of axioms rather than concepts.

2.1 Intuition and Assumptions

Intuitively, we can say that a module should contain information about a coherent subtopic that can stand for itself. This requires that the concepts within a module are semantically connected to each other and do not have strong dependencies with information outside the module. These considerations imply the need for a notion of *dependency* between concepts that needs to be taken into account. There are many different ways in which concepts can be related explicitly or implicitly. At this point we abstract from specific kinds of dependencies and choose a general notion of dependency between concepts. The resulting model is the one of a weighted graph $O = \langle C, D, w \rangle$ where nodes C represent concepts and links D between concepts represent different kinds of dependencies that can be weighted according to the strength of the dependency. These dependencies can be reflected in the definitions of the ontology or can be implied by the intuitive understanding of concepts and background knowledge about the respective domain. Looking for an automatic partitioning method, we are only interested in such kinds of dependencies that can be derived from the ontology itself. This leads us to the central assumption underlying our approach:

> **Assumption:** Dependencies between concepts can be derived from the structure of the ontology.

Depending on the representation language, different structures can be used as indicators of dependencies. These structures can be subclass relations between classes, other

relations linked to classes by the range and domain restrictions or the appearance of a class name in the definition of another class. It can be argued that a method purely based on the structure of the ontology is not able to capture important dependencies that could be found by analyzing the names of classes and the logical definitions of concepts. Our motivation for ignoring these additional aspects is of a practical nature: we want our method to scale up to large ontologies that contain hundreds of thousands of concepts. A semantical investigation of the dependencies of such ontologies would suffer from the same scalability problems we hope to solve with out partitioning method. Therefore, the question addressed in this paper is not so much about an optimal method for determining modules but about how far we can get with a rather simple and scalable method based on the structure of the ontology.

2.2 Partitioning Steps

Our method consists of two tasks that are executed in five independent steps. The first task is the creation of a weighted graph from an ontology definition. This is done in two steps: extraction of the dependency structure and determination of the weight of the dependency. The second task concerns the identification of modules from the dependency graph. The first step in this task is the detection of strongly related sets of concepts. In the following we discuss the techniques currently used in these steps.

Step 1: Create Dependency Graph: In the first step a dependency graph is extracted from an ontology source file. We implemented a PROLOG-based tool that reads OWL and RDF schema files using the SWI semantic web library [10] and outputs a graph format used by the networks analysis tool Pajek [2] that we use for detecting related sets of nodes. The tool can be configured to use different kinds of dependencies. Currently it can extract dependencies corresponding to the subclass hierarchy and dependencies created by the domain and range restrictions in property definitions.

Step 2: Determine strength of Dependencies: In the second step the strength of the dependencies between the concepts has to be determined. Following the basic assumption of our approach, we use the structure of the dependency graph to determine the weights of dependencies. In particular we use results from social network theory by computing the proportional strength network for the dependency graph. The strength of the dependency of a connection between a node c_i and c_j is determined to be the proportional strengths of the connection. The proportional strength describes the importance of a link from one node to the other based on the number of connections a node has. In general it is computed by dividing the sum of the weights of all connections between c_i and c_j by the sum of the weights of all connections c_i has to other nodes (compare [4], page 54ff):

$$w(c_i, c_j) = \frac{a_{ij} + a_{ji}}{\sum_k a_{ik} + a_{ki}}$$

Here a_{ij} is the weight preassigned to the link between c_i and c_j - in the experiments reported below this will always be one. As a consequence, the proportional strength

used in the experiments is one divided by the number of nodes c_i is connected to. The intuition behind it is that individual social contacts become more important if there are only few of them. In our setting, this measure is useful because we want to prevent that classes that are only related to a low number of other classes get separated from them. This would be against the intuition that classes in a module should be related. We use node **d** in Figure 1 to illustrate the calculation

