Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat)

STRUCTURE-BASED PARTITIONING OF SEMANTIC WEB ONTOLOGIES

eingereicht am Fachbereich Mathematik und Informatik
der Freien Universität Berlin von

Gökhan Coşkun

Tag der Disputation: 1. Oktober 2014

1. Gutachter: Prof. Dr. Robert Tolksdorf
2. Gutachter: Prof. Dr. Heiner Stuckenschmidt

Yesterday I was clever, so I wanted to change the world.
Today I am wise, so I am changing myself.

Jalal ad-Dīn Muhammad Rumi

ABSTRACT

Component-based development of large and complex software systems by small well defined building blocks improves the comprehension as well as the management and leads to reusable software modules and a scalable overall system. Accordingly, designing ontologies in a modular way is intuitively promising in order to benefit from the same advantages. However, the status quo is that the most publicly available ontologies are monolithic. For that reason the number as well as the size of available ontologies has increased with the growing utilization during the last years. In order to improve the efficient usage (e.g. through distributed and scoped reasoning for reasoners), to simplify the maintenance (e.g. through refactoring support) and to allow reusable components (e.g. through increased human understandability) there is a need to partition large ontologies into well-sized building blocks in a (semi-) automatic way. Especially from the viewpoint of the Semantic Web reusability is a crucial issue because an agreed common semantic model allows easy data integration and interoperability.

Considering ontologies as networks of concepts connected through properties, utilizing network analysis techniques is a promising approach to analyze and partition ontologies. As a very well established discipline in science there are a lot of sophisticated methods, algorithms and tools for network analysis available. This work is driven by the belief that these methods can be modified and applied to ontologies, so that the ontology structure can be used to analyze the content and to identify regions, which can be seen as network "communities" representing subdomains of the ontology. Furthermore, the analysis of the structure enables a first evaluation of the usability by allowing different views, so that existing ontologies can be easier comprehended by ontology engineers. This is very important because refactoring and reusing existing models assume that these models are understood.

In this regard, an adaptable structure-based ontology partitioning framework has been designed and implemented that utilizes community detection algorithms from the field of social network analysis. According to the motivation of the partitioning, the framework provides different configurable parameters. By this means the optimal solution for a certain motivation can be achieved. The proposed framework has been evaluated with a gold-standard approach for two concrete ontology partitioning cases. On the one hand, it was analyzed how term chunks from ontology documentation pages of thirteen ontologies can be reconstructed. On the other hand, it was investigated how the modules of four selected modular built ontologies can be reidentified.

For both cases, 480 different combinations of configurations have been applied on each ontology. The performance of the framework has been measured with F-Measure similarity function applied on the reference model and the produced partitions. This resulted in very good as well as very bad results. For that reason, the next problem was to define a strategy to select the best configuration for the partitioning process based

on the structure of the ontology and the motivation for partitioning. Two different approaches have been used in this regard. Firstly, the results with all ontologies and all configurations have been analyzed statistically. The values for the different parameters, which led to the best results, have been selected. Secondly, assuming that similar ontologies should be partitioned alike, each new ontology that should be partitioned has been compared to already partitioned ontologies with a distance metrics based on structural metrics. After the most similar ontology was identified, the configuration leading to the best results for the already known ontology has been applied on the new ontology.

With both approaches similar tools could be outperformed significantly, whereas the similarity based approach led to minimally better results than the statistic approach. The overall result is that for both reconstructing term chunks as well as modular ontologies the reference models could be reproduced up to sixty percent. Even though this value is twice as good as the performance of the similar tools, this does not justify a fully automatic approach for ontology partitioning. However, it could be demonstrated that with the proposed framework at least a semi-automatic approach for ontology partitioning can be realized, that creates an acceptable first result that should be refined manually.

ACKNOWLEDGMENTS

After years of work on this doctoral thesis, it is now time to come to an end. Pursuing a doctoral degree is really a tough endeavor. Therefore, I am deeply moved to write this section and express my sincere gratitude to all the people surrounded and supported me during this long run.

First of all, I am grateful to Prof. Dr. Robert Tolksdorf and Prof. Dr. Adrian Paschke for giving me the opportunity to be part of their team. Together with Prof. Dr. Heiner Stuckenschmidt, they motivated, challenged, and supported me in scientific discussions. Through their guidance and constructive critics, I felt impelled to keep on track, to work focused, to follow scientific rigor, and - last but not least - to become creative.

I also want to thank my former colleagues Mario Rothe, Markus Luczak-Rösch, Ralf Heese, Radoslaw Oldakowski, Olga Streibel, Ralph Schäfermeier, Sascha Todor, and Kia Teymourian for building a constructive and harmonic working environment and having a great time. Working with you for almost five years was really a gift and an honor.

Furthermore, I want to express a profound gratitude to my lovely family. I am grateful for having parents, who taught me to have deep respect for education and supported the natural curiosity that at the end caused me to become a scientist. I also thank my parents in law, my brother and his family as well as my brothers in law for making life uncomplicated and worth living.

Finally, I want to emphasize that I feel appreciation and gratitude more than I can articulate and even words can express for being married with my lovely wife Ebru and having two healthy and cute children, my son Yakub Emir and my daughter Sevde Yaren. You really make me happy and fill my life with joy!

TABLE OF CONTENTS

1 INTRODUCTION

Component-based development of large and complex software systems from small well-defined building blocks improves comprehension as well as management and leads to reusable software modules and a scalable overall system (Sametinger, 1997). Accordingly, designing ontologies in a modular way is expected to benefit from the same advantages (Stuckenschmidt & Klein, 2004). However, the status quo is that most publicly available ontologies are monolithic. For that reason, there are plenty of large and complex ontologies online available. Table 1-1 shows some size properties of popular ontologies.

Name	Number of classes	Number of properties	File size	Lines of code
FOAF[1]	19	67	44 KB	605
SIOC[2]	17	86	50.2 KB	851
DBPedia[3]	394	1,748	747.6 KB	4237
NCI Cancer[4]	27,652	70	34.4 MB	932,712
Gene Ontology[5]	52,904	41	103.9 MB	2,009,934

Table 1-1: Popular ontologies along with some size properties. (The number of classes and the number of properties have been extracted with Protege[6].)

The constantly growing number of online available ontologies for various domains (d'Aquin, Baldassarre, et al., 2007; Ding & Finin, 2006) increases the probability that reusable ontologies with acceptable quality are available. Since reusability is a crucial issue for the success of ontologies and especially for the Semantic Web, this is a highly desired trend. In fact, it is broadly accepted that efficient and effective reuse promises many benefits. Primarily, it enables saving necessary investment costs by avoiding the reconstruction of already existing ontologies. In order to understand the complexity of

[1] http://xmlns.com/foaf/spec/ last access April 25th 2012
2 http://rdfs.org/sioc/spec/ last access April 25th 2012
3 http://wiki.dbpedia.org/Downloads37 last access April 25th 2012
4 http://www.mindswap.org/2003/CancerOntology/nciOncology.owl last access April 25th 2012
5 http://www.geneontology.org/GO.downloads.ontology.shtml last access October 10th 2013
6 http://protege.stanford.edu/ last access April 25th 2012

ontology development and to assess the possible benefit of ontology reuse, it is essential to know that the importance of ontologies lies in the deep problem and domain analysis during the development process. According to Chandrasekaran et al., a "good" analysis clarifies the structure of the domain knowledge (Chandrasekaran, Josephson, & Benjamins, 1999). However, a good analysis, which is only one part of the overall ontology creation process, is a very cumbersome and time-consuming activity. Apart from saving investment costs, ontology reuse is expected to ensure a certain level of quality. The reason for this is that the longer an ontology exists and is reused, the more review processes it has gone through. Furthermore, in the context of the Semantic Web ontologies are considered the shared knowledge of distributed information systems (Bontas, Mochol, & Tolksdorf, 2005; Bontas & Mochol, 2005). In this regard, ontology reuse is also expected to support interoperability and system integration.

Due to these important advantages, ontology reuse is recommended in most ontology engineering methodologies. However, the initially described issue of monolithically created large and complex ontologies leads to substantial problems exacerbating ontology reuse. Even though most ontology engineering methodologies mention the reuse of existing ontologies as a possible starting point, none of them provide detailed descriptions about the reuse process. Especially the analysis of discovered candidate ontologies and the decision if and to which extend they can be reused are still open research questions. Since decision making is always a matter of knowledge, the main question is how candidate ontologies can be comprehended fully. For that reason, appropriate size and complexity reduction methods and techniques become crucial. This especially holds for large and complex ontologies, whose development requires high effort. Therefore, their reuse would lead to much more benefit than the reuse of simple and easy to create ontologies.

In order to support perception and to accelerate comprehension, it is important to visualize an ontology in an appropriate way (Dzbor & Motta, 2008). For ontologies with hundreds and thousands of concepts, it is impossible for the human mind to comprehend the whole content. In this regard, suitable support systems lower the burden to understand complex ontologies through reduction and projection techniques (Dzbor & Motta, 2008), e.g. by highlighting only relevant concepts or by reducing the complexity through partitioning. The latter is of particular interest because even the specification document of the Friend of a Friend (foaf) (Brickley & Miller, 2010) vocabulary, which is in comparison to the ontologies listed in Table 1-1 rather small, contains a grouping of the concepts (illustrated in Figure 1-1). Such a grouping is treated as a particular application for ontology partitioning. Similar term chunks have been used in the documentations of other ontologies like the Music Ontology (Raimond et al., 2012), the Atom Activity Streams Ontology (Minno & Palmisano, 2010), and the Semantic Web Conference Ontology (Möller, Bechhofer, & Heath, 2009). Since they are

all about the same size like FOAF, they emphasize how fast increasing size and complexity exceeds the cognitive abilities of humans and important an appropriate presentation is.

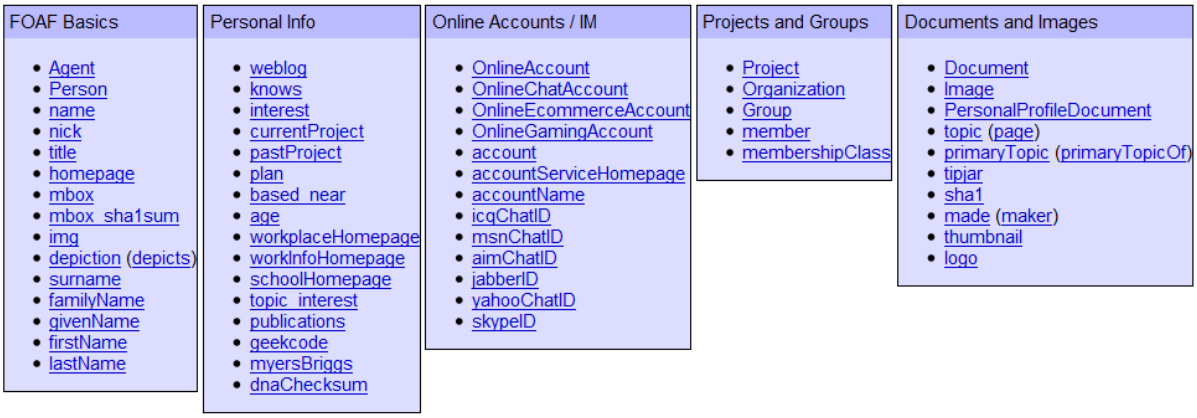


Figure 1-1: Term chunks of the FOAF vocabulary in the specification.

In addition to ontology reuse, processes like interlinking, refactoring, maintenance and management are also depending on comprehension and would benefit from size and complexity reduction as well. It must be kept in mind that each one of these processes has different demands, e.g. regarding the size and the number of partitions to be created. Thus, this work's objective is the realization of size and complexity reduction through an adaptable ontology partitioning framework. The targeted outcome is a support system that accelerates interlinking and reuse (e.g. through ontology documentation support) and simplifies maintenance (e.g. through refactoring and modularization support).

1.1 Approach and Contribution

A partitioning process to create building blocks from monolithic ontologies has to meet different requirements than a partitioning process for grouping of concepts into subdomains as in Figure 1-1. Apparently, different motivations for partitioning have different demands regarding the created partitions. Therefore, an ontology partitioning system must be either optimized only for one motivation or has to be adaptable. Since the former can be realized as a fixed configuration of the latter, this work tackles the

development of an adaptable ontology partitioning framework that can be configured for different use cases.

For the realization of such a framework, a structure-based approach and the utilization of network partitioning techniques have been chosen. The main rationale for this decision was the fact that there are various sophisticated methods and tools for network partitioning available. In this regard, considering ontologies as networks of concepts connected through properties, the adoption of network measures and network analysis techniques is a promising approach to analyze and partition ontologies. The assumption driving this work is that these techniques can be modified and applied on the structure of ontologies to create partitions.

During the realization of the framework, basic research questions have been tackled in depth. In the following, these questions are formulated along with brief descriptions of the provided solutions:

1. *Which goals do ontology partitioning processes have and what are the expectations of different goals on the partitioning process?*
   This question is discussed in Chapter 3 in depth, while the focus is primarily on two goals. Firstly, partitioning for grouping the concepts of an ontology for the documentation page. Secondly, partitioning to improve the maintainability.

2. *What is the structure of an ontology and how can it be represented as a graph?*
   In Section 5.2, the structure and its representation are discussed and two different possible graph-representations are described. The first one is the standard RDF graph and the second one is a more class-centric representation similar to entity-relationship-diagrams.

3. *How can community algorithms be applied on these graph representations of the structure and which parameters are possible to allow configuring the partitioning process?*
   In fact, the core contribution of this work is the answer to this question. The whole Chapter 4 describes the proposed framework. The basic notion for communities in social network is the Modularity function which is described in Section 5.4.1. In order to allow configurable partitioning, two approaches have been proposed which are novelties in the field of ontology partitioning. The first one is a weighting function based on the properties. It is described in Section 5.5. By this means, the semantic is taken into account during the partitioning process. The second one is an extension of the Modularity function with the size and the number of the created partitions. This is described in Section 5.6.

4. *How are the results of a partitioning process evaluated?*
   The evaluation of ontology partitioning is an ongoing research field. Even though there are some proposals, it will be an open issue for a while because even the field of ontology evaluation - without taking modularity into account - is still an open research question. (It is broadly accepted that the quality of an ontology depends on its application field. That means that ontology evaluation has to be done in the context of a concrete application.) Since the addressed problems in this work are neither ontology evaluation nor the evaluation of ontology partitioning, they have been bypassed by utilizing existing reference partitionings which are created by humans (mostly by the authors of the ontology). Considering these reference partitionings as gold standards, the problem of the evaluation has been solved by reducing it to the calculation of the similarity between the reference models and the framework's outcome. This is presented in detail in Section 6.1 and in Section 6.2.

The contribution of this work is an adaptable ontology partitioning framework that utilizes community detection algorithms from the field of social networks analysis. According to the goal of the partitioning, the framework provides different configurable parameters. During the work on this thesis, some parts of the contributions have been presented at workshops and conferences and have been published within the proceedings of those events. In the following these papers are listed:

- Coskun, G. 2008. Ontologiemodularisierung im Unternehmenskontext Einleitung Anforderungen an Ontologieentwicklung im Unternehmenskontext Ontologie-modularisierung im Unternehmenskontext. (Berlin, 2008).

- Coskun, G. 2010. Structure-based Analysis and Modularization of Ontologies. Doctoral Consortium of the Future Internet Symposium 2010 (Berlin, 2010).

- Coskun, G., Heese, R., Luczak-Rösch, M., Oldakowski, R., Paschke, A., Schäfermeier, R. and Streibel, O. 2009. Corporate Semantic Web Towards Deployment of Semantic Technologies in Enterprises. The Second Canadian Semantic Web Working Symposium (CSWWS 2009) (Kellowna, BC, Canada, 2009).

- Coskun, G., Heese, R., Luczak-Rösch, M., Oldakowski, R., Paschke, A., Schäfermeier, R. and Streibel, O. 2009. Towards a Corporate Semantic Web.

International Conference on Semantic Systems (I-SEMANTICS ’09) (Graz, Austria, 2009).

- Coskun, G., Luczak-Rösch, M., Heese, R. and Paschke, A. 2009. Applying Ontology Modularization for Corporate Ontology Engineering. International Conference on Semantic Systems (I-SEMANTICS ’09) (Graz, Austria, 2009).

- Coskun, G., Luczak-Rösch, M., Heese, R. and Paschke, A. 2009. Survey of Ontology Modularization for Corporate Ontology Engineering. The Second Canadian Semantic Web Working Symposium (Kellowna, BC, Canada, 2009).

- Coskun, G., Rothe, M. and Paschke, A. 2012. Ontology Content "At A Glance". 7th International Conference on Formal Ontology in Information Systems (Graz, Austria, 2012), 147–159.

- Coskun, G., Rothe, M., Teymourian, K. and Paschke, A. 2011. Applying Community Detection Algorithms on Ontologies for Indentifying Concept Groups. 5th International Workshop on Modular Ontologies (Ljubljana, Slovenia, 2011).

- Heese, R., Coskun, G., Luczak-Rösch, M., Oldakowski, R., Paschke, A., Schäfermeier, R. and Streibel, O. 2010. Corporate Semantic Web – Semantische Technologien in Unternehmen. Datenbank-Spektrum. 10, 2 (Aug. 2010), 73–79.

- Paschke, A., Coskun, G., Heese, R., Luczak-Rösch, M., Oldakowski, R., Schäfermeier, R. and Streibel, O. 2010. Corporate Semantic Web: Towards the Deployment of Semantic Technologies in Enterprises. Canadian Semantic Web - Technologies and Applications. W. Du and F. Ensan, eds. Springer US. 105–131.

- Teymourian, K., Coskun, G. and Paschke, A. 2010. Modular Upper-Level Ontologies for Semantic Complex Event Processing. Proceeding of the 2010 conference on Modular Ontologies Proceedings of the Fourth International Workshop WoMO 2010 (2010), 81–93.

1.2 Methodology and Thesis Outline

Research in Information Systems can be categorized into two paradigms (Hevner, March, Park, & Ram, 2004). The first one is called Behavioral Science Paradigm and corresponds to research in natural sciences, which means that it comprises knowledge-producing activities (March & Smith, 1995).

> *"It seeks to develop and justify theories (i.e., principles and laws) that explain or predict organizational and human phenomena surrounding the analysis, design, implementation, management, and use of information systems."* (Hevner et al., 2004).

The second one is called Design Science Paradigm and corresponds to an engineering approach (Hevner et al., 2004). It comprises knowledge-using activities to solve problems and to improve information systems (March & Smith, 1995). Thus, it is sometimes referred to as Improvement Research. The Design Science Paradigm is accepted to be the best suitable paradigm for this work. It was used to structure the main research activities and narratives of this thesis. Takeda et al. propose a design cycle, which describes the different steps of a design processes (Takeda, Veerkamp, Tomiyama, & Yoshikawam, 1990). It is illustrated in Figure 1-2.

The first step is the decision on a concrete problem from a set of problems that is to be solved. From the knowledge about the problem, a solution is inferred through abduction and a suggestion is defined. In the next step, the suggested solution is developed. During the development phase, different new problems might occur, which then cause the creation of new cycles. Within the evaluation phase the performance of the developed solution is measured and the achieved improvement is quantified. Finally, in the conclusion phase the insights are analyzed and new problems as well as further investigations are discussed.

The structure of this thesis is based on this design cycle. The first four chapters enable the reader to understand the general motivation of this work (Chapter 1), the fundamentals of the problem area (Chapter 2), the problem details in depth (Chapter 3), and finally existing solution attempts (Chapter 4). The suggestion and the development are then presented in Chapter 5, where low-level problems which came up during the development phase and are caused by the approach are discussed. In Chapter 6 the analysis of the framework's performance is presented with respect to the aforementioned concrete motivations for partitioning ontologies. The main focus is on the influence of the different parameters and the role of the Modularity function. The comparison of the proposed framework with existing solutions is described in Chapter 7.

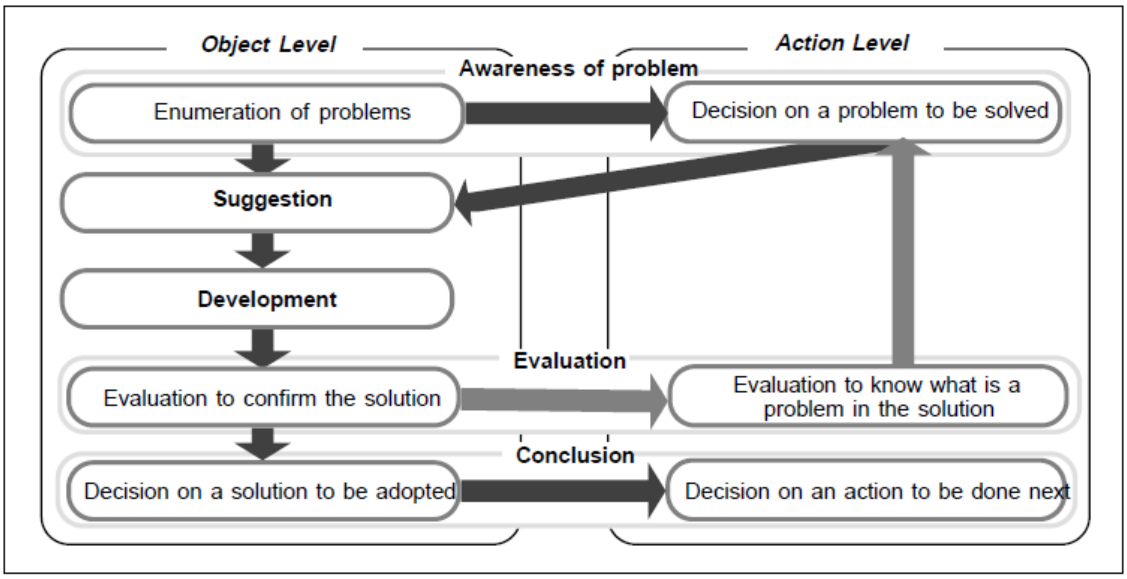


Figure 1-2: Design cycle for structuring the research activities according to the Design Science Paradigm. It has been applied in this research and used to structure the narratives of this document.

Through this comparison, the contribution of this work to the problem area becomes clear. Finally, a summary and an outlook are presented in Chapter 8.

For the sake of convenience for the reader and to offer the highest possible clarity about the structure of this document, each chapter starts with a brief abstract and closes with a short chapter summary. The former clarifies the structure as well as the motivation of the chapter whereas the latter repeats the most important chapter insights. Due to their general view, the first and last chapters do not follow this structure.

2 FUNDAMENTALS

In this chapter the fundamentals of this work will be presented. Only with a clear understanding about the fundamentals it is possible to follow the ideas and understand the outcome. In this regard, Section 2.1 starts with an in-depth analysis of the term ontologies and its understanding in Semantic Web. In contrast to the field of Artificial Intelligence (AI), where the term ontology was firstly adopted, Semantic Web brought new challenges. While the focus in AI was mainly on the expressivity and support for sophisticated reasoning processes, the Web environment demands more agility, scalability, and collaborative methods due to its open, large-scale, volatile, and participatory nature.

In Section 2.2 Ontology Engineering is introduced, which is the main research area this work pursues to contribute. The most important methodologies from this field are described focusing on ontology reuse and maintenance. Thereby, the trend from heavy-weight methodologies towards more agile methodologies allowing collaboration and flexibility is discussed.

A summary about the content of this chapter is provided in Section 2.3. Finally, as a completion for the fundamentals and for the sake of clarity, a terminology section has been added to the appendix of this document that can be seen as a part of this chapter. It provides an overview about the most important concepts of this thesis with their definitions.

2.1 Ontologies in Semantic Web

The vision of Semantic Web is to extend the World Wide Web by bringing structure to the meaningful content, so that it allows computers and people to better work in cooperation. The basic technologies for this vision, which are illustrated as a layer stack in Figure 2-1, are the Extensible Markup Language (XML), Resource Description Framework (RDF) and Ontologies. XML is an easy to understand and human-legible language which is used for bringing structure to documents by enabling users to create their own tags. RDF allows expressing meaning through triples following the form of subject-predicate-object statements. By using Uniform Resource Identifiers (URI) RDF makes sure that used concepts are tied to unique definitions accessible on the Web. Finally, an ontology - defined as a specification of a representational vocabulary for a shared domain of discourse (Gruber, 1993) - is a formal definition of concepts and their relations. Comprising taxonomy and inference rules, well designed ontologies enable powerful knowledge expression in a machine readable way, which in turn is the main goal of the Semantic Web vision. In this regard the success of Semantic Web depends significantly on the success of ontologies.

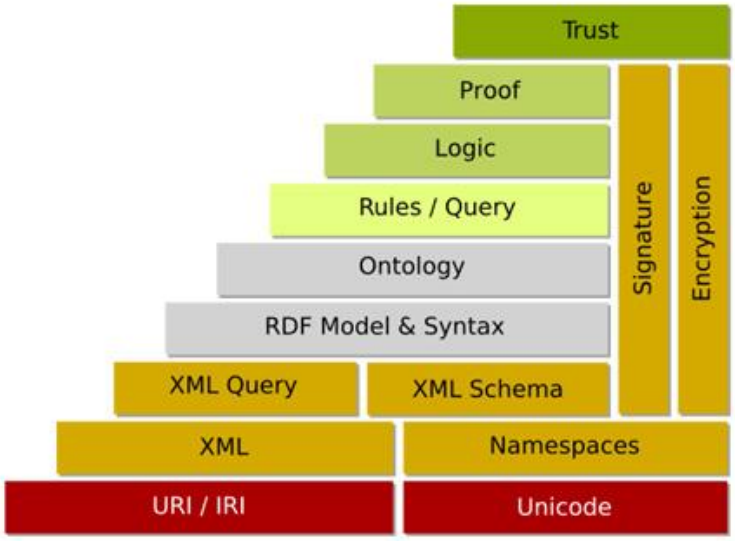


Figure 2-1: Visualizing of the layers of the Semantic Web.(The picture was taken from [7])

[7] http://www.semanticgroup.org/ last access October 11th 2013

In contrast to XML and RDF the concept ontology is a very abstract one. The term ontology has its origin in philosophy, where it is defined as the systematic study of being as such, that means each entity in the universe. Its goal is to describe the terms of entity classification. Due to its descriptive nature, computer scientists in the field of Artificial Intelligence (AI) adopted this term. They used it to refer to the basic models of their software systems, which form an abstract description of a part of the world. (These models are mostly referred to as the knowledge base of the software system.) However, while the philosophical understanding of this term is widely acknowledged, there are different views on ontologies in computer science. According to the online summit "Ontology, Taxonomy, Folksonomy: Understanding the Distinctions" from 2007 the spectrum of artifacts that are called "ontologies" covers folksonomies, taxonomies, thesauri, conceptual models, and formal logic-based models. A clarification of the different views on the term ontology is necessary.

Poli and Obrst distinguish between the philosophical perspective and the computer science perspective (Poli & Obrst, 2010). While the former is called ontology as categorical analysis, the latter is called ontology as technology. In contrast, the ontology community distinguishes between lightweight ontologies, which are mainly taxonomies, from heavyweight ontologies, which model the domain in a deeper way and provide more restrictions on domain semantics (Gomez-Perez, Fernandez-Lopez, & Corcho, 2004). Initially, basic logic formalisms were considered for modeling heavyweight ontologies with powerful expressivity. In AI first-order logic languages as Ontolingua (Gruber, 1993) and description logic languages as KL-ONE (Brachman & Schmolze, 1985) and Loom[8] were developed. During the Semantic Web initiative different mark-up languages were defined for representing and expressing ontologies. While the Web Ontology Language (OWL) and the RDF Schema (RDFS) are broadly accepted by the Semantic Web community as ontology representation languages, methodologies how to create and maintain ontologies are still objects of investigation.

2.1.1 Need for Ontologies

The content of the Web is growing continuously since its invention. Through a significant shift of its perception, which is called Web 2.0, this growth got accelerated even more. This shift made every Web user to a potential content producer by using tools like blogs, wikis, social networks, multimedia sharing platforms etc. The apparent ubiquity of the Web, which nowadays gets an additional boost by modern mobile devices, made it an important part of everyday life. Companies as well as government departments understood how important the Web is to get in touch with the customers.

[8] http://www.isi.edu/isd/LOOM/ last access July 16th 2012

Online product catalogues, product comparison sites, online encyclopedias and much more Web-based services demonstrated the importance of public data and its simple accessibility.

The experience with the success of freely accessible data leads to high expectations of appropriate combination and integration of this data. The Web provides numerous data sources from very different domains (e.g. health care, movie database, geolocations, and weather information). To gain added value from these information sources, appropriate integration techniques are the key factor of success. In this regard, ontologies have become necessary to support the integration and management of Information Systems, which become large and complex (Di Maio, 2009). One essential benefit is the support of knowledge reuse and communication. Even very small ontologies - when well-formed and properly grounded - can be reused, linked to, referenced, and incorporated or at least intersected into larger ontologies (Di Maio, 2009).

2.1.2 Linked Data

The vision of the Semantic Web (Berners-Lee, Hendler, & Lassila, 2001) published in 2001 depicted the future of the World Wide Web. Being a vision, it described potential benefits of existing technologies on an abstract level. After short time, it was clear that it was necessary to define a first step for the progress towards the vision. A new paradigm came up called Linked Data which is a collection of best practices to publish data and enable the first step of the Semantic Web from document-oriented web of to the Web of Data. This paradigm consists mainly of the following four principles (Berners-Lee, 2006):

1. Use URIs as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, RDFS, SPARQL)
4. Include links to other URIs so that they can discover more things.

The most important and at the same time the most difficult principle to implement is the fourth one (Auer, Lehmann, & Ngonga Ngomo, 2011). In order to interlink a data source with another data source it is necessary to discover a suitable candidate data source which contains similar content and to understand its structure. While the first part of this process is supported by online ontology libraries as Ontolingua and OntoSelect and search engines as Swoogle[9], Watson[10] and Ontosearch[11] the second part

[9] http://swoogle.umbc.edu last access July 16th 2012

is still a critical open issue. (There are more than 26 billion of triples in the Linked Open Data cloud, which is illustrated in Figure 2-2, but the interlinking is still less than 5\% [12].) Recent research activities in this field are mainly focused on automatic linking of entities in different knowledge bases. That means that individuals in different sources are analyzed and links between them are discovered (Volz, Bizer, Gaedke, & Kobilarov, 2009).

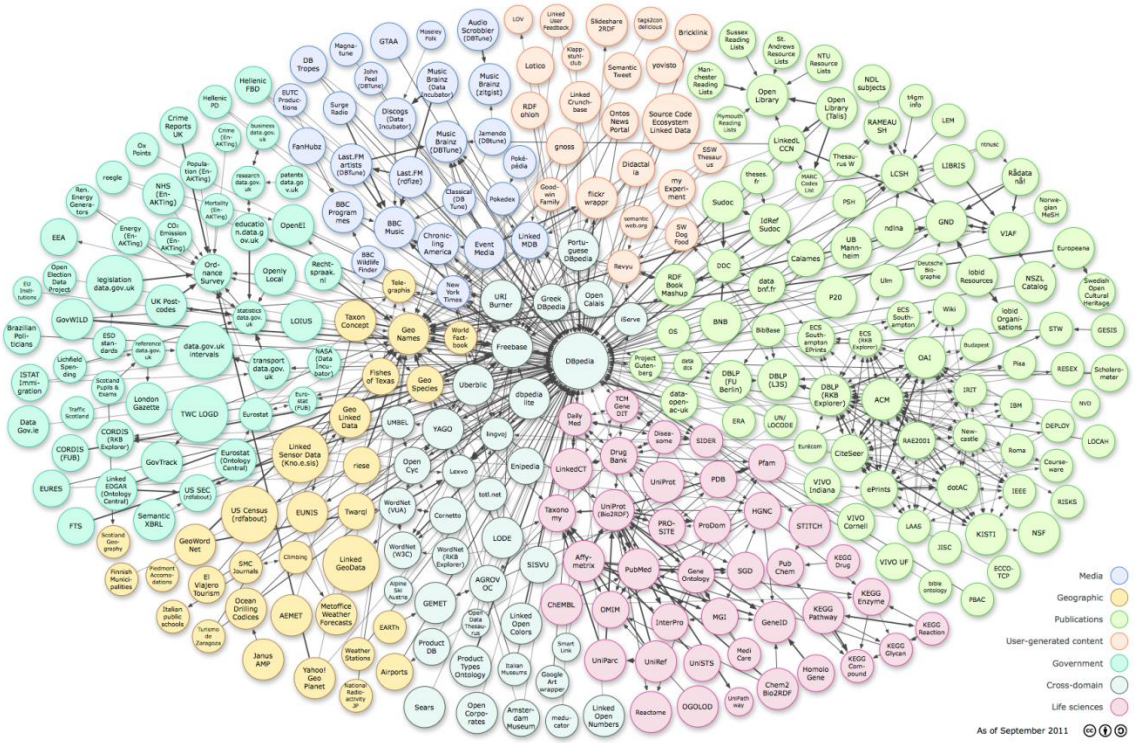


Figure 2-2: Visualization of the Linked Open Data cloud[13] as of September 2011.

