Ontology Refinement – Towards Structure-Based Partitioning of Large Ontologies

WonderWeb:
Ontology Infrastructure for the Semantic Web

Heiner Stuckenschmidt, Michel Klein
Vrije Universiteit Amsterdam
email: heiner@ cs.vu.nl, michel.klein@cs.vu.nl

Identifier	Del 22
Class	Deliverable
Version	1.0
Date	27.6.2004
Status	Draft
Distribution	Public
Lead Partner	VUA

WonderWeb Project

This document forms part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2001-33052.

For further information about WonderWeb, please contact the project co-ordinator:

Ian Horrocks
The Victoria University of Manchester
Department of Computer Science
Kilburn Building
Oxford Road
Manchester M13 9PL
Tel: +44 161 275 6154
Fax: +44 161 275 6236
Email: wonderweb-info@lists.man.ac.uk

Contents

1 Motivation

1.1 Why: Use Cases for Modularization

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 [7] 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 of these ontologies causes 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 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 require 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 of 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 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 a large ontology is treated as a single monolithic model. 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 avoiding 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 part of the model is 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 is will be easier to judge the completeness and consistency of the model. Validated modules could be published early while other parts of the ontology is still under development.
- The existence of modules will enable the use of software tools 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 [6, 8, 3] that 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 we refer to as *modularization* or *partitioning* is discussed in the remainder of this paper.

1.2 What: A Definition of Modularization

The first problem we encounter when approaching the problem of modularization is the lack of a proper definition of this task. While the notion of module is quite well understood in the area of software engineering, it is not clear at all what are the characteristics of an ontology module. Instead of adopting one of the notions of module proposed in the papers mentioned above, we will try to keep our definition of module as general as possible to make our approach applicable with respect to different concrete implementations of ontology modules.

Modules and Partitions We start out definition with the notion of an ontology. In order to be consistent with a wide range of different languages and models, we simply define an ontology to be a set of concepts $C = \{c_1, \cdots, c_m\}$ abstracting from the fact that concepts can be defined and related with each other in many different ways. We chose concepts as the basis for our definition, because every model of ontological knowledge we are aware of contains the notion of a concept in one or the other way.

We can now define modules $m_1, \cdots, m_n$ of an ontology to be sets of concepts from C that are:

- non-empty: $m_i \neq \emptyset, i = 1, \cdots, n$
- contained: $m_i \subseteq C, i = 1, \cdots, n$

This definition is general enough to cover most common situations that deal with modules. In many cases it may be too general in the sense that it does not provide useful criteria for deciding what subset of concepts should form a model. In this paper we therefore concentrate on the notion of partitioning of an ontology that is a special kind of modularization where the modules form a partition of the original ontology, thereby also satisfying the following two criteria:

- covering: $\bigcup_{i=1}^{n} m_i = C$
- disjoint: $i \neq j \Longrightarrow m_i \cap m_j = \emptyset, i = 1, \cdots, n$

These criteria provide the basis for the partitioning methods presented and evaluated in the rest of this paper which are therefore not applicable in cases where not a partitioning, but a different way of modularization is required.

Modularization and Partitioning Based on the definitions above the modularization task can now defined as the task of finding an assignment of classes to modules. For this purpose we introduce an assignment relation $\alpha \subset C \times \mathbb{N}$ into the set of natural numbers. Each number represents a modules and $\alpha(c,k)$ states that the concept c belongs to module m_k. We write $\alpha(c,0)$ to state that c has not been assigned to a module. Each assignment mapping induces a set of modules containing the assigned. The definition of an assignment guarantees that every induced set of modules satisfies the properties discussed above.

In the same way we can define a partition using a special assignment mapping $\alpha : C \rightarrow \mathbb{N}^+$ which is a function into the natural numbers without zero. Being a function, this assignment guarantees that induced modules are disjoint, excluding the zero from the range of the assignment guarantees that the modules cover the original set of concepts. In the following, we consider the task of finding such an assignment function that induces a partition.

1.2.1 How: Intuition and Method

The key question connected to modularization is about criteria for determining the assignment of concepts to modules. This requires a good intuition about the nature of a module that goes beyond the formal criteria given above.

Intuitively, we can say that a module should contain information about a coherent subtopic that can be stand for itself. This requires that the concepts within a module are semantically connected to each other and do not semantically depend on 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 extend our model with a general notion of dependency between concepts. The resulting model of an ontology 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.

There are many different ways in which concepts can depend on each other. 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 need leads us to the central assumption underlying our approach:

> `Assumption`: Dependencies between concepts can be derived form 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.

2 An Automatic Partitioning Method

In this paper we make an initial proposal for an automatic partitioning method based on the structural dependencies between concepts in an ontology. In the current setting, the method uses the subclass hierarchy as the dependencies between concepts, however, it can easily be extended to other types of dependencies. We chose this simple setting in order to be able to better study the behavior of the method before using more complex models. In the following we discuss the basic steps of the method and propose a iterative strategy for determining modules with minimal user input.