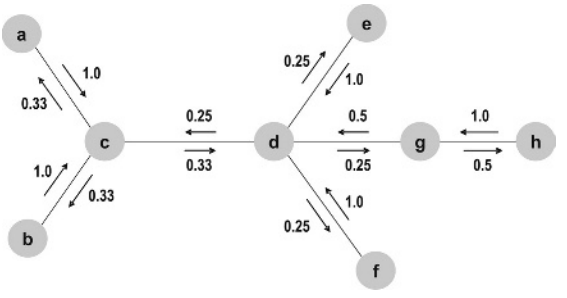

Fig. 1. An Example Graph with proportional strength dependencies

of weights using the proportional strength. The node has four direct neighbors, this means that the proportional strength of the relation to these neighbors is 0.25 (one divided by four). Different levels of dependency between **d** and its neighbors now arise from the relative dependencies of the neighbors with **d** (the proportional strength is non-symmetric). We see that **e** and **f** hoaving no other neighbors completely depend on **d**. The corresponding value of the dependency is 1. Further, the strength of the dependency between **g** and **d** is 0.5, because **g** has two neighbors and the dependency between **b** and **d** is 0.33 as **b** has 3 neighbors.

Step 3: Determine Modules: The proportional strength network provides us with a foundation for detecting sets of strongly related concepts. For this purpose, we make use of an algorithm that computes all maximal line islands of a given size in a graph [1].

Definition 1 (Line Island). *A set of vertices $I \subseteq C$ is a line island in a dependency graph $G = (C, D, w)$ if and only if*

- *I induces a connected subgraph of G*
- *There is a weighted graph $T = (V_T, E_T, w_T)$ such that:*
  - *T is embedded in G*
  - *T is an maximal spanning tree with respect to I*
  - *the following equation holds:*

$$\max_{\{(v,v')\in D | (v\in I \wedge v'\notin I)\vee(v'\in I\wedge v\notin I)\}} w(v,v') < \min_{(u,u')\in E_T} w(u,u')$$

Note that for the determination of the maximal spanning tree the direction of edges is not considered.

This criterion exactly coincides with our intuition about the nature of modules given in the introduction, because it determines sets of concepts that are stronger internally connected than to any other concept not in the set. The algorithm requires an upper and a lower bound on the size of the detected set as input and assigns an island number to each node in the dependency graph. We denote the island number assigned to a concept c as $\alpha(c)$. The assignment $\alpha(c) = 0$ means that c could not be assigned to an island.

We use different sets of nodes in the graph in Figure 1 to illustrate the concept of a line island. Let us first consider the set $\{\mathbf{a}, ..., \mathbf{f}\}$. It forms a connected subgraph. The maximal spanning tree of this set consists of the edges $\mathbf{a} \xrightarrow{1.0} \mathbf{c}$, $\mathbf{b} \xrightarrow{1.0} \mathbf{c}$, $\mathbf{c} \xrightarrow{0.33} \mathbf{d}$, $\mathbf{e} \xrightarrow{1.0} \mathbf{d}$, and $\mathbf{f} \xrightarrow{1.0} \mathbf{d}$. We can see however, that this node set is not an island, because the minimal weight of an edge in the spanning tree is 0.33 and there is an incoming edge with strength 0.5 ($\mathbf{g} \xrightarrow{0.5} \mathbf{d}$). If we look at the remaining set of nodes $\{\mathbf{g}, \mathbf{h}\}$, we see that it fulfills the conditions of an island: it forms a connected subgraph, the maximal spanning tree consists of the edge $\mathbf{h} \xrightarrow{1.0} \mathbf{g}$ and the maximal value of in- or outgoing links is 0.5 ($\mathbf{g} \xrightarrow{0.5} \mathbf{d}$). This set, however, is not what we are looking for because it is not maximal: it is included in the set $\{\mathbf{d}, ..., \mathbf{h}\}$. This set is a line island with the maximal spanning tree consisting of the edges $\mathbf{e} \xrightarrow{1.0} \mathbf{d}$, $\mathbf{f} \xrightarrow{1.0} \mathbf{d}$, $\mathbf{g} \xrightarrow{0.5} \mathbf{d}$ and $\mathbf{h} \xrightarrow{1.0} \mathbf{g}$ where the minimal weight (0.5) is higher than the maximal weight of any external link which is $\mathbf{c} \xrightarrow{0.33} \mathbf{d}$. Another reason for preferring this island is that the remaining node set $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ also forms a line island with maximal spanning tree $\mathbf{a} \xrightarrow{1.0} \mathbf{c}$, $\mathbf{b} \xrightarrow{1.0} \mathbf{c}$ and the weaker external link $\mathbf{c} \xrightarrow{0.33} \mathbf{d}$.