The pragmatic approach of Linked Data leads to a more Web-centric view on ontologies than a semantic-centric view. That means that ontologies are no longer expected to be semantically enriched and expressive models. In Linked Data ontologies are considered as published schemas represented in RDF, RDFS or OWL. That does not

[10] http://watson.kmi.open.ac.uk/WatsonWUI last access July 16th 2012
[11] http://www.ontosearch.org last access July 16th 2012
[12] http://lod-cloud.net last access July 16th 2012
[13] Figure taken from http://lod-cloud.net/ (last access February 28th 2013)

mean, that heavyweight semantics is now allowed, but it is not a requirement. This is a contrast to the understanding of ontologies in the initial vision of the Semantic Web. This main rationale for this is the commonly accepted consensus that semantic technologies are not mature enough at this stage to be really useful in productive settings (Roberta, Alexandre, Vincent, & Carlo, 2007). Therefore, Linked Data focuses on the publishing of raw data on the global Web by postponing the semantic aspect to the future.

2.1.3 Definitions

Due to different understandings of the term ontology within the Computer Science community, it is important to clarify the meaning of this term in the context of this work. This is necessary to avoid misunderstandings and ambiguity in the rest of this document.

According to Di Maio ontologies are conceptual and semantic models devised to support various intelligent functions including information representation, processing and retrieval in network-supported and Web-based environments (Di Maio, 2009). Since this work is done in the broader field of the Semantic Web research activities and especially with the pragmatic view of the Linked Data initiative, the understanding of the term ontology is as follows:

> *Each publicly available vocabulary represented in RDF, RDFS or OWL is in this work treated as an ontology.*

However, as this definition comprises a broader spectrum, some distinctions are necessary to classify the different types of ontologies. One distinction is based on the expressivity level. Ontologies which comprise just concepts, concept taxonomies, relations between concepts and properties are called light-weight ontologies. On the other hand heavy-weight ontologies comprise axioms and constraints in addition to the aforementioned aspects. Furthermore, a second classification can be done based on the scope of the ontology. (Gomez-Perez et al., 2004) organizes ontologies into the following categories:

- **Top Level / Upper Level Ontologies** are generic models and applicable in various domains. They define general concepts and relationships between them and are standardized by international standardization committees (e.g. Suggested Upper

Merged Ontology [14] (SUMO) by the IEEE Standard Upper Ontology Working Group[15]).

- **Domain Ontologies** are domain-specific models, which describe the knowledge of a particular domain (e.g. electronic, life science, financial) and are being developed by communities of domain experts (e.g. the Gene Ontology by the Gene Ontology Consortium[16]).

- **Task Ontologies** in contrast to domain ontologies, task ontologies are models which focus on a particular task like scheduling. They define concepts and relations describing tasks needed for solving concrete problems.

- **Domain-Task Ontologies** are task ontologies which are made for one domain and are only applicable in this domain.

- **Method Ontologies** are specifying concepts and relations about methods which are necessary to do a task.

- **Application Ontologies** are concrete and application-dependent models which are made for a particular application to solve concrete problems.

Figure 2-3 illustrates an aspect of the categorization of ontologies by comparing scope and abstraction based on three selected categories. It shows that the number of ontologies within an abstraction level increases with getting more concrete. This is caused by the generality of abstract ontologies, which make them applicable across several domains and lead to more reusability.

2.2 Ontology Engineering

In order to provide some structural guidance for the ontology creation process some Ontology Engineering (OE) methodologies have been proposed. Ontology Engineering or sometime referred to as Ontological Engineering *"refers to the set of activities that*

[14] http://suo.ieee.org/SUO/SUMO/index.html last access April 29th 2012
[15] http://suo.ieee.org last access April 29th 2012
[16] http://www.geneontology.org last access April 29th 2012

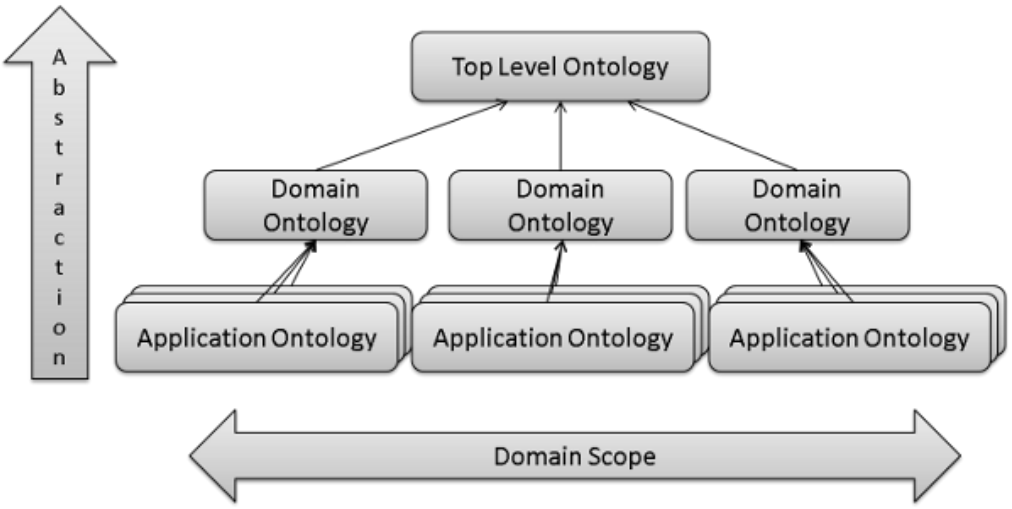


Figure 2-3: Comparison of the abstraction and the scope of ontologies. The more abstract an ontology is, the less alternatives it has.

concern the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them" (Gomez-Perez et al., 2004). A methodology is defined as *"a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed*". There are already a lot of existing methodologies in literature. According to Di Maio "*the main purpose and goal of adopting such structured approaches to systems development are*

- *to make the development process of complex systems more manageable and to increase the effectiveness of the communication between analysis, designers, and programmers (which is supposed to reduce the risk of errors an flaws)*

- *to maximize the chances that the system under development actually meets the business requirements by providing a means of mapping and translating requirements directly into systems functionality.*'' (Di Maio, 2011)

In this regard ontology engineering *"is a specialized set of activities requiring depth and breadth of understanding, knowledge, and skills"* (Di Maio, 2011). The overall ontology development process consists of three different activity groups (Gomez-Perez et al., 2004). To the first kind of activities belong management activities like scheduling of tasks to be performed, control of planed actions, and quality assurance. The second kind of activities is development oriented activities, which in turn can be divided into three subgroups: pre-development activities like environment and feasibility study;

development activities like specification, conceptualization, formalization, and implementations; post-development activities like maintenance and (re)use. Finally, support activities are the third kind of activities which are knowledge acquisition, evaluation, integration, documentation, merging, alignment, and configuration management.

The range of ontology engineering methodologies widened during the last years mostly inspired by the knowledge engineering and software engineering disciplines and often evolved from each other. Therefore approaches differ in their relationship to software engineering and knowledge engineering, in details referring to the composition of ontology engineering and application development, the range of users interacting in ontology engineering tasks, and the degree of lifecycle support.

One important difference between the existing methodologies is the degree of details regarding the specification. Some lightweight methodologies can be seen as an outline, while heavyweight methodologies range from the knowledge level to the representation formalism used. As this work has its scope on treating the complexity regarding ontology reuse and maintenance, this section presents only a selected part of the state-of-the-art in ontology engineering methodologies ranging from two heavy-weight methodologies (METHONTOLOGY and On-To-Knowledge) to two collaborative methodologies (DILIGENT and the NeOn Methodology) and two light-weight methodologies (COLM and *"Just Enough Ontology Engineering"*). The selection is supposed to demonstrate the relevance of maintenance and reuse for the whole spectrum of ontology engineering methodologies. For an exhaustive description and comparison refer to (Gomez-Perez et al., 2004)}.

2.2.1 METHONTOLOGY

METHONTOLOGY (Fernandez, Gomez-Perez, & Juristo, 1997) can be considered as the most comprehensive Ontology Engineering methodology in literature. In fact, the aforementioned distinction of the overall ontology development process into three activity groups is based on METHONTOLOGY. Having its root in intelligent agent technologies, it has an AI perspective on ontologies. Therefore, it targets the construction of ontologies at the knowledge level. It is an approach to build ontologies from scratch, reuse or re-engineer existing ones. The central structure of its processes have been adopted from the IEEE standard for Software Engineering, which was assumed to be more mature than any new developed process for this special purpose. That yields to the lifecycle illustrated in Figure 2-4. It comprises three central activities of management (scheduling / planing, control and quality assurance), five activities of development (specification, conceptualization, formalization, implementation, and maintenance), and five activities of support (knowledge acquisition, integration,

evaluation, documentation, and configuration management). These activities underrun a cyclic lifecycle which allows for the iterative release of evolving ontology prototypes.

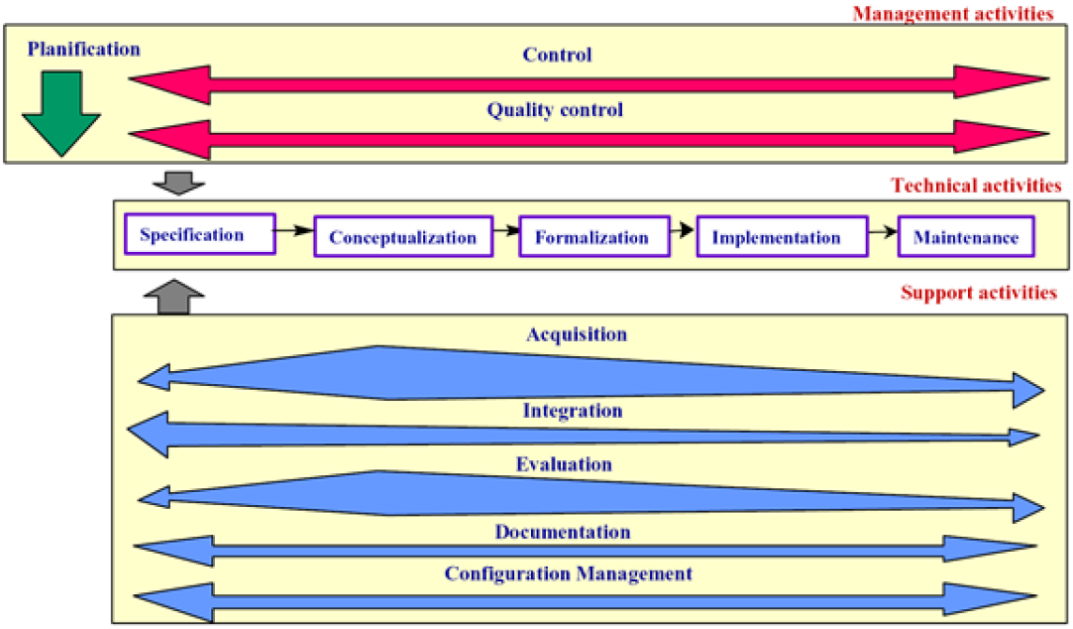


Figure 2-4: The development process of METHONTOLOGY comprises three central activities of management, five activities of development, and five activities of support (Corcho, Fernández-López, Gómez-Pérez, & López-Cima, 2003).

Regarding documentation METHONTOLOGY proposes to create a requirements specification document after specification, a knowledge acquisition document after knowledge acquisition, a conceptual model document after conceptualization, a formalization document after formalization, an integration document after integration, an implementation document after implementation, and finally an evaluation document after evaluation. But it is not described how these documents should be created and how they should look like.

The reuse process is divided into the following four steps:

1. find candidate ontologies
2. the content and the granularity
3. select the ontologies to be reused
4. evaluate the selected ontologies

As mentioned before, METHONTOLOGY does not have its roots in Semantic Web, but in Agent Technologies. For that reason, it addresses knowledge workers and experienced ontology engineers and it assumes the developers to be familiar with ontologies in general. Therefore, it is not described in detail how these steps (for this work especially steps 2. and 3. are very relevant) should be executed. Additionally, it is not described how the proposed documents should support the reuse and re-engineering processes.

2.2.2 On-To-Knowledge

The On-To-Knowledge Methodology (OTKM) proposes a more application-oriented ontology development process than METHONTOLOGY (Staab, Studer, Schnurr, & Sure, 2001). Its overall goal is to build a knowledge management system for enterprise systems^, which enables the integration of knowledge from different knowledge sources. Ontologies are considered to be the means to switch from a document-oriented view to a knowledge-oriented view. In this regard, the ontology development process is seen as a part of an "overarching methodology for introducing knowledge management systems" (Staab et al., 2001). Figure 2-5 illustrates the proposed process.

In OTKM the actual development of the ontology begins in the kickoff phase, when the outcome of the feasibility study justifies it. In the kickoff phase, it is mentioned that ontologies might be reused, but it is not described in detail how this should be done. In the following phase, the refinement phase, it is mentioned that appropriate visualization of the ontology content (e.g. as a mindmap) might be helpful for the ontology engineer. Furthermore, it is stated that during the refinement each step should be documented without explaining how to do it.

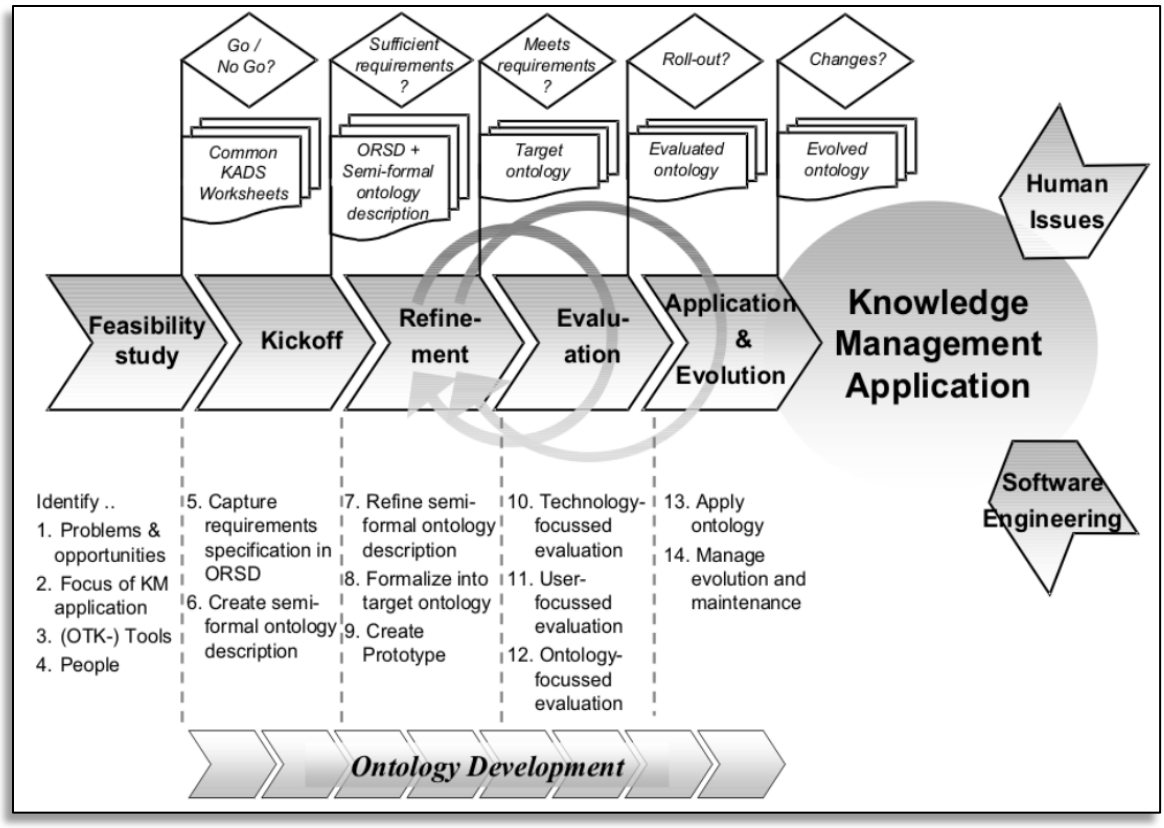


Figure 2-5: Abstract overview of the On-To-Knowledge Process (Sure, Staab, & Studer, 2009)

2.2.3 Diligent

METHONTOLOGY as well as OTKM are heavyweight methodologies for creating ontologies for knowledge-based systems in a centralized manner. With the Semantic Web vision and the Linked Data initiative ontologies became an enabler for information integration in large-scale and highly distributed environments like the World Wide Web. Additionally, the Web 2.0 trend eliminated the separation between the role of the information producer and the role of the information consumers towards a more participatory and collaborative Web.

Thus, the demands on a methodology for the development of ontologies in the sense of Semantic Web and Linked Data were different. This led to a new methodology called DILIGENT (Pinto, Staab, & Tempich, 2004) which stands for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. Figure 2-6 illustrates the interactions of different participants in DILIGENT, which respects the large distribution of ontology

engineering in Web-scaled settings and the totally disparate skill level of process stakeholders.

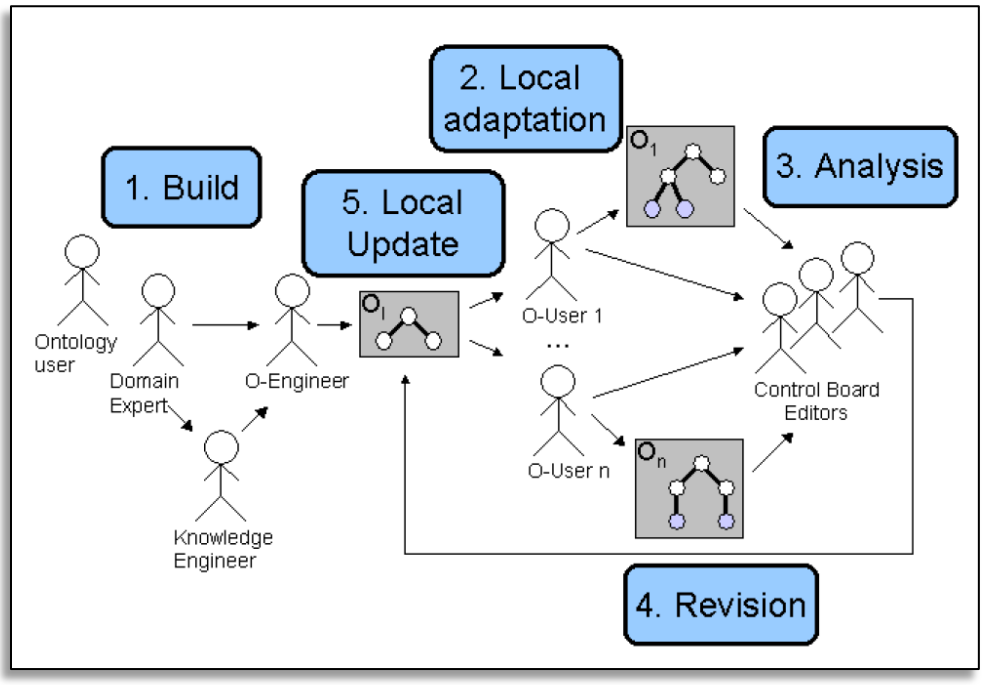


Figure 2-6: The DILIGENT setting of roles and functions (Pinto et al., 2004)

An ontology consensus is reached by an argumentation-based approach following a dedicated argumentation model (the DILIGENT argumentation ontology). Every individual is free in adapting the central ontology consensus locally and modify this adaption for its own purposes. A so-called control board analyzes the local modifications and adapts the shared central ontologies accordingly. The evolution of the consensual model is depending on these local adaptions. A lifecycle is underrun in DILIGENT, which enables an iterative evolution of the central consensual ontology while the detailed process phases (build, local adaption, analysis, revision, and local update) concentrate on reaching this human-centered consensus by argumentation about concept.

The focus in DILIGENT is the argumentation-based coordination of the evolution of the ontology. By assuming a simple core ontology at the beginning of the process created by a small group it implicitly mentions modularity (core ontology and its extensions) without providing any further details. It is also not described how the control board is supposed to do the analysis of the local modifications and how to maintain the ontology according the outcome from the analysis.

2.2.4 NeOn Methodology

One of the most comprehensive and promising methodologies is the so-called NeOn methodology (Suárez-Figueroa et al., 2008). It was created within the context of a research project with 14 European contributors which run for several years. It was driven by the assumption that former methodologies provide detailed process description but lack the appropriate "style and granularity" as it is known from software engineering methodologies. The NeOn methodology attends to facilitate guidelines for building individual ontologies by reuse and re-engineering of other domain ontologies or knowledge resources and for plugging in continuously evolving ontologies. The following nine approaches for ontology creation are described, which are called scenarios:

1. Building ontology networks from scratch without reusing existing knowledge resources.
2. Building ontology networks by reusing and reengineering non ontological resources.
3. Building ontology networks by reusing ontological resources.
4. Building ontology networks by reusing and reengineering ontological resources.
5. Building ontology networks by reusing and merging ontological resources.
6. Building ontology networks by reusing, merging and reengineering ontological resources.
7. Building ontology networks by reusing ontology design patterns.
8. Building ontology networks by restructuring ontological resources.
9. Building ontology networks by localizing ontological resources.

Figure 2-7 illustrates these different approaches in association with processes and activities. The addressed target groups of the NeOn methodology are software developers as well as ontology practitioners which should be enabled to build ontology networks by use of ontology building platforms (e.g. NeOn Toolkit, Protege, or TopBraid Composer). The definition of process phases and activities is accompanied by a description of purposes, inputs and outputs, involved actors, methods, techniques, and tools used for their execution.

Obviously, the NeOn methodology has a strong focus on reuse. Six different criteria for reuse are defined, which mainly consist of the estimation of the necessary effort. However, this demands for a first analysis of different aspects of the ontology. Therefore, it becomes clear that the focus is mainly on very large ontologies, since it is implicitly assumed that there are enough resources to do the necessary analysis and the mentioned estimation.

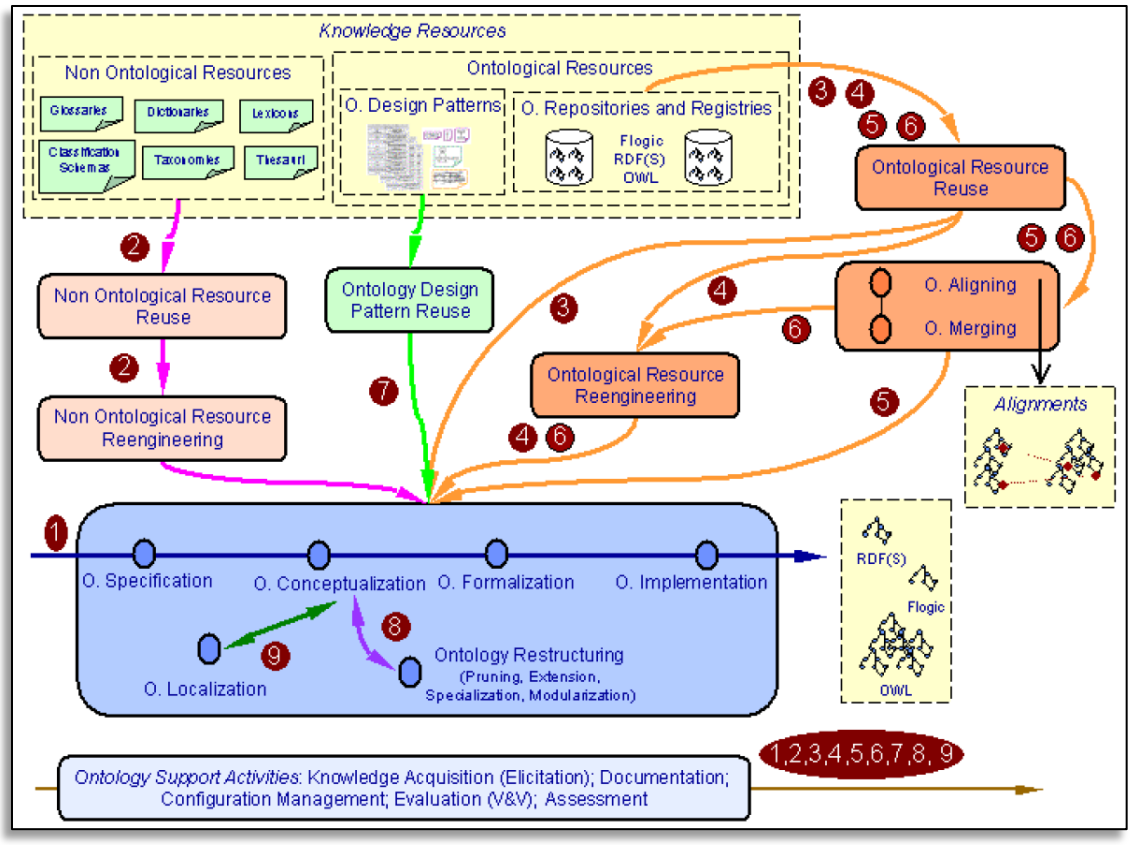


Figure 2-7: Ontology development processes in the NeOn methodology (Suárez-Figueroa et al., 2008)

2.2.5 Just Enough Ontology Engineering

The "Just Enough" approach was introduced by Ed Yourdon in (Yourdon, 2006),which he commented as follows:

> "*Today, we're too busy to spend much time thinking about anything, and we're also far too busy to read more than a couple hundred pages of the bare essentials on any topic. What we want is just enough – enough to give us the basic idea, enough to get us started, enough to give us a grounding in the fundamentals.*"[17]

[17] http://yourdon.com/publications/ last access on July 19th 2012

This inspired Di Maio to adapt the “just enough” approach to Ontology Engineering. The Just Enough Ontology Engineering (JEOE) methodology (Di Maio, 2009) describes the essential activities for Ontology Engineering without describing in detail a sequence to perform. The following activities are discussed

1. Identifying Stakeholders
2. Defining the Purpose / Goal of the Ontology
3. Outlining Requirements
4. Identifying and Surveying Existing Knowledge Sources
5. Scoping the Ontology
6. Evaluation and Testing
7. Definition
8. Implementation
9. Deploying
10. Testing and Validation
11. Publishing
12. Maintenance and Reuse

In most of these aspects (especially in 4. and 12) ontology reuse and interlinking is mentioned as an important feature of ontologies. Di Maio states in (Di Maio, 2009)

> “*Although it may not always be possible to reuse an existing ontology, it should always be possible to reuse at least some parts of it - at least the parts of it that are public, declared, explicit, or easily acquired. Other ontologies, even when substantially different from the one we are trying to develop, should always be acknowledged and referenced, at least for completeness*.”

At some parts Di Maio mentions an ontology specification document but does not get clear how this should be created and structured. Such a document is an essential requirement for efficient reuse, understandability and continuous refinement of the ontology. While understandability is one top level requirements for the accessibility if ontologies by different stakeholders, continuous refinement is an important claim of JEOE, because it assumes an ongoing incremental development method to adapt the ontology to the changing reality or the view it is based on.

2.2.6 Corporate Ontology Lifecycle Methodology

The Corporate Ontology Lifecycle Methodology (COLM) illustrated in Figure 2-8 is an innovative lifecycle model for continuously evolving ontologies in corporate contexts (Luczak-Rösch & Heese, 2008). The main intention is to provide an intuitive understanding of raising costs per iteration and of the duration and effort spent in each process phase.

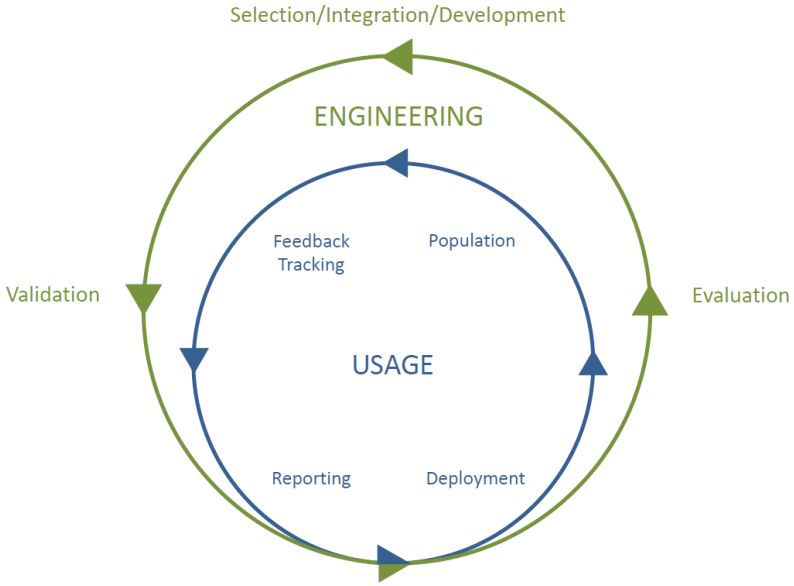


Figure 2-8: Corporate Ontology Lifecycle Methodology

The seven phases of the two-part cycle refer either to the outer cycle as selection/development/integration, validation, evaluation, or to the inner circle as deployment, population, feedback tracking, and reporting. The outer cycle represents pure engineering tasks, which is an expert-oriented environmental process. The inner constitutes the ontology usage, which is a human-centered concurrent process.

Starting the process at selection/development/integration means to start the knowledge acquisition and conceptualization, to re-use or re-engineer existing ontologies, or to commission a contractor to develop an ontology. The result of this phase is an ontology, which is validated against the objectives. At the intersection point between the engineering and the usage cycles the ontology engineers and the domain experts decide whether the ontology suites the requirements or not. If this is approved the ontology is deployed and in use by applications. Then the ontology is populated, which means that a process for instance generation from structured, semi-structured

and unstructured data runs up. Throughout the whole feedback tracking phase, formal statements about users' feedback and behavior are recorded. A reporting of this feedback log is performed at the end of the usage cycle. That means that all feedback information, which was collected until a decisive point, is analyzed respecting internal inconsistencies and their effects to the currently used ontology version. The usage cycle is left and the knowledge engineers evaluate the weaknesses of the current ontology with respect to the feedback log. This point may also be reached, when the validation shows that the new ontology is inappropriate to the specification. The lifecycle starts again with the implementation of the results of the evaluation.

2.3 Chapter Summary

The importance of the World Wide Web for everyday life as a ubiquitous information source is increasing continuously. Likewise, the importance of ontologies (with its roots in the field of artificial intelligence) for the future of the WWW rises. This trend caused different ontology engineering methodologies to come up within a short period of time. They have different requirements and expectations and therefore differ in various aspects like their relationship to knowledge engineering or software engineering, their level of detail, their application dependency etc. However, presently, there is not a methodology which gains broader acceptance. Stakeholders, especially corporations, have a well-founded caution by the decision to use ontologies and to apply one methodology, because "undertaking an OE project can be a bigger risk than other IT projects simply because the complexities and uncertainties are higher." (Di Maio, 2009).

Existing OE methodologies lack of a sound balance between the pragmatics of cost-effective processes as demanded by corporations and the sophisticated theoretical basis for academics. It is important to focus on common aspects of various Ontology Engineering methodologies and to exhaustively work on basic processes and concrete techniques with keeping the mentioned pragmatics in view. This work is doing so by concentrating on modularity and documentation of ontologies for reuse and maintenance from a complexity reduction perspective. The targeted contribution is expected to be applicable within each OE methodology and provide techniques fulfilling the demands of corporate settings.

3 ONTOLOGY PARTITIONING

In this chapter the problem of structure-based partitioning ontologies is analyzed. Different aspects and dimensions of the research problem are discussed. Its main goal is to provide insights into the low level problems this work is faced with and to understand the requirements the targeted framework needs to fulfill.