2.1 Basic Method

Our method consists of two 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. This task includes the detection of strongly related sets of concepts and the handling of unassigned concepts. In the following we discuss the techniques currently used in these steps.

Step 1: Create Ontology 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 [9] 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. Figure 1 shows the dependency graph created from an OWL version of the UMLS-semantic network using only the class hierarchy.

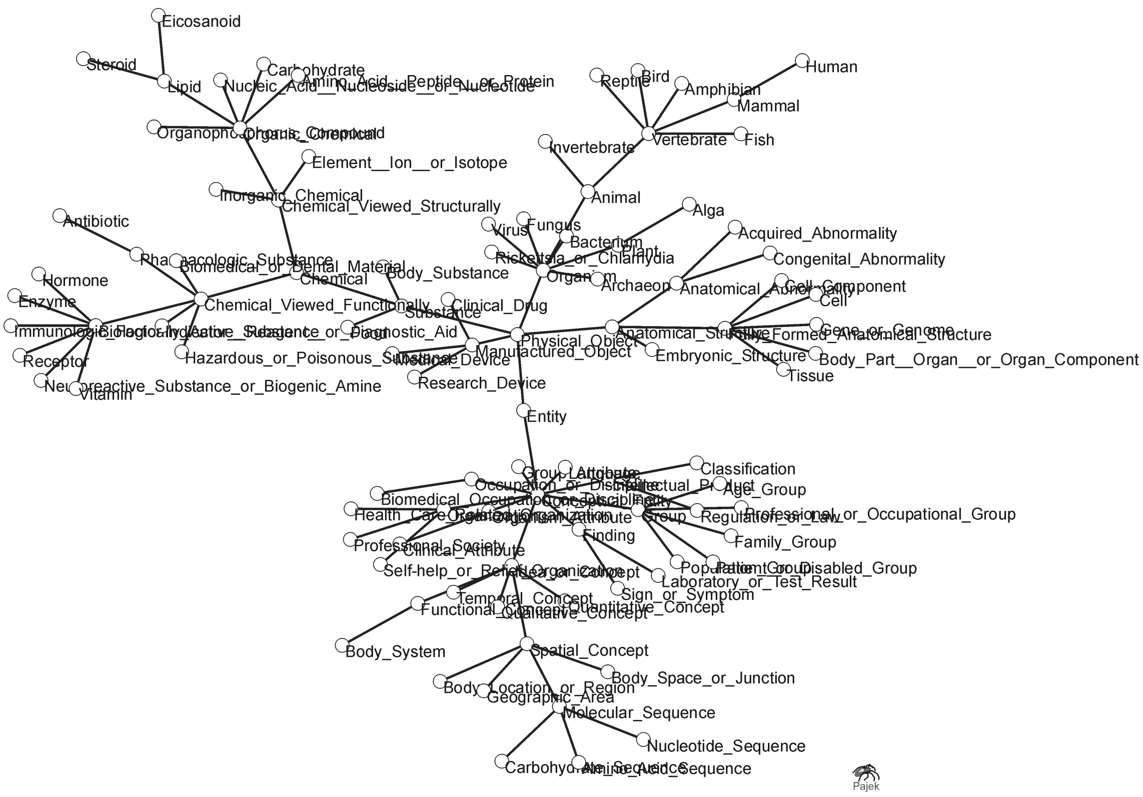

Figure 1: Class Hierarchy graph for the Entity-Related part of the UMLS semantic network

Step 2: Determine Strength of Relations 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 proportional strength p_{ij} of a connection between a node c_i and c_j describes the importance of a link from one node to the other based on the number of connections a node has (a_{ij} is the weight preassigned to the link between c_i and c_j) [4]:

