Towards Structural Criteria for Ontology Modularization

Anne Schlicht, Heiner Stuckenschmidt

University of Mannheim, Germany
{anne,heiner}@informatik.uni-mannheim.de

Abstract. Recently, the benefits of modular representations of ontologies has been recognized by the semantic web community. Existing methods for splitting up models into modules either optimize for completeness of local or for the efficiency of distributed reasoning. In our work on semantics-based P2P systems, we are also concerned with the additional criteria of robustness or reasoning in cases where peers are unavailable and with ease of maintenance. We define a number of structural criteria for modularized ontologies and argue why these criteria are suitable for estimating efficiency, robustness and maintainability. We apply the criteria to a number of modularization approaches and discuss the trade-offs made. Based on the discussion we propose a general quality measure for modular representations in the context of our use case.

1 Motivation

The problem of modularizing ontologies in the sense of splitting up an existing ontology into smaller, interconnected parts has recently been discussed by a number of researchers (see for example [10, 1, 9]). These approaches differ significantly in terms of the concrete goal of the modularization and consequently in terms of the criteria used to determine a good modularization. In our work we are concerned with the automatic modularization of ontologies to support a distribution of knowledge in a P2P network in such a way that following requirements are fulfilled:

Efficiency Reasoning in the distributed system should be efficient. This also requires that communication costs, which are known to be a major bottleneck in distributed systems are minimized.

Robustness In a P2P network, single peers can be temporarily or permanently unreachable. The impact of such failures on the completeness of reasoning should be minimized.

Maintainability Ontologies evolve over time and the corresponding changes need to be propagated through the network. The number of changes and the required costs for computing necessary changes should be minimized.

It is obvious that these are conflicting requirements that need to be balanced in order to determine an optimal strategy for modularization. Similar problems

have been addressed in the area of deductive databases [8, 3] where authors are concerned with an optimal distribution of datalog rules in a network. Similar work has also been done in the area of theorem proving [6] where approaches for an optimal partitioning of propositional knowledge bases have been proposed. In both cases, the only criterion was the efficiency of reasoning. As mentioned above, we are also concerned with robustness and maintainability. This means that existing approach do not find the best solution for our problem. Solving the corresponding optimization problem optimally is impractical for a number of reasons. First of all, there is no reasoning infrastructure available that could be used to check the criteria mentioned above. Second, an optimal solution can only be computed with respect to statistics about the use of the system which are currently not available. Finally, even if we had a reasoning infrastructure and usage statistics it is easy to see that computing an optimal solution is computational infeasible as it involves reasoning about ontologies which itself is intractable and also spans a combinatorial search space which adds complexity to the approach.

Our solution to this problem is the following: instead of directly checking the criteria mentioned above, we propose a number of structural criteria that can be checked by looking at the modularized ontology without actually performing reasoning. These structural criteria that are the main contribution of this paper are based on general knowledge about factors that contribute to computational complexity, robustness and ease of maintenance. In particular, we propose the following structural criteria that are discussed in more detail in the following section:

Connectedness of Modules The efficiency of distributed computation is known to be heavily influenced by the amount of communication necessary. This effort can be estimated by looking at the degree of interconnectedness of the generated modules. The connectedness also has an influence on the robustness as the number of connections that are potentially cut when a peer in unavailable.

Size and Number of Modules The size and number of modules created has a strong influence on the robustness. If most of the information is in a small number of large modules, the unavailability of one of these modules will have a big impact on the completeness of the system. If the information is more or less equally distributed across modules, this impact will be much smaller. On the other hand, a modularization that produces a very high number of very small modules on the other hand will lead to efficiency problems.

Redundancy of Representation A common way of improving efficiency and to improve robustness is to use redundant representations. In an extreme case the complete ontology could be put in each module. This redundant information increases the maintenance effort and should be avoided. Further it has been shown in [4] that reasoning on non-redundant representations of parts of the complete model can lead to performance improvements.

In the remainder of this paper, we first discuss the concrete structural criteria proposed in more details. We then report from some experiments in which we compared different existing partitioning approaches using the structural criteria. In particular, we used our own approach described in [10] and the approach of Cuenca-Grau [1] which is implemented in the SWOOP ontology editor. Based on the results of the comparison, we draw some conclusions about the trade-offs made by the different approaches and propose a unified quality measure for modularizations based on the structural criteria that seems to be suited for our specific use case.