Ontologies are complex information models possessing different aspects. Understanding ontologies completely means understanding all aspects of them and the relations among them. Therefore, it is analyzed which aspects ontologies have before it is analyzed how ontologies can be decomposed into different partitions with respect to these aspects. For that reason this chapter starts in Section 3.1 with a brief overview of the aspects of an ontology and how partitioning can be understood according to each aspect. Keeping these aspects in mind, the terms size and complexity are discussed in Section 3.2, because partitioning is a decomposition process for reducing both. It is emphasized that in order to quantify the performance of a partitioning process measures are required for both.

The next question to clarify is how concrete partitioning should be done. Two different motivations for partitioning namely reuse and maintenance are then discussed in the Sections 3.3 and 3.4, respectively. Both motivations demand different levels of knowledge about the ontologies and therefore have different requirements on the involved users, the used tools, the partitioning process as well as the identified partitions. After these sections, it becomes clear that a one-fits-all solution for ontology partitioning is not possible. Rather an adaptable framework is necessary which adapts towards the need of different motivations in terms of configurable parameters. Therefore, it is investigated which properties can be derived from existing examples, which can be potentially used as parameters in the adaptable partitioning framework. Finally this chapter is closed with a summary in Section 3.5.

3.1 Aspects of Ontologies

According to Vrandecic ontologies have six different aspects, namely the vocabulary, syntax, structure, semantic, representation, and context (Vrandecic, 2010). The meaning of the term ontology partitioning depends on the considered aspect. The following discussion should provide a brief insight how partitioning could be understood with respect to each aspect.

Vocabulary

The vocabulary is a glossary of terms, which are names representing different concepts, properties and instances of the domain to be modeled. Within the Semantic Web names are mostly URI references (e.g. http://xmlns.com/foaf/0.1/Person) and, which consist of a namespace (http://xmlns.com/foaf/0.1/) and a local name (Person). Through URIs these names are globally unique and can be looked up through HTTP. Partitioning a vocabulary would for example mean to create different glossaries, which in fact is a simple list of terms.

Syntax

The syntax represents the ontology in a serialized form, which enables saving it in a document. In Semantic Web the mostly used syntax is RDF/XML. Possible other syntactical representations are the N3 notation (Berners-Lee, 2005) and Turtle (Beckett & Berners-Lee, 2008). A partitioning of the syntactical representation can be considered as splitting up a file into different pieces (e.g. line-based).

Structure

The structure of ontologies is mainly made up of the relations between the terms of the vocabulary, which can be hierarchical relations or property relations. Partitioning the structure is a research problem, which is the primary focus of this work.

Semantic

Ontologies contain axioms and reasoning rules to express semantics. This is the distinguished aspect of ontologies, because this feature separates ontologies from data and information models. It allows for inference that makes implicit information explicit. Thus, ontologies are useful means to represent knowledge. In this regard, partitioning of the semantics can be considered as dividing the knowledge into different knowledge bases.

Representation

The representation of ontologies is defined as the relation between the semantics and the structure of ontologies (Vrandecic, 2010). There are different methods how the semantics can be represented structurally. This important issue is tackled in this work deeply in Section 5.2.

Context

The context is the setting in which the ontology was created and is used. This ranges from the ontology engineering methodology used to the concrete application environment in which the ontology is utilized. Therefore, it can be divided into different aspects like organizational aspects, methodological aspects, technological aspects, etc.

3.2 Size and Complexity

The terms size and complexity are not trivial in the context of ontologies. For each one of the mentioned ontology aspects different size and complexity measures are possible. Size can be e.g. defined as the amount of terms in the vocabulary, the number of lines of code in the syntactical representation, the diameter in the ontology structure, the number of axioms of the semantics. Although there are different possible definitions for size, most of them are simple to calculate and easy to understand.

Complexity, on the contrary, is not that easy to define as a measure. In (IEEE, 1990) complexity is defined as follows:

> *"The degree to which a system or component has a design or implementation that is difficult to understand and verify."*

Accordingly, a complexity measure needs to be able to calculate the mentioned degree. As size has an influence on the comprehensibility it can also be seen as an indirect complexity measure. However, there are other properties regarding the mentioned aspects, which have an influence on the comprehensibility of ontologies. Figure 3-1 exemplifies this by illustrating three different syntactical representations of a small ontology. Although each of them contains the same semantic content they provide different levels of readability and hence different levels of comprehensibility.

In terms of lines of code and the amount of symbols in a document, the N3 notation (part b) of Figure 3-1) could be identified as the "shortest" representation and therefore the best understandable one. But the prefix definition could confuse readers, who are not familiar with it. Furthermore, a “good” graphical visualization simplifies the

comprehension, because with mental images memorizing is easier than memorizing just syntactical representations (Barclay, 1973). Therefore, many readers could agree on the graphical visualization (third part of Figure 3-1) as the best comprehensible presentation of the ontology. However, long URIs enlarge the illustration and exacerbate the visual perception.

Well-defined size and complexity measures are necessary to compare the ontology at the beginning with the generated partitions after the partitioning process. For that reason, they are essential to measure the quality of partitioning processes. Additionally, these measures can be used to optimize the partitioning process with respect to them, if the expected partitions can be described in terms of size and complexity. Concrete measures on this are introduced and discussed in Section 5.3.

For the sake of clarity, it is important to mention at this point that in the field of computer science the term complexity is mainly used in the context of computational complexity which seeks to classify computational problems in terms of required resources (e.g. time and memory space) for the computation process. The definition of the complexity is based on properties of the input parameters (e.g. number of elements in case of a list). In this regard, size and complexity of ontologies have an effect on the computational processes applied on ontologies. That's why in OWL 2 three different profiles are defined, namely OWL 2 EL, OWL 2 QL, and OWL 2 RL.

> *"Each profile is defined as a syntactic restriction of the OWL 2 Structural Specification, i.e., as a subset of the structural elements that can be used in a conforming ontology, and each is more restrictive than OWL DL. Each of the profiles trades off different aspects of OWL's expressive power in return for different computational and/or implementational benefits. OWL 2 EL enables polynomial time algorithms for all the standard reasoning tasks; it is particularly suitable for applications where very large ontologies are needed, and where expressive power can be traded for performance guarantees. OWL 2 QL enables conjunctive queries to be answered in LogSpace (more precisely, AC^0) using standard relational database technology; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to access the data directly via relational queries (e.g., SQL). OWL 2 RL enables the implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples."*
> (Hitzler, Krötzsch, Parsia, Patel-Schneider, & Rudolph, 2009)

a)

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns="http://www.coskun.org/dissexample.rdf#">
  <foaf:Person rdf:about=
        "http://www.coskun.org/dissexample.rdf#gokhancoskun">
    <foaf:mbox rdf:resource="mailto:gokhan@coskun.org"/>
    <foaf:homepage
          rdf:resource="http://www.coskun.org/gokhan"/>
    <foaf:family_name>Coskun</foaf:family_name>
    <foaf:givenname>Gökhan</foaf:givenname>
  </foaf:Person>
</rdf:RDF>
```

b)

```
@prefix :     <http://www. coskun.org/dissexample.rdf#">#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