$$p_{ij} = \frac{a_{ij} + a_{ji}}{\sum_k a_{ik} + a_{ki}}$$

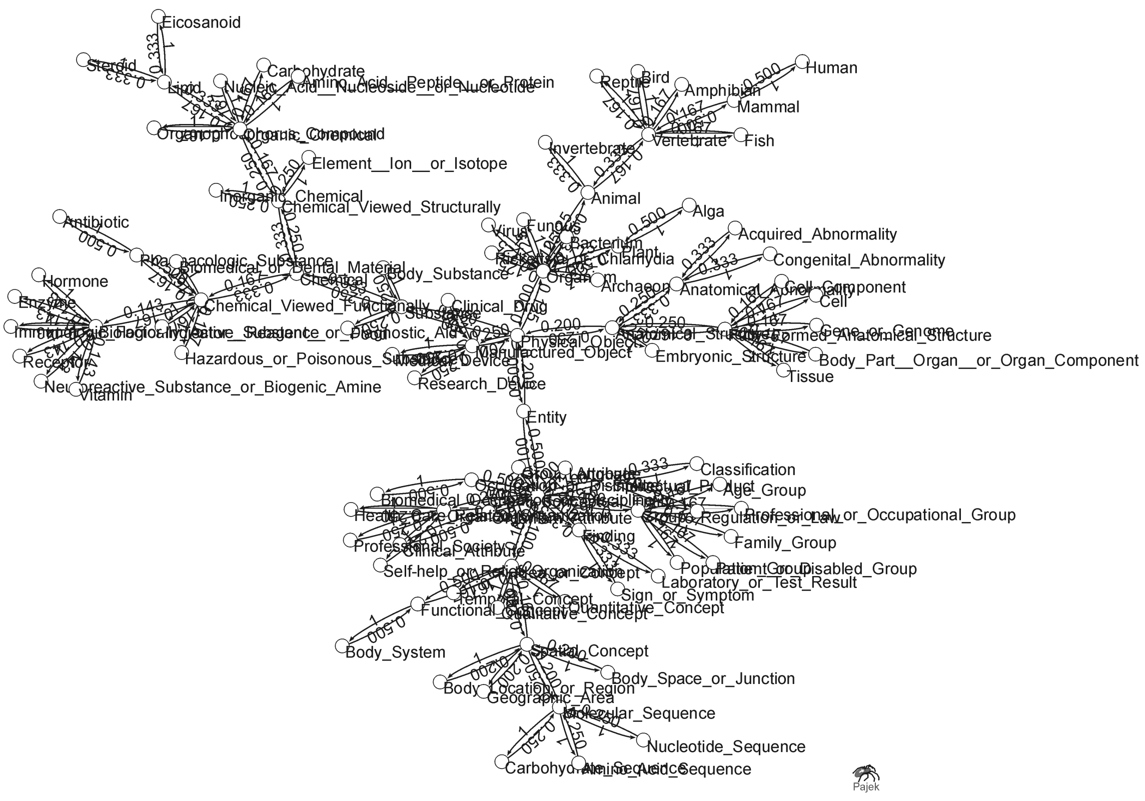

Figure 2: Relative Strength Networks for the Class Hierarchy Graph

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 present 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. Figure 2 shows the proportional strength network for the UMLS dependency graph.

Step 3: Determine Concept Islands The proportional strength network provides us with a foundation for detecting sets of strongly related concepts. For this purpose, we make use of the 'island' algorithm implemented in the Pajek tool. A set of vertices $I \subseteq C$ is a line island in network if and only if it induces a connected subgraph and the lines inside the island are stronger related among them than with the neighboring vertices. In particular there is a spanning tree T over nodes in I such that [1]

$$\max_{(u,v)\in V, v\notin T} w(u,v) < \min_{(u,v)\in T} w(u,v)$$

This criterion exactly coincides with our intuition about the nature of modules given in the introduction. The algorithm requires an upper and a lower bound on the size of the detected set as input. Figure 4 shows the result of determining islands of a size between 5 and 15 nodes.