Using these steps as a starting point we carried out a number of experiments in order to test our hypothesis that the structure of the ontology can actually be used to create a sensible partitioning of large ontologies. The most relevant of these experiments are reported in the following.

3 Experiments I: Iterative Partitioning

The different steps described above lead us to a partitioning of the input ontology into modules that satisfy the formal conditions mentioned in the previous section. We applied the steps to a number of ontologies available in OWL or RDF schema, including the SUMO [6] the NCI cancer ontology [5] as well as DICE, a large medical ontology used in one of our projects. Due to lack of space, we do not discuss the results of these initial experiments in details, but concentrate on the problems we identified. The complete results of all experiments can be found at the website that accompanies this article.[1]

One of the main problems with the approach as described above is the fact that we have to determine the size of modules that we want to be generated. The reason is that the optimal size of modules heavily depends on the size and the nature of the ontology.

[1] See `http://swserver.cs.vu.nl/partitioning/`.

In some preliminary experiments we found a module size of one to ten per percent of the size of the complete ontology works quite well for some example ontologies that had between 1000 and 2000 concepts. This heuristic, however, did not work for larger or more fragmented ontologies. In some cases a bad choice of the upper and lower bound for the size of modules also led to an extremely high number of unassigned nodes, i.e. nodes that were not part of any 'island'. In order to avoid these problems, we defined an iterative partitioning strategy that avoids some of the problems mentioned above. In the following, we present the iterative partitioning strategy and summarize the results of applying this strategy to real world ontologies.

3.1 Iterative Partitioning

The idea of the iterative assignment algorithm is to not prescribe the size of modules to be generated but to let them be determined solely by the island criterion given in the last section. A way of doing this is to set the lower bound to 1 and the upper bound to $s-1$ where s is the size of the complete ontology. Reducing the limit by one forces the algorithm to split up the ontology in some way as the complete model exceeds the upper size limit for an island. Choosing a limit that is just one below the size of the complete ontology does not further restrict the selection of islands. This way we get the most natural grouping of concepts into strongly dependent sets. Even in this case where we do not restrict the size of island it can still happen, that nodes cannot be assigned to islands. Therefore we have to perform the expansion step afterwards in order to assign these nodes to a module.

Step 4: Assign Isolated Concepts: Our experiments showed that leftover nodes can occur at different places in the graph and are not necessarily related. Therefore we chose to assign them to existing modules. The assignment is based on the strength of the relations to nodes already assigned to a module. In particular leftover nodes are assigned to the island of the neighboring node they have the strongest relation to. In cases where all neighboring nodes are unassigned as well, these nodes are assigned first.

As a result of this strategy the islands found by the algorithm can significantly differ in size. In particular, we often get large islands that cover most of the ontology. In order to get modules of a reasonable size, we iteratively apply the algorithm to islands that are too large to be useful modules. Often this applies only for a single large island, but there are also cases especially in the case of very large ontologies where the algorithm has to be applied recursively on different parts of the ontology. Nodes in islands that are small enough get a unique number assigned and form a module of the ontology.