     :gokhancoskun a foaf:Person ;
          foaf:givenname "Gökhan" ;
          foaf:family_name "Coskun" ;
          foaf:homepage <http://www.coskun.org/gokhan> ;
          foaf:mbox <mailto:gokhan@coskun.org> .
```

c)

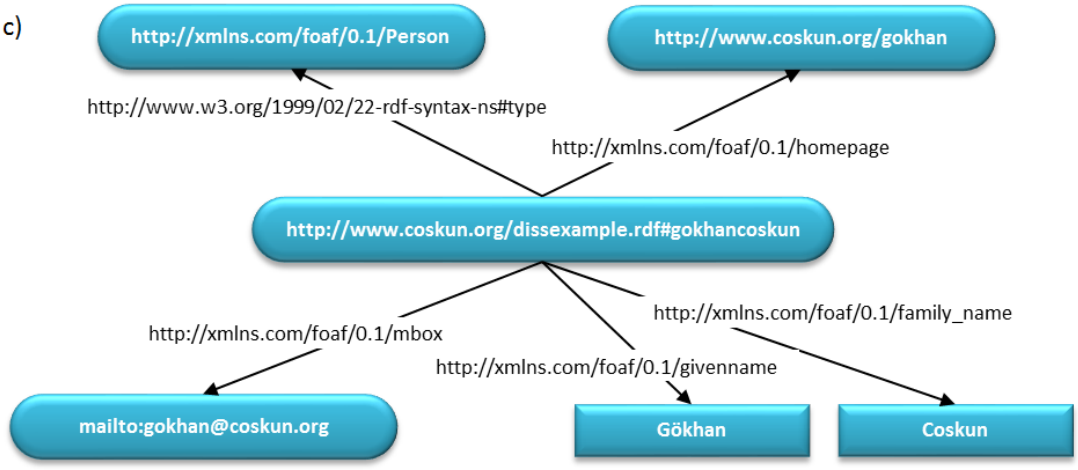


Figure 3-1: Different approaches to serialize an ontology, a) RDF/XML b) N3 Notation c) RDF Graph

However, as mentioned above this categorization is based on syntactic restrictions and therefore very simple to do by checking the used constructs within an ontology. This type of complexity is not suitable for the envisioned partitioning system.

3.3 Partitioning for Ontology Reuse

Development strategy in IT seeks for abstraction, encapsulation and reusability in various levels. This caused different paradigms like Object-oriented Programming, Agent-oriented Software Engineering, Aspect-oriented Programming and different technologies like middleware and application containers. The distinction between the program logic and the information model is suggested throughout these approaches. The reason for this is twofold. The first one is avoiding dependency between the model and the programming logic. The second one is to allow for reusable components. Based on the initially understanding of ontologies as semantically enriched information models of knowledge-based systems, reusability is an inherent feature of ontologies. According to (Dzbor & Motta, 2008) "*the reuse of existing, possibly imperfect, ontologies becomes the key engineering task*."

Different abstraction levels for ontologies lead to the usability-reusability tradeoff problem, which was described in (Klinker, Bhola, Dallemagne, Marques, & McDermott, 1991). Increasing usability leads to reduced reusability, while increasing reusability causes reduced usability. Applied to ontologies, this means, that the more abstract ontologies are the more reusable but less usable they become. This is illustrated in Figure 3-2.

A survey of different case studies regarding ontology reuse is provided and analyzed by Simperl along with two additional case studies described in detail (Simperl, 2009). An important insight is that in the surveyed case studies ontology reuse is considered as a manual process. However, a consensus on the need for tools supporting this process was also observable**.** It is important to mention that from the usability point of view such tools must not be designed for high skilled users with extensive ontology engineering experience, but need to keep "normal" users in view (Dzbor & Motta, 2008). In fact, most existing domain ontologies seem to be created by developers with no experience in formal knowledge representation (Dzbor & Motta, 2008).

The mentioned survey provides an analysis of the applied methodologies in the case studies (Simperl, 2009). It is said that "*activities such as ontology assessment, integration, translation, and customization seem to be relevant across case studies*". The reuse process commonly starts with the intention to utilize ontologies in an envisioned IT system. That means that the developer has an application and a domain in mind.

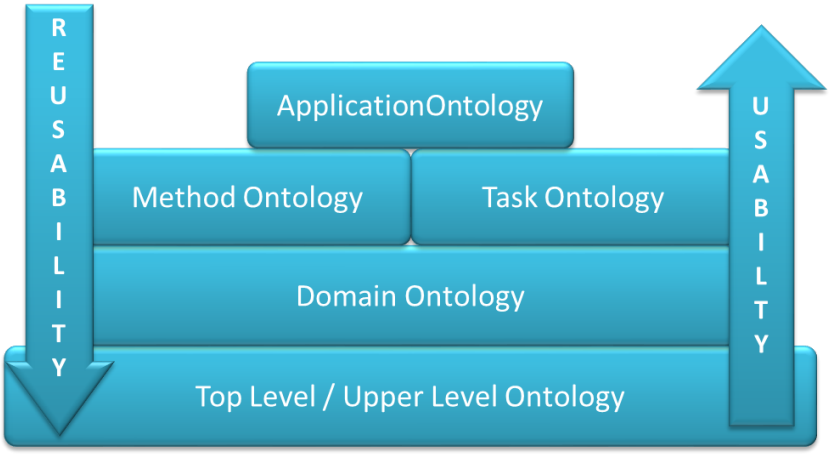


Figure 3-2: The usability-reusability tradeoff problem. The more abstract ontologies are the more reusable but less usable they become.

Based on this the developer starts searching for candidate ontologies, which might be reused. Different search engines (d'Aquin, Sabou, et al., 2007) and ontology libraries (D'Aquin & Noy, 2012) are available to support this discovery process.

Having obtained a list of potential candidate ontologies an analysis and decision taking step has to be done. The coverage and level of detail of each candidate has to be analyzed, in order to answer the question, if an ontology is reusable for the targeted system. If it is, the second question is, to which extend it can be reused and whether it needs to be customized. Reuse can range from an inspiring input up to the complete adoption without any customization. It is also possible that a candidate is reused partially, which would assume some modularization step to be taken. It is very important that these decisions are taken quickly and correctly. If the analysis process cannot be done efficiently or the decision is made wrong, the reuse effort would lead to waste of time and resources, although its primary motivation was to save both.

3.3.1 Reuse Support through Ontology Documentation

Careful documentation of the development process and the created artifact is broadly accepted as an important means to support reuse. It is frequently used in the field of Software Engineering, where tools like JavaDoc are very popular. Whereas, in the field of Ontology Engineering the lack of good documentation makes reuse difficult, because the decision on the applicability of candidate ontologies becomes a time-consuming process. On the other hand, the process of documentation is an additional

effort for the ontology developer which still lacks of an appropriate support system to create documentations automatically or semi-automatically.

However, there are some simple or premature tools available, which support the documentation process. Inspired by the success of JavaDoc for code written in Java, OWLDoc[18] is a tool that generates frame-based HTML pages with three areas. An example is visualized in Figure 3-3.

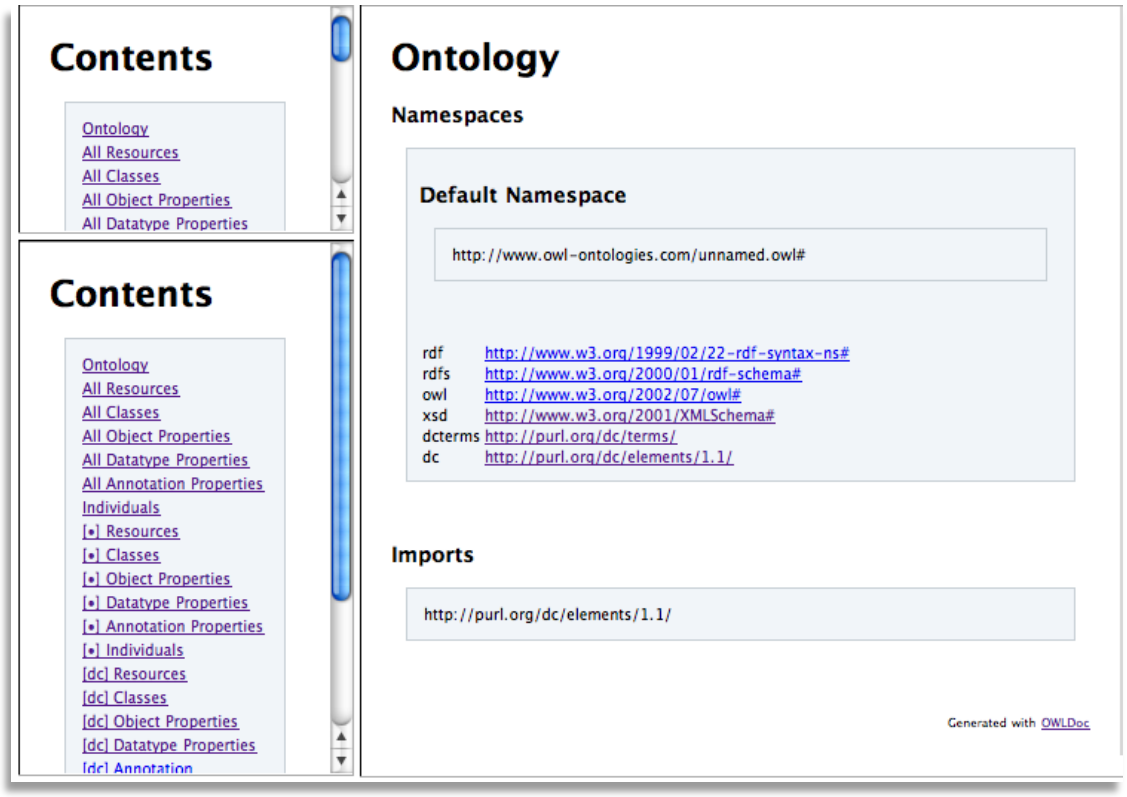


Figure 3-3: An example of a documentation page created with OWLDoc

It allows for navigating quickly to a specific resource and for obtaining information about it like comments, labels, type etc. When a class is chosen the main frame shows information such as superclasses and disjoint classes, while in case of properties information as superproperty, domain and range are shown. This kind of representation is useful to get detailed information about a single concept and its connections to other

[18] http://www.co-ode.org/downloads/owldoc last access on August 14th 2012

concepts. However, it does not provide an overview about the ontology and its structure as a whole.

More recent documentation tools such as Neologism (Basca, Corlosquet, Cyganiak, Fernández, & Schandl, 2008), SpecGen[19] and VocDoc[20] create one HTML page containing detailed information about the classes and the properties. Additionally, these HTML pages also contain meta-information like version information, changelog, authors, namespaces, license information, and referenced external ontologies. This kind of information is either at the beginning of the document or at the end. The details about the ontology and its concepts are at the main part of the document. Before the main part begins, there is a brief section with an alphabetically sorted list of classes and properties, which is called "overview" or "at a glance". Neologism extends this section with a graphical visualization (illustrated in Figure 3-4), which is very useful in case of very small ontologies. Since the whole ontology is visualized without any reduction, even though the ontology is large, the visualization gets confusing very quickly.

In the documentations of FOAF e.g. the "at a glance" section is extended with a manually created grouping of concepts (called summarization in (Jannink et al., 1999)), which is actually a partitioning of the vocabulary**.** (This grouping was depicted in Figure 1-1). It is a good introduction so the reader can understand, what the ontology is about and can decide very quickly, if the content covers relevant concepts for her or his purpose. Therefore, this illustration addresses most likely users who are looking for a reusable ontology and want to decide quickly, if a closer look makes sense. In consideration of the fact that the documentation of FOAF comprises about 40 printed pages, it becomes clear how important such a support is and how much time it can save. Additionally, it emphasizes that even in case of rather small ontologies there is a need for breaking down the complexity for documentation purposes, where chunking terms is one promising means to do this. But in case of large ontologies the partitioning of the vocabulary has to be extended with a reduction step limiting the number of groups as well as the size. This is very important because chunking terms for documentation purposes need to take cognitive capabilities of humans into consideration**.**

3.3.2 Term Chunks in Existing Documentations

For the understanding how such term chunks should be created and to derive - if possible – parameters for the envisioned partitioning framework, existing examples have been searched for. After an exhaustive search for online available ontology

[19] https://bitbucket.org/wikier/specgen/wiki/Home last access on August 14th 2012
[20] http://kantenwerk.org/vocdoc last access on August 14th 2012

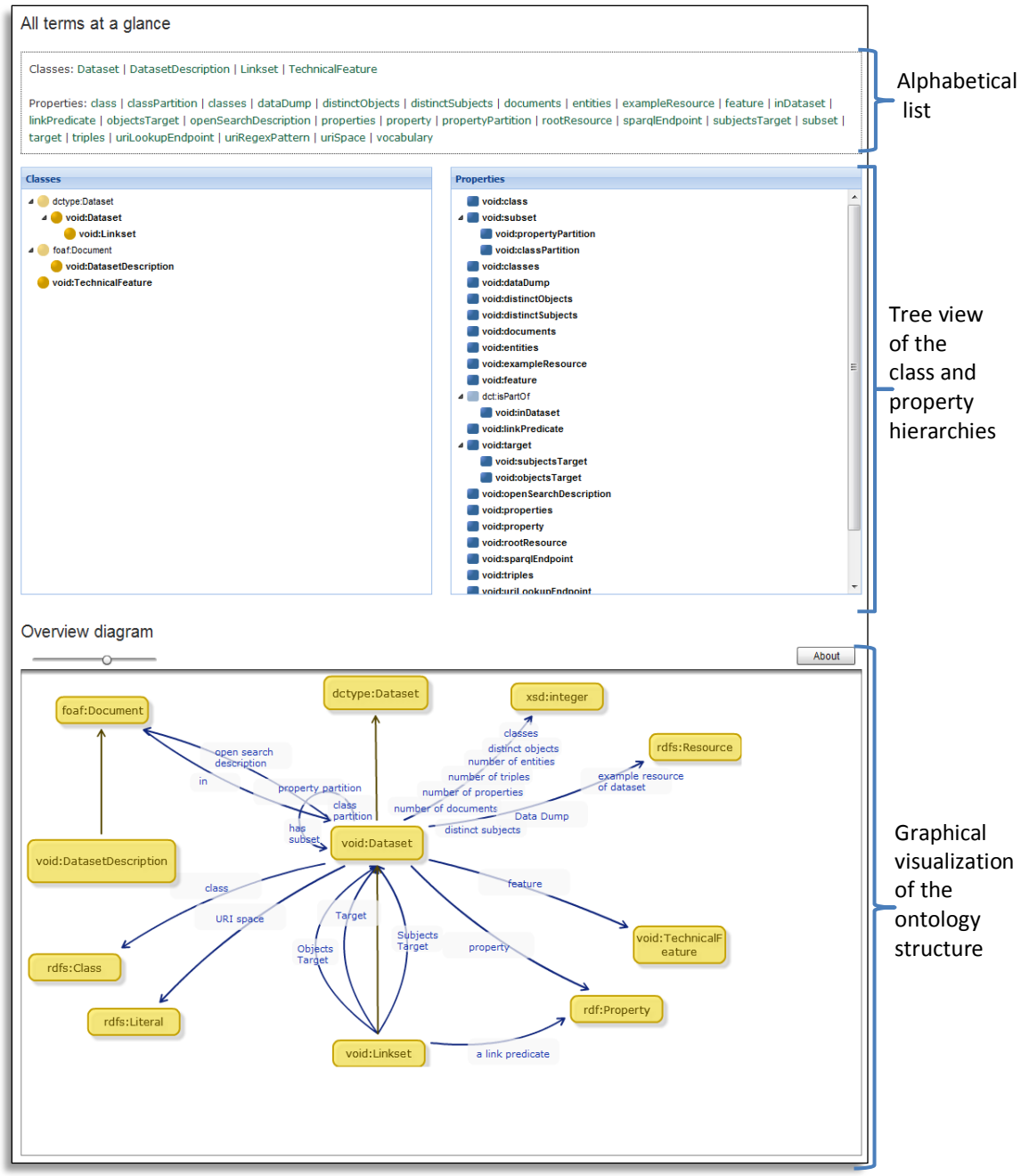


Figure 3-4: "At a glance" section of Neologism

Documentations, where concepts have been grouped to support the understanding, thirteen ontologies were found. In the following these ontologies are briefly described along with some noticeable aspects of their history and the groups.

Atom Activity Streams

The Atom Activity Streams ontology (aair) is the RDF representation of the Atom Syndication Format (Nottingham & Sayre, 2005). The ontology was created by two responsible authors and a contributor within an EU research project. The initial version was proposed in December 24th in 2009 and the last update was in March 3rd 2010. The documentation of the ontology (Minno & Palmisano, 2010) contains four groups, which have the same style as the groups illustrated in Figure 1-1. They contain only classes and are alphabetically sorted. The four groups are primarily the subtrees of the four root classes "Actor", "Object", "Verb", and "Context". The groups are named according to these classes. However, there is also a class within a group that is not the subclass of one of the four mentioned classes.

Biographical Ontology

The Biographical Ontology (bio) contains terms to describe events that are part of a person's life. It was created by two authors initially proposed in March 7th 2003. The last update was in June 14th 2011 which was the 15th version. The documentation (Davis & Galbraith, 2010) contains five groups of concepts. These groups are comma separated lists of concepts as illustrated in Figure 3-5. One group comprises classes only and the other four groups consist only of properties. One group contains the properties of the class "Person", whereas one group contains only the properties of the class "Event". The other two property groups contain properties relating an event to an agent or an event to an event respectively.

Collaborative User Resource Interaction Ontology

In the Collaborative User Resource Interaction Ontology (curio) classes and properties are defined to describe discussions on collaboration platforms. It allows the management of user-generated content on these platforms. The ontology was created within an EU research project by a single author. The initial version was created in September 14th 2009 and the last update was in February 1st 2011, which was the sixth revision. The documentation of curio (Burel, 2011) contains four groups of concepts, which have the same style as the groups illustrated in Figure 1-1. The groups comprise classes as well as properties. The class names are alphabetically sorted while the properties of a class are listed directly after the class name. The first group contains general classes and the groups "Annotation", "Documents", and "Events" represent

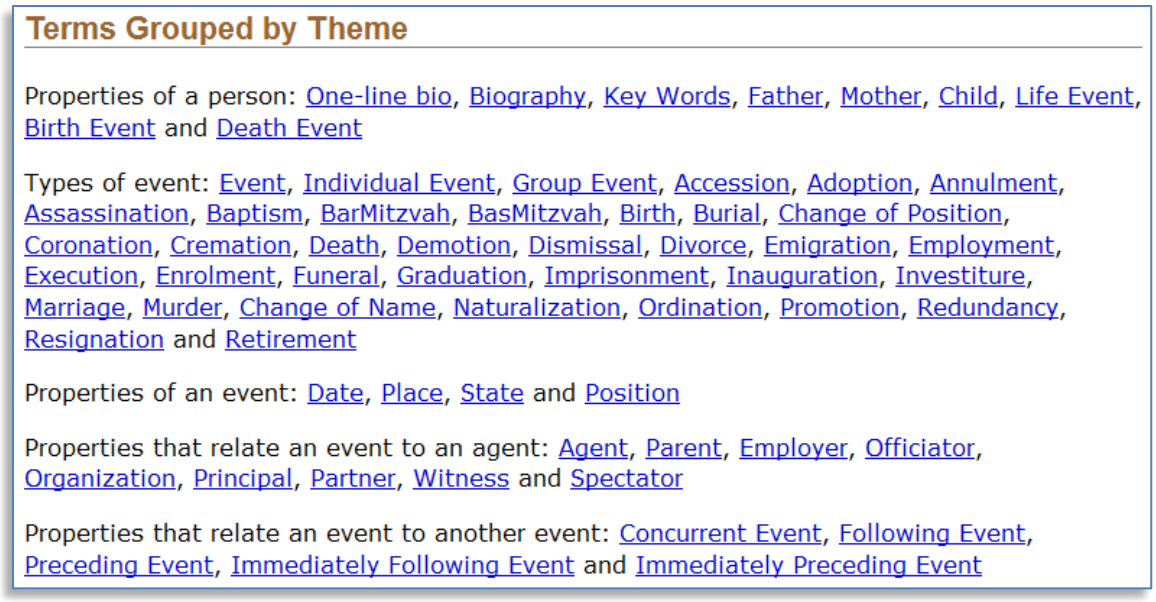


Figure 3-5: Term chunks of the Biographical Ontology in the documentation page

mainly subtrees of the class hierarchy refining classes of the first group. While the "Documents" group contains the class "Document" and the group "Events" contains the class "Event" there is not a class named "Annotation".

Enterprise Competence Organization Schema

The Enterprise Competence Organization Schema (ecos) defines terms to describe the competences of organizations within a global market. It was created by two authors as a part of a PhD thesis. The initial version was proposed in September 1st 2009 and was not further developed. Within the documentation page (Khilwani & Harding, 2009) the classes and properties are grouped into four different groups, which have the same style as the groups illustrated in Figure 1-1. The groups are representing the subtrees of the concepts "General", "Business", "Specific", and "Record" while these classes themselves are not listed in the group. The classes are at the beginning of the list and the properties are at the end. The properties are in the same group where the corresponding domain class is. Some properties have several domain classes which are in different groups. In this case these properties are mentioned several times.

Friend of a Friend Vocabulary

The Friend of a Friend vocabulary (foaf) is a very popular and broadly used ontology to publish personal profile information and links to friends in a social network manner. The initial version was created in 2000 and it was updated frequently. The last update was in August 9th 2010. It was created by two authors. The documentation of version 0.97 (Brickley & Miller, 2010) contains five term chunks (Figure 1-1) while four groups contain properties as well as classes and one group contains properties only. The groups do not represent subtrees of the ontology nor are the properties in the same group as their domain classes. In fact, the most properties belong to the classes "Agent" and "Person" but they are distributed over all groups. Three groups are named according to the classes they contain ("Online Accounts", "Project and Groups", "Documents and Images"). The group "FOAF Basics" seems to contain the most relevant terms for this ontology and the group "Personal Info" contains mainly properties of the class "Person".

Although the successor version of the documentation has only two groups, it is ignored in this work. Because on the one hand one group is still divided into three subgroups and on the other hand the newer grouping does not match to the motivation of dividing the concepts according to subdomains.

GI2MO Types Taxonomy

The GI2MO Types Taxonomy (gi2mo) contains terms to annotate ideas in an idea management system. It has one author, who created it in the context of a Spanish research project. The initial version was proposed in May 30th 2011. This version was revised only once and the latest version is from June 10th 2011. The documentation page (Westerski, 2011) contains four groups comprising properties as well as classes. The terms are organized in a tree structure as illustrated in Figure 3-6. However, it is not clear in the visualization if a child node is a subclass or a property.

Music Ontology

The Music Ontology (music) was created to allow the description of music in a broader sense. The initial version was created in December 21st 2006. About seven authors worked on this ontology and updated it several times. The last version is from August 12th 2012, which is the 15th version. The documentation page (Raimond et al., 2012) contains 23 groups with the same visualization style like in Figure 1-1. There are groups comprising classes only and properties only as well as groups that comprise both. The property only groups contain properties of only one particular class while in most cases these groups are named according to the particular domain class. In some cases this class itself is also listed in the group.

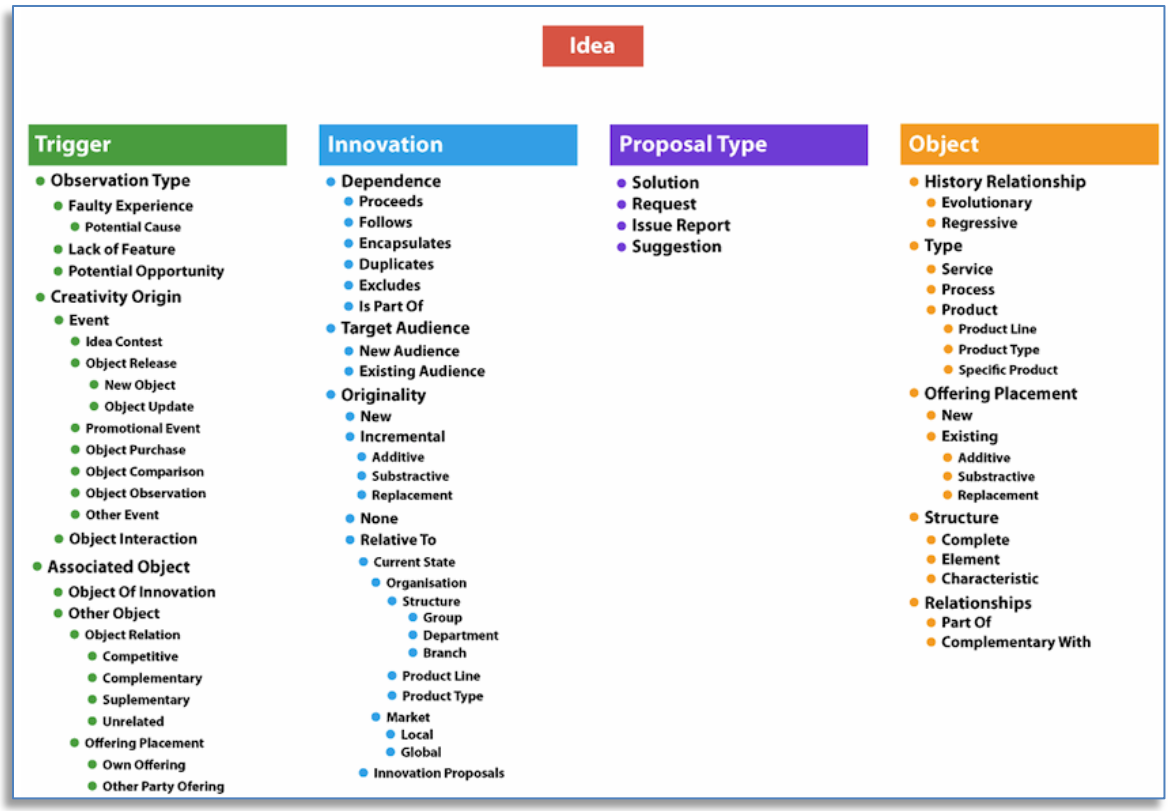


Figure 3-6: Term chunks of the GI2MO ontology in a tree structure

W3C Prov Ontology

The W3C Prov Ontology (provo) defines the concepts of the Prov Data Model[21] of the W3C in OWL that aims at the description of provenance information about entities, activities, and authors involved in the creation of data. The initial version was created in December 13th 2011 by three authors within the context of the work done by the W3C PROV Working Group. It was updated six times. The latest version is from April 30th 2013. The documentation page (Lebo, Sahoo, & McGuinness, 2013) contains three groups of concepts which are sown in Figure 3-7. Each one of these groups contains two subgroups, one group for the classes and one group for the properties. The first group is named "Starting Point classes and properties", the second group "Expanded classes and properties" and the third group is "Qualified classes and properties". The order of the classes does not show any rule, whereas the properties show a noticeable order. The first properties in each properties' subgroup are properties of the classes of the very

[21] http://www.w3.org/TR/2013/REC-prov-dm-20130430/ last access 22nd August 2013

first classes' subgroup. These properties are followed by the properties of the classes of the same group. Along with the naming of the groups this indicates a grouping according to the importance of the different classes and properties.

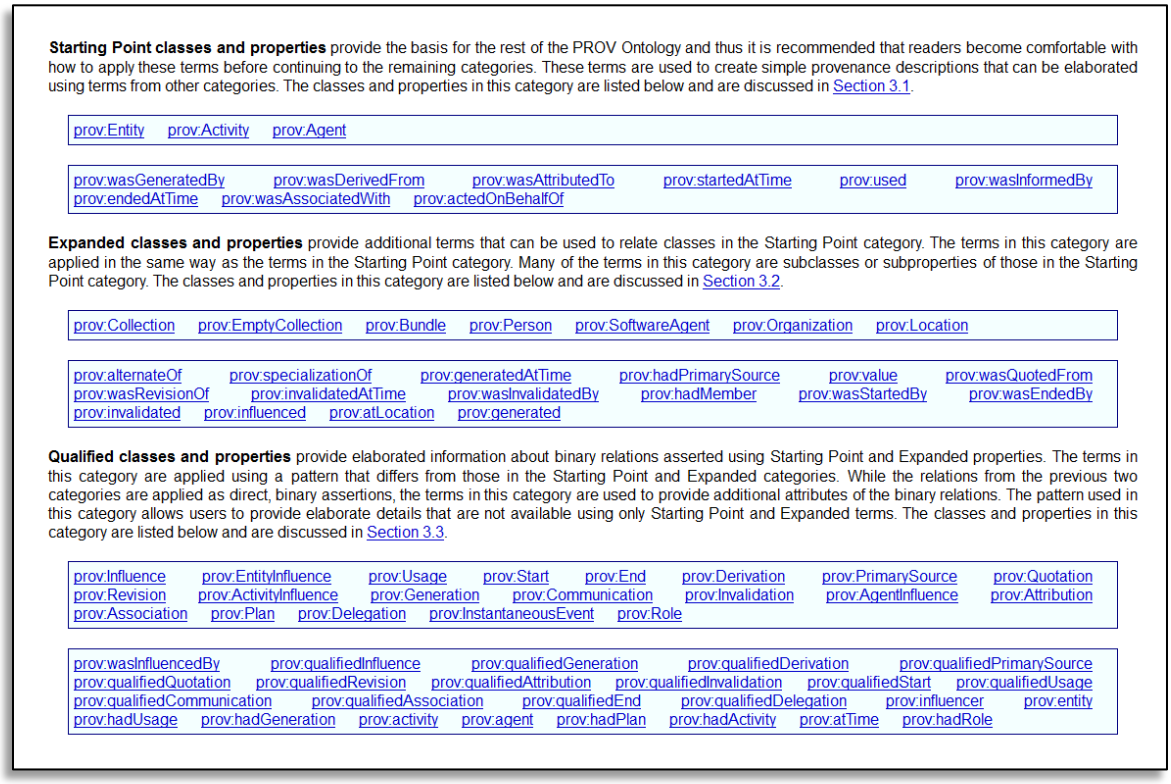


Figure 3-7: Term chunks of the provo ontology

Online Presence Ontology

The Online Presence Ontology (opo) defines classes and properties to describe the online presence of users. The initial version was created in 2008 by one author and was updated several times. The latest version is from July 1st 2010 and has the version number 0.52. The documentation page (Stankovic, 2010) contains two groups with properties, one group with classes and instances, and one group with classes, properties and instances which are marked as such. The property groups contain only properties belonging to one particular domain class. The terms are alphabetically sorted. The groups are visualized as simple HTML lists as illustrated in Figure 3-8.

OnlinePresence properties

- avatar,
- currentAction,
- currentLocation,
- customMessage,
- declaredBy (which Agent),
- declaredOn (which account),
- duration,
- hasPresenceComponent,
- intendedFor (which audience),
- publishedFrom (which device/application),
- startTime.

Online Presence Components

- Findability (instances: PubliclyFindable, ConstrainedFindability),
- Notifiability (instances: AllNotificationsPass, NotificationsConstrained, NotificationsProhibited),
- OnlineStatus (has a name and its components).

Online Status Components

- Activity (instances: Active, Inactive, ProlongedInactive),
- Contactability (instances: FreelyContactable, ConstrainedContactability),
- Disturbability (instances: Available, DoNotDisturb),
- Visibility (instances: Visible, Invisible).

Sharing Space properties

properties shared by the members of a Sharing Space, bounding them together

- attendedOrganisedEvent,
- basedNear,
- closestFriendsOf,
- commonInterest,
- currentlyIn,
- familyOf,
- schoolHomepage,
- workplaceHomepage.

Figure 3-8: Term chunks of the Online Presence Ontology

Provenance Vocabulary Core Ontology

The Provenance Vocabulary (pvc) provides classes and properties for the description of provenance information for the particular case of Web data. It is an extension of provo. The first version was created in August 25th 2009 by two authors and was updated six times. The latest version is from March 14th 2012. The documentation page (Hartig & Zhao, 2012) provides a graphical illustration of the ontology structure as shown in Figure 3-9. The whole ontology is divided into three parts, which are considered as three term chunks. Since this is a graphical representation, there is no sorting of the classes or properties possible.

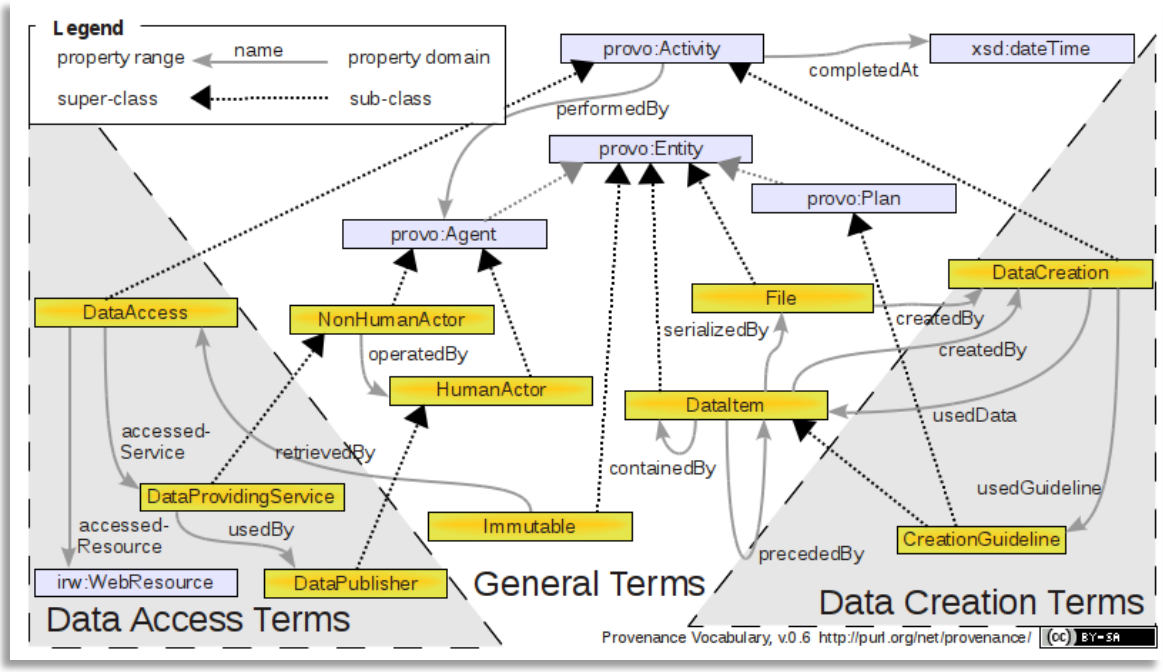


Figure 3-9: Term chunks of the pvc ontology

Premis

The Premis OWL Ontology[22] (premis) is a provenance metadata ontology. It is based on the OASIS reference model PREMIS – Data Dictionary for Preservation Metadata (Guenther et al., 2011). The initial version was created in 2011 and the latest version is from September 2012, which is named as version 2.2 and declared as draft. In the documentation page[23] five UML-like diagrams are shown. Four diagrams are

[22] http://premisontologypublic.pbworks.com/w/page/45987067/FrontPag last access 22nd August 2013
[23] http://premisontologypublic.pbworks.com/w/page/46121028/Diagrams last access 22nd August 2013

representing subdomains and one diagram is depicting the relatedness of the other four diagrams. Hence, the latter group is considered as an artificial group and therefore ignored in this work.

Represent Rights Data Ontology

The Represent Rights Data Ontology (rrdonto) provides classes and properties definitions for describing the intellectual property value chain. The first version was created in October 4th 2006 by three authors. The latest version has the version number 6.3 and was created in July 15th 2007. Within the documentation page there are four groups for classes only and one group for all properties, as illustrated in Figure 3-10 (Gauvin, Delgado, & Rodríguez, 2007). The first group contains classes, whose subtrees are represented in the other groups.

Figure 3-10: Term chunks of the Represent Rights Data Ontology

Semantic Web Conference Ontology

The Semantic Web Conference Ontology (swco) defines concepts to describe academic conferences, focusing on the particular conferences the European Semantic Web Conference and the International Semantic Web Conference. It was first created in May 31st 2007 by three authors. It was updated several times. The latest version is numbered 895. It is dated to May 11th 2009. In the documentation page (Möller et al., 2009) there are five term chunks in the documentation with classes only which have the same style as the groups shown in Figure 1-1. The groups are primarily representing subtrees and are named according to the particular root class. The latter itself is in some cases within the group. The classes are alphabetically sorted.

3.3.3 Properties of Existing Term Chunks

The brief descriptions of the different term chunks unveil that there is no unique understanding how a term chunk should look like. While in some cases only classes are grouped in other cases properties are also included in the groups. In some cases the groups correspond to subtrees of the class hierarchy and in other cases the groups are horizontal cuts of the hierarchy. In some cases class names are used as titles for the group while the class itself is not included in the group. In other cases these classes are also part of the group. There is also no unique style how to represent the term chunks. In some cases the concepts are grouped in html lists and in some cases the concepts are grouped as textual lists or as graphical representations. Table 3-1 shows some relevant properties of the different term chunks.

Ontology	Number of groups in docu	Mean group size	Groups alphabetically sorted	Properties in groups	Property only groups	Mixed Groups
aair	4	9.75	✓	-	-	-
bio	5	12.8	-	✓	4	-
curio	4	12.5	✓	✓	-	✓
ecos	4	11.75	-	✓	-	✓
foaf	5	12.6	-	✓	1	✓
gi2mo	4	21.25	-	✓	-	✓
music	23	6.52	-	✓	10	✓
opo	4	6.5	✓	✓	2	✓
provo	3	13.66	✓	✓	-	✓
pvc	3	9.33	na	✓	-	✓
premis	4	14.5	na	✓	-	✓
rrdonto	5	8.4	-	✓	1	-
swco	5	5.8	✓	-	-	-

Table 3-1: Overview of some important properties from the existing term chunks in the documentations.

However, besides the differences there are also noticeable aspects which indicate some kind of consensus. First of all, the ontologies are all about the same size. Even though they can be considered as rather small ontologies, they all contain the

aforementioned term chunks within their documentation pages. That points to the fact that all authors agree on the usefulness of such a grouping even at that size. Secondly, in most cases the groups contain classes and properties, while in almost every case the names of the classes start with capital letters and the names of the properties start with small letters. This helps to distinguish between them even though they are in the same group.

In order to understand, how partitioning should be done to create term chunks the previously described documentations have been analyzed in terms of partition size and count. Figure 3-11 visualizes the distribution of the group size.

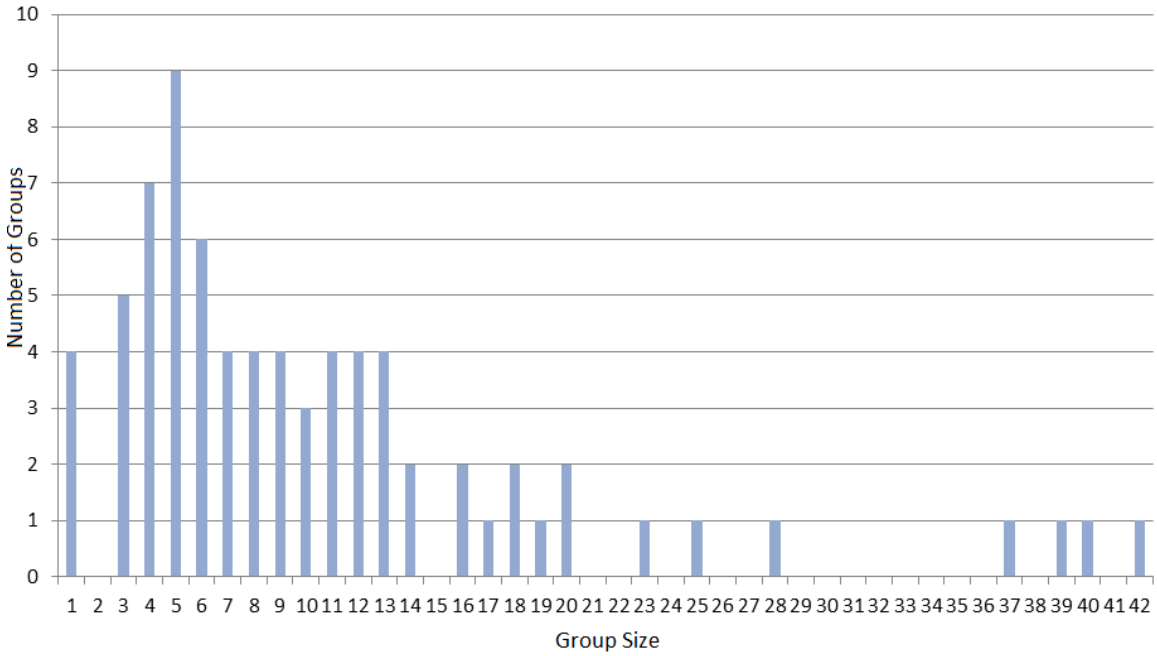


Figure 3-11: Distribution of the number of elements per term chunks

The arithmetic means of the group size is 11.18 with a standard deviation of 3.19 (28% of the arithmetic means). This leads to an optimal group size of about 8 to 14 elements. However, the number of analyzed documentations is not statistically significant. It is important to double check this value with other findings. Since term chunks are created to support the comprehension, cognitive science has been identified as the most appropriate research area to look for additional information.

In order to understand the content of a group, it is important to memorize its elements for a short term of period. According to Miller this is done in the immediate memory which has a capacity of about seven (Miller, 1956). A later work by Baddeley

showed that this capacity is increased to 15 if the elements to memorize are related (Baddeley, 2003). Since these values do not contradict - they even correlate - with the calculated range, an interval of 8 to 14 elements per group is accepted to be very appropriate. This is especially justified, if one considers that elements of one group are expected to be more related to each other than to elements from other groups.

The arithmetic means of the number of groups for the thirteen ontologies is 5.6 with a standard deviation of 2.6. This would mean that 3 to 9 groups per documentation is a good guidance interval. But after removing the value for music from this analysis because of its significant difference, the arithmetic means of the number of groups is 4.16 with a much lower deviation of about 0.56 (13.5% of the arithmetic means). This value is considered to be acceptable due to two arguments. On the one hand this value does not contradict with the aforementioned finding from cognitive science. Because term chunks have a higher complexity than the elements of the groups and they are less related to each other than the elements of one term chunk. Therefore memorizing them requires more mental capacities. On the other hand the quite small deviation indicates a strong consensus and can be accepted as a best practice. So the guidance value for the number of groups per documentation is three to five. This is considered to be acceptable.

3.4 Partitioning for Ontology Maintenance

In traditional development methods of knowledge-based systems, creating ontologies are heavyweight processes, including detailed domain and application analysis. Before the ontology is deployed, it passes through different tests until it achieves a satisfying maturity level. Maintenance in this case might be a rather small issue, as it is very unlikely that further refinement apart from small corrections is necessary, while the ontology is in use. However, in the context of agile ontology engineering methodologies, maintenance needs special attention, because in agile processes the ontology is permanently within a refinement process. In this case, maintenance can be considered as equal to forward engineering forming the overall evolution process. In this regard, maintenance does not include only low-level activities like adding new elements, updating, refining, merging, and removing existing elements it also comprises the complete refactoring (Gasevic et al., 2011).

The overall goal of maintenance is to improve the quality of an ontology or to adapt it to new requirements. The higher the complexity as well as the size of an ontology is, the more difficult is the maintenance process. So an ontology has a maintainability characteristic, which depends on its size and complexity. Regarding software quality, there is an ISO standard for Systems and Software Quality Requirements and Evaluation

called SQuaRE (Internation Organization for Standardization, 2011) which defines the maintainability of a software product as a combination of the following five subcharacteristics: modularity, reusability, analyzability, modifiability, and testability. This approach was adapted by Duque-Ramos et al. to ontologies in OQuaRE, where the maintainability is defined as:

> *"The capability of ontologies to be modified for changes in environments, in requirements or in functional specifications. Some subcharacteristics are modularity, reusability, analyzability, changeability, modification stability and testability."* (Duque-Ramos, Fernández-Breis, Stevens, & Aussenac-Gilles, 2011)

In SQuaRE as well as in OQuaRE modularity is mentioned as the first subcharacteristic and is, therefore, accepted as the most important one. In fact, the other subcharacteristics seem to depend to some degree on the modularity. At least for reusability and analyzability, this dependency has been shown in the previous section. However, in contrast to a user, who seeks for a reusable ontology, the maintenance task demands a user who is already familiar with the ontology. It is not sufficient for the maintainer to understand merely the ontology completely, but it is mandatory to understand the modeled domain and the application context. Moreover, it is essential to understand the effect of the change on the whole ontology and on the application which makes use of it. This difference demands for distinguishing between partitioning for accelerating ontology reuse and partitioning for simplifying ontology maintenance. Therefore, previously elaborated properties of term chunks do not hold for ontology modules. This notion is additionally supported by the fact that having experience with an object improves the capacity to recognize meaningful structures about it (Shneiderman, 1977). That means that the cognitive limits for the number of partitions as well as the size of them are different than the limits mentioned in the previous section. This makes clear that partitioning, aiming at the improvement of the maintenance task, has different requirements and has to be tackled on its own.

3.4.1 Existing Modular Ontologies

Obviously, modularity is one important quality aspect for artifacts. Well done modularity is expected to increase flexibility and to simplify maintainability. Therefore, different ontology engineering methodologies mention or address modular design of ontologies. However, most online available ontologies were not built following methodologies at all or at least strictly and are therefore mostly monolithic (Luczak-Rösch, 2011). With an appropriate partitioning tool, such ontologies can be modularized

afterwards. In order to understand, how this can be done and do derive requirements, analyzing the structural properties of existing modular ontologies is accepted to be a reasonable strategy. Four ontologies have selected for this analysis, which are briefly described along with some noticeable properties as follows.

Collaborative User Resource Interaction Ontology