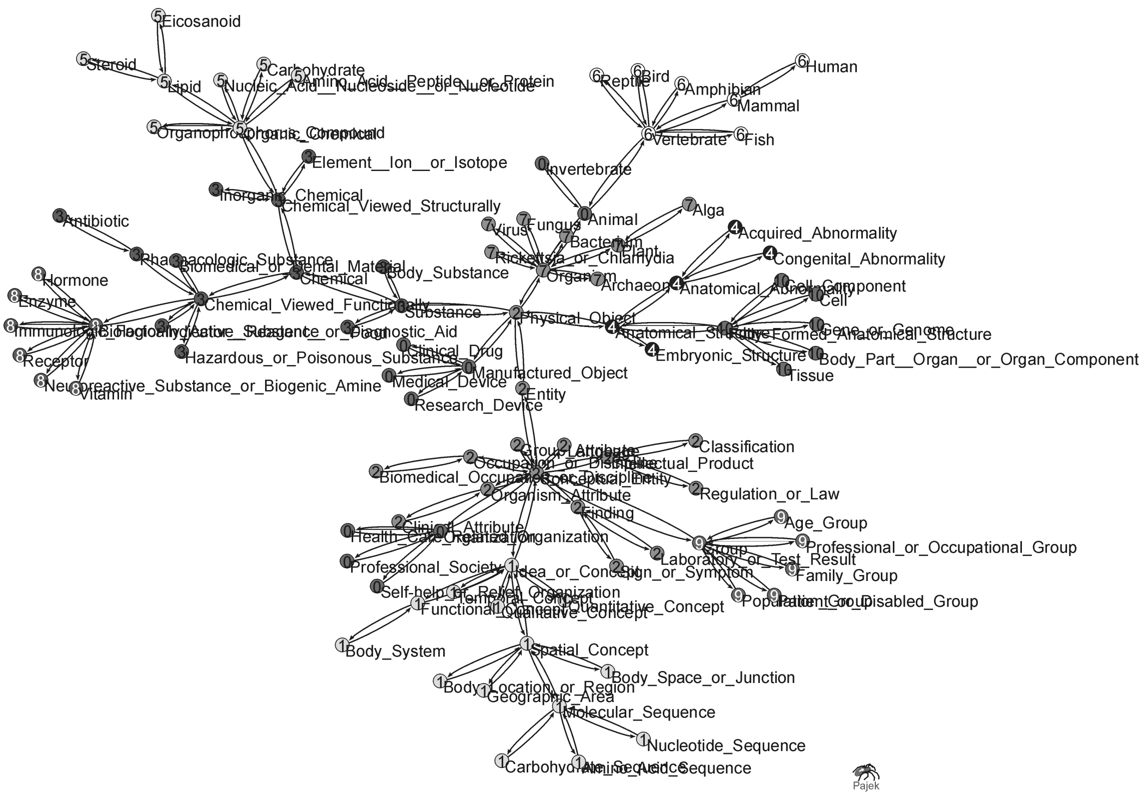

Figure 3: Islands of size between 5 and 15 (10 Islands, 6 unassigned nodes)

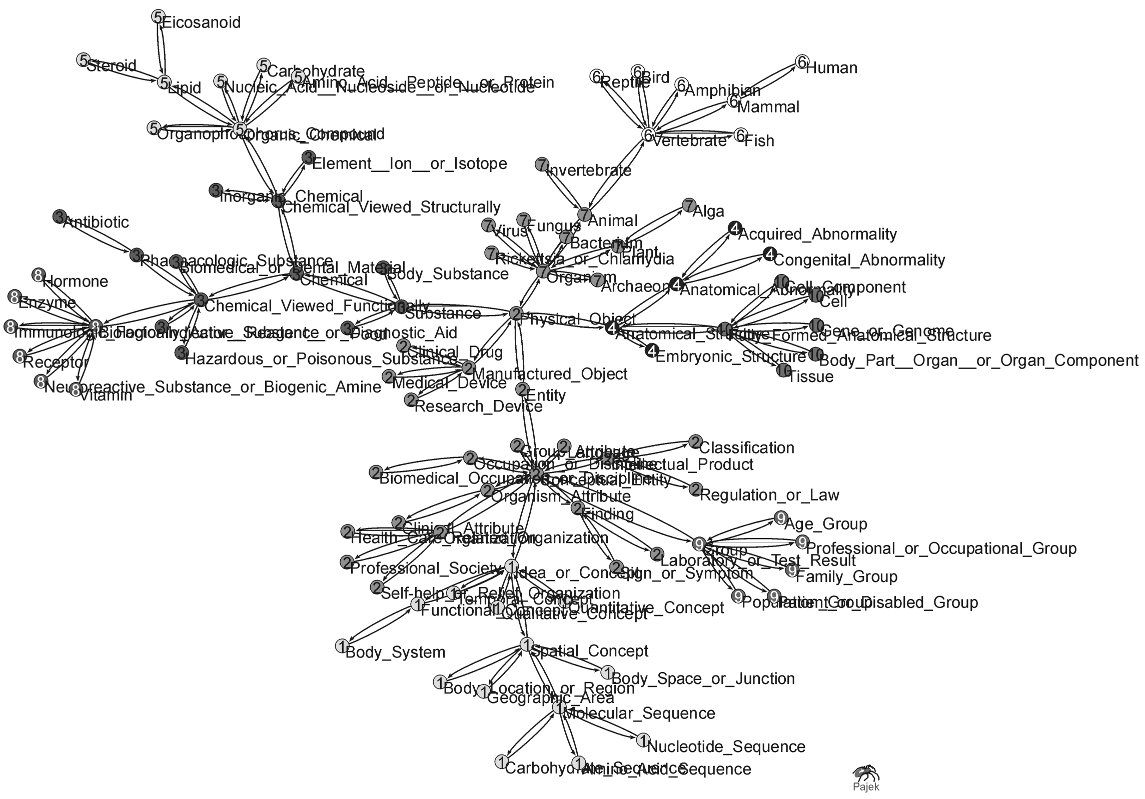

Figure 4: Module assignment after expanding the islands

Step 4: Assign Isolated Concepts Depending on the nature of the dependency graph it may happen that some nodes cannot be assigned to an island. In figure 4 there are 6 of these unassigned nodes marked with a zero (four concepts related to organizations as well as the concepts animal and invertebrate). As our definition of a partitioning does not allow unassigned classes, we have to assign these concepts to a modules as well. The example shows that leftover nodes can occur at different places in the graph and are not necessarily related. Therefore we chose assign them to existing modules. The assignment is based on the strength of the relation to nodes already assigned to a module. In particular leftover nodes are assigned to the island of a that neighboring node they have the strongest they have the strongest relation to. In cases where all neighboring nodes are unassigned as well, these nodes are assigned first. The result of this assignment for the example ontology is shown in figure 4. You can see that the concepts relating to organizations have been assigned to module 2 (rooted at the general concept entity) and the concepts animal an invertebrate are assigned to module 7 (organisms).

2.2 Discussion

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. 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. 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 that in turn created quite large modules with little internal coherence after reassigning them as described in step 4.