Algorithm 1 shows the corresponding labelling algorithm that takes a graph and labels the nodes of the graph with numbers that correspond to a partitioning assignment. The algorithm also needs the upper limit of the size of a module as input in order to determine when to stop the iterating. The counter is used in the recursive calls to make sure that modules have unique numbers. When starting the iteration, the counter has to be set to zero.

Algorithm 1 Partition

```
Require: limit: integer
Require: ontology: graph
Require: counter:integer
  CURRENT := ontology
  if |current| > limit then
    MIN := 1
    MAX := |current| − 1
    CANDIDATES := islands(min,max,current)
    for all module ∈ candidates do
      Expand(module,current)
      Partition(limit,module,counter)
    end for
  else
    COUNTER := counter + 1
    for all c ∈ current do
      α(c) := counter
    end for
    return counter
  end if
```

3.2 Evaluation of Partitions

Looking at the result of the example application we get a first idea about the strengths and weaknesses of the algorithm. We can see that the algorithm generates some modules that meet our intuition about the nature of a module quite well. In some cases subtrees that could be considered to form one module are further split even if the complete subtree does not exceed the upper size limit. This can be explained by an unbalanced modelling of the ontology as subtrees tend to be split up at concepts with a high number of direct subclasses compared to its sibling classes. This phenomenon often reflect a special importance of the respective concept in the ontology that also justifies the decision to create a separate model for this concept. The iterative strategy frees us from determining a lower bound for the size of modules. As a result, however, the algorithm sometimes create rather small modules. This normally happens when the root concept of a small subtree is linked to a concept that has many direct subclasses. For the result of the partitioning method these subsets are often pathological because a coherent topic is split up into a number of small modules that do not really constitute a sensible model on their own.

When inspecting the dependencies in the relevant parts of the hierarchy, we discovered that most of the problematic modules have very strong internal dependencies. In order to distinguish such cases, we need a measure for the strength of the internal dependency. The measure that we use is called the 'height' of an island. It uses the minimal spanning tree T used to identify the module: the overall strength of the internal dependency equals the strength of the weakest link in the spanning tree.

$$height(I) = \min_{(u,u') \in E_T} w(u, u')$$

We can again illustrate the the concept of height using the example from figure 1. WE identifies two islands, namely $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ and $\{\mathbf{d}, ..., \mathbf{h}\}$. As the maximal spanning tree of the first island consists of the two edges $\mathbf{a} \xrightarrow{1.0} \mathbf{c}$, $\mathbf{b} \xrightarrow{1.0} \mathbf{c}$, the height of this island is 1.0. In the maximal spanning tree of the second island the edge $\mathbf{g} \xrightarrow{0.5} \mathbf{d}$ is the weakest link that therefore sets the height of the island to 0.5.

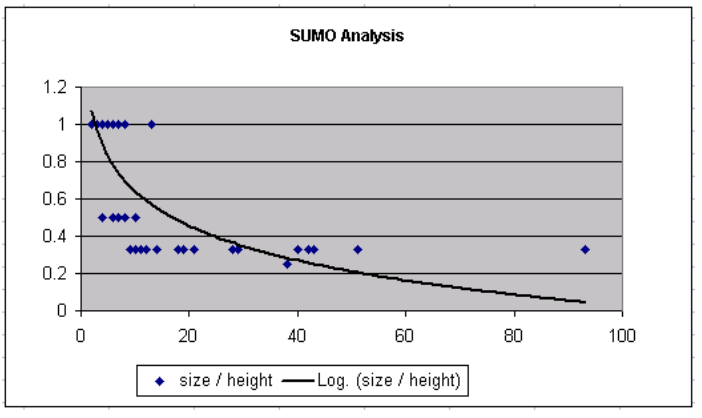

Fig. 2. Sizes and heights of partitions in the SUMO ontology

We found many cases where generated modules that do not make sense had an internal dependency of strength one. In a post-processing step this allows us to automatically detect critical modules. While for the case of an internal strength of one we almost never found the corresponding module useful in the context of the original ontology, it is not clear where to draw the line between a level of internal dependency that still defines sensible modules and a level that overrules important dependencies to concepts outside the module. In our experiments we made the experience that a threshold of 0.5 leads to good results in most cases.[2]