The aforementioned ontology curio has three extensions, which are all created by the same author and are importing the core ontology with the "owl:imports" concept. The CURIO Annotations Vocabulary[24] defines 16 terms. The revision number of the latest version is 0.4 dated to February 1st 2011. The CURIO Documents Vocabulary[25] defines four terms and CURIO General Resources Vocabulary[26] defines eight terms. The latest versions of both have the revision number 0.2b. In the previous analysis only the core ontology (which defines 22 terms and has the revision number 0.5 dated to February 1st 2011) has been used, because it was the only one that had a term chunk in the documentation. Its initial version was proposed September 14th 2009. Merging all modules (curioMerged) leads to an ontology with 30 classes and 20 properties.

Semantically-Interlinked Online Communities

The Semantically-Interlinked Online Communities[27] (sioc) ontology provides definitions of classes and properties for the integration of online social communities. The documentation page lists 13 author names and two editor names from different organizations. The current version has the revision number 1.35 and is dated to March 25th 2010. The change log lists 29 entries starting from April 7th 2005. The core ontology which contains 91 terms is extended by the following three modules:

1. SIOC Access Module[28] with four terms
2. SIOC Types Module[29] with 38 terms, and the
3. SIOC Services Module[30] with eight terms.

None of the three modules contain additional version information. These module import the core ontology with the "owl:imports" concept. The merged ontology (siocMerged) comprises 53 classes and 88 properties.

[24] http://purl.org/net/curio/annotations# last access 26th September 2013
[25] http://purl.org/net/curio/documents# last access 26th September 2013
[26] http://purl.org/net/curio/resources# last access 26th September 2013
[27] http://sioc-project.org/ last access 27th September 2013
[28] http://rdfs.org/sioc/access last access 27th September 2013
[29] http://rdfs.org/sioc/types last access 27th September 2013
[30] http://rdfs.org/sioc/services last access 27th September 2013

SPICE Mobile Ontology

The SPICE Mobile Ontology[31] (spice) describes the mobile communications domain. It was created in the context of an EU research project by a consortium comprising nine academic as well as industry partners. The core ontology, in which 19 terms are defined, is extended by the following eight extensions[32]:

1. Services with 92 terms
2. Service Context with 30 terms
3. Profile with 30 terms
4. Presence with 63 terms
5. Context with 55 terms
6. Distributed Communication Sphere with 209 terms
7. Content with 67 terms, and
8. Privacy with 23 terms.

Apart from the Services and the Content extensions, all extensions import the core ontology with the "owl:imports" concept. The Privacy extension additionally imports the Profile extension as well. The merged ontology (spiceMerged) contains 354 classes and 234 properties.

Financial Industry Business Ontology

The Financial Industry Business Ontology[33] (fibo) is defined by the Object Management Group[34] (OMG) and the Enterprise Data Management (EDM) Council[35]. It contains definitions for financial terms, in order to provide terminological support that allows transparency in the global financial system. The draft version from June 1st 2013[36] of the foundations ontology comprises 23 modules. These modules are strongly connected to each other through the "owl:imports" concept. They create a dense dependency graph which is illustrated in Figure 3-12. The number of terms defined by the different modules is shown in Table 1-1 with an additional distinction between classes (c) and properties (p). The merged ontology (fiboMerged) contains 84 classes and 109 properties.

[31] http://ontology.ist-spice.org/ last access 27th September 2013
[32] http://ontology.ist-spice.org/spice_ontologies_files.htm last access 27th September 2013
[33] http://www.omg.org/hot-topics/fibo.htm last access 27th September 2013
[34] http://omg.org/ last access 27th September 2013
[35] http://www.edmcouncil.org/ last access November 13th 2013
[36] http://www.omgwiki.org/OMG-FDTF/doku.php last access 27th September 2013

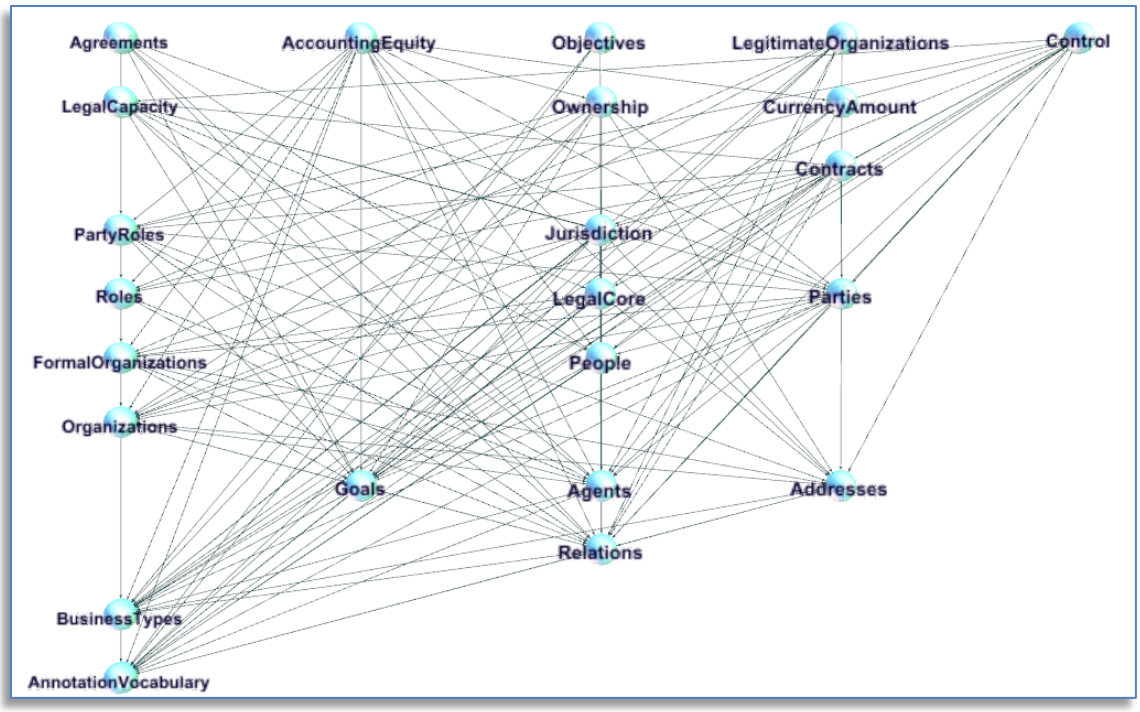


Figure 3-12: Dependency graph of the modules of fibo

Module	C	P	S
People	7	11	18
Agents	1	0	1
CurrencyAmount	6	6	12
AccountEquity	8	1	9
Ownership	3	2	5
Control	3	0	3
AnnotationVocabulary	0	10	10
Relations	0	44	44
BusinessTypes	0	14	14
LegitimateOrganization	5	0	5
Addresses	2	2	4
Organitzation	1	1	2

Module	C	P	S
FormalOrganization	3	0	3
Objectives	1	1	2
Goals	1	1	2
Roles	2	1	3
PartyRoles	2	1	3
Aggreements	1	0	1
Contracts	17	10	27
Parties	1	1	2
LegalCore	5	0	5
LegalCapacity	8	2	10
Juristdiction	7	1	8
Sum	**84**	**109**	**193**

Table 3-2: Size properties of the fibo modules (C: number of classes, P: number of properties, S: sum)

3.4.2 Properties of Modular Ontologies

In contrast to the analysis of existing term chunks, the small number as well as the diversity of the four ontologies do not allow to extract common aspects of modularity. Nonetheless, some relevant and interesting insights can be drawn. Firstly, the different revision numbers of the curio modules proof that modularity allows independent development and is, therefore, an important aspect for maintenance support. This supports the aforementioned SQuaRE and OQuaRE approaches which defined modularity as one subcharacteristic of maintainability. Secondly, the dependency between the modules are defined by using the "owl:import" construct. An important insight regarding the dependency is that it seems to be common practice to have one very abstract "core" module. All other modules or at least most of them import this core module but are not connected between each other. This holds for curio, spice and sioc, where the dependency between the modules can be described as a star-like structure. However, fibo's dependency graph on the contrary can be rather seen as a tree-like structure with several abstraction levels. Apparently, the overall structure of the dependency graph seems to be mainly formed by the domain it represents and the relation between the modeled level of detail and the spectrum of content. Thirdly, the number of modules and the number of terms per module differ stronger than in the case of term chunks. Table 3-3 shows the average module size, the standard deviation of the module size and the number of modules for the mentioned four ontologies. The small number of analyzed ontologies along with the strong deviation do not justify further discussion regarding the number and the size of modules as it was done in Section 3.3.3.

Ontology	Average Module Size	Standard Deviation	Number of modules
curio	12.5	6.5	4
sioc	35.25	29.25	4
spice	65.33	38.22	9
fibo	8.39	6.68	23

Table 3-3: Properties of the modularization for the analyzed ontologies

3.5 Chapter Summary

In this chapter the problem of ontology partitioning has been tackled in depth on a conceptual level. After clarifying aspects of ontologies, the concepts size and complexity, whose growth is mentioned as the main problem demanding for appropriate partitioning techniques, have been discussed in detail. It becomes clear that concrete definitions for both are difficult. The mentioned definition for complexity clarifies that the issue with increasing size and complexity depends on the limits of perception and comprehension.

Furthermore, partitioning to support ontology reuse and partitioning to support ontology maintenance were discussed thoroughly. And again, the essential insight was that comprehensibility and the cognitive limits play a crucial role in the overall process of partitioning.

The overall insights of this chapter can be summarized as follows.

- Ontologies can be partitioned in terms of different aspects, while the structure that is in the focus of this work is one of these aspects. For that reason there might be other partitioning approaches targeting at other aspects.

- Size and complexity are difficult to define. Since partitioning is a decomposition process to overcome problems caused by the growth of both, it is essential to provide well-defined metrics to measure the quality of the outcome and to compare it with the original ontology. Keeping the initial sentence in mind, such metrics can only be some indications, but not formal and mathematical definitions for them.

- Partitioning an ontology is a goal-oriented process. Various tasks in the lifecycle of an ontology like reuse, maintenance, interlinking, and processing depend significantly on knowledge and demand for different levels of expertise about the ontology at hand. Therefore, the partitioning process depends on the motivation and needs to be done according the requirements of the concrete goal.

- Numerical properties of existing term chunks are quite similar. The number of groups as well as the number of elements per group within all analyzed documentation pages unveil that there is some kind of a common understanding. Double checked with values from cognitive science regarding

working memory, three to five groups per documentation seem to very appropriate, whereas the groups should have about eight to 14 elements.

- The number of partitions as well as the size of ontology modules depend strongly on the concrete ontology. In contrast to term chunks, modular built ontologies show a strong deviation in the number of modules as well as the size of modules. Therefore, ontology modularization seems to be very ontology dependent regarding structural properties.

4 RELATED WORK

After the research problem has been understood in depth, related work is now analyzed. It is investigated, how the mentioned problems and requirements were tackled by other researchers. This enables to understand existing approaches, proposals and outcomes of research efforts. Furthermore, the clarification of the state-of-the-art reveals benefits and limitations of existing solutions. It, additionally, depicts open problems and research questions.

Since this work applies a structure-based approach for partitioning, this chapter starts in Section 4.1 with a brief literature survey discussing how the structure of an ontology is defined in related work and which structural metrics are used for which purposes.

In Section 4.2 relevant works from the research field of module extraction are presented. The rationale behind this is twofold. On the one hand, this section should clarify the relation and emphasize the difference to ontology partitioning. On the other hand, it is important to see, how the structure of an ontology is defined and used for module extraction.

In Section 4.3 existing ontology partitioning efforts are presented, while the closest related work is discussed in Section 4.3.1 which uses the same approach for the same goal as this work does. In this regard, this work can be seen as an extension of them. Section 4.3.2 provides a brief description of a logic-based approach to ontology partitioning.

The summary in Section 4.4 underlines the most important insights from this chapter and provides a comparative view on the different understanding of ontology structure. Additionally, it positions this work within the environment of the mentioned related work.

4.1 Structural Representation and Metrics

Structural representation of ontologies and structural metrics are used in several ontology analysis techniques. The motivations of these techniques range from detecting central concepts to deriving patterns for synthetic creation of artificial ontology structures. In order to understand, how structural representations of ontologies are created and which structural metrics are used in which context, a detailed literature work has been conducted. In the following, a survey is provided, which describes existing work that utilizes some kind of a structural representation of ontologies and structural metrics. (Note: concrete mathematical definitions of the metrics are not provided in this section as most of them will be discussed in detail in Section 5.3.)

Structural Pattern

Theoharis et al. state that there is a need to benchmark repositories and query language implementations (Theoharis, Tzitzikas, Kotzinos, & Christophides, 2008). This work is driven by the idea that the success of the Semantic Web depends on the existence of ontologies for advanced querying and reasoning services. The realization of benchmark requires means to create synthetic ontologies comprising schemas as well as data. For that reason, the authors analyzed the structure of 83 selected ontology schemas with more than 100 classes for structural patterns. The focus was primarily on power-law degree distribution. They distinguished between the property graph and the subsumption graph of the schema. In both cases, the classes and literals were represented as nodes and the properties as edges. The outcome of this analysis is that the most of the analyzed schemas approximate a power-law for degree distributions in the property graph and also in the subsumption graph. This indicates the existence of central concepts forming a core.

Node Importance

In the following papers different notions of structural centralities have been used. The assumption in these works is that important concepts are in the core of the ontology and have, therefore, high centrality values. Graves et al. make use of the RDF graph as an undirected labeled graph to represent the ontology (Graves, Adalı, & Hendler, 2008). No differentiating is made between schema and instances. For ranking the nodes the closeness centrality values are used. This method is called Node Centrality Ordering (noc-order).

AKTiveRank is a system which is motivated to facilitate reusing existing ontologies (Alani & Brewster, 2006). It aims at improving search engines by ranking ontologies based on structural properties of the search terms within the whole ontology. Four different measures are defined, which are calculated separately by ignoring the

instances. These are namely "class match measure", "density measure", "semantic similarity measure" and "betweenness measure". Apart from the first one, they are all structural measures.

Hoser et al. introduce "Semantic Network Analysis" (SemNA) to analyze ontologies for the purpose of reuse and re-engineering (Hoser, Hotho, Jäschke, Schmitz, & Stumme, 2006). Different notions of node centrality are used, namely degree centrality, betweenness centrality and eigenvector centrality. The named entities of an ontology were represented as nodes in the graph. For an RDF statement this means that each subject, each predicate, and each object becomes a node, where the edges were directed from subject to predicate and from predicate to object. Experiments with the Semantic Web for Research Communities[37] ontology and the Suggested Upper Merged Ontology[38] (SUMO) are presented and the results are discussed.

Measuring Complexity

Apart from finding particular elements within an ontologies, structural metrics are also proposed to measure properties of the whole ontology. Yang et al. propose ten metrics to measure the complexity of ontologies (Yang, Zhang, & Ye, 2006). The main purpose is to analyze the evolution of ontologies and extract patterns from their changes. Through these patterns the future maintenance should be supported. This work uses the subsumption hierarchy and ignores other properties. The proposed metrics are divided into the categories "primitive metrics" and "complexity metrics". The first category consists of the "Total number of concepts", "Total number of relations", and the "Total number of paths" as well as "the longest path length for concept c_i", "the average path length for concept c_i", "the max path length of an ontology", and "the average path length of an ontology". The second category consists of "the average relation per concept", "the average paths per concept" and finally "the ratio of max path length to average path length of an ontology". After calculating these metrics for different versions of three ontologies, the authors detected some similarities between the changes of values for different ontologies. However, the term complexity is used completely different than it was defined in Section 3.2 because understandability and perception of humans are not considered at all.

In contrast, Zhang et al. make use of the term complexity in the same sense as it is used in this work (Zhang, Li, & Tan, 2010). It is advocated that increasing complexity has a negative impact on the quality of ontologies due to cognitive limits of maintainers. For measuring this complexity eight metrics are proposed. They are categorized in "ontology-level metrics" and "class-level metrics". The first category comprises the following metrics: "size of vocabulary", "edge node ratio", "entropy of graph." The

[37] http://ontoware.org/swrc/ last access April 12th 2014
[38] http://www.ontologyportal.org/ last access April 12th 2014

second category consists of the following metrics: "number of children of a class", "depth of inheritance of a class", "in-degree for a class", "out-degree for a class". The underlying structural representation of an ontology is formally described with graph-transformation rules and corresponds the RDF graph representation. Applied on different ontologies, it has been shown that these metrics allow good differentiation. However, apart from the motivation, the relation between cognitive limits and proposed complexity metrics is not investigated deeper. This work lacks of concrete threshold values for the different metrics based the cognitive capabilities, even though this expectation is raised.

Ontology Evaluation

Ontology evaluation is the process of determining the quality of an ontology. As mentioned previously, it is broadly accepted that ontology evaluation can be done only in context of a concrete application. However, several ontology evaluation approaches make use of a structure-based approach.

Ning and Shihan propose such an approach and define the following six structural metrics: "concept quantitiy", "property's expectation", "property's standard deviation", "tree balance", "concept connectivity", and "key concept quantity" (Ning & Shihan, 2006). On the one hand, for the "tree balance" metric the subsumption hierarchy is used. On the other hand, for the "concept connectivity" a proprietary graph is created which is not clearly described. Only the following sentence is provided: "If a concept has an object property whose value is an instance of another concept, an edge will be drawn between these two concepts." Regarding the metric "key concept quantity", an edge weight function is proposed which depends on the number of subclasses and relations of a concept. The concepts are then assigned the sum of the values of their relations. The concepts are then ranked accordingly and a previously defined proportion is then considered as key concepts. Furthermore, the authors obviously assume that there is some knowledge about the abstract structure of a domain of interest. This becomes clear, when they advocate that the representation of a domain should have a structure, which satisfies the structure of the domain. Additionally, the evaluation that is provided in the paper is a kind of discussion by the authors based on the values of the proposed metrics.

Based on a similar approach (Tartir, Arpinar, Moore, Sheth, & Aleman-meza, 2005) propose two categories of metrics to assess the quality of an ontology. Firstly, the schema metrics contain the following three metrics: "relationship richness", "attribute richness", and "inheritance richness". Secondly, they propose nine instance metrics: "class richness", "average population", "cohesion", "importance", "fullness", "inheritance richness", "relationship richness", "connectivity", "readability". This work has a strong focus on the instances of an ontology. It does not provide any scale to

categorize an ontology based on the values of the different metrics. That means that the user has to decide on its own, if a concrete value for a metrics indicates a good quality or a bad quality. Closing the paper with a similar discussion as in the previously mentioned paper (Ning & Shihan, 2006) makes clear that the proposed metrics are some kind of a support for a user. The user has to look at the values of the different metrics and has to build up his or her own mind.

Ontology Modularity

Within software modularity the terms cohesion and coupling are broadly accepted as good indicators for the quality of modularization. Cohesion refers to the degree of interconnections between elements of one module. Coupling on the contrary, is a metric that indicates the degree of connections between different modules. Due to the broad acceptance of these metrics in software engineering, attempts have been made to adopt them in the field of ontology engineering.

Yao et al propose three metrics to measure the cohesion of an ontology module (Yao, Orme, & Etzkorn, 2005). Based on the hierarchy tree, the proposed metrics are "number of root classes", "number of leaf classes", and "average depth of inheritance tree of leaf classes". In a follow-up paper, the authors proposed additional metrics to measure coupling, namely "number of external classes", "reference to external classes", and "referenced includes" (Orme, Yao, & Etzkorn, 2006). In both cases the metrics are simple counts without taking the overall size or the complexity of the ontology into account. The authors did comparisons with the ratings of eighteen human evaluators for 33 example ontologies. A correlation could be detected which indicates appropriateness of the proposed metrics.

Analyzing the network structure of an ontology as a basis for partitioning the class hierarchy into disjoint and covering set of concepts is presented by Stuckenschmidt (Stuckenschmidt, 2006). Its main goal is to support distributed maintenance, selective reuse and efficient reasoning. The ontology is preprocessed in the same way as in SemNA(Hoser et al., 2006).

4.2 Module Extraction

Module extraction refers to the activity to extract only a part of an ontology. The original ontology is reduced to a focused part. This presumes that there is a definition which part - which concepts and properties - is relevant and have to be extracted. In most cases the module is created by traversal. That means that a small module with the given concepts is created and extended step by step. The outcome is a new ontology document, which can exist in parallel to the original one.

4.2.1 GALEN Segmentation

Web ontology segmentation proposed by Seidenberg and Rector is a method which was developed for the GALEN ontology of medical terms (Seidenberg & Rector, 2006). Its primary goal is to create classifiable segments in order to enable reasoners to work properly even with large ontologies. In this regard, segments are targeted, which are as small and focused as possible, but also containing enough information on the same time. This method is optimized for densely connected and large ontologies with more than 1000 concepts. These ontologies should have at least (on average) one restriction asserted per concept.

A pre-requirement for this method is a normalized ontology that is an ontology in which primitive classes do not have more than one primitive superclass. The basic principle followed by this algorithm is that any concept and property participating directly or indirectly to the definition has to be included. The segmentation algorithm starts with one or more concepts, given by the user. The extraction is created by traversal around the given and related concepts. All classes upwards and downwards, superclasses and subclasses respectively, are taken into the extraction, without merging superclases for avoiding destruction of semantic accuracy. Only properties which are used in the class hierarchy are traversed together with their superproperties. Properties which are not used in the class hierarchy or are not a direct superproperty of such a property as well as sibling concepts are not included in the extraction. In the following steps all superclasses of already included concepts are also traversed and included in the extraction.

4.2.2 Extraction for Reuse

The technique presented in by Doran et al. aims at achieving an efficient way for reuse of large ontologies (Doran, Tamma, & Iannone, 2007). Having an ontology engineering perspective, the modularization process allows the ontology engineer some control in order to identify which part of the ontology to separate. A definition for ontology module is given as following: "An ontology module is a reusable component of a larger or more complex ontology, which is self-contained but bears a definite association to other ontology modules, including the original ontology." Three requirements of ontology modules are given. The first requirement is self-containment. That is, each module should be transitively closed with respect to an arbitrary given set of relations. Being concept centered is the second requirement of an ontology module. A module is supposed to describe a given start concept with enough information. The last requirement is that a produced module should be consistent. This approach exploits abstract graph models instead of logical formalism based representation, which

makes it language independent. It reduces extraction of an ontology module to traversal of a graph starting at a given point. It follows the idea that elements should be included which make a reference to the initial element, no matter directly or indirectly, explicitly or implicitly.

4.2.3 Extraction for knowledge selection

A modularization technique is proposed by d'Aquin et al. which is part of a process called knowledge selection (d'Aquin, Sabou, & Motta, 2006). This process refers to the integration of ontology selection with modularization in order to retrieve relevant ontology modules. Being strongly tight to a particular application the modularization technique is derived by following four previously determined application requirements.

The first requirement states that the criteria for modularization should reflect the criteria of selection. In other words, the module which is to be produced should be the smallest part of the original ontology which covers all terms used during the ontology selection. The second requirement aims at maximizing the number of potentially utilizable candidates. For that reason it demands that no assumption in respect of the ontology should be made, neither language nor quality. Minimal user interaction is the third requirement. This is of high importance for the examined semantic web browsing scenario. The user just wants to brows and does not care anything about the ontology selection and module extraction mechanism. Due to the semantic nature of ontologies the last requirement states that the extracted module should also include implicit information which can be inferred from the original ontology should be contained in the extracted module.

The module extraction algorithm is given by inclusion criteria for concepts, properties, individuals and assertions. In general, the algorithm works by including upwards. In order to keep the module size small, not all superclasses are included only the most specific common superclasses. Using inference during the modularization process allows for including implicit knowledge.

4.2.4 Traversal View Extraction

Although the mechanism presented by Noy and Musen is called view extraction, it can be seen as another approach for module extraction (Noy & Musen, 2004). Inspired by database views, the notion of ontology view is introduced which is a specific portion of an ontology. In contrary to the previously presented methods, traversal view extraction is strongly interactive. The user has to define a starter concept, a list of

relationships that are relevant and should be traversed and the maximum distance per relation.

The connection between the created view and the original ontology is kept by the so called boundary. Excluded concepts which are referenced by included concepts are part of the boundary. This allows the user to extend the view interactively and iteratively by choosing concepts to include from the boundary.

4.2.5 Logic-Based Modularization

As mentioned in Section 3.1, an ontology has different aspects. Semantics is one of these aspects. It is utilized in by Grau et al. for extracting modules (Grau, Horrocks, Kazakov, & Sattler, 2008). This logic-based approach targets at supporting the reuse of a subvocabulary $\mathcal{SV}$, which is part of a candidate ontology $\mathcal{O}$. A module $\mathcal{OM}$ is that part of the ontology $\mathcal{O}$ ($\mathcal{OM} \subseteq$) which extends a second ontology $\mathcal{P}$ ($\mathcal{OM} \cup \mathcal{P}$) regarding a subvocabulary $\mathcal{SV}$ as if the whole candidate ontology would be added to $\mathcal{P}$ ($\mathcal{O} \cup \mathcal{P}$). In this regard, $\mathcal{M}$ added to $\mathcal{P}$ has to have the same meaning regarding $\mathcal{SV}$ as if the whole ontology $\mathcal{O}$ is added to $\mathcal{P}$. This relation is called deductive conservative extension. $\mathcal{O}$ added to $\mathcal{P}$ is a conservative deductive extension of $\mathcal{OM}$ added to $\mathcal{P}$ regarding $\mathcal{SV}$ because it does not add any entailment regarding $\mathcal{SV}$. It is shown that computing a module as defined is undecidable. Two different approximation methods are proposed which are based on the concept of locality. The experiments with existing large ontologies illustrate that this approach might lead to modules, which are very unbalanced in terms of size. This is the case when this approach is applied to ontologies which contain strong semantic dependencies.

4.3 Ontology Partitioning

Following the idea that any system has the property of near-complete decomposability, ontology partitioning presumes that it is possible to find groups of objects within a given ontology which have a closer relationship to each other than to the other objects. In contrast to module extraction, ontology partitioning assumes that every necessary piece of information for partitioning is in the ontology itself. Therefore, it considers the ontology as a whole and partitions it completely without having any additional input like important concepts etc. However, as elaborated in Section 3, different motivations of partitioning have different expectations regarding the outcome. For that reason, configurations like expected number of partitions or the expected number of elements per partition are essential. Ontology partitioning tries to keep the

original semantic as complete as possible and produces new subontologies, which optimally should be interrelated somehow. Ideally, the original ontology should be reconstructible through merging the partitions.

4.3.1 Structure-based Partitioning