3 Assignment Strategy

In order to avoid the problems caused by a wrong choice of upper and lower bond for the module size, we designed an algorithm that iterates over steps 3 and 4 of the process and generates 'natural' islands that are not influenced by the choice of a particular size. In the following we describe the algorithm and exemplify its effect using the same example as above.

3.1 Iterative Algorithm

The idea of the iterative assignment algorithm is to not prescribe the size of modules to be generated but to let them 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 extension step afterwards in order to assign these nodes to a module.

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 are assigned to a unique number 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 $c \in current$ do
      $\alpha(c)$ := counter
    end forreturn counter
  end if
```

3.2 Partitioning Example

We now illustrate the labelling algorithm described above using the UMLS model we also used to describe the different steps of our method. We chose an upper limit of 20 for the size of modules. For the relatively small example model, the algorithm only needs three iterations to determine modules. We discuss the result of the different iterations in the following.

Iteration 1 In the first iteration the algorithm already determines three islands that are smaller than 20. These islands consist of concepts related to biological active substances, different age groups as well as the subtree rooted at the concept concept or idea. While it could be argued that biologically active substances could be included in a larger module on substances the other two modules and in particular the one about ideas and concepts clearly contain concepts that are related and sufficiently different from the other concepts for form a module on their own.

Iteration 2 After removing the modules found in the first step iterating steps three and four results in another island of size lower than 20 namely the subtree rooted at the concept organism. This island is a good example of a very natural module found by the algorithm as the different kinds of organisms clearly form a coherent subtopic within the ontology.

Iteration 3 The third iteration already produces a partition of the remaining concepts into islands that are all of the required size thereby ending the iteration. Figure 7 shows the result of the third iteration that contains the following additional modules:

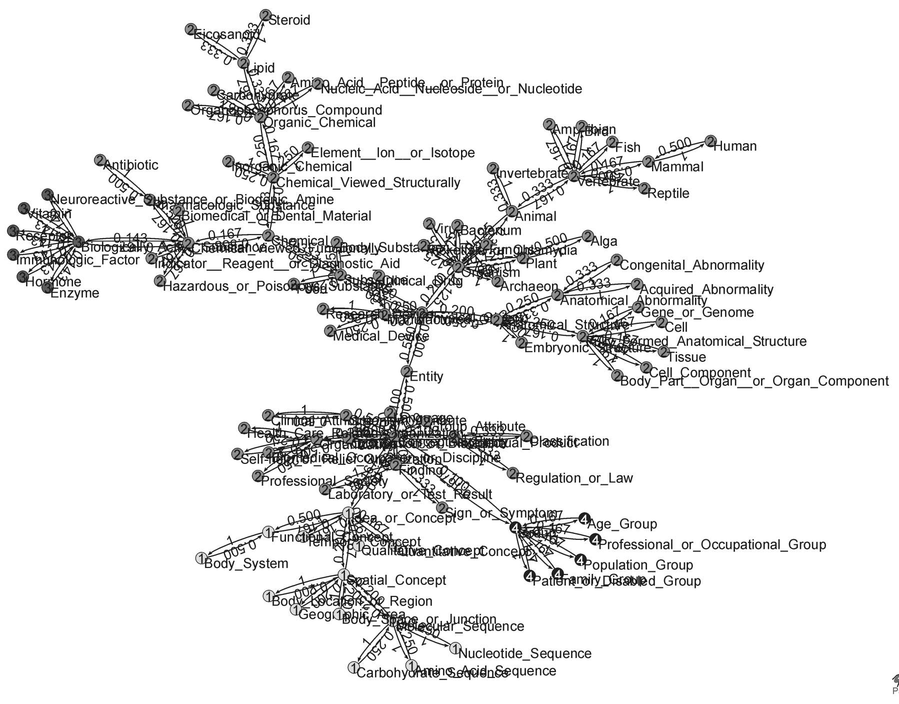