Figures 2 and 3 show the results of comparing the size and the height of computed islands. The plots clearly show a correlation between these properties. We also see that—except for one case—islands with a height of one are quite small.[3] The results of these experiments provided us with sufficient evidence that the height of an island is a useful criterion for judging the quality of a module. In successive experiments reported below we used this result to improve the partitions created by the iterative strategy. The results of these experiments are reported below.

[2] Note that due to the calculation of the dependency value, the internal strength is always of the form $\frac{1}{n}$.

[3] The exception is a part of the NCI ontology that lists all countries of the world and therefore contains 1 class with more than 200 subclasses

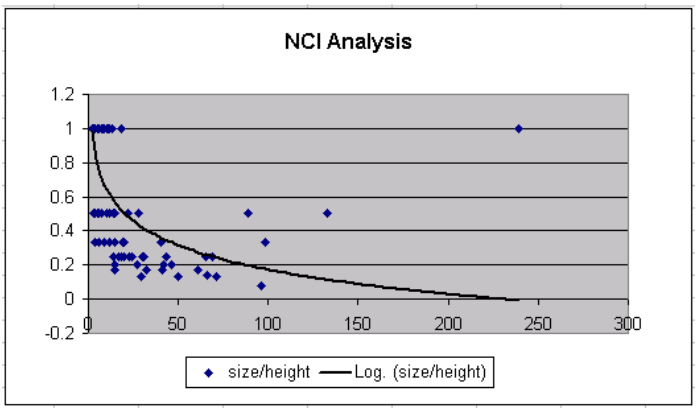

Fig. 3. Sizes and heights of partitions in the NCI ontology

4 Experiments II: Manual Post-processing

As described above, on of the findings in the first experiments was the strong correlation between size of modules and the degree of internal dependency. Further, we found out that small modules were unnatural in most cases. In a second experiment, we wanted to find out whether this result can be used to 'repair' the result of the straightforward partitioning.

Step 5: Merging: All modules with a height of 1.0 or 0.5 are merged into adjacent modules with a lower height. In many cases, there is only one adjacent module to merge with. In cases where more than one adjacent module exist, we decided for the most natural choice. In principle, the strength of the dependencies between the modules can be used to also determine a candidate for merging automatically.

The merging process was done manually using the functionality of the Pajek tool and some supporting scripts. In the following, we present and discuss the results of this experiment for the SUMO and the NCI administration ontology.

4.1 The SUMO Ontology

The partitioning of the SUMO ontology resulted into 38 partitions of which 17 had a height of lower than 0.5 (see complete results at `http://swserver.cs.vu.nl/partitioning/`). These 17 partitions and their sizes after the merging process are listed in table 1. We see that the topics of the modules—which we derived from the top concept of the respective module—represent sensible subtopics of the overall ontology, which are not too close to each other. The average size of the modules is 37 concepts. Comparing this list to the modules created in the first experiment, we see that the use of the height criterion for merging enabled us to get rid of most of the problems that had occurred. An example is a high number of small modules for different kinds of measures that were created by the partitioning algorithm and that are now contained in

Table 1. Modules Generated for the SUMO Ontology

	Topic	Size
1	Text	14
2	Biological Attribute	12
3	Substance	38
4	Intentional Process	33
5	Linguistic Communication	36
6	Declaration	21
7	Making	111
8	Artifact	20
9	Real Number	14
10	Self-Connected Object	24
11	Language	23
12	Proposition	11
13	Relation	51
14	Constant Quantity	77
15	Agent	40
16	Entity (Top-Level)	51
17	Corpuscular Object	54

the module 'Constant Quantity'. The only problematic modules are the ones concerned with actions. The module 'making' is quite large and contains different types of actions that a human knowledge engineer would not necessarily put in the same module. Looking at the original graph we see that this part of the hierarchy is quite tangled which makes it difficult to find split points.