Structure-based partitioning is a partitioning method for large light-weight ontologies, that is ontologies mainly consisting of a class hierarchy (Stuckenschmidt & Klein, 2004). It aims at enabling easier browsing and exploring the hierarchy by dividing the class hierarchy into disjoint and covering sets of concepts. The basic assumption which is made by this method is that the structure of an ontology allows the derivation of the dependencies among concepts. The authors state that their method is not able to capture important dependencies that could be found by analyzing the names of classes and the logical definitions of concepts. Nevertheless, they advocate that for scalability issue such a simple approach is important. The partitioning method consists of five different phases. In the first phase a dependency graph is created. Dependencies corresponding to the subclass hierarchy or domain and range restrictions in property definitions are used in this regard. Based on the constructed graph the strength of the dependencies are determined in the second phase. Metrics from social network theory are used to compute and assign strength values to the network. In the third phase the mentioned strength value are used for determining modules. The fourth phase is necessary because some single nodes can be left over during the third phase. To assign leftover nodes to existing modules, the relation between a leftover node and a neighboring node, which is already assigned to a module, is taken into account. Finally, in the last phase created modules are merged manually.

This basic algorithm was extended and refined towards more flexibility (Schlicht & Stuckenschmidt, 2006). The connectedness of modules, size as well as number of modules and redundancy were defined as structural criteria for modularized ontologies which allow for estimating efficiency, robustness and maintainability. In a later work a sixth step called Axiom Duplication was added, which allows for overlapping modules (Schlicht & Stuckenschmidt, 2007). The intention behind this extension is to reduce the connectedness between modules. Finally, a method was presented that is used to automatically select optimal parameters in order to maximize the quality of the result (Schlicht & Stuckenschmidt, 2008a). This technique is implemented in a freely accessible tool called PATO.[39] Experiments with existing ontologies showed promising results to support reasoning and visualization of ontologies. The former was evaluated by

[39] http://web.informatik.uni-mannheim.de/anne/Modularization/pato.html last access 24th December 2012

calculating the communication cost between the partitions for a distributed resolution process (Schlicht & Stuckenschmidt, 2008b). The latter is evaluated through some kind of anecdotal evidence and was not supported with findings from cognitive science, even though it is about perception.

4.3.2 Partitioning Using ϵ Connections

Grau et al. propose a method for automatic partitioning of OWL ontologies is (Grau, Parsia, Sirin, & Kalyanpur, 2005a), which is based on ϵ -Connections (Kutz, Lutz, Wolter, & Zakharyaschev, 2004). In a previous work the authors propose an abstract syntax extension of OWL following the ϵ Connections formalism which allows to express a new kind of properties (Grau, Parsia, & Sirin, 2004). These so called link properties enable to create a link between entities of two different ontologies. The partitioning is done by transforming an OWL knowledge into ϵ Connections so that relevant subdomains of the ontology are represented in different components. These components are formally proven to contain the minimal set of atomic axioms necessary in order to maintain crucial entailments. Even though the algorithm is formally sound, tests have shown that it suffers from scalability issues (Grau, Parsia, Sirin, & Kalyanpur, 2005b). Ontologies containing a top ontology could not be partitioned properly. Similar to the logic-based approach for module extraction described in Section 4.2.5, this technique tends to create one very large partition and few very small partitions. This technique is implemented in an ontology editing tool called SWOOP[40].

4.4 Chapter Summary

This chapter provides an overview about existing related work in the research area this thesis pursues to contribute. It clarifies earlier scientific efforts and their outcome addressing the same basic research questions. The insights from this chapter can be summarized as follows.

Obviously, focusing on the structure of ontologies is broadly accepted as a good approach for various problems in Ontology Engineering. These problems range from finding central concepts to the overall quality assessment of ontologies. However, the conducted literature work unveil that each work is based on its own notion of structure and metrics. In most cases there are open questions like: Are the metrics based on the inheritance tree only? Is it assumed that the artificial upper concept "Thing" exists? Are

[40] http://code.google.com/p/swoop/ last access 24th December 2012

instances, reasoning and the import mechanism of OWL taken into account? Furthermore, the wording that is used is ambiguous or even confusing, as in some papers object type properties are called "relations" or "properties" and data type properties are called "attributes", whereas in other works object type properties are called "relations" and data type properties are called just "properties". In some cases, it is even unclear how the structure and the metrics are defined. Table 4-1 provides an overview about the different structural approaches in literature.

Section 4.2 clarifies the difference between module extraction and ontology partitioning. The descriptions of existing work depicted that most techniques utilize a traversal approach. In doing so, the focus is primarily on the hierarchical relations. Furthermore, the process of module extraction requires additional inputs by the user like number of hierarchical levels and types of relations to traverse. Therefore, the process of module extraction is rather interactive and demands more time and effort from the user than ontology partitioning as targeted in this thesis.

The logic based approaches for module extraction and ontology partitioning presented in Section 4.2.5 and Section 4.3.2 respectively underpin the relevance and importance of the structure-based approach of this work. Both examples led to modules and partitions respectively, which were very unbalanced in terms of size. This in turn is not acceptable, if reuse and maintainability support is envisioned and the focus is on perception and cognition instead of reasoning. Moreover, the work presented in Section 4.3.1 shows that even reasoning can be improved with structure-based partitioning and does not necessarily demand for logic-based approach.

Finally, the closest related work that has been presented in Section 4.3.1 applies a structure-based approach for ontology partitioning. The main motivations are improving reasoning and supporting the maintenance process through suitable visualization. The evaluation of the former is more sophisticated than the evaluation of the latter, which is some kind of a discussion. Therefore, it becomes apparent that the former has a higher priority for the authors. Furthermore, visualization of ontologies is about perception the ontology content. This in turn demands the consideration of cognitive capabilities of users, which is totally omitted. In fact, this issue is tackled in this thesis extensively. Thus, this work can be considered as an extension of the related work presented in Section 4.3.1.

	Goal	Graph Structure
(Theoharis et al., 2008)	Derive common structural features of schema graphs enabling the creation of synthetic ontology schemata for benchmark tests	Classes and literals are nodes, properties are edges. Additional distinction between subsumption graph and property graph
(Graves et al., 2008)	Find central node in RDF graph.	RDF graph
(Alani & Brewster, 2006)	Rank ontologies for search engines based on search terms.	RDF graph
(Hoser et al., 2006)	Analyzing ontology structure for reuse and re-engineering	Named entities as nodes. Edges between class and superclass, property and superproperty, property and domain class as well as between property and range class.
(Yang et al., 2006)	Extracting structural patterns from the evolution in order to support further maintenance	The subsumption hierarchy is primarily used as the structure of an ontology
(Zhang et al., 2010)	Measuring the complexity of an ontology from a cognitive perspective	RDF graph
(Ning & Shihan, 2006)	Measuring different structural properties which can be used in manual evaluation	Distinction between the subsumption hierarchy and the property graph. The latter is not concretely described
(Tartir et al., 2005)	Measuring different structural properties with focus on the population. The values are used in a manual evaluation process	This work does not propose any graph-based metrics.
(Yao et al., 2005).	Adopting the notion of cohesion on ontologies	Only the subsumption hierarchy is used
(Orme et al., 2006)	Adopting the notion of coupling on ontology modules	Only count metrics are proposed, therefore not graph representation is used
(Stuckenschmidt, 2006)	Analyzing the network structure of an ontology as a basis for partitioning the class hierarchy	Named entities as nodes. Edges between "related" concepts

Table 4-1: Ontology structure in literature

5 ADAPTABLE ONTOLOGY PARTITIONING FRAMEWORK

This chapter represents the core of this thesis. It describes the main contribution, which is an adaptable framework for partitioning ontologies in a structure-based manner. The conceptual model of the proposed framework is introduced in Section 5.1. The functionality of each component and their interdependencies are described from a high level perspective. The following sections present the concrete solutions elaborated for low level research questions.

Section 5.2 clarifies what is understood by the structure of ontologies and describes concrete techniques to create a structural representation. Section 5.3 provides definitions of structural metrics like size and complexity, which are necessary for the partitioning and evaluation process. Section 5.4 describes community detection algorithms from the social network analysis field and how they can be used to create ontology partitions. In this regard, the algorithms which have been integrated into the framework are introduced in detail.

In Section 5.5, it is described how this framework takes the inherent semantics of ontologies into consideration. This is done by the introduction of a weight function which assigns weights to the edges of the structural representation based on the meaning of the edges.

As discussed in Chapter 3, different motivations for partitioning have different requirements on the created partitions. Section 5.6 describes how this framework considers the number as well as the size of partitions during the partitioning process and becomes an adaptable partitioning framework. Finally, this chapter is closed with a chapter summary describing briefly the main aspects of this chapter.

5.1 Conceptual Model

The adaptable ontology partitioning framework contains different functional components which interact with each other. Figure 5-1 illustrates the conceptual model of the framework.

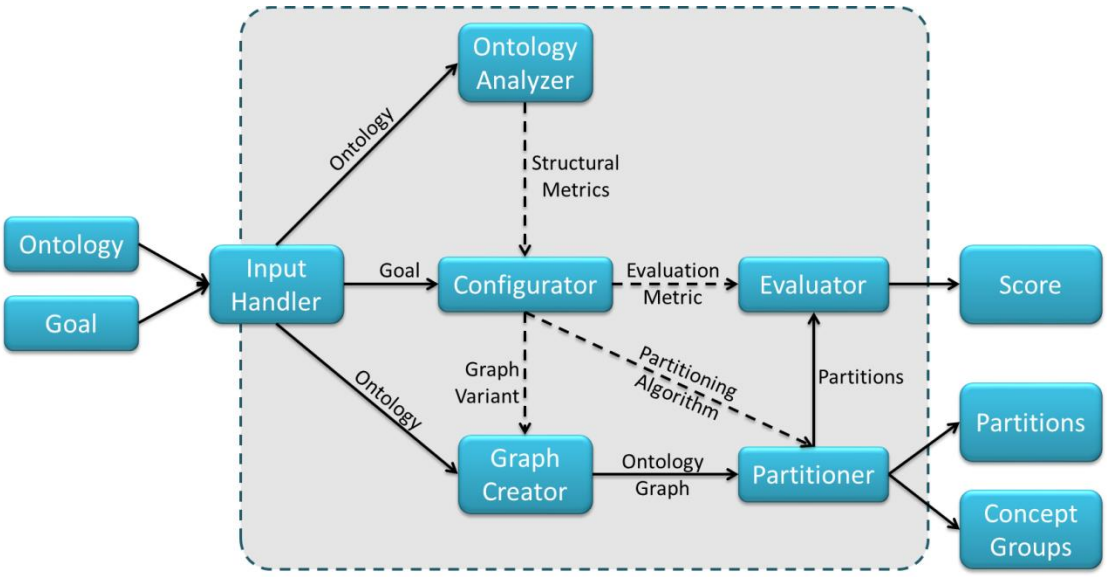


Figure 5-1: Conceptual model of the adaptable ontology partitioning framework. (Dashed arrows represent passing of configuration information. The continuous lines represent passing of user input or an intermediate result.)

Being adaptable for different goals, the ontology partitioning framework requires two input parameters: the ontology and the goal. The Input Handler provides a user interface and allows for uploading the ontology and setting the goal of the partitioning by selecting between "partitioning for creating term chunks" and "partitioning for creating a modular ontology". The Input handler forwards the ontology to the Ontology Analyzer and to the Graph Creator and the goal to the Configurator. The Ontology Analyzer analyses the ontology and passes a map of structural metric values to the Configurator. Based on the goal and the ontology category the Configurator tells the Graph Creator the Graph Variant to use, the Evaluator the evaluation metrics to apply, and the Partitioner the partitioning algorithms to run. The Graph Creator extracts a graph representation and forwards it to the Partitioner, which then partitions the ontology. The partitions are sent to the Evaluator which calculates an overall score for

the partitioning process. The output is a set of partitions or term chunks together with a score. Figure 5-2 illustrates the partitioning process in a step-by-step manner.

Figure 5-2: Presentation of the overall partitioning process in a step-by-step maner

5.2 From Description Logics to Structural Representation

Although the basic logics of ontologies are clearly defined and OWL is established as the de facto standard, the structure of ontologies is defined differently in various existing works (Section 4.1). In a broad sense, the structure of an ontology can be defined as "a set of ontology properties which are not part of the semantic content". Thus, two ontologies describing two different domains might have a similar structure, even though their semantic content is very different. On the other hand, two ontologies describing the same domain might have very different structures, depending on the level of detail, the scope, the perspective of the author etc.

The mostly used language for representing ontologies from a logic-based perspective is Description Logics (DL). OWL and its successor OWL 2 are mainly based on DL. An ontology $\mathcal{O}$, as a DL knowledge base, is defined as $\mathcal{O} = <\mathcal{T}, \mathcal{A}>$, where $\mathcal{T}$ is the schema (TBox), and $\mathcal{A}$ the set of individuals (ABox). The set $\mathcal{C}$ of all class (concept) names in conjunction with the set $\mathcal{P}$ of all property (role) names create the signature Sig($\mathcal{O}$) of $\mathcal{O}$. Based on this it is possible to distinguish among the TBox structure and the ABox structure and to define basic structural properties like the overall number of elements they contain and the number of elements belonging to different types like classes and properties etc. Moreover, complex structural properties can be extracted from the different relationships like hierarchical and domain-range-relationships which exist between these elements. In fact, the structural representation of these interrelations is essential to detect closeness and dependency between the elements and to group them into partitions.

5.2.1 The RDF Graph

The primary data model of the Semantic Web is RDF. As a directed and labeled multigraph, RDF can be considered as one way to represent the structure of an ontology. It enables representing information as triples following the form <subject, predicate, object>. The graph syntax of RDF maps triples to graphs where the subjects and the objects are nodes and the predicates are directed edges (from subject to object). Any ontology expressed in OWL can be translated to an RDF graph and this RDF graph again can be translated to the original OWL ontology (Patel-Schneider & Horrocks, 2004). However, that does not mean that an RDF graph has the same expressivity as an OWL ontology. In fact, an RDF graph can be parsed to an OWL ontology only, if it satisfies the restrictions, which are defined for representing OWL ontologies as RDF graphs. Primarily, this means that the concepts which are defined in OWL have to be used.

The plain RDF graph representation of an OWL ontology makes no distinction between the ABox and the TBox nor between classes, properties, and individuals. It is also not possible to distinguish between local elements and external elements which are defined in other ontologies. The latter issue can be overcome with the assumption that the URIs of local elements belong to the namespace of the ontology, whereas external elements belong to other namespaces.

5.2.2 Class-centric Graph

A second way to represent an ontology structure is the creation of a class-centric graph. In this graph only classes are represented as nodes connected by properties, where the edge direction is from the domain class of the property to the range class of the property. This representation is similar to classic entity-relationship-diagrams. As such it is not lossless. That means, it is not always possible to recreate the original ontology from the class-centric graph representation. However, the main rationale for this representation is that classes are the major objects of an ontology, while the properties can be seen as extensions of those classes relating them with each other. By representing properties as edges and ignoring individuals completely, the class-centric graph is a focused representation enabling better detection of the interdependencies between classes.

As illustrated in Figure 5-3 the class-centric graph is much smaller and therefore less complex than the RDF graph. In this case the set of classes equals the set of nodes, while the set of properties equals the set of edges.

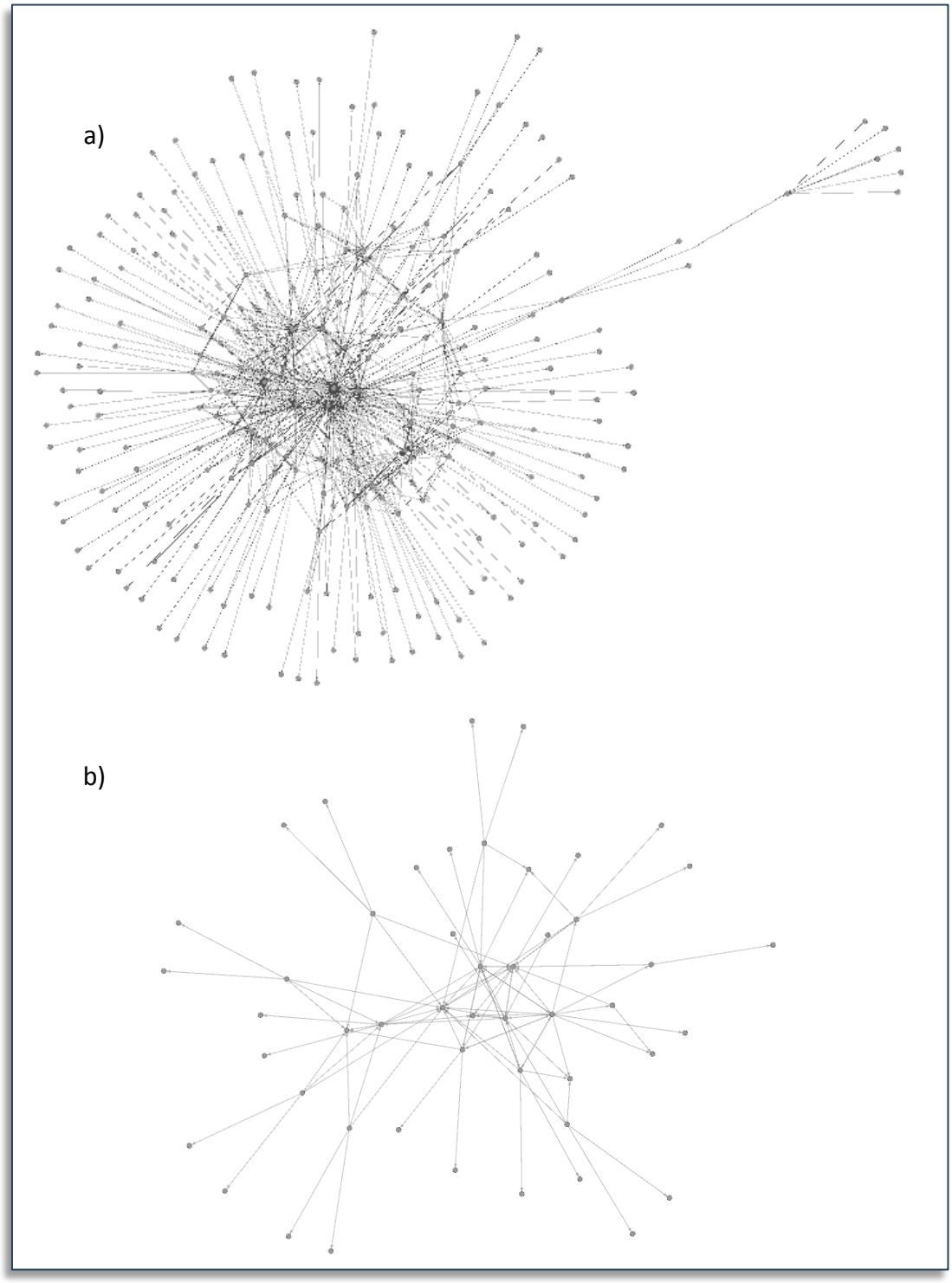


Figure 5-3: Two different structural representation of the Friend-of-a-Friend ontology visualized as graphs. In a) the plain RDF graph is visualized, whereas in b) the class-centric graph is shown.

5.3 Structural Metrics

An adaptable structure-based framework needs metrics to measure the structural properties of an ontology in order to adapt according to them. For the actual partitioning process such metrics are also required for measuring the strength of the connections between the elements and to identify partitions. Furthermore, structural metrics are also necessary to calculate an evaluation score for the partitioning process. That means that metrics are necessary to measure the difference between the original ontology and the outcome of the partitioning process. As described in Section 4.1, numerous metrics are already introduced and used. Each work, however, is based on its own understanding of ontology structure. In most cases there are open questions like: Are the metrics based on the inheritance tree only? Is it assumed that the artificial upper concept "Thing" exists? Are instances, reasoning and the import mechanism of OWL taken into account? Furthermore, the wording that is used is confusing, as in some papers object type properties are called "relations" or "properties" and data type properties are called "attributes", whereas in other works object type properties are called "relations" and data type properties are called just "properties".

In order to avoid ambiguity, formal definition for the metrics to measure the structural features of an ontology are provided in this section. The formulas are based on the basic notation used in Section 5.2. Some metrics have been proposed before. However, there is also a set of new metrics, which are introduced in this work. For the description, the wording which is commonly used in the documentation of OWL is applied. The metrics are classified into the following categories:

- **Size Metrics** are used to count the number of different elements of an ontology.

- **Hierarchy-based Metrics** are used to measure features of the inheritance tree like depth and paths.

- **Complex Metrics** give insights into numerical relations between different types of elements and the hierarchical properties. These metrics unveil the overall complexity of the ontology

The only assumption for these metrics is the existence of the artificial upper class "Thing". This class should be inserted if the ontology does not already contain it. It is inserted as the superclass of all classes, which do not have an explicitly defined superclass. Apart from that assumption, it is important to keep in mind that depending on the loading strategy one metric might have different values for the same ontology. The loading strategy has three important properties. The first one is how imports within

an ontology are treated. That means, if only the explicitly formulated content of an ontology file is loaded or if imported ontologies are loaded as well. Secondly, it is part of the loading strategy, if a reasoner is applied on the loaded ontology, which materializes the ontology. If this is the case, it needs to be clarified what kind of reasoner is applied and which expressivity level or inference rules have been used. Finally, the third property is how individuals are treated. For the analysis of the ontology schema individuals might be irrelevant and could be ignored.

In order to have a quick overview about the metrics, Table 5-1 lists all defined metrics at the end of this section. It provides information about metrics which have been introduced in previous work along with the local names or abbreviations. It enables to see which metrics were already used and which have been introduced in this work.

5.3.1 Size Metrics

In various works in literature on ontology structure, size is a common structural property. Schlicht and Stuckenschmidt as well as d'Aquin et al. propose size as one structural criteria for evaluating ontology modularization techniques (d'Aquin, Schlicht, Stuckenschmidt, & Sabou, 2009; Schlicht & Stuckenschmidt, 2006). Oh et al. it is stated that

> "*size is an important metric, because it has a strong influence on the maintainability, robustness, and evolution of the application relying on it.*" (Oh, Yeom, & Ahn, 2010)

On the one hand, large modules tend to lose flexibility regarding their evolution and exploitation (d'Aquin et al., 2009). On the other hand, too many and too small modules are not appropriate as well. Because they would not cover one domain sufficiently and would demand even more management effort in order to keep all modules synchronized and consistent.

Although size seems to be a very trivial property at the first glance, due to the complexity of ontologies there is a set of different properties which can be regarded as a dimension of the ontology size. Zhang et al. define for example the size of a vocabulary as the number of named entities (Zhang et al., 2010), whereas d'Aquin et al. define size as the number of all elements without a restriction, if they are named or not (d'Aquin et al., 2009). For the sake of clarity, a set of definitions for different size metrics are provided in the following. These metrics are mainly based on the different types of elements which are illustrated in Figure 5-4.

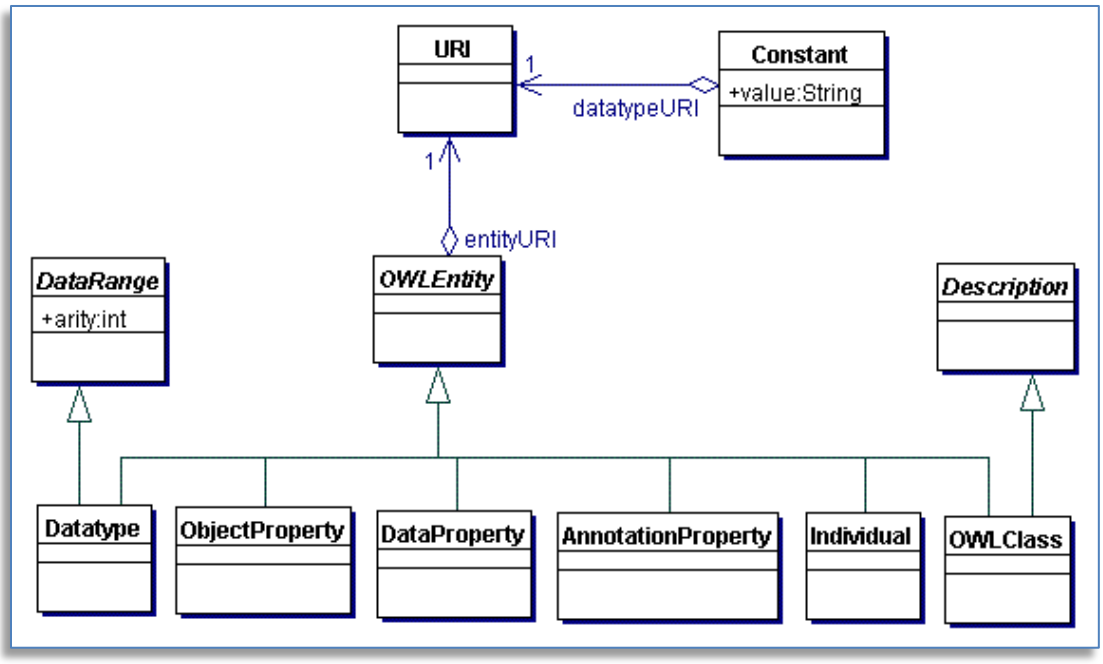


Figure 5-4: UML class diagram representing the entities defined in OWL. (Picture is taken from (Motik, Patel-Schneider, & Horrocks, 2008))

Number of named classes

Classes are the basic elements of ontologies. Thus, they are mostly defined at the very beginning of the ontology creation process (e.g. called glossary in METHONTOLOGY; see Section 2.2.1). The number of named classes (NoNC) defined in an ontology indicates the level of detail or the spectrum of an ontology. This metric is also used in (YANG et al. 2006; Ning & Shihan 2006a) and is defined as

$$NoNC = |\, \boldsymbol{C}_N \,| \qquad \boldsymbol{C}_N \; is \; the \; set \; of \; all \; defined, named \; classes \qquad (1)$$

Number of anonymous classes

Classes which do not have a label are called anonymous classes and are represented as blank nodes in RDF. Indicating that something exists without assigning a label for it they are mainly used for representing complex concepts. Thus, they have an important impact on the complexity of ontologies. The comprehension of anonymous classes within the ontology is not easy, because they have no own meaning but get a meaning through the combination of other classes. That means that anonymous classes add

semantics to the ontology by linking other elements of the ontology. The number of anonymous classes (NoAC) is an important metric to measure the size and is defined as

$$NoAC = |\ \boldsymbol{C}_A\ | \qquad \boldsymbol{C}_A\ is\ the\ set\ of\ all\ anonymous\ classes \tag{2}$$

Number of classes

The sum of the number of anonymous classes and the number of named classes is the overall number of classes (NoC) that are defined within an ontology.

$$NoC = NoNC + NoAC \tag{3}$$

Number of external classes

Classes which are defined in an external ontology but are used to define classes and properties within an ontology are external classes. The number of external classes (NoEC) indicates the dependency of an ontology from other ontologies. The higher the number of external classes the more complex the ontology is, because external classes might demand for understanding of additional ontologies. Usually, the elements of an ontology have the same namespace. Therefore elements with a namespace other than the defined one are considered as external elements. This metric is also used in (Orme et al., 2006) and is defined as

$$NoEC = |\ \boldsymbol{C}_E| \quad \boldsymbol{C}_E\ is\ the\ set\ of\ external\ classes \tag{4}$$

Number of data type properties

Data type properties extend classes with well-defined data types like xsd:string. This allows to add literals to individuals of those classes. The higher the number of defined data type properties the more detailed an ontology is and the more knowledge it contains. The number of data type properties (NoDP) which are defined within an ontology is defined as

$$NoDP = |\ \boldsymbol{P}_{DT}\ | \qquad \boldsymbol{P}_{DT}\ is\ the\ set\ of\ all\ data\ type\ properties \tag{5}$$

Number of object type properties

Object type properties are class extensions which are connecting two classes with each other. On the individuals level this means that individuals of one class might be

connected with corresponding individuals from the other class. A high number of object type properties defined within an ontology might be an indication for dense connectivity between the classes and high cohesion for the overall ontology. The number of object type properties (NoOP) which are defined within an ontology is defined as

$$NoOP = |\, \boldsymbol{P}_{OT} \,| \qquad \boldsymbol{P}_{OT}\ is\ the\ set\ of\ all\ object\ type\ properties \tag{6}$$

Number of properties

The sum of the number of data type properties and the number of object type properties which are defined within an ontology is the number of properties (NoP). Considering properties as extensions of classes the number of properties may indicate the level of detail for the classes, the complexity of the classes, and the complexity of the domain. This metric is defined as

$$NoP = NoDP + NoOP \tag{7}$$

Number of annotation properties

Annotation properties allow to add meta-information to an ontology to support the user for better understanding the ontology or to understand its relation to other ontologies. Because annotation properties do not have semantics they are ignored in any reasoning processes. Therefore annotation properties are not taken into account in the calculation of the overall number of properties (NoP). The number of defined annotation properties (NoAP) is defined as

$$NoAP = |\, \boldsymbol{P}_{AP} \,|\ \ \boldsymbol{P}_{AP}\ is\ the\ set\ of\ all\ defined\ annotation\ \text{properties} \tag{8}$$

Number of annotations

By using annotation properties meta-information are added to ontologies. Standard annotation properties like rdfs:comment and rdfs:label are used to improve the readability of an ontology. A similar metric is also used in (Tartir et al., 2005) which is called "Readability" and considers only rdf:comment and rdfs:label properties. The number of used annotation (NoA) is defined as

$$NoA = |\, AN \,| \qquad AN\ is\ the\ set\ of\ all\ \ annotations \tag{9}$$

Number of individuals

Classes and properties of ontologies can be instantiated by concrete individuals. The number of individuals (NoI) within an ontology indicates the number of concrete entities within a domain of interest. Hence, this metric can be considered as an overall size metric for the ABox and as such for the whole domain. This metric is defined as

$$NoI = |\boldsymbol{A}| \qquad \boldsymbol{A}\ is\ the\ set\ of\ all\ individuals \tag{10}$$

Number of named entities

The named entities of an ontology are those elements which have concrete labels. It is the sum of the number of named classes, the number of properties, and the number of individuals. The number of named entities (NoNE) was already used in (H. Zhang et al. 2010) where it was considered as the overall the size of the ontology vocabulary. It is defined as

$$NoNE = NoNC + NoP + NoI \tag{11}$$

Number of references to external classes

In contrast to NoEC, the number of references to external classes (NoREC) is not counting the number of distinct used external classes but the number of the references to those. The more references to external classes exist the higher is the connection density among local and external elements and the stronger is the dependency of the ontology on external classes. Therefore this metric measures the degree of coupling. It is also used in (Orme et al., 2006) and is defined as

$$NoREC = \left|R_{\boldsymbol{C_E}}\right| \quad R_{\boldsymbol{C_E}}\ is\ the\ set\ of\ references\ to\ external\ classes \tag{12}$$

Number of Imported Ontologies

OWL defines an import mechanism which allows to incorporate external ontologies completely into a new ontology. The number of imported ontologies (NoIO) is the number of explicitly included ontologies at the beginning of the description of the new ontology. Because the external ontologies are imported completely any increase in this metric might have a huge impact on the overall size and complexity of an ontology. Together with NoEC and NoREC this metric is an additional coupling metric that is also used in (Orme et al., 2006) and is defined as

$$NoIO = |\, \boldsymbol{O}_i \,| \quad \boldsymbol{O}_i \; is\; the\; set\; of\; included\; ontologies \tag{13}$$

5.3.2 Hierarchy-based Metrics

Hierarchical properties are a specific kind of transitive object type properties which are connecting two classes with each other. This kind of properties is expressing inheritance relation between two classes. The set of individuals of a class is a subset of the individuals of the corresponding superclass. Since a cyclic inheritance relation would lead to an equivalence relation the inheritance structure within an ontology should avoid cycles and should build a tree structure. Within the inheritance tree each class has a unique position. Utilizing the position as well as generic properties of trees such as depth and path the following hierarchy-based structural metrics are defined:

Number of root classes

Classes which do not have an explicitly defined superclass are subclasses of the Thing-class. They are the roots of the different branches of the inheritance tree. Hence the number of root classes (NoRC) indicates the spectrum of an ontology. This metric is also used in (Yao et al. 2005) and is defined as

$$NoRC = |\, \boldsymbol{C}_{RC} \,| \quad \boldsymbol{C}_{RC} \; is\; the\; set\; of\; all\; root\; classes \tag{14}$$

Number of leaf classes