Figure 5: Result of the first Iteration

- Entity
- Organization
- Device
- Anatomical Structure
- Fully Formed Anatomical Structure
- Substance
- Organic Chemical

Most of these modules make sense, the only questionable results are the separation of fully formed anatomical structures from anatomical structures and the separation of organic chemical from substance. We will analyze and discuss these results in more details in the next conclusions of this section.

3.3 Conclusions

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

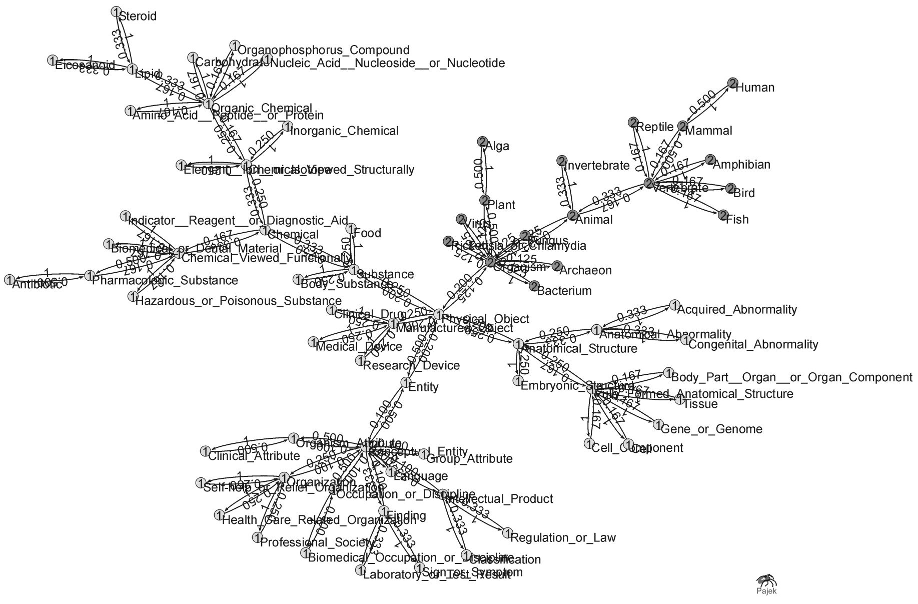

Figure 6: Result of the second Iteration

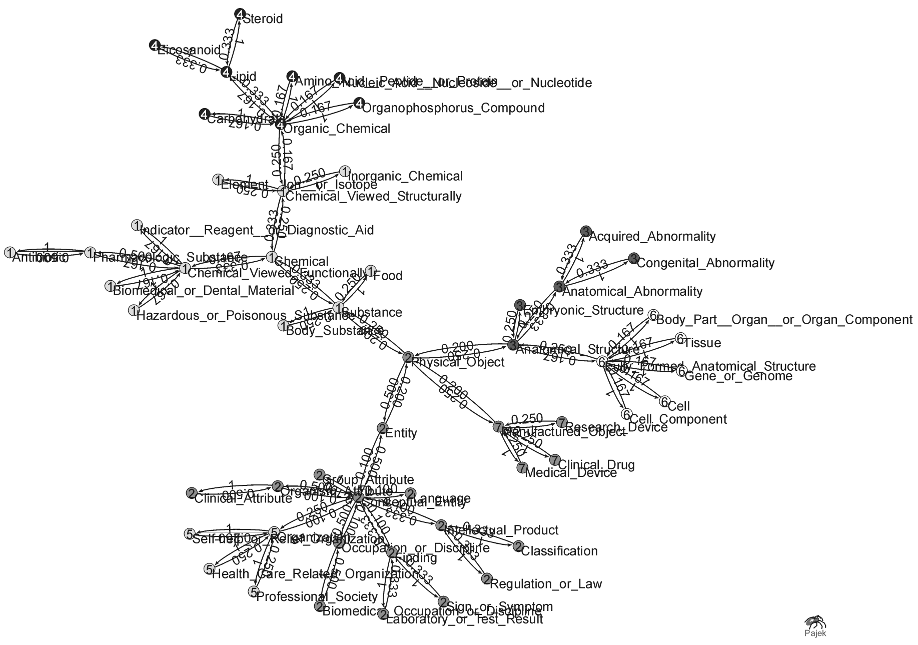

Figure 7: Result of the third Iteration

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. In our example the manufactured objects module for example only contains four concepts. 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 coherent topic are split up into a number of small modules that do not really constitute a sensible model on their own. In the following we describe a post-processing strategy that detects problematic modules and tries to merge them to create a more sensible partitioning.

4 Experiments

In the following, we summarize the results of applying the partitioning strategy to real world ontologies. 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. In our example the manufactured objects module for example only contains four concepts. 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 coherent topic are split up into a number of small modules that do not really constitute a sensible model on their own.

4.1 Evaluating Modularization

When inspecting the dependencies in the relevant parts of the hierarchy, we discovered that most of the concerned modules have very strong internal dependencies. Such a high degree of dependency between a rather small number of nodes separates them from the rest of the graph rather early. In order to detect such cases, we analyze the strength of the internal dependency of generated modules by looking at the minimal spanning tree T used to identify the module and the strength of its links. The over all strength of the internal dependency equals the strength (called the 'height') of the weakest link in the

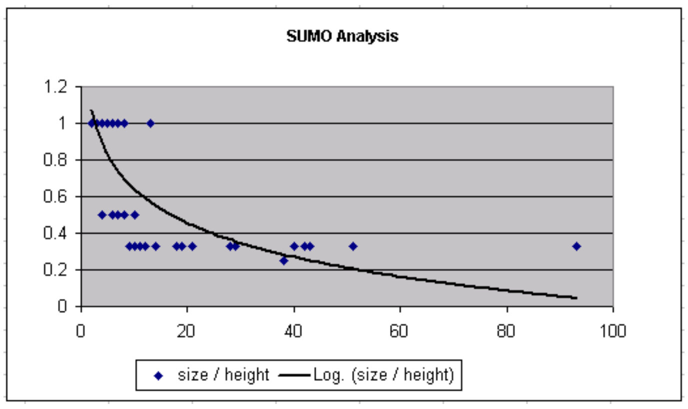

Figure 8: Sizes and heights of partitions in the SUMO ontology

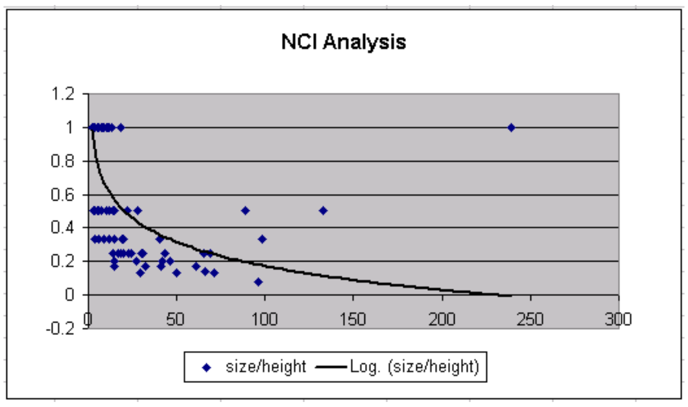

Figure 9: Sizes and heights of partitions in the NCI ontology

spanning tree [1]

$$height(m) = \min_{(u,v)\in T} w(u,v)$$