More striking than the pure list of concepts, however, is the module graph created in the second experiment. The graph shown in Figure 4 provides a very useful overview over the different topics covered by SUMO. We think that this representation is more useful for getting an overview of the content of SUMO than any visualization based on the individual concepts.

4.2 The NCI Cancer Ontology

In a second experiment, we took the partitioning created for the administrative part of the NCI Cancer ontology and removed partitions with a height of 1 or 0.5 by merging them with partitions of a lower height. The initial partitioning contained 108 modules many of which were quite small (compare Figure 3. Often, clearly related parts of the hierarchy had been cut up into a number of small modules. A good examples are cell lines. The initial partitioning distinguished between six different cell lines in addition to the general module about cell lines. Each of these cell line modules contained less than ten concepts. A similar situation could be observed in connection to medical occupations like different kinds of surgeons and nurses. The manual merging process based on the height of the partitions created a single cell line partition. For the case of surgeons the result is less optimal as different kinds of occupations had to be merged into a single module when using the height criterion.

Only 38 of the modules had a height of lower than 0.5. Table 2 lists these 38 modules and their respective sizes after the merging process. The average size of a module is 57 which higher than for SUMO but still a reasonable value. We have to notice, however that the variance is much higher as we find module sizes between 4 and 268. Most of

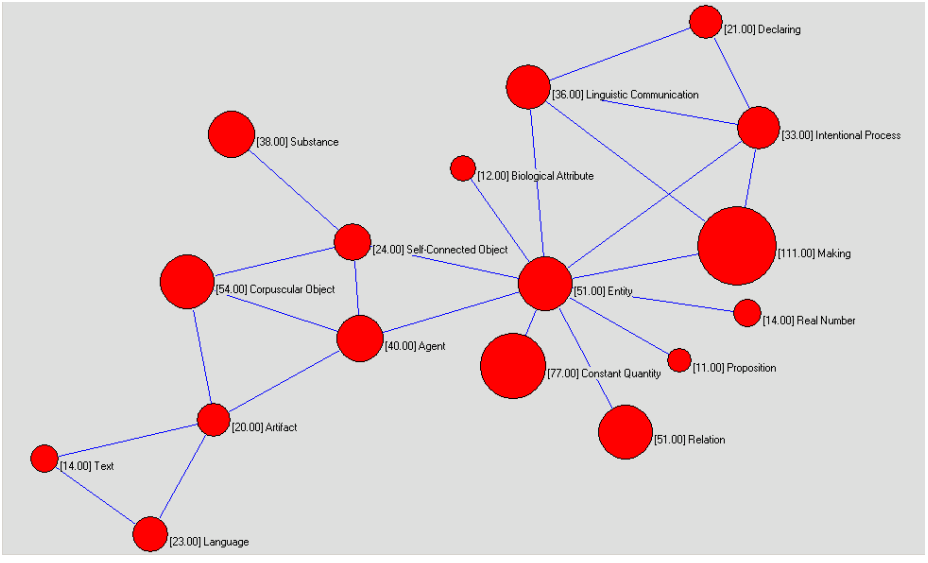

Fig. 4. Module Graph of the SUMO Ontology after Merging

the extreme cases can be explained by the special structure of the ontology. The concept conceptual entity for example has about 100 direct subconcepts that create a rather large module. The same holds for the concept country. More problematic are the modules 'clinical sciences' and occupation or discipline' that are rather large and heterogenous. In future work, we will have to analyze these modules in detail and determine a strategy for avoiding these problems by adjusting the dependency measure or the merging process.