2 Structural Criteria

For the sake of simplicity, we consider an ontology to be a set of axioms $\{A_1, \cdots, A_n\}$. These axioms can for instance be rules, concepts definitions in description logics or axioms in any other logical language. We consider the number of axioms n to be the size of the ontology. The task of partitioning an ontology $\mathcal{O}$ can now be described as the process of splitting up the set of axioms into a set of modules $\{M_1, \cdots, M_k\}$ such that

- each M_i is an ontology
- $\bigcup_{i=1}^{k} M_i = \mathcal{O}$

This definition leaves room for a large variety of different modularizations, regardless whether they are suited for the distribution scenario described in the preceding section. In order to judge whether a given modularization is a good one with respect to the goals of the distribution, we define a number of structural quality criteria that indicate whether a given modularization is likely to meet the goals of the distribution.

2.1 Size

The first set of criteria is concerned with the size of the modules created. Analysing the influencing factors for defect density in software engineering it was discovered that there is a strong correlation to the size of the software modules [5, 2] Since ontologies are mainly handmade by ontology engineers, a process that is very similar to programming, the defect density of ontologies is determined by similar human capabilities. The optimal size of software modules is quoted to be about 200-300 logical lines of code. The closest correspondent to logical lines in ontologies are descriptions, the basic building blocks of class axioms depicted in the W3C Web Ontology Language Reference [7].
In order to have a good distribution over the network, we want the sizes of the modules clustered around the optimal size. Normally, we would measure this

using the standard deviation. In this particular case, however, we want to take into account the possibility for improving the distribution of axioms after the initial modularization step. In particular, we do not want to punish the creation of modules that can easily be merged with others to form modules of wanted size. Consider the example of a partitioning consisting of three large modules and many modules that contain only one or two axioms. The standard deviation would be very high although this partitioning can easily be optimized by merging the small modules into the big ones. For obtaining a criterion that measures the main algorithm independent of optimization we define the size indicator in terms of the fraction of axioms contained in modules of appropriate size.

Definition 1 (Appropriateness of module size). *According to the correlation to defect density we define a function* appropriate *that maps module sizes* $n_i = |M_i|$ *to values in* $[0, 1]$. *The shape of this function reflects the defect density, at the size limits the value is zero and the value of the optimal size is one. Furthermore the derivative is zero at the size limits to avoid discontinuity. A function that meets these requirements is depicted in figure 1.*

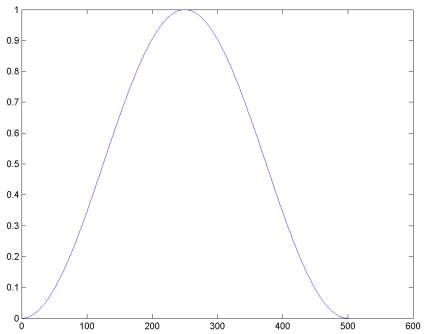

Fig. 1. The *appropriate* function.

The formal definition of this function is

$$appropriate(x) = \frac{1}{2} - \frac{1}{2}\cos(x \cdot \frac{\pi}{250})$$

Based on the appropriateness of the individual module sizes we obtain a global indicator for partitionings by averaging over the contained axioms:

$$size = \frac{\sum_{i=1}^{k} n_i \cdot appropriate(n_i)}{\sum_{i=1}^{k} n_i}$$

size is required to be greater than zero to ensure existence of at least one acceptable module.

2.2 Redundancy

Apart from the size of individual modules we rate the blow up of the ontology due to duplication of axioms. As mentioned above, redundant representations can be used to improve efficiency but increase the effort needed to maintain the distributed model. We therefore use the degree of redundancy in terms of the fraction of duplicated axioms with respect to n.

Definition 2 (Redundancy). *The number of duplicated axioms should be limited.*

$$redundancy = \frac{(\sum_{i=1}^{k} n_i) - n}{\sum_{i=1}^{k} n_i}$$

The requirements regarding the sizes of modules are rather weak due to the fact that partitioning algorithms commonly produce partitionings that fail anyway.

2.3 Connectedness

Another set of criterion is concerned with the interaction of the axioms in the different modules. In order to be able to make assertions about this interaction, we use the notion of a axiom graph. The axiom graph of an ontology is a labeled graph $\langle N, E, s, t\rangle$ where the set of nodes N is the set of axioms of the ontology.