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 useful level of internal dependency required for a model and a level where important dependencies to the outside are overruled. In our experiments we made the experience that a threshold of 0.5 leads to good results in most cases [1]. Figures 8 and 9 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[2]. The results of these

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

[2]The exception is a part of the NCI ontology that lists all countries of the world and therefore contains

	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

Table 1: Modules Generated for the SUMO Ontology

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.

4.2 Manual Post-Processing

As described above, on result of 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 by merging modules with a height of 1.0 or 0.5 with adjacent modules with a lower height. This merging process was done manually using the functionality of the Pajek tool and some supporting scripts. In many cases, there is only one adjacent module to merge with. In cases, where more than one adjacent module exists, 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. In the following, we present and discuss the results of this experiment for the SUMO and the NCI administration ontology.

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

more than 200 concepts

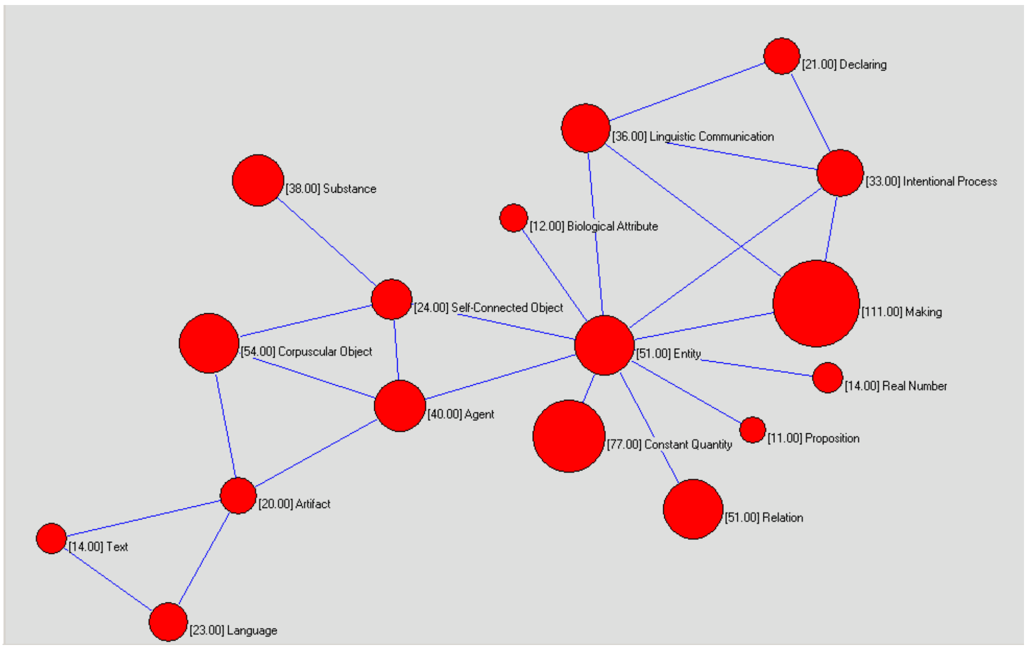

Figure 10: Module Graph of the SUMO Ontology after Merging

merging process are listed in table 1. We see that the topics of the modules that we derived from the top concept of the respective module represent sensible subtopics of the overall ontology that 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 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 shows that this part of the hierarchy is quite tangled which on one hand 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 10 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.

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 9. Often, clearly related parts of the hierarchy had been cut up into a number of small modules. A

	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

Table 2: Modules Created for the NCI Administration Ontology

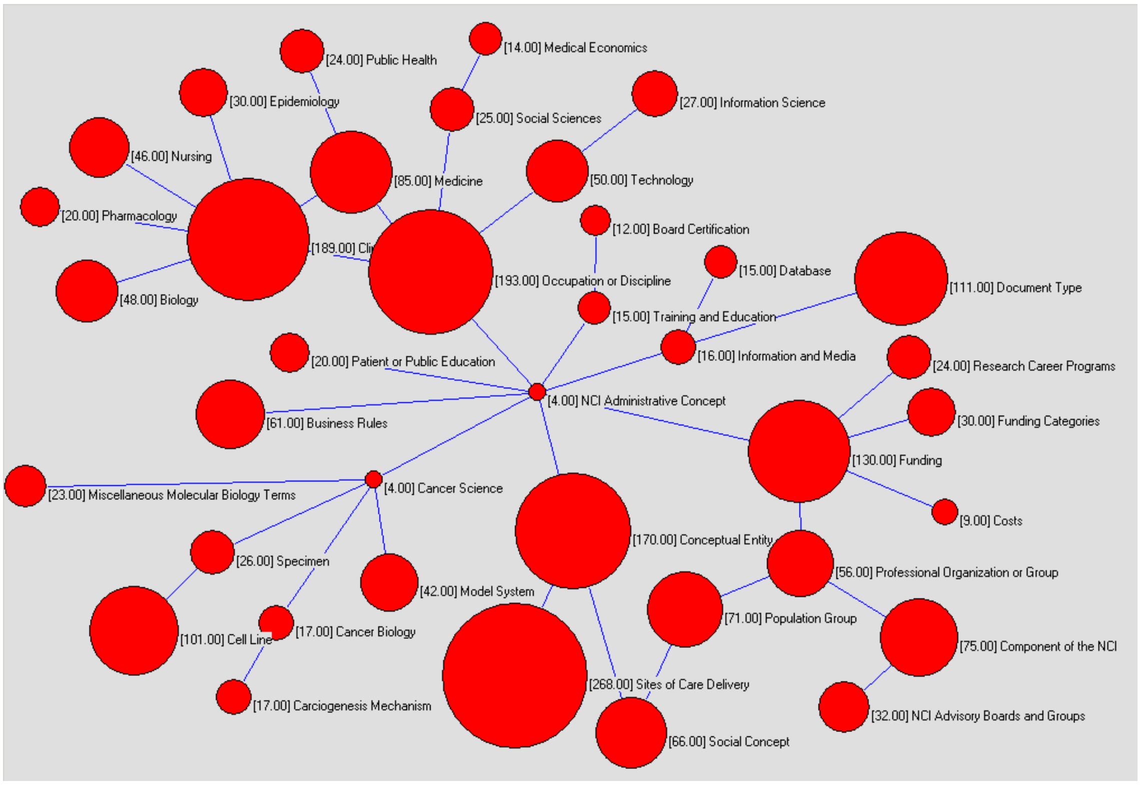
[24.00] Public Health
[14.00] Medical Economics
[30.00] Epidemiology
[25.00] Social Sciences
[27.00] Information Science
[46.00] Nursing
[85.00] Medicine
[50.00] Technology
[20.00] Pharmacology
[189.00] Cli