5 Discussion

In this paper we describe a method for structure-based ontology partitioning that is practically applicable to very large ontologies. We show that a modularization based on structural properties of the ontology only already results in modules that intuitively make sense. Because modularizing an ontology essentially is a modelling activity, there is no "golden standard" to compare our results with. To come up with a more validated assessment of the quality of our modules, we need to do an experiment in which we compare the result of human modularization with our result[4].

An advantage of the method described in this paper is that there are no arbitrary choices – apart from the maximum size of a module which in many cases can be derived from the limitations of the technology used – that have to be made in advance (e.g. the number of modules required). However, there are several choices we made

[4] Creating a human modularization might be difficult in practice, as one of the major reasons for partitioning is that a human is not able to overlook the ontology as a whole.

Table 2. Modules Created for the NCI Administration Ontology

	Topic	Size
1	Biology	48
2	Pharmacology	20
3	Epidemiology	30
4	Clinical sciences	189
5	Medicine	85
6	Public Health	24
7	Occupation or Discipline	193
8	Social Sciences	25
9	Medical Economics	14
10	Technology	50
11	Information Science	27
12	NCI Administrative Concept	4
13	Training and Education	15
14	Board Certification	12
15	Information and Media	16
16	Database	15
17	Media / Document Type	111
18	Business Rule	61
19	Patient or Public Education	20
20	Nursing	46
21	Funding	130
22	Funding Categories	30
23	Research Career Programs	24
24	Costs	9
25	Professional Organization	56
26	Component of the NCI	75
27	NCI Boards and Groups	32
28	Population Group	71
29	Social Concept	66
30	Conceptual Entity	170
31	Sites of Care Delivery	268
32	Cancer Science	4
33	Model System	42
34	Miscellaneous Terms	23
35	Cancer Biology	17
36	Carciogenesis Mechanism	17
37	Specimen	26
38	Cell Line	101

when designing the method that need to be discussed in the light of the experiments. The first one is the choice of using the ontology structure as a basis for determining modules. We think that the experiments reported here show that we can achieve good results even with a minimal approach that uses the concept hierarchy only. It remains to be investigated whether the use of more structural information than the hierarchy and produces better results. We plan to investigate the performance of our method on graphs that include dependencies resulting from user defined relations and from the use of concepts in definitions and axioms.

Another direction for future research is the dependency measure. Currently, we use the proportional strengths, a measure adopted from social network theory that is only based on the direct connections of a node giving each connection an equal importance. It is possible that the development of dependency measures specifically for ontologies could improve our results. There are two directions in which such measures can be developed.

- **Context-aware measures:** dependency measures that do not only use the direct relations of a node but also look at relations further away and the "importance" of nodes. Measures of this type could vary in the depth in which they take other relations into account, the weight that is given to them, and the way in which the importance of nodes is calculated.
- **Semantics-based measures:** measures that use the semantics of relations for determining the dependency. Possible measures of this type are ones that give "isa"