The first type of a connection is defined in terms of shared symbols. Two nodes are connected by an edge if they share a non-logical symbol (i.e. a predicate name). Note that axioms that share more than one non-logical symbol are connected by one edge per shared symbol. Each node is labeled with the axiom it represents, each edge is labeled with a symbol shared by the connected nodes. In the axiom graph, we distinguish intra-module and inter-module edges. An edge e with $s(e) \in M_i, t(e) \in M_j$ is an inter-module edge if $i \neq j$ and an intra-module edge if $i = j$. We denote the number of all inter-module edges as E_X. Based on this notion of a axiom graph, we can now define the following criteria.

Definition 3 (Connectedness). *The number of symbols shared between axioms in different modules should be as small as possible. For comparing different partitioning algorithms we consider the fraction of inter-module edges with respect to the total number of edges:*

$$connected_{\cup} = \frac{|E_X|}{|E|}$$

Since communication cost increases with the number of extern edges a low connected value implies that the modules are sparsely connected and therefore indicates a good partitioning.

This criterion of small shared language was mentioned in some approaches to partitioning knowledge bases [6]. It is a appropriate indicator for communication cost as long as the modules are disjunct or rarely intersecting. With a large amount of axioms shared by two or more modules this type of edges fails to reflect module connection. Consider the following example:

Partitioning 1	Partitioning 2
$A_{1,1} : C \sqsubseteq D \sqcap E$	$A_{1,1} : C \sqsubseteq D \sqcap E$
$A_{1,2} : E \doteq A \sqcap B$	$A_{1,2} : E \doteq A \sqcap B$
$A_{2,3} : H \doteq G \sqcap \exists r.E$	$A_{2,3} : H \doteq G \sqcap \exists r.E$
	$A_{2,2} : E \doteq A \sqcap B$

The axiom A_2 is duplicated in the partitioning on the right to reduce communication cost. Since modules 1 and 2 in partitioning 2 do not need to communicate about A_2 the connected-indicator should decrease when duplicating the axiom. The axiom graph does not support distinguishing this type of extern edge because it does not contain them.

For defining a connectedness indicator for partitionings with intersecting modules we first define the partition graph $\langle N_M, E_M, s, t\rangle$ where the set of nodes is the set of axioms $\{A_{i,l} \in \mathcal{O} \mid A_l \in M_i\}$ with duplicates identified by the modules containing them and the set of edges is $E_M \subseteq \{e \mid s(e) = A_{i,l}, t(e) = A_{j,m}, i \neq j\}$. Apart from the intra-axiom extern edges there is another inaccuracy in the first connectedness indicator when used for intersecting modules. For example two axioms that are contained in every module of a partitioning induce 2^k edges although they do not cause any communication cost. Formally we attempt to consider only the extern edges $E_{MX} = \{e \mid s(e) = A_{i,l}, t(e) = A_{j,m}, A_l \notin M_i, A_m \notin M_j\}$ and denote the set of edges as $E_M = \{e \mid s(e) = A_{i,l}, t(e) = A_{j,m}, i \neq j, l = m \vee (A_l \notin M_i, A_m \notin M_j)\}$. Note that in general it is not defined whether an edge e of the axiom graph is an extern edge because it depends on the modules connected by this edge not only on the axioms. For illustrating consider two axioms A_1 and A_2 both of them contained in two modules but only one contained in a third module. With respect to modules one and two the edge would be discarded but w.r.t. modules one and three it is a proper extern edge.

Definition 4 (Connectedness of intersecting modules). *For comparing different partitioning algorithms that generate intersecting modules we compare the number of extern edges to the number of edges in the partition graph:*

$$connected_{\cup} = \frac{|E_{MX}|}{|E_M|}$$

In case of disjunctive modules this coincides with the definition of $connected_{\dot{\cup}}$

2.4 Relative Distance

The above indicator depicts the number of direct dependencies between modules. For characterizing the communication effort caused by separation of the ontology into modules we compare the average path length of the partition graph to the average path length of the axiom graph. The average path length (*apl*) of a graph is the distance between two nodes averaged over all pairs of nodes in the graph. Since we can not guarantee reachability and are only interested in the relation of the two average path lengths we elide shortest paths with $dist(a, b) = \infty$.

$$apl_{<\infty}(\langle N, E, s, t\rangle) = \frac{1}{|N|(|N|-1)} \sum_{\substack{a,b \in N \\ dist(a,b)<\infty}} dist(a, b)$$

For comparison of these paths a formal definition of the distance between axioms in the partition graph is required. $dist(A_l, A_m)$ is the length of the shortest directed path in the axiom graph from axiom A_l to axiom A_m.