Classes which do not have an explicitly defined subclass are leafs of the inheritance tree. Classes which neither have an explicitly defined superclass nor a subclass can be considered as leaf classes and root classes at the same time. Therefore there might be an intersection between the set of root classes and the set of leaf classes. The number of leaf classes (NoLC) is an indication for the level of detail. It is also used in (Yao et al. 2005) and is defined as

$$NoLC = |\, \boldsymbol{C}_{LC} \,| \quad \boldsymbol{C}_{LC} \; is\; the\; set\; of\; all\; leaf\; classes \tag{15}$$

Number of inheritance paths per class

The path in an inheritance tree is the sequence of inheritance relationships between a class and one corresponding superclass. Since in OWL a class may have more than one direct superclass (multiple inheritance) each class may have more than one root classes and therefore several paths to its corresponding root class(es). The amount of different

paths from a class to its corresponding root classes is the number of inheritance paths per class ($NoIP_i$). If a class has more than one path it inherits from different branches of the tree. That in turn exacerbates the comprehension of the class. Therefore this metric is a measure for the complexity of a class. It is also used in (Yang et al., 2006) and is defined as

$$NoIP_i = \left| path\left(\boldsymbol{C}_i, \boldsymbol{C}_{RC_j}\right)\right| \quad \boldsymbol{C}_{RC_j}\ are\ the\ root\ classes\ of\ class\ C_i \qquad (16)$$

Average depth of a class

The number of inheritance relationships between two classes is the length of a path. (The path length between a class and its direct superclass is 1.) The sum of the length of different paths from a class to its corresponding root classes divided by the number of paths is called the average depth of the class ($ADoC_i$). A high value for this metric means that a class has many superclasses from which it inherits. For that reason a high value for this metric indicates high complexity. This metric is also used in (Yang et al., 2006) and is defined as

$$ADoC_i = \frac{\sum_j length(\,path\left(\boldsymbol{C}_i, \boldsymbol{C}_{RC_j}\right))}{NoIP_i} \qquad \boldsymbol{C}_{RC_j}\ as\ in\ NoIP_i \qquad (17)$$

Maximum depth of a class

The longest path from a class to its corresponding root classes is the maximum depth of a class ($MDoC_i$). The higher the value for this metric the more interdependent superclasses a class has. That means a high value indicates a complex inheritance chain which increases the complexity of the class. This metric is also used in (Yang et al., 2006) and is defined as

$$MDoC_i = MAX\left(path\left(\boldsymbol{C}_i, \boldsymbol{C}_{RC_j}\right)\right) \qquad \boldsymbol{C}_{RC_j}\ as\ in\ NoIP_i \qquad (18)$$

Number of inheritance paths

The sum of the amount of different paths from all classes to their corresponding root classes is called the number of inheritance paths (NoIP). The more inheritance paths an ontology contains the more independent inheritance chains exist. A high value for this metric indicates increased complexity of the ontology. It is defined as

$$NoIP = \sum_{i}^{NoC} NoIP_i \quad (19)$$

Average path per class

The number of inheritance paths divided by the number of classes is the average path per class. If an ontology contains single inheritance only the value for this metric will be 1. As mentioned before multiple inheritance is more difficult to comprehend. Therefore an increased value for this metric indicates higher complexity of the ontology. This metric is also used in (Yang et al., 2006) and is defined as

$$APpC = \frac{NoIP}{NoC} \quad (20)$$

Depth of an ontology

The longest inheritance path within the inheritance tree of an ontology is the depth of the inheritance tree (DoO). The value for this metric represents the highest leaf-root distance within an ontology. It might indicate the level of detail of an ontology, because the longer this distance is the finer is the modeling granularity between the leaf class and its corresponding root class. This metric is also used in (Yang et al., 2006) and in (Duque-Ramos et al., 2011). It is defined as

$$DoO = MAX\ (MDoC_i) \quad (21)$$

Average depth of leaf classes

The sum of the length of different paths from each leaf class to their corresponding root classes divided by the number of paths is called the average depth of an ontology (ADoLC). This metric was introduced in (Yao et al., 2005) and was used in (Duque-Ramos et al., 2011) where the artificial root class THING ($C_{RCj} = THING$) was used. However, this modification just increases the value by one. ADoLC is defined as

$$ADoLC = \frac{\sum_{ij} length(\ path\ (\boldsymbol{C_{LCi}}, \boldsymbol{C_{RCj}})\ )}{NoIP} \quad (22)$$

Number of children

The number of direct subclasses of a class is called number of children (NoC_i). The more subclasses a class has the higher is its importance, because it has a connecting function

between more elements and a change on this class will affect all its subclasses. This metric is also used in (Zhang et al., 2010) and is defined as

$$NoC_i = |subclass(\boldsymbol{C}_i)| \quad (23)$$

Number of subclass relations

All explicitly defined subclass relations within an ontology is called the total number of subclass relations (NoSC). Since all edges in the inheritance tree are subclass relations this metric measures the number of edges in the tree. For that reason it can be seen as a size metric for the inheritance tree. It is also used in (Yang et al., 2006) and is defined as

$$NoSC = \sum_i^{NoC} |subclass(\boldsymbol{C}_i)| \quad (24)$$

Average number of subclass relations

The average number of explicitly defined direct subclasses per class is the average number of subclasses (AoSC). In the inheritance tree this corresponds to the edge to node ratio. A high value indicates high fan-out of the inheritance tree. That means a high number of different branches. This metric is also used in (Tartir et al., 2005), in (Yang et al., 2006) and in (Duque-Ramos et al., 2011). It is defined as

$$AoSC = \frac{NoSC}{NoC} \quad (25)$$

Average number of subclasses per class

The average number of subclasses within a subclass-tree of a class is the average number of subclasses per class ($AoSC_i$). This metric measures the centrality and the importance of a class in the inheritance tree. The higher the value the more branches the subtree will have. This metric indicates the level of abstraction of a class. It also used in (Tartir et al., 2005) and is defined as

$$AoSC_i = NoSC\ (subtree(\boldsymbol{C}_i)) \quad (26)$$

Tree Impurity

Single inheritance relations lead to a tree structure. In contrary, multiple inheritance causes a graph structure. Because a tree with n nodes has n-1 edges the value for this

metric measures the number of multiple inheritance relations which differentiate the inheritance structure from a tree. Since single inheritance leads to a linear inheritance direction it is easier to comprehend than multiple inheritance which demands for following the inheritance from parallel point of views. Hence the higher the value for this metric is the more compact but at the same time more complex the inheritance structure becomes. This metric is also used in (Zhang et al., 2010) and is defined as

$$TIP = \ NoSC - NoC + 1 \quad (27)$$

5.3.3 Complex Metrics

Combinations of the aforementioned basic and hierarchical metrics lead to more complex metrics which give a deeper insight into the overall structure of ontologies. These combinations can be considered as normalized metrics which do not depend on the value of one metric but are measuring the relation between different metrics.

Object type property distribution

The object type property distribution (OTPD) is the average number of object type properties per class. Since object type properties are connecting two classes with each other, a high value is an indication for dense connectivity between the classes of an ontology and might be an indication for high cohesion. Dense connectivity between classes increases the complexity of an ontology, because it necessitates to comprehend the relationships between the different classes. This metric is also used in (Li, Motta, & D'Aquin, 2010) and in (Duque-Ramos et al., 2011). It is defined as

$$OTPD = \frac{NoOP}{NoC} \quad (28)$$

Object type propertys' standard deviation

The object type property standard deviation (OTPSD) is the average deviation of the number of properties of a class different from the OTPD. It indicates how the distribution of the properties over the classes are. A high value of OTPSD is a strong indication for an unbalanced distribution of the data type properties over the classes. This means, that most of the data type properties belong to few classes which can be considered as central and important classes. If the object type properties are distributed equally over the classes the value will be 0. This metric is defined as

$$OTPSD = \sqrt{\frac{\sum_i^{NoC}(\boldsymbol{P}_{OTC_i} - OTPD)}{NoC}} \quad (29)$$

Data type property distribution

The datatype property distribution (DTPD) is defined as the average number of data type properties per class. The higher the number of data type per classes is the more detail is known about that class. For that reason, this metric can be seen as an indication for the level of detail of an ontology and hence the amount of included knowledge described. It is also used in (Tartir et al., 2005) and in (Duque-Ramos et al., 2011). It is defined as

$$DTPD = \frac{NoDP}{NoC} \quad (30)$$

Data type propertys' standard deviation

This metric is similar to OTPSD but belonging to data type properties and not to object type properties. It measures the distribution of the defined data type properties over the classes. If few classes have many data type properties while the most classes have less or none data type properties, the value of for this metric will be high. In this case, the classes with the most data type properties can be considered as the most detailed classes and therefore as the most important classes in the ontology. This metric is defined as

$$DTPSD = \sqrt{\frac{\sum_i^{NoC}(\boldsymbol{P}_{DTC_i} - DTPD)}{NoC}} \quad (31)$$

Properties distribution

Data type properties and object type properties increase the level of detail of an ontology. The more properties an ontology contain the more powerful is its semantic expressivity. That means that the ontology is able to represent more knowledge which leads to more complex ontologies. The distribution of the properties over the classes of the ontology (PD) measures the average property per class within an ontology. Therefore it can be seen as a measure for the complexity of an ontology. This metric is also used in (Duque-Ramos et al., 2011) and is defined as

$$PD = \frac{NoP}{NoC} \quad (32)$$

Properties' standard deviation

The properties' standard deviation (PSD) measures how the number of property per class differs from the average number of properties per class. A high value for this metrics indicates an unbalanced distribution of the properties over the classes. If this is the case few classes will have the most properties which can then be considered as the most important and central classes of an ontology. This metric is defined as

$$PSD = \sqrt{\frac{\sum_i^{NoC}(\boldsymbol{P}_{C_i} - PD)}{NoC}} \quad (33)$$

Datatype to objecttype property ratio

Object type properties connect two different classes which each other while data type properties are extensions of a class of simple types. The ratio between the number of defined data type properties and the number of defined object type properties (DOR) reveals the authors' preferred way of adding details to classes. A high value might indicate a domain comprising mainly objects with simple attributes and low interdependency between the elements. This metric is defined as

$$DOR = \frac{NoDP}{NoOP} \quad (34)$$

Annotation distribution

The distribution of the annotations within an ontology (AD) indicates the richness of an ontology with respect to meta-information. A high value for this metric might indicate a complex domain as well as a complex ontology which is difficult to understand. On the other hand the annotations might be helpful in understanding the content of the ontology. This metric is also used in (Duque-Ramos et al., 2011) and is defined as

$$AD = \frac{NoAP}{NoC} \quad (35)$$

Property richness

The property richness (PR) was introduced in (Tartir et al., 2005) and is defined as the ratio between the defined properties connecting classes and data types and the subclass

relations between classes. If most connection between classes are subclass relation the overall structure of the ontology will be more tree like, whereas a high number of properties will lead to a higher value for this metric indicating a more complex graph structure of the ontology. This metric is defined as

$$PR = \frac{NoP}{NoSC+NoP} \quad (36)$$

Metric	(Yao et al., 2005)	(Tartir et al., 2005)	(Yang et al., 2006)	(Orme et al., 2006)	(Ning & Shihan, 2006)	(Zhang et al., 2010)	(Duque-Ramos et al., 2011)
NoNC			TNOC		Concept Quantity		
NoAC							
NoC							
NoEC				NEC			
NoDP							
NoOP							
NoP							
NoAP		Rd					
NoI							
NoNE						SOV	
NoREC				REC			
NoIO				RI			
NoRC	NOR						
NoLC	NOL						
$\mathbf{NoIP}_i$			TNOP				
$\mathbf{ADoC}_i$			$\bar{\lambda}_i$				
$\mathbf{MDoC}_i$			λ_i			DIT	
$\mathbf{NOIP}_{LC_i}$							
NOIP							
APpC			ρ				
DoO			Λ				DITOnto
ADoLC	ADIT-LN						LCOMOnto
$\mathbf{NoC}_i$						NoC_c	
NoSC			TNOR				
AoSC		IR_s	μ				NOC/CBOOnto
$\mathbf{AoSC}_i$		IR_c					
TIP						TIP	
OTPD					PE		INROnto
OTPSD					PSD		
DTPD		AR					NOMOnto
DTPSD							
PD							WMCOnto
PSD							
DOR							
AD							ANOnto
PR		RR					

Table 5-1: Overview of the mapping of the metrics to related work

5.4 Community Detection Algorithms

People and their interrelations like friendship and colleague are so called social networks. These networks can be represented by graphs, where people are nodes and the relations are edges (Newman, 2003). A group of people who have more interrelations between each other than to other people are called communities. Within the graph representation such communities built subgraphs which have a higher density than the overall graph. Therefore, detecting communities in social networks corresponds to the problem of finding subgraphs which are internally densely connected but have fewer connections between each other. A straightforward way to find those subgraphs would be to calculate each possible partitioning and to measure to what extend they comply with being a community. The partitioning with the best result would then be chosen. However, this approach leads to two different problems.

5.4.1 Modularity Metric

Firstly, a metric has to be defined which measures the community degree of a graph partitioning. The requirements for this measure are perfectly described by Reichardt and Bornholdt as follows:

> "[…] communities are understood as groups of densely interconnected nodes that are only sparsely connected with the rest of the network. Any quality function for an assignment of nodes into communities should therefore follow the simple principle: group together what is linked, keep apart what is not. From this, we find four requirements of such a quality function: it should (i) reward internal edges between nodes of the same group (in the same spin state) and (ii) penalize missing edges (nonlinks) between nodes in the same group. Further, it should (iii) penalize existing edges between different groups (nodes in different spin state), and (iv) reward nonlinks between different groups." (Reichardt & Bornholdt, 2006)

Since a community is defined as a set of nodes with high density the metric has to be based on the distribution of the density over the graph. In this regard Newman and Girvan define a so called modularity metric for simple graphs in the following way: A is an adjacency matrix representing a simple graph where A_{vw} equals 1 if the node v and the node w are connected and A_{vw} equals 0 if v and w are not connected (Newman & Girvan, 2004). With c_v indicating that v belongs to c_v and $\delta(c_v, c_w)$ = 1 if v and w belong to the same community (means $c_v = c_w$) and $\delta(c_v, c_w)$ = 0 if v and w belong to

different communities the following equation is the ratio between the in-community edges and the overall number of edges (ICER)

$$ICER = \frac{\sum_{vw} A_{vw}\, \delta(c_v, c_w)}{\sum_{vw} A_{vw}} = \frac{1}{2m} \sum_{vw} A_{vw}\, \delta(c_v, c_w) \quad (37)$$

Since this would equal 1 in case of only one community comprising the whole graph it is not an accurate metric yet. It is necessary to set this metric in relation to the distribution of the edges in case of random networks. In random networks the probability that an edge exists between the nodes v and w equals $k_v k_w / 2m$ with $k_v = \sum_w A_{vw}$ being the number of incident edges upon v. This leads to the following metric for modularity (MOD):

$$MOD = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) \quad (38)$$

This metric measures how the ratio of in-community edges over the overall number of edges differs from the edge distribution within a random edge distribution. A higher density is represented by a positive number while a lower density stands for a lower density. In case of MOD = 0 the edge distribution corresponds to a random edge distribution. A value above 0.3 is considered to indicate significant community structure (Clauset, Newman, & Moore, 2004). With some modification the equation can be simplified to the following form (Clauset et al., 2004):

$$MOD = \sum_i \left(e_{ii} - a_i^2 \right) = Tr\, \boldsymbol{e} - \|\boldsymbol{e}^2\| \quad (39)$$

In the formula $\boldsymbol{e}$ is a symmetric *k* x *k* matrix representing the connections between *k* subgraphs of a graph. e_{ij} corresponds to the ratio of edges between the subgraphs i and j over the overall number of edges. The expected connectivity between the subgraphs in a comparable random network is a_i^2 so that $(e_{ii} - a_i^2)$ corresponds to the deviation of fraction of edges within the subgraph i from the expected value. Furthermore, $\|\boldsymbol{x}\|$ is the sum of all elements of the matrix $\boldsymbol{x}$ and the Trace $TR\, \boldsymbol{e} = \sum_i e_{ii}$ corresponds to the fraction of edges that are within subgraphs.

Newman shows that this metric is also applicable on weighted graphs by transforming the graph into a multigraph and introducing a unit r to handle non-integer values (Newman, 2004a). However, in case of directed graphs this metrics needs a modification that is shown in (Leicht & Newman, 2008). The metric needs to reflect that $e_{vw} = e_{wv}$ is not valid in a directed network and that the probability $k_v k_w / 2m$ has to

be changed to $k_v^{in} k_w^{out}/m$. Applied on the modularity metric from (39) this leads to the following modularity metric for directed networks ($\mathrm{MOD_d}$):

$$MOD_d = \frac{1}{m} \sum_{vw} \left[A_{vw} - \frac{k_v^{in} k_w^{out}}{m} \right] \delta(c_v, c_w) \qquad (40)$$

5.4.2 Partitioning Algorithms

The second main problem is that calculating each possible partition for a graph is computationally a very complex task, because it means to find all possible subsets for the set of nodes and then to find all possible subsets for the set of edges for each of the subsets of nodes. Even the complexity of finding all possible subsets for a set with *n* elements is 2^n. Brandes et al. show that maximizing the modularity is computationally an NP-complete problem (Brandes et al., 2007). Therefore, there have been different proposals seeking to find the best solution without calculating every possible partitioning.

In classic graph partitioning approaches the targeted number of partitions is known and the main goal is to divide the original graph into this number of partitions. Thereby the main focus is on minimizing the connections between the modules and if possible to obtain similar sized partitions. An example is the division of a computational problem for parallel computing in a computer cluster. In this regard partitioning is a kind of a preprocessing for the actual goal.

On the contrary, in the social network analysis approach the goal is to detect the intrinsic partitioning that is supposed to exist in a social network naturally (Newman & Girvan, 2004). These approaches are called hierarchical clustering techniques because they create a hierarchical structure (Xu & Wunsch, 2005). This is mostly represented by a specific tree representation called dendrogram. An example is shown in Figure 5-5. The leaf nodes are representing the nodes of the graph and the intermediate nodes are representing join-points where the elements (nodes or communities) from lower levels are merged together to build increasingly larger communities. That means that each level represents an intermediate partitioning result. Hierarchical techniques can be divided into two different categories.

Firstly, agglomerative approaches start with creating partitions which contain just one node. These clusters are then merged to bigger clusters. Regarding the visualization in Figure 5-5 an agglomerative algorithm would progress from the right part of the tree to the left part. The challenge is to find in each step the next two elements or clusters to merge. For that purpose metrics are necessary which measure the relatedness between them so that the next “meaningful” merge is detected.

Secondly, divisive approaches are working in the opposite direction, namely from the left part to the right part of the Figure 5-5. That means that divisive approaches create one big cluster at the beginning containing all elements. In every step each cluster from the previous step is divided into two new clusters. Therefore, in each step the challenge is to find two subgraphs which have low interrelations.

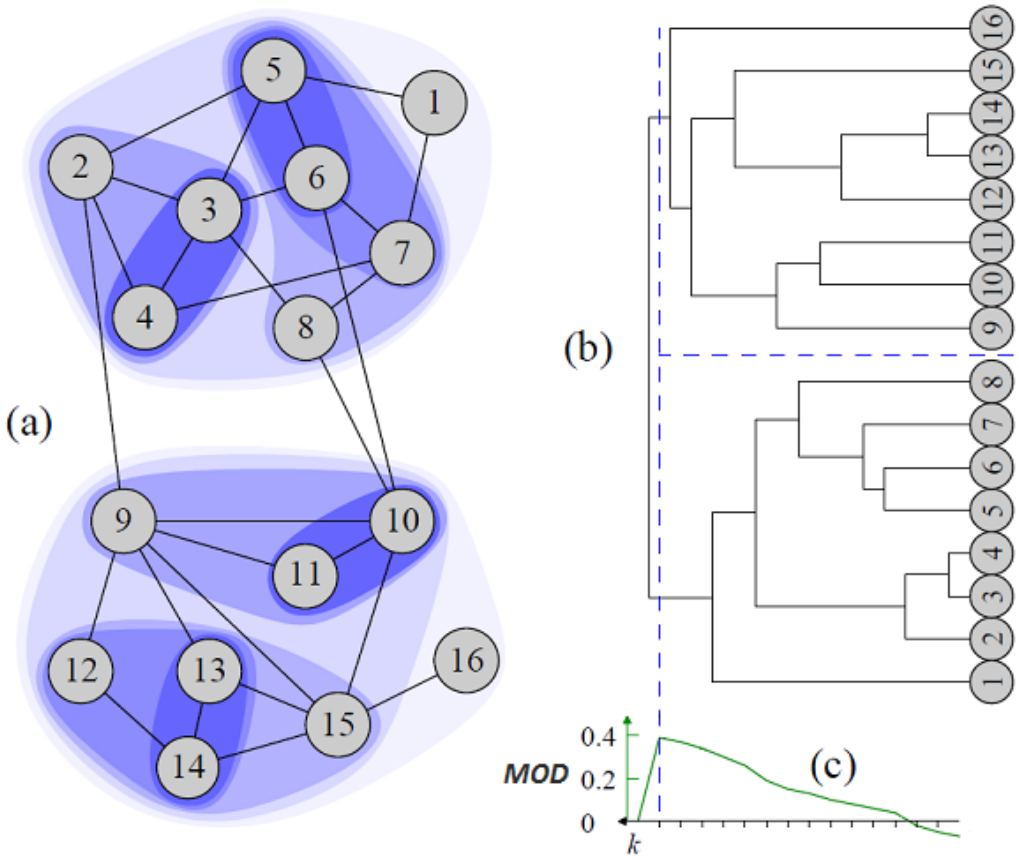


Figure 5-5: Graphical illustration of community detection algorithms. In a) an example for a network with community structure is visualized. Part b) visualized a dendrogram representing a hierarchical clustering for the visualized graph. Finally, c) shows the modularity function for the dendrogam is shown. The maximum of MOD is achieved at the level with two partitions. (This figure is taken from (Pons & Latapy, 2006) with minor modifications.)

There are various agglomerative and divisive techniques which partition a graph based on different metrics. A comprehensive description can be found in (Fortunato, 2010). For this work the following simple algorithms have been chosen which are described briefly:

Edge Betweenness Community

The Edge Betweenness Community (ebc) algorithm introduced by Newman and Girvan is a divisive hierarchical clustering algorithm which focuses on the edges (Newman & Girvan, 2004). Its basic idea is that a network comprises densely connected communities which in turn are sparsely connected. By calculating the shortest paths between each node pair the edge with the highest betweenness, which is likely to be connecting two communities, can be identified and removed. In each step of this algorithm the betweenness of each edge is calculated and only the one with the highest betweenness is removed.

Fast Greedy Community

The Fast Greedy Community (fgc) algorithm was initially introduced by Newman (Newman, 2004b). It is an agglomerative hierarchical algorithm that actually is an optimizing algorithm which uses the modularity function as a fitness function. In each step the next best merge is chosen by taking only connected nodes into consideration. This focus leads to an efficient algorithm. The performance was even improved by Clauset et al., which makes use of a more efficient data structure, because the matrix is reduced after each step (Clauset et al., 2004).

Walktrap Community

Pons and Latapy are proposing an algorithm, which is based on the idea that "random walks on a graph tend to get trapped into densely connected parts corresponding to communities" (Pons & Latapy 2006). For that reason, this algorithm is called Walktrap Community (wtc). Based on the degree of the nodes a probability function for random walk routes is calculated. These routes are then used to define a distance measure between nodes. This algorithm is an agglomerative hierarchical clustering algorithm.

5.5 Weighting Semantic Relations

Even though relations between people might be of different types like friend or colleague, in classic graph representations of social networks the edges have no type. In contrast, the edges between nodes in an ontology are of different types which stand for different semantic relations. That means that each edge has a meaning. This additional information is worth to be taken into consideration within the partitioning process. Because the main goal is to divide an ontology into subdomains so that the concepts within a partition are stronger related semantically to each other than to concepts from other partitions. E.g. an "equivalentClass" relation between two classes A and B

expresses semantic equivalence. Therefore, it is a strong indication that A and B belong to the same subdomain and should not be separated in the partitioning process.

In order to insert this additional information into the partitioning process, quantification is necessary which measures the strength or weakness of a relation between classes. For that purpose it is proposed to introduce a weight function that assigns weights to edges based on their semantics. These weights are then taken into account within the partitioning algorithm or in the modularity function or in both.

For the definition of a weight function the first question to clarify is the following: Which properties should be taken into consideration? Instead of defining a weight for each possible property of RDFS and OWL, it is decided to select the most used properties. For that purpose, all statements of LOD ontologies have been extracted and the predicates have been analyzed. Table 5-2 shows the results of this analysis.

Property	Used in % of all statements	Used in # of ontologies
rdf:type	17.2	301
rdfs:label	11.1	283
rdfs:rest	6.6	108
rdfs:first	6.6	108
rdfs:isDefinedBy	6.6	166
rdfs:comment	5.5	282
rdfs:range	5.3	264
rdfs:domain	5.1	257
owl:class	4.6	236
rdfs:subClassOf	3.2	242

Table 5-2: Usage analysis of the RDFS and OWL properties. The first column shows, in how many of the overall number of statements the corresponding property has been used. The second column shows, in how many ontologies the corresponding property occurs.

The most used properties are rdfs:type (used in 17.2% of all statements) for instance definition and rdfs:label (used in 11.1% of all statements) for adding labels to defined resources. Both properties are used in most of the analyzed ontologies. The properties rdfs:rest, rdfs:first and rdfs:isDefinedBy are each used in 6.6 % of all statements. However, these properties are used in only one third to one half of all ontologies. This indicates that a large number of ontologies do not contain these properties at all, whereas in other ontologies they appear extensively. The annotation property

rdfs:comment is used in 5.5% of all statements and appears in 282 ontologies. The rdfs:range and the rdfs:domain properties which are used for defining properties are used in 5.3% and 5.1 of all statements and appear in 264 and 257 ontologies, respectively. The definition of a class with owl:class is used in 4.6% of all statements and appears in 236 ontologies, while the definition of a subclass with the rdfs:subclassOf property is used in 3.2% of all statements and appears in 242 ontologies.

For the weighting function the annotation properties rdfs:label and rdfs:comment have been ignored, because they are used for human readers and do not add semantics to the ontology. The properties rdf:type, rdfs:isDefinedBy and owl:Class are ignored because they do not play any role in the schema of an ontology. The properties rdfs:first and rdfs:rest have been also ignored. On the one hand they are not used in about two third of all ontologies and would not have any effect in most cases. On the other hand they do not really add significant semantics to the schema, apart from bringing order to a sequence of resources. The left three properties are used for the definition of the proposed weighting function which contains three different categories for weights.

Non-hierarchical relations

The first category of properties are those properties which are - based on the previous analysis - accepted to have the most impact, namely rdf:range and rdf:domain. Despite the fact that the owl:equivalentClass is not one of the most used properties and was consequently not listed in Table 5-2, it is also considered within this category. This has been done because the equivalence between two classes is considered as a very strong relation indicating that two classes having this relation should be part of the same partition.

Due to the equivalence it represents, the owl:equivalentClass property is weighted with the highest possible value, which is 100 in the proposed system. The rdf:domain property creates a connection between a property and the class, which possesses this property. On the contrary, the rdf:range property defines the possible range of values for the property. For that reason, the former property is accepted to be a stronger relation than the latter.

As a concrete quantification of the weight of both relations two third of the maximum value for rdf:domain and one third for rdf:range have been selected arbitrarily. The rationale behind this is twofold. Firstly, the difference between the weights should be great enough. Secondly, the values should be defined without investing too much effort to have a first result for the partitioning with the weighting functions quickly. However, it should be mentioned, that numerical relations between the weights do not claim to be statements about the semantic similarity. An rdf:domain property is not semantically twice as strong as an rdf:range property.

Property	Description	Weight
owl:equivalentClass	This relation indicates the equivalence between two classes and is therefore assigned with highest possible value for the weight. (This corresponds to the idea that the similarity between a leaf class in a hierarchy and its superclass is almost a equivalence and is therefore assigned with the highest possible weight, too.)	100
rdfs:domain	The domain of a property is the class the property belongs to. That means that properties are mainly elements of their domain classes and depend on them. Therefore the weight for a domain edge is quite high and is defined as two third of the maximum possible weight.	66
rdfs:range	The range of datatype properties are basic data types like string. These are ignored because they do not relate different classes of one ontology. The range relation between an object type property and the range-class is considered as less important than the domain relation. Therefore the half of the weight of domain relation is assigned to range relations.	33

Table 5-3: Weights for non-hierarchical properties. The maximum possible value for a weight is 100 which is assigned to the owl:equivalentClass property. The other properties are assigned with two third and one third of the maximum value, respectively.