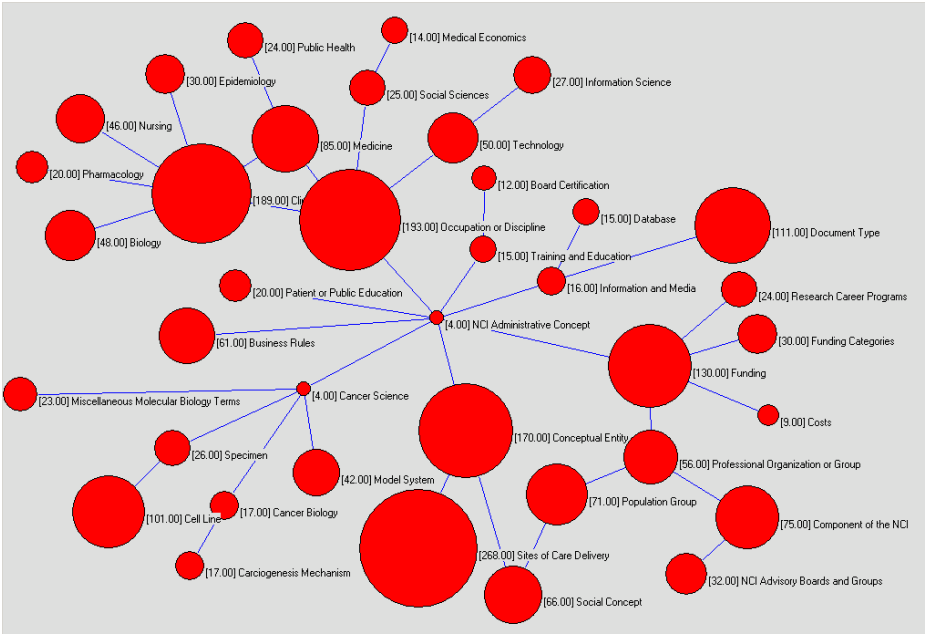

Fig. 5. Module Graph of the NCI Administration Ontology after Merging

relations a higher weight than other relations and measures that give a higher initial value to subrelations than their superrelations.

Experiments with other dependency measures should tell us whether such measures results in better modules than the ones that result from the basic measure that we used.

Another choice involves the level of abstraction at which we create the partitioning, i.e. the number of modules produced and their respective size. There are two parameters that control the abstraction level:

1. the termination point of the partitioning process, and
2. the criteria for merging modules.

In our current approach, we use the size of the module as termination criterium. This criterium can be seen as subjective; however, often the simple fact that the size of an ontology exceeds a certain threshold is the major motive to start modularizing an ontology. Using the size of the modules as termination criterium therefore is defendable. For the merging process, we currently use the coherence as criterium. For both parameters, we could experiment with the other measures than the one currently used. Further research is necessary to tell what measures perform well in which situations and what are useful threshold values for specific ontologies.

Besides experimenting with the different factors discussed above, we plan to work on automating the partitioning process as a whole. Ideally, this would result in tool that partitions an ontology and allows to adapt the abstraction level on the fly. For this to happen, we first need to do more work on the merging process and create a method that precisely describes how to perform the merging.

References

1. V. Batagelj. Analysis of large networks - islands. Presented at Dagstuhl seminar 03361: Algorithmic Aspects of Large and Complex Networks, August/September 2003.
2. V. Batagelj and A. Mrvar. Pajek - analysis and visualization of large networks. In M. Jünger and P. Mutzel, editors, *Graph Drawing Software*, pages 77–103. Springer, 2003.
3. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-owl: Contextualizing ontologies. In *Proceedings of the 2nd International Semantic Web Conference ISWC'03*, Lecture Notes in Computer Science, pages 164–179, Sanibal Island, Florida, 2003. Springer Verlag.
4. R.S. Burt. *Structural Holes. The Social Structure of Competition.* Harvard University Press, 1992.
5. Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The national cancer institute's thesaurus and ontology. *Journal of Web Semantics*, 1(1), 2003.
6. I. Niles and A. Pease. Toward a standard upper ontology. In Chris Welty and Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.
7. H. Stuckenschmidt and M. Klein. Integrity and change in modular ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'03*, pages 900–905, Acapulco, Mexico, 2003. Morgan Kaufmann.
8. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
9. R. Volz, A. Maedche, and D. Oberle. Towards a modularized semantic web. In *Proceedings of the ECAI'02 Workshop on Ontologies and Semantic Interoperability*, 2002.
10. J. Wielemaker, G. Schreiber, and B. Wielinga. Prolog-based infrastructure for RDF: performance and scalability. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The Semantic Web - Proceedings ISWC'03, Sanibel Island, Florida*, pages 644–658, Berlin, Germany, october 2003. Springer Verlag. LNCS 2870.