Definition 5 (Relative Distance). *Let* $dist(A_{i,l}, A_{i,m})$ *be the minimal number of extern edges on a path from* $A_{i,l}$ *to* $A_{i,m}$*. The inter-module communication needed to access connected axioms should be minimal. In particular the average path length of the partition graph should be short compared to the average path length of the axiom graph.*

$$effort = \frac{apl_{<\infty}(\langle N_M, E_M, s, t\rangle)}{apl_{<\infty}(\langle N, E, s, t\rangle)}$$

$$= \frac{\frac{1}{|N_M|(|N_M|-1)} \sum\limits_{l,m \leq n} \sum\limits_{\substack{i,j \\ dist(A_{i,l}, A_{i,m})<\infty}} dist(A_{i,l}, A_{i,m})}{\frac{1}{n(n-1)} \sum\limits_{\substack{l,m \\ dist(A_l, A_m)<\infty}} dist(A_l, A_m)}$$

The definitions made above provide us with the possibility to compare different modularization approaches in terms of the values for the indicators $size, redundancy, connected$ and *effort* that can be determined experimentally for different approaches. They also serve as a basis for developing a parameterized partitioning algorithm that optimizes the results according to these requirements.

3 Experiments

As mentioned in the introduction, the structural criteria introduced in the previous sections contradict each other and force modularization methods to make certain trade-offs. We started analyzing these trade-offs in a number of experiments one of which we report in the following. The goal of this experiment was to analyze how existing algorithms trade-off size and connectedness. It is obvious that minimizing connectedness will normally lead to a small number of large modules. The size criterion on the other hand favors modularizations that consist of a number of equally large modules. As the basis for the experiment we chose two real world ontologies from the medical domain. The DICE ontology has been developed at the Amsterdam Medical Center to describe concepts of clinical medicine. The statistics of the DICE and GALEN ontologies are given below:

Ontology: DICE
Language: $\mathcal{ALC}(D)$
Classes: 2543
Properties: 50
SubClass Statements: 3919
Max. Depth of Class Tree: 9
Min. Depth of Class Tree: 1
Avg. Depth of Class Tree: 5.06
Max. Branching Factor: 259
Min. Branching Factor: 1
Avg. Branching Factor: 6.49

Ontology: GALEN
Language: $\mathcal{SHF}$
Classes: 2749
Properties: 413
SubClass Statements: 1978
Max. Depth of Class Tree: 13
Min. Depth of Class Tree: 1
Avg. Depth of Class Tree: 5.35
Max. Branching Factor: 766
Min. Branching Factor: 1
Avg. Branching Factor: 5.69

In the experiments we automatically slit up the DICE ontology using two existing modularization tools: the modularization method implemented in the SWOOP ontology editor and the PATO tool for ontology partitioning developed at the Vrije Univeristeit Amsterdam. We applied the modularization methods and compared the values for size and connectedness. Since SWOOP and PATO partition symbols (concept and property names) instead of descriptions we adapted the *appropriate* function to this domain. The ratio descriptions to symbols is about five descriptions per symbol so we applied *appropriate* with an optimal size of 50 symbols instead of 250 descriptions.

3.1 SWOOP

In a recent paper Cuenca-Grau and others propose a modularization method that aims at producing modules that are self contained in the sense that all inferences about the signature contained in a module can be made solely on the basis of local reasoning [1]. The method consists of three basic steps:

Safety Check The above mentioned guarantees for completeness of local reasoning can only be given if the ontology satisfies certain safety conditions (for details see [1]. These conditions are checked in the first step. The DICE ontology that we used in our experiments satisfies these conditions.

Partitioning In the second step, the definitions in the ontology are partitioned into disjoint sets. The algorithm starts with a single partition. In a partitioning step it non-deterministically creates new partitions and checks whether parts of the definitions can be moved to that new partition without violating the completeness condition. The later is checked using the structure of the concept definitions and the relation to concepts already placed in other partitions.

Module Generation In the third step, the partitioning created in the second are used to determine the actual models. This is done by merging partitions into overlapping modules. At this stage, redundancy can potentially be introduced into the ontology.

Applying the Method to the DICE ontology resulted in 146 modules most of which (122) only contained a single concept definition. The majority of the definitions (2373 out of 2543) ended up in a single module. The results for the GALEN ontology where even more extreme. It was partitioned in two modules with one of them containing only a single definition. This clearly reflects the extreme strategy implemented in SWOOP to always prefer local reasoning over a good distribution of module sizes. This strategy is also reflected in a very good value for connectedness and a very bad one for size.