[12.00] Board Certification
[48.00] Biology
[193.00] Occupation or Discipline
[15.00] Database
[111.00] Document Type
[15.00] Training and Education
[16.00] Information and Media
[20.00] Patient or Public Education
[24.00] Research Career Programs
[4.00] NCI Administrative Concept
[61.00] Business Rules
[30.00] Funding Categories
[130.00] Funding
[23.00] Miscellaneous Molecular Biology Terms
[4.00] Cancer Science
[9.00] Costs
[170.00] Conceptual Entity
[26.00] Specimen
[56.00] Professional Organization or Group
[42.00] Model System
[101.00] Cell Line
[17.00] Cancer Biology
[71.00] Population Group
[75.00] Component of the NCI
[268.00] Sites of Care Delivery
[17.00] Carciogenesis Mechanism
[32.00] NCI Advisory Boards and Groups
[66.00] Social Concept

Figure 11: Module Graph of the NCI Administration Ontology after Merging

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 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 heterogenous. IN future work, we will have to analyze these modules in detail and determine a strategy for avoiding the problem by adjusting the dependency measure or the merging process.

5 Discussion

In this report we describe a method for structure-based ontology partitioning that is practically applicable to very large ontologies. We show that a modularization based only on structural properties of the ontology 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[3].

An advantage of the method described in this paper is that there are no arbitrary choices that have to be made in advance (e.g. the number of modules required). However, there are several choices we made 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 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" 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:

[3]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.

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. 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. Jnger 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. Burt. *Structural Holes. The Social Structure of Competition.* Harvard University Press, 1992.

[5] J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, J. Oberthaler, and B. Parsia. The national cancer institute's thesaurus and ontology. *Journal of Web Semantics*, 1(1), 2003.

[6] 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.

[7] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[8] 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.

[9] 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.