DICE		GALEN	
size	*connected*	*size*	*connected*
0.0007822223	0,0443	~ 0	0

It should be mentioned that partitionings produced by SWOOP are not always this extreme. Another selection is reported in [1].

3.2 Pato

PATO is a tool for automatically partitioning ontologies into disjoint sets of concepts based on structural criteria [10]. The method underlying PATO consists of the following steps:

Step 1: Create Dependency Graph: In the first step a dependency graph is extracted from an ontology source file. The idea is that elements of the ontology (concepts, relations, instances) are represented by nodes in the graph. Links are introduced between nodes if the corresponding elements are related in the ontology, e.g. because they appear in the same definition.

Step 2: Determine strength of Dependencies: In the second step the strength of the dependencies between the concepts has to be determined. This actually consists of two parts: First of all, we can use algorithms from network analysis to compute degrees of relatedness between concepts based on the structure of the graph. Second, we can use weights to determine the importance of different types of dependencies, e.g. subclass relations have a higher impact than domain relations.

Step 3: Determine Modules The proportional strength network provides us with a foundation for detecting sets of strongly related concepts. This is done using a graph algorithm that detects minimal cuts in the network and uses them to split the overall graph in sets of nodes that are less strongly connected to nodes outside the set than to nodes inside.

Step 4/5: Improving the Partitioning In the last steps the created partitioning is optimized. In these steps nodes leftover nodes from the previous steps are assigned to the module they have the strongest connection to. Further, we merge smaller modules into larger ones to get a less scattered partitioning. Candidates for this merging process are determined using a measure of coherence.

The PATO method can be tuned to a given problem using a number of different parameters. First of all, the user can select which kind of syntactic features are used to create the links in the dependency graph in step 1. Further, the user can chose the maximal size of islands created in step 3. Finally, a threshold value can be selected that determines which modules are merged in step 5. The lower the threshold, the more modules are merged.

In an experiment, we computed the values for size and relatedness of the modularization generated by PATO using different settings. Since the modules generated by Pato do not intersect *redundancy* is always zero. The table below shows the result based on a dependency graph created from subclass links and the occurrence of relation names in concept definitions for different threshold values.

DICE			GALEN		
thres.	*size*	*connected*	*thres.*	*size*	*connected*
0.1	0.0000	0.3720	0.1	0.0796	0.4695
0.2	0.1122	0.5410	0.2	0.1393	0.4860
0.3	0.2747	0.5724	0.3	0.2740	0.4900
0.4	0.3068	0.6003	0.4	0.2823	0.4941
0.6	0.2940	0.6092	0.6	0.2903	0.4955
1.0	0.2940	0.6092	1.0	0.2903	0.4955
1.1	0.2319	0.7010	1.1	0.3073	0.5663

We can see that setting the threshold parameter generally trades size quality for connectedness of the resulting partitioning[1]. But choosing a threshold of 1.0 is favorable over not merging at all (thres.=1.1).

4 Discussion

Our experiments showed that using the structural criteria proposed in this paper, it is possible to analyze the strategies of different modularization algorithms. we could see that algorithms that generate sparsely connected modules often fail completely with regard to their size. Evidently partitioning algorithms trade communication cost for feasibility of module sizes. While one value is highly optimized, the other degrade and make the partitioning less useful for our purpose. The partitioning method in SWOOP is an example of a method that focusses on the reduction of communication costs. In particular, it was designed to ensure that no communication is necessary at all to answer queries about the content of a module. The above criteria inspire the development of an algorithm with balanced indicator values. For algorithms with very low *size* value the idea is weakening the requirements for module autonomy to enable feasible sizes. Another approach to moderate indicator values is by means of a parameterized partitioning algorithm. For example in PATO we identified the favorable configuration of links and the best value for the threshold by the resulting values for *size* and *connected*. In the DICE experiments, this optimal trade-off could be observed at a threshold value of 0.4 (compare figure 2).

A drawback of the current implementation of PATO is the fact, that the optimal parameters have to be determined by hand beforehand. A straightforward idea is to automatically determine the optimal parameters on the basis of the criteria described in this paper. This can be done in terms of solving an optimization problem that tries to optimize the overall quality of the modularization. What we need for this is an overall quality value based on the individual criteria. Such an overall measure can be achieved using the following definition of quality.

Definition 6 (Quality). *The combined indicator for partitioning quality.*

[1] note that for size large and for connectedness small values are preferred

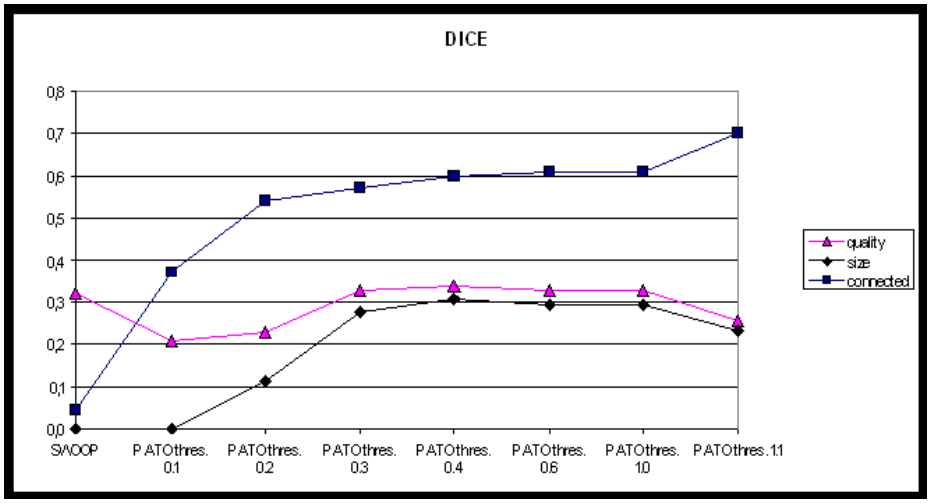

Fig. 2. Size vs. Connectedness

$$quality = \frac{1}{4}(size + 3 - redundancy - connected - effort)$$

The importance of the different factors depends on the application which can be taken into account by additional weights. For example if the application requires a particularly low connectedness we would change the equation to $quality = \frac{1}{6}(size + 5 - redundancy - 3 \cdot connected - effort)$. *For our experiments we put zero weight on redundancy and effort by applying* $quality = \frac{1}{3}(2 \cdot size + 1 - connected)$.

For comparing different partitioning algorithms the separate values of $size, redundancy, connected$ and *effort* are more significant whereas automatic optimization is facilitated by an integrated indicator. Detecting the pareto-optimal configurations is another possibility if adequate weights can not be assigned to the quality indicators. Applying this method we identify automatically the set of configurations, that would be optimal with weights set accordingly. The final configuration is then selected from this set manually.

Currently, PATO does not partition axioms but symbols (concept and property names) that appear in the given set of axioms. For conversion to partitioning axioms a allocation routine is required that determines the partitioning of axioms depending on a given partitioning of concept and role names. There are multiple options for this allocation routine. Combining PATO with an allocation routine we obtain a parameterized partitioning algorithm that can be applied by means of the quality indicator and a common greedy procedure for parameter assignment. As the results show, however, the current algorithms are far from being optimal for the task at hand. Therefore, future work will include modifications to the existing partitioning method in PATO to better fit the requirements of ontology modularization for P2P systems.

Acknowledgement

This work was partially supported by the German Science Foundation in the Emmy-Noether Program under contract Stu 266/3-1.

References

1. Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, 2006.
2. Les Hatton. Reexamining the fault density-component size connection. *IEEE Software*, 14(2):89–97, 1997.
3. Seung Jin Lim and Yiu-Kai Ng. Vertical fragmentation and allocation in distributed deductive database systems. *Information Systems*, 22(1):1–24, 1997.
4. Bill MacCartney, Sheila McIlraith, Eyal Amir, and Tomas Uribe. Practical partition-based theorem proving for large knowledge bases. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.
5. Yashwant K. Malaiya and Jason Denton. Module size distribution and defect density. *issre*, 00:62, 2000.
6. S. McIlraith and E. Amir. Theorem proving with structured theories. In B. Nebel, editor, *Proceedings of IJCAI'01*, pages 624–634, San Mateo, August 2001. Morgan Kaufmann.
7. Guus Schreiber Mike Dean. Owl web ontology language reference. http://www.w3.org/TR/2004/REC-owl-ref-20040210/.
8. Mukesh K. Mohania and Nandlal L. Sarda. Rule allocation in distributed deductive database systems. *Data Knowledge Engineering*, 14(2):117–141, 1994.
9. Julian Seidenberg and Alan Rector. Web ontology segmentation: Analysis, classification and use. In *Proceedings of the 15th international World Wide Web Conference*, Edinburgh, Scotland, 2006.
10. H. Stuckenschmidt and M. Klein. Structure-based partitioning of large concept hierarchies. In D. Plexousakis, S. McIllraith, and F. van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference ISWC*, Lecture Notes in Computer Science, 2004.