Inheritance relations

The second category is the inheritance relation which is defined as subClassOf in RDF-Schema. Even though this relation has always the same meaning its impact on the semantics of an ontology depends on the particular position in the subsumption hierarchy. An inheritance relation between the THING class and another class does in fact add no semantics to an ontology. In contrast, an inheritance relation between a leaf class and its parent class connects the leaf class to all its ancestor classes in the hierarchy. By inheriting all the properties of the ancestor classes the leaf class is also connected to all properties of the ancestor classes. Therefore, an inheritance relation at the top of the subsumption hierarchy is less important for the semantics of an ontology than an inheritance relation at the bottom.

The subsumption hierarchy in an ontology can be considered as a category system in the classic sense. In a very early work (Rosch, 1978) it has been shown that in a category hierarchy a so called basic level exists where the most basic category cuts can be made.

People tend to prefer categories at that level like car or chair instead of more abstract ones like furniture or vehicle or more concrete ones like sports car or kitchen chair. The categories at the basic level are considered as a kind of natural categorization where the increase of the similarity between the elements of a category reaches its maximum. More abstract categories are classified as superordinate categories with elements which share just a few properties. More concrete categories are called subordinate categories with very similar elements which have more common properties than distinctive ones and might be combined.

In Figure 5-6 the similarity between the elements of a category tree is illustrated in relation to the hierarchy level. The derivative of the similarity function shows that the highest gradient is reached at the basic level.

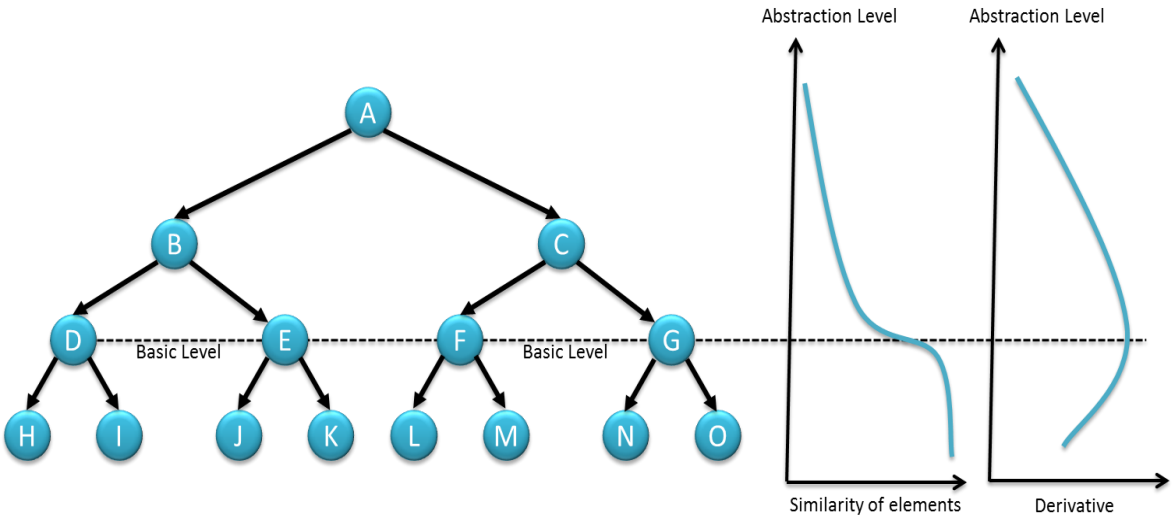


Figure 5-6: Illustration of the basic level in a category tree. The dashed line illustrates the location of the basic level within the tree and the graphs.

Since there are no databases of basic level concepts a weight function is proposed which considers the basic level as a function of the height of an element within the subsumption hierarchy. Additionally, since ontologies can be of different categories as described in Section 2.1.3 and illustrated in Figure 2-3, the basic level can be at different depth of the taxonomy hierarchy. For example in case of abstract top level ontologies the basic level might be closer to leaf classes, whereas in case of more concrete domain ontologies the basic level might be closer to the upper classes. Therefore, different functions are proposed to weight the subClassOf as illustrated in Figure 5-7. In case of the function Base3 the basic level is expected to be exactly at the middle of the

taxonomy hierarchy. In case of Base1 and Base2 the basic level is expected to be at higher levels and in case of Base4 and Base5 at lower levels of the taxonomy hierarchy. For comparison reasons the Linear and Static functions are defined additionally.

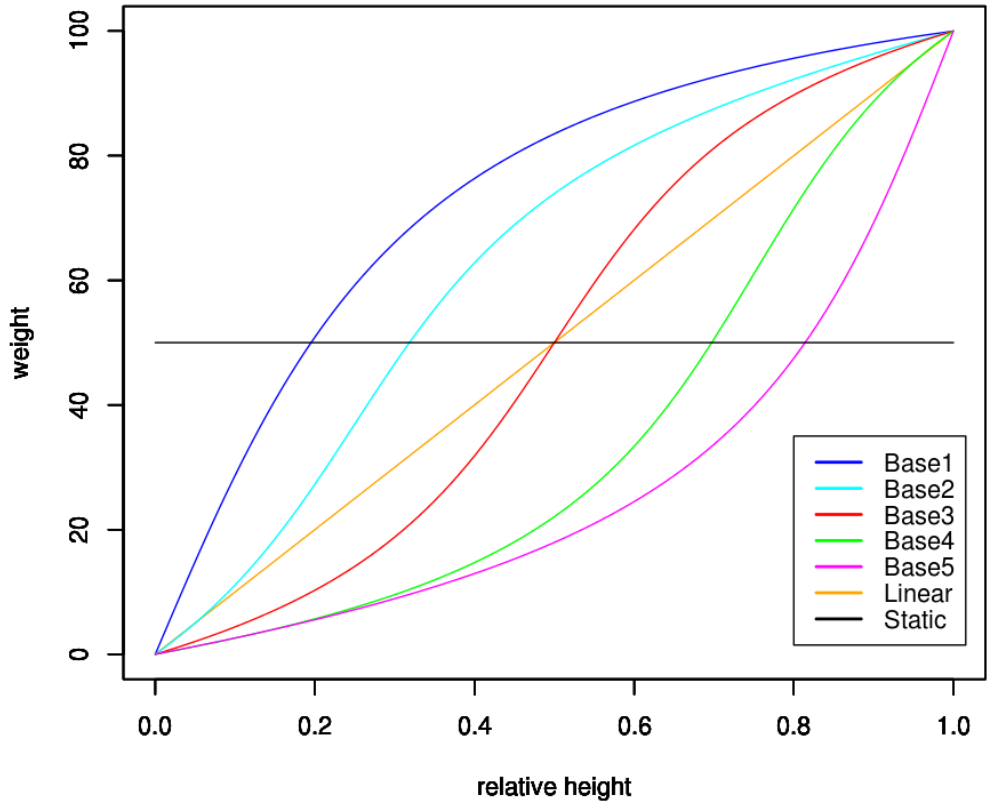


Figure 5-7: Various functions for weighting the subClassOf relation which assume different locations of the basic level within the subsumption hierarchy

Defined properties

The properties which are defined in an ontology are represented as edges in the class-centric representation of an ontology. This holds for the instantiations of the properties in the ABox even in the plain RDF graph representation. As these edges are somehow representing a domain as well as a range relation they are weighted with the mean value of both of them. This means that these edges are weighted with the half of the maximum weight, which is 50.

5.6 Parametric Partitioning

As discussed in Section 3 the optimal number and size of partitions depend on the goal of the partitioning. Community detection algorithms, however, do not take parameters for the envisioned number of communities or their size into account some. Since this framework provides partitioning in a goal oriented manner and there are obviously cases, in which the number and the size of partitions matter, this framework needs to allow these values as inputs for the overall partitioning process. These values are used as parameters for the partitioning process and the score value for selecting the best partitioning is extended. That means, that besides the modularity value for a partitioning the number and size of partitions are taken into consideration during the process of finding the community structure. This is done by an extension of the score function that is used to select the best partitioning. For that purpose the following function is defined for calculating the score of a concrete partitioning p

$$Score_p = \frac{w_{mod} * Mod_p + w_{PS} * PS_p + w_{NP} * NP_p}{w_{mod} + w_{PS} + w_{NP}} \quad (41)$$

Where PS_p is the average value for the score values PS_{pp} that are assigned to the partitions and NP_p is the score value for the number of partitions in a partitioning. These functions are defined as follows

$$PS_{pp} = e^{-\frac{1}{2} * \left(\frac{group\ size - expected\ group\ size}{allowed\ variance}\right)^2} \quad (42)$$

$$PS_p = \frac{\sum_i^{|p|} P}{|p|} \quad (43)$$

$$NP_p = e^{-\frac{1}{2} * \left(\frac{number\ of\ partitions - expected\ number\ of partitions}{allowed\ variance}\right)^2} \quad (44)$$

The Figure 5-8 shows the shape of the used basic function for both score functions. If the size of a group equals the optimal size of a group given by the user the PS_{pg} value of the group is the maximum value 1. If the difference between the actual partition size and the given partition size equals the allowed variance the assigned score is about 60% of the allowed maximum value. If the difference is greater the score value decreases rapidly.

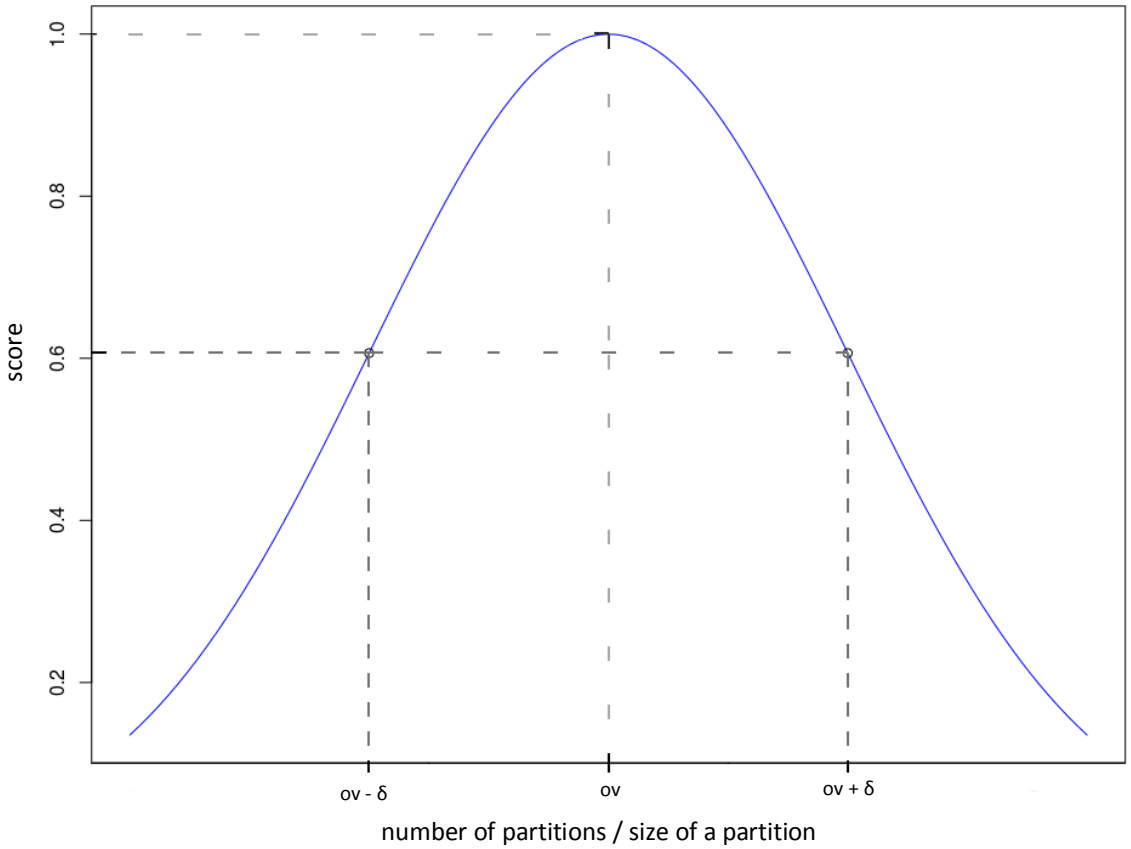


Figure 5-8: Illustration of the score function for the modified Modularity function. It is used to assign scores based on the number of group and based on the size of a group respectivly. (ov is the optimal value, δ is the allowed variance)

Since the function for NG_p is defined with the same basic function, the behavior of it corresponds to the behavior of the PS_{pg} function.

The overall score for a partitioning is the weighted sum of the three different score functions as shown in the Equation (43). Through the ability to set the values for the weights w_{mod}, w_{PS}, and w_{NP} as well as for the optimal number of partitions and optimal size for a partition the user is able to configure the partitioning process to obtain better results according to the needs of the user.

5.7 Chapter Summary

In this chapter the proposed adaptable partitioning framework has been described in detail. This artifact, in fact, is the core contribution of this thesis which follows the design science paradigm. For the sake of clarity and to avoid ambiguity it has been discussed in detail what the structure of an ontology is and how metrics based on that structure can be defined. This lead to a comprehensive list of possible metrics allowing insight into the structure of an ontology. Furthermore, community detection algorithms from the field of social network analysis and the modularity function have been presented. Being the means of partitioning an ontology, it is substantial to understand how they work and how they can be modified for the realization of an adaptable partitioning framework.

An analysis of the most used predicates in LOD ontologies' statements unveiled that the most used properties which add semantics to an ontology and are broadly used are the subClassOf and the domain-range relations. Based on this, a semantic-based weight function has been proposed, which is a novelty in the field of ontology partitioning and modularization. In fact, this weight function is of essential importance for the framework. Therefore, it is treated as one major contribution of this thesis.

Finally, an extension of the modularity function has been described that takes the number as well as the size of partitions into account. This is driven by the insight that the motivation for partitioning leads to different expectations regarding the number of partitions and the size of partitions as discovered in Section 3.

6 PARAMETER ANALYSIS

The proposed adaptable ontology partitioning framework comprises different parameters which can be configured. To achieve the best possible partitioning for an ontology regarding a concrete goal, it is important to find the best configuration of the framework. For that purpose, the performance of the framework with respect to the different parameters is analyzed in this chapter. On the one, hand the framework has been used with various configurations to reconstruct existing term chunks from ontology documentations. On the other hand, the framework has been used to reconstruct modularization of modular built ontologies. The method for the analysis is presented in Section 6.1. In fact, the created partitions are evaluated through a gold standard approach. The similarity between the reference models and the produced partitions are calculated with the F-Measure metric which is describe in Section 6.2

In order to analyze the impact of the different parameters, the ontologies have been partitioned with 480 different configurations for each ontology. The selected combinations are presented in Section 6.3. Section 6.4 and Section 6.5 present the results for reconstructing term chunks and for reconstructing modular ontologies, respectively. It is analyzed in detail, how the different algorithms and parameters influence the results for the different ontologies. Finally, this chapter is closed with a chapter summary in Section 6.6.

6.1 Analysis Methodology

The basic assumption that has driven the proposed framework is the notion that the partitioning process depends significantly on the goal. Hence, the quality of a partitioning has to be evaluated with respect to that goal. In other words, a concrete partitioning for a particular ontology might be very good for improving the maintainability, whereas it might be very bad with respect to the task of supporting the decision process about the reusability of an existing ontology. Thus, the proposed framework has to be analyzed and evaluated for different partitioning goals separately. However, this leads to the question of which different motivations are possible for a partitioning process and how a partitioning for a particular goal can be evaluated. Without claiming to be exhaustive, the main goals of partitioning considered in this work are partitioning to improve maintainability and partitioning to create term chunks for ontology documentations, as discussed in Section 3.3 and Section 3.4.

Even though there are different approaches in literature for the evaluation of ontology partitioning (some metrics have been described in Section 4.1), there are still no established methods. This especially holds for the evaluation of a partitioning in a goal-oriented manner. For that reason, the evaluation method that is chosen for this work is, to find manually created existing ontology partitionings. Thereby, it is of essential importance that the goal is obvious and to try to reconstruct them. The rationale for this is the assumption that manmade partitioning for particular goals can be considered as the optimal solution and as reliable reference models for the particular goal. The creators' expertise and experience with the proposed ontology is regarded as the justification for the quality of the partitioning.

This assumption leads to the following evaluation methodology for ontology partitioning. First of all, a set of existing ontology partitionings which have been created with the mentioned motivations have been looked for. (Found ontologies are described in Section 6.4 and Section 6.5). Secondly, these ontologies have been partitioned with various configurations, which are considered to be significantly different to understand the framework's behavior with respect to different configurations. (These configurations are described in Section 6.2.) Thirdly, the results of the partitioning process have been compared to the reference models and a similarity score has been calculated. (The function to calculate the similarity is described in Section 6.3.) Finally, the overall results have been analyzed with respect to different parameters of the configuration, the properties of the ontologies, and the properties of the different reference models. (This is done for reconstructing term chunks and for reconstructing modular ontologies in Section 6.4.2 and in Section 6.5.2 respectively.)

It is important to understand that this method's critical drawback is its dependency on the reference models. For being expressive, it is important to have enough reference

models which are representative and which allow to derive more general statements about the partitioning framework. The lack of enough partitioned ontologies for term chunks or for maintenance purpose, however, is a strong indication for the lack of good support for partitioning. This in fact advocates the necessity of such a framework and justifies this work.

6.2 Similarity with F-Measure

In order to calculate the similarity between the reference model and the constructed partitioning, a metric called F-Measure that has already been used in this context by Stuckenschmidt has been applied (Stuckenschmidt, 2006). This measure is a pair-based metric based on the widely applied precision and recall functions. Precision in this work is defined as follows:

$$precision = \frac{number\ of\ generated\ pairs, that\ are\ correct}{number\ of\ all\ generated\ pairs} \quad (45)$$

It indicates the fraction of the generated pairs that are correct with respect to the reference model. And recall is defined as follows:

$$recall = \frac{number\ of\ generated\ pairs, that\ are\ correct}{number\ of\ all\ correct\ pairs} \quad (46)$$

Recall indicates the fraction of all correct pairs that have been correctly reconstructed by the partitioning process. These functions are combined in the F-Measure function as follows:

$$F - Measure = \frac{2 * recall * precision}{recal + precision} \quad (47)$$

In fact, being the harmonic means of the precision and the recall function the F-Measure is an average value for both mentioned fractions.

6.3 Configuration of the Framework

As described in Chapter 5, the proposed framework has different configurable parameters: two graph representations, three algorithms, and eight weight functions. (Besides the seven weight functions presented in Section 5.5 it is also possible to run the partitioning without any edge weight, which means equal weight for all edges.) Furthermore, through defining values for the variables described in Section 5.6, the size as well as the number of partitions can be considered for calculating the modularity score. For the analysis of the framework's performance, five different values for the variables described in Equation (42) have been chosen as shown in Table 5. The rationale for the defined values is to see, how increasing weight of the modularity function (w_{mod}) influences the overall result. The higher this value is the less important are the number of partitions and their average size. Therefore, this exposes the importance of the modularity function in comparison with the size and number of partitions.

Variable	s1	s2	s3	s4	sNone
w_{mod}	2	5	10	50	1
w_{PS}	1	1	1	1	0
w_{NP}	1	1	1	1	0

Table 6-1: Configurations for the size related parameters of the framework

Besides the concrete configurations of the partitioning process, there are also different properties regarding the loading mechanism of an ontology. The import of external ontologies adds some kind of uncertainty due to possible availability problems and different version problems of the external ontologies. Therefore, during the performance analysis all imports of external ontologies have been ignored. Only the content of the ontology file at hand has been used. Regarding the applied inference mechanism during the loading process, the decision is to run each possible configuration twice per ontology. In the first case, no inference is executed, which means that only explicit formulated content of the ontology has been taken into consideration. In the second case, an OWL Full reasoner has been applied on the loaded content and the ontologies have been materialized. Together with the mentioned parameters this leads to 480 possible configurations of the partitioning process.

As elaborated in Section 3.3.3 and in Section 3.4.2, the expected number of partitions and their sizes depend on the motivation of the partitioning process. Therefore, the four

variables *"expected group size"* and *"allowed variance"* in Equation (43) and "*expected number of partitions"* and *"allowed variance"* in Equation (45) depend on the goal of the partitioning. Since this work focuses on term chunks for ontology documentations on the one hand and modular ontologies on the other hand, the mentioned variables have to be set differently.

Firstly, the framework's performance with respect to the creation of term chunks is analyzed. In this regard, the guidance values elaborated in Section 3.3.3 are used. The variable "$expected\ group\ size$" in equation (43) is set to eleven and the "$allowed\ variance$" is set to three. That leads to the interval of eight to 14. Furthermore, the "$expected\ number\ of\ partitions$" in Equation (45) is set to four and the "$allowed\ variance$" is set to one. It is worth mentioning that in case of different ontologies with a higher variance in size, a normalization of these values based on the overall size of the ontologies should be done by keeping the mentioned cognitive limits in view. However, in case of the ontologies at hand this is not necessary, because the used ontologies are about the same size in terms of classes and properties.

Secondly, it is analyzed how the framework performs for creating modular ontologies. Again, previously elaborated findings from Section 3.4.2 are used for the configuration. In contrast to the previous analysis, the small number of use cases as well as their diversity regarding size and number of partitions do not allow to extract one common value to be used with all of them. In fact, in case of partitioning for constructing modular ontologies, the optimal number as well as the size of partitions seem to be domain and ontology dependent. Therefore, these values are expected as input values for a partitioning defined by the user. Since the maintainer is expected to be familiar with the ontology, such an approach is considered to be acceptable and not an unrealistic expectation. Finding accurate values for the *"expected number of partitions"* and the *"allowed variance"* in Equation (45) like in the first analysis, is not possible. Therefore, the actual number of modules is selected for the first variable. The second variable is defined arbitrarily as one fifth of the first parameter. This leads to the following intervals for the number of partitions to be created by the algorithms and the following values for the variables

- curio: 3 - 5 (*"expected number of partitions"* = 4, *"allowed variance"* = 1)
- sioc: 3 - 5 (*"expected number of partitions"* = 4, *"allowed variance"* = 1)
- spice: 7 - 11 (*"expected number of partitions"* = 9, *"allowed variance"* = 2)
- fibo: 19 - 27 (*"expected number of partitions"* = 23, *"allowed variance"* = 4)

6.4 Reconstructing Term Chunks

In Section 3.3.2 thirteen ontologies have been described which have documentation pages containing term chunks. In order to understand, how well the proposed framework is able to create term chunks, it was analyzed to which extend the system is able to recreate those existing groups. For the performance analysis, each one of the thirteen ontologies has been partitioned with 480 different combinations of the parameters. The resulting partitions have been compared with the existing term chunks with the F-Measure similarity function presented in Section 6.2. The overall performance of the proposed system aggregated for all ontologies with all possible configurations is shown in Figure 6-1.

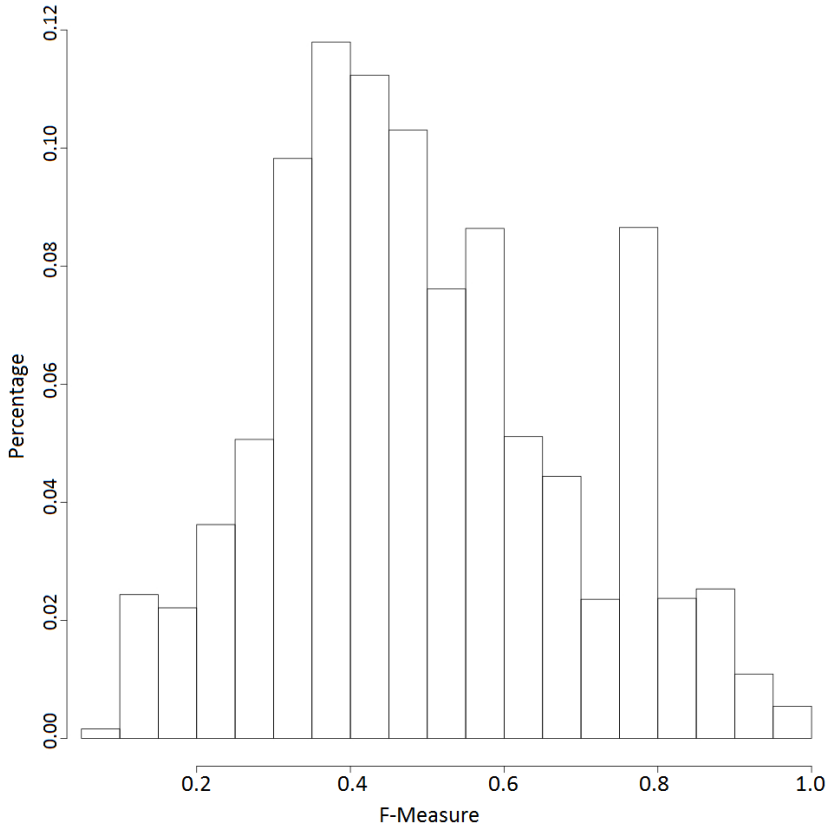


Figure 6-1: Distribution of the overall results for reconstructing term chunks. The results for all ontologies with all possible configurations are aggregated.

The shape of the distribution looks like a Gaussian normal distribution. The peek at 0.8 might be caused because of the low number of ontologies and reference models.

There is no reasonable argument that causes this peek. However, this distribution does not allow further insights about the framework, besides that it produces very well as well as very bad results and that it produces in most cases results between 0.3 and 0.7 with an average of 0.496.

Since the basic idea of organizing the concepts in groups differs from use case to use case as discussed in Section 6.4, the next question is how the different configurations perform depending on the different ontologies. Therefore, the performance of the framework has been analyzed for each ontology separately. The result of this analysis is shown in Figure 6-2.

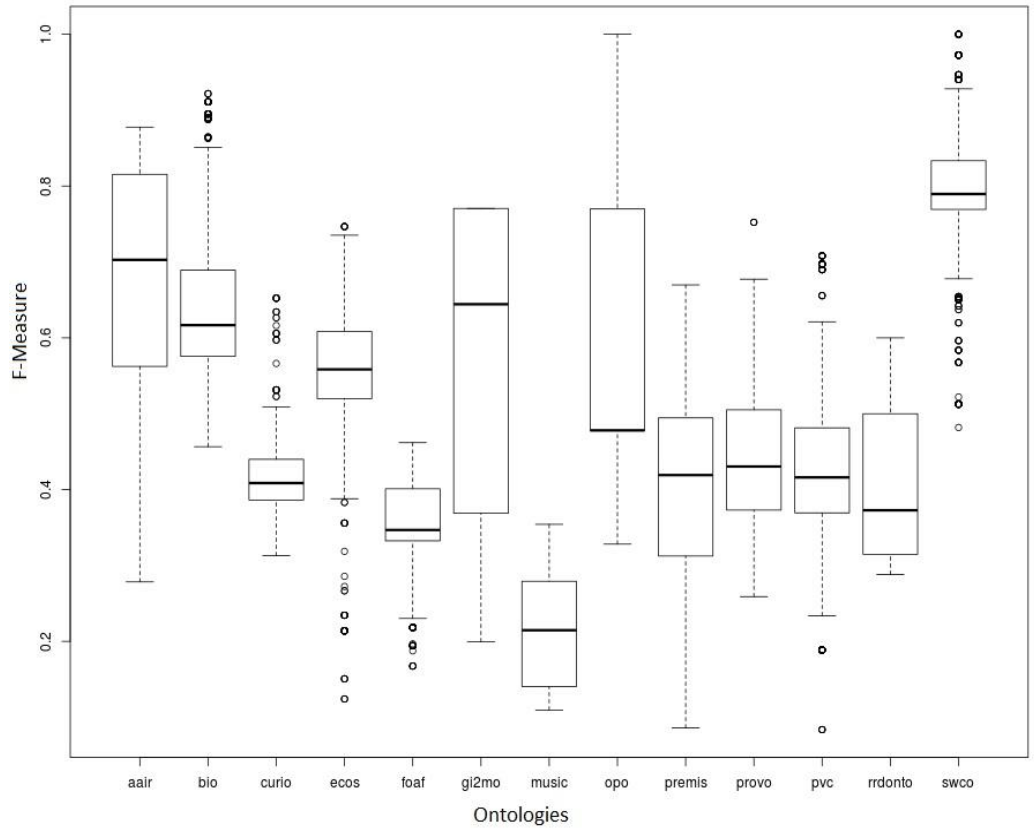


Figure 6-2: Distribution of the results for the different ontologies. Each box of this box-and-whisker plot represents the range of the F-Measure values of the partitionings of one concrete ontology with 480 different configurations. The concrete ontology is named at the x-axis. (E.g. the first box in this figure shows that the F-Measure values for the different partitions created out of aair with 480 configurations range from about 0.3 to 0.9 with an average of 0.7.)

It becomes clear that the partitioning performance depends significantly on the ontology and the reference model. While there are ontologies, which have very good results for each possible configuration as in case of swco, there are other ontologies, which lead to very bad results with all possible configurations as in case of music. Furthermore, the difference in the width of the spectrum for the F-Measure values indicates that communities are detected, which differ significantly. A wide spectrum indicates that there are more possible partitionings, while a narrow spectrum indicates less possible partitionings of an ontology. Accordingly, curio, ecos, foaf, and swco seem to allow a low number of different partitionings.

The main question in this chapter is how the different parameters influence the results. Therefore, the performance was analyzed with one fixed parameter whereas all other parameters have been changed. Figure 6-3 shows the overall results for the different configurations. Each box of this box-and-whisker plot stands for the range of the F-Measure values with a fixed parameter, which is shown at the x-axis, whereas the other parameters have been changed.

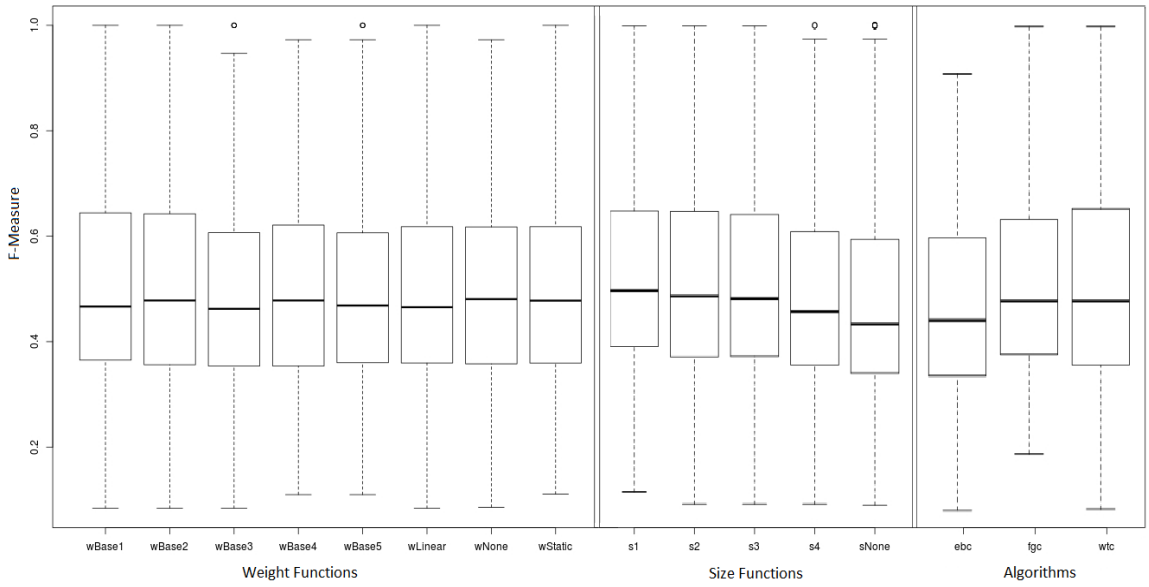


Figure 6-3: Distribution of the results for different fixed parameters. Each box of this box-and-whisker plot represents the range of the F-Measure values of the partitionings created with one fixed parameter for all ontologies. (E.g. the first box represents the range of the F-Measure values for the partitionings created with the weight function wBase1, whereas all other configurations have been changed and applied on all ontologies.)

The distributions of the results for the different parameters are rather similar. This means, that there is no concrete parameter that performs better than the other configurations. Obviously, each parameter is equally important for the quality of the partitioning process. They all lead to very good as well as very bad results with average values between 0.4 and 0.5. Since this analysis did not expose any well-performing parameter it was also analyzed how these parameters influence the best results. For that purpose, the best ten results for each ontology have been selected. Figure 6-4 shows the share of the different parameters on the aggregation of the best ten results for each ontology.

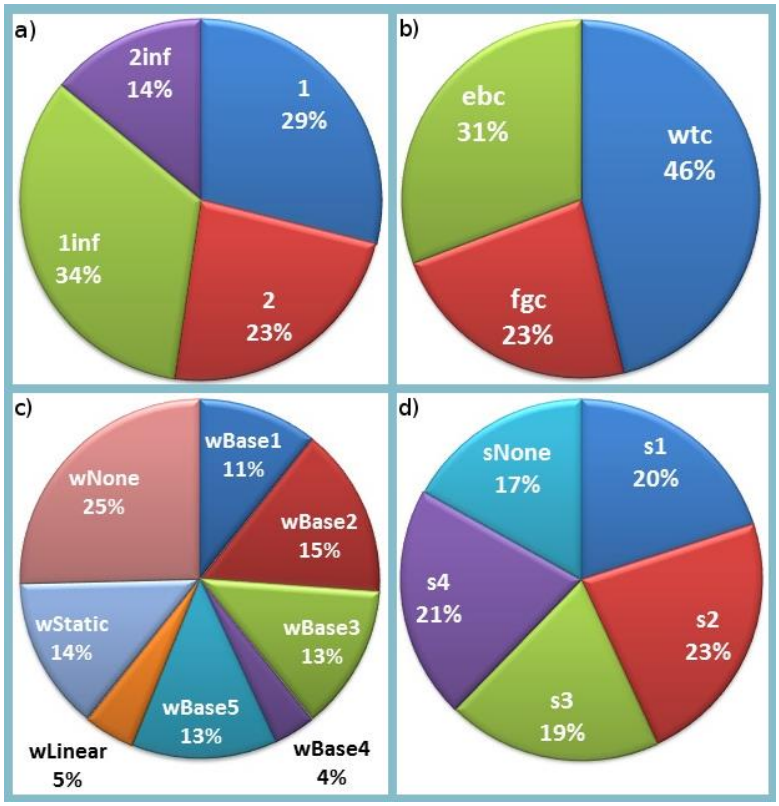


Figure 6-4: Share of the different concrete parameter values on the best ten results for all ontology. This figure provides an overview of the different configurations that lead to the best ten results for each ontology . a) shows the fraction of the different graph variants, b) shows the fraction of the different algorithms, c) shows the fraction of different weight functions, and d) shows the fraction of the different size functions.

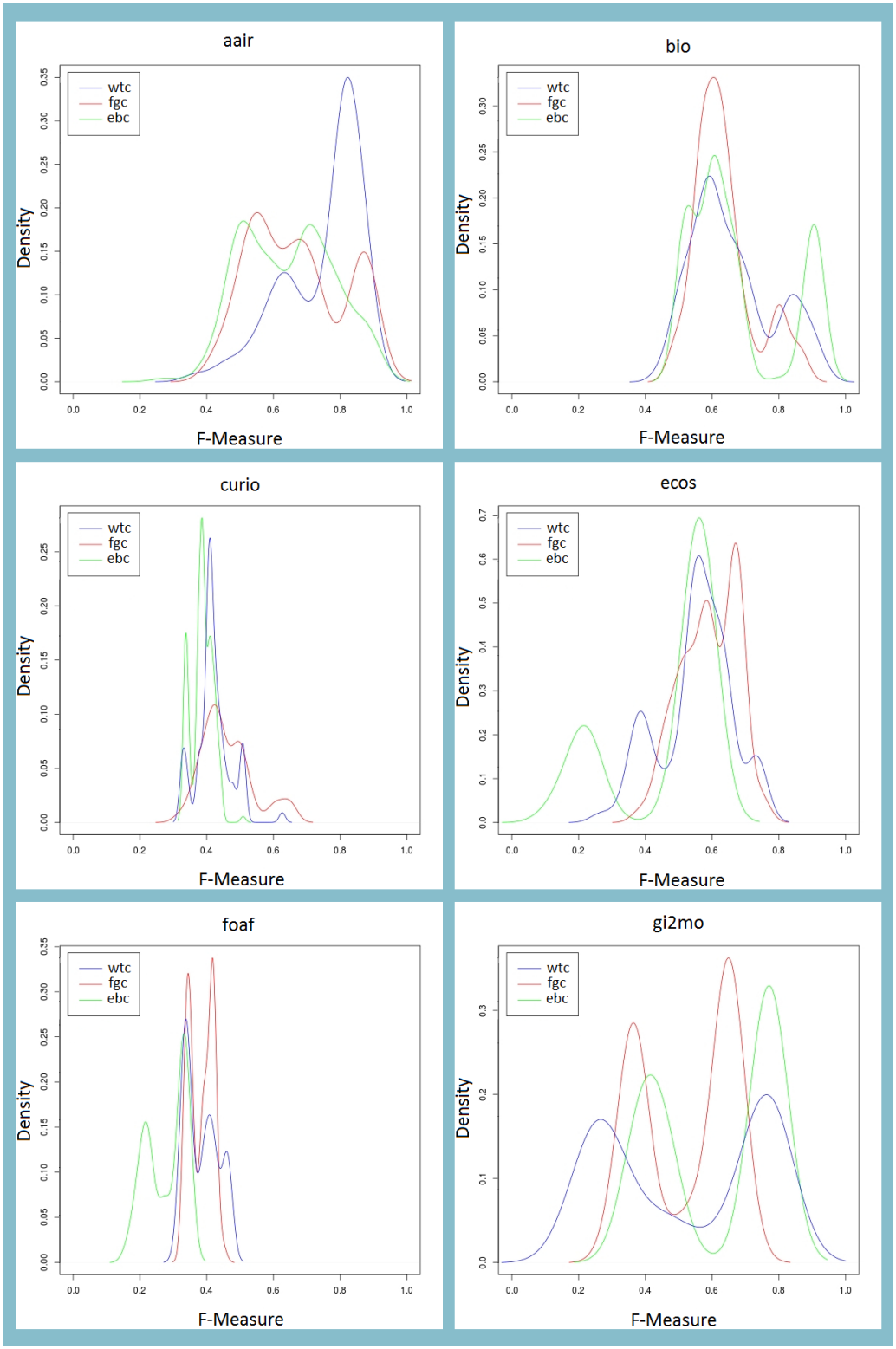
Regarding the different graph representation variants as shown in Figure 6-4 a), there is a considerable difference visible. On the one hand, the RDF-based graph representations (1 and 1inf) lead to about two third of the best results and therefore perform better than the class-centric graphs (2 and 2inf). On the other hand, inference improves the results for the RDF-based graph (1inf) whereas it leads to worse results for the class-centric graph (2inf).

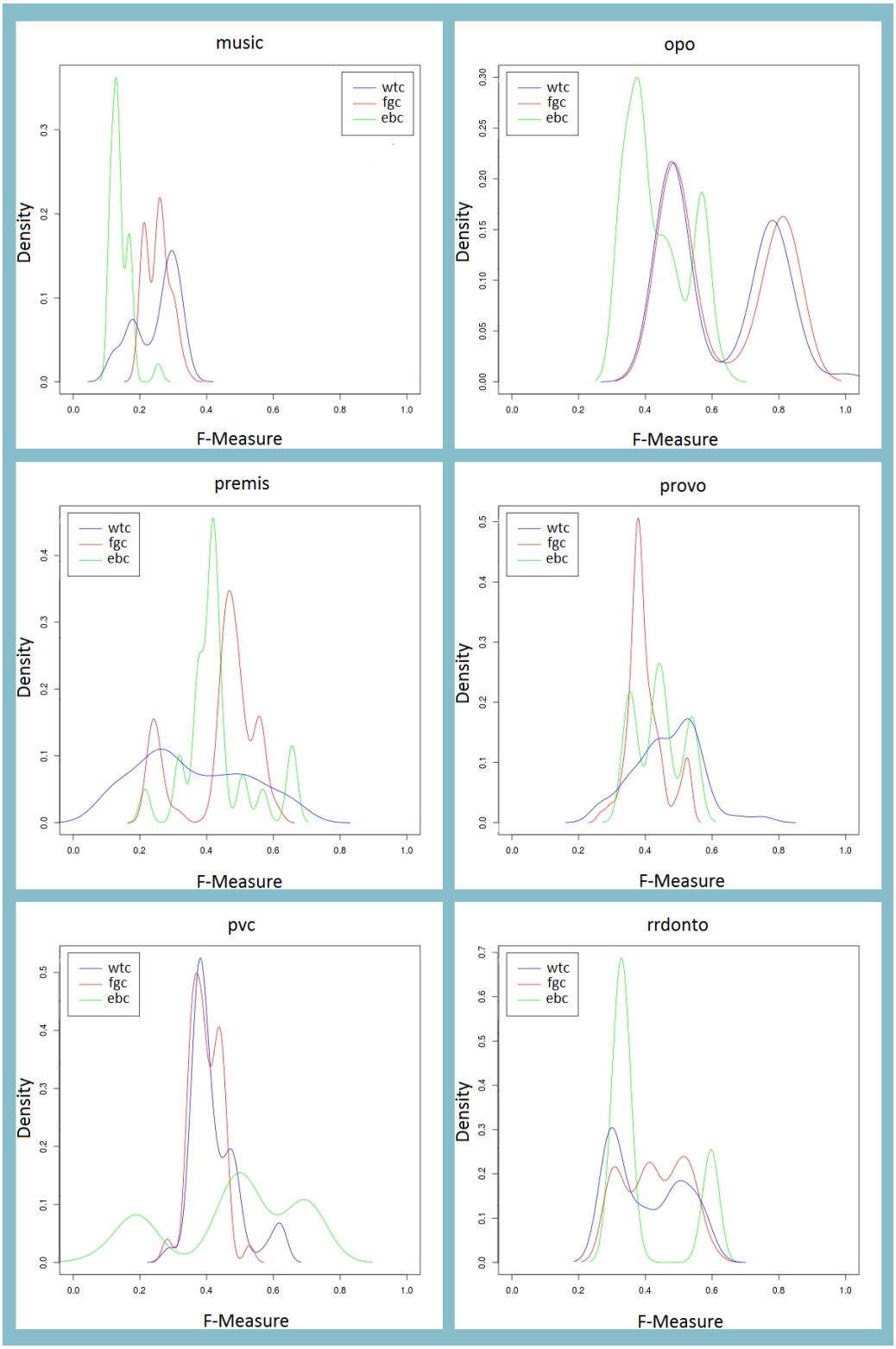
The share of the different size functions on the best ten results as shown in Figure 6-4 d) are quite similar. The size function s2 has the highest share, but each function leads to approximately one fifth of the best results. That means that in case of a random selection of a size function the probability to find the best value for this parameter is about 0.2.

The difference between the weight functions, on the contrary, is bigger. As shown in Figure 6-4 c), assigning no weights to the edges (wNone) leads to one quarter of the best results. The weight functions wBase1, wBase2, wBase3, wBase5 and wStatic have similar shares on the best ten results (11%-15%), whereas the weight functions wLinear and wBase4 lead in only one of 20 cases to results which are within the top ten.

The share of the different algorithms on the best ten results is even more significant (Figure 6-4 b). Approximately one half of the best results have been produced with the wtc algorithm. The ebc algorithm leads to about one third, whereas the fgc algorithm leads to about one quarter of the best results.

In a probability-based selection of the properties, the best choice would be to use the graph variant 1inf, the weight function wNone, the wtc algorithm, and the size function s2, because these values provide the highest probability to lead to the best possible result. However, the significant difference between the algorithms - especially the high share of the wtc algorithm - justifies a closer look on their performance for each ontology separately. Therefore, Figure 6-5 provides one diagram for each ontology and allows comparing the behavior of each algorithm.


aair
bio
curio
ecos
foaf
gi2mo
wtc
fgc
ebc
Density
F-Measure


music
opo
premis
provo
pvc
rrdonto
wtc
fgc
ebc
Density
F-Measure

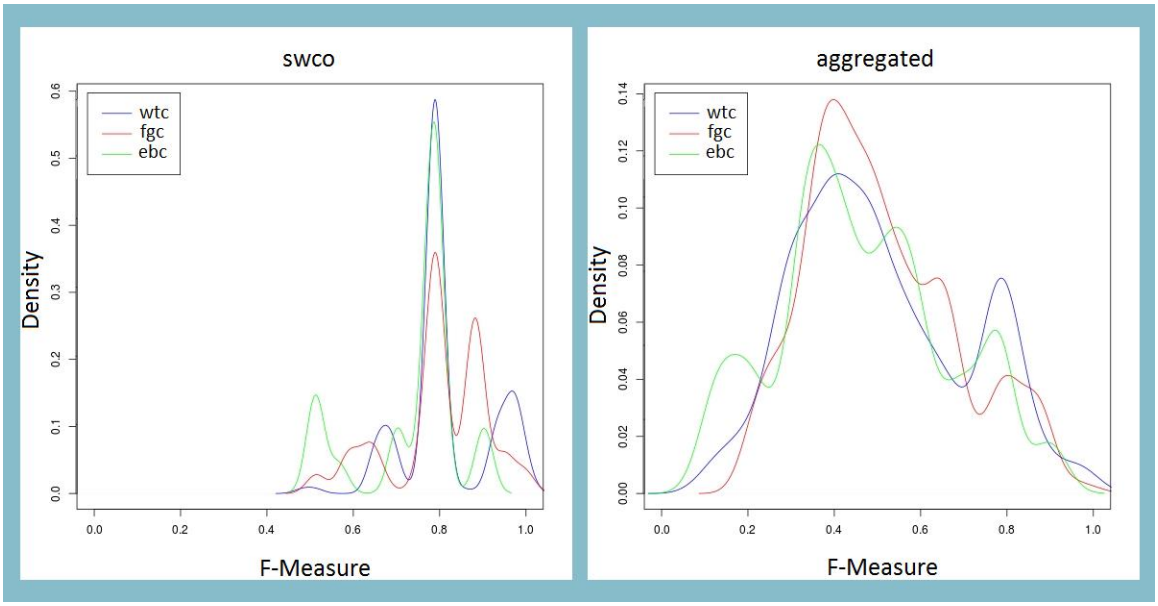


Figure 6-5: Distribution of the results for all three algorithms for each ontology and for all ontologies aggregated.

As already seen in Figure 6-3, the spectrums of the different algorithms are about the same size. That means that each algorithm creates very good as well as very bad results. The peaks, however, are obviously at different points within the graph and have different heights. Therefore, this analysis emphasizes again that the performances of the different algorithms depend significantly on the ontology. Obviously, the structural properties of ontologies play a significant role in the performance of the different algorithms. An indication of this is the following. Apart from the probability based selection of the parameters as described previously, a more sophisticated method would be to select the parameters based on the structural properties of the ontology. In this regard, known ontologies would be seen as a training set and every ontology, that is to be partitioned, would be compared to the training set. The best configuration of the most similar ontology from the training set would be selected. In order to determine the similarity between the ontology at hand and the ontologies from the training set, the structural metrics, which were presented in Section 5.3, could be used.

6.4.1 The Role of the Modularity Function for Term Chunks

As described in Section 5.4.1, community detection algorithms are approximation functions seeking to optimize the modularity function. The application of these algorithms in this work is based on the assumption that the relation of subdomains of an ontology to the whole ontology is similar to the relation of communities to the whole social network. That means that the density within a partition is higher than the overall density. If this is the case, a partitioning with a high modularity value should lead to high F-Measures values. In order to analyze if this assumption can be proofed, the relation of the F-Measure score is compared to the modularity score. Figure 6-6 shows plots of the F-Measure results with respect to the modularity score for each ontology and one for the aggregation of all results.

The following observations are possible for the relation between the F-Measure results and the Modularity score.

- The diversity between the different ontologies is again too high to make general statements. While there seems to be a linear relation between the F-Measure and the Modularity for opo and swco, which indicates increasing F-Measure values for higher modularity values, for bio the relation seems to be the opposite. In case of bio, the highest values for F-Measure are at the lowest values for the Modularity value.

- In most cases, the highest F-Measure values are at the higher Modularity values (aair, curio, foaf, gi2mo, music, opo, provo, rrdonto, swco). That means that the constructed partitionings, which have the highest similarity compared to the reference models, have mostly high Modularity values. However, since there are other partitionings with high Modularity scores which have very low F-Measures it is not possible to state that the Modularity function is a good indication for good partitioning. However, it is possible to say that partitionings with bad Modularity scores are probably not good.

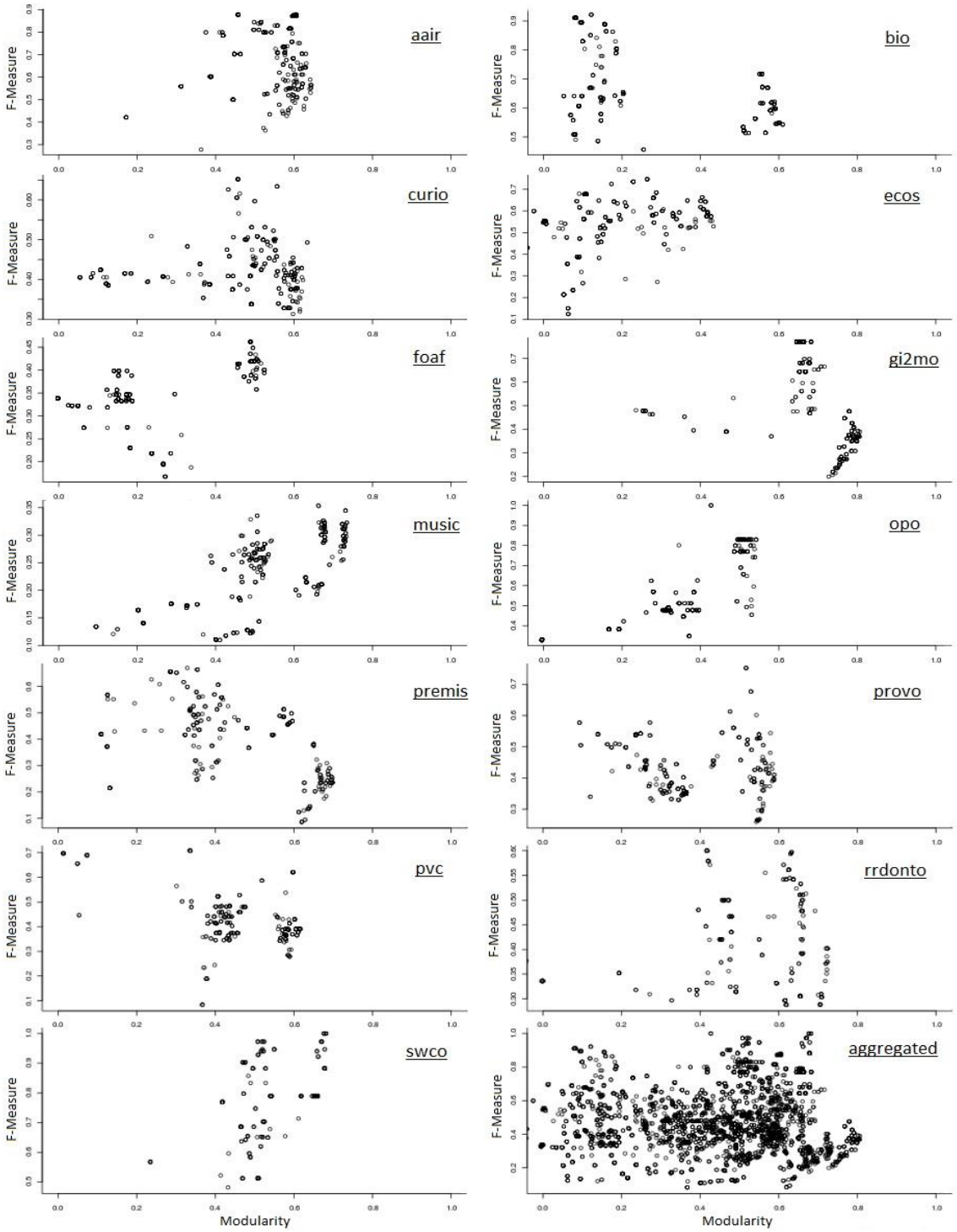


Figure 6-6: F-Measure values above the modularity values for each ontology and for the aggregation of all results.

6.5 Reconstructing Modular Ontologies

The performance in reconstructing merged modular ontologies is accepted to be an indication for the overall performance of the proposed framework with respect to ontology partitioning for creating modular ontologies. Therefore, four modular built ontologies, which were presented in Section 3.4.1, were used to analyze the parameters of the proposed framework. The analysis has been done similar to the previous analysis described in Section 6.4. The overall performance for each ontology can be seen in Figure 6-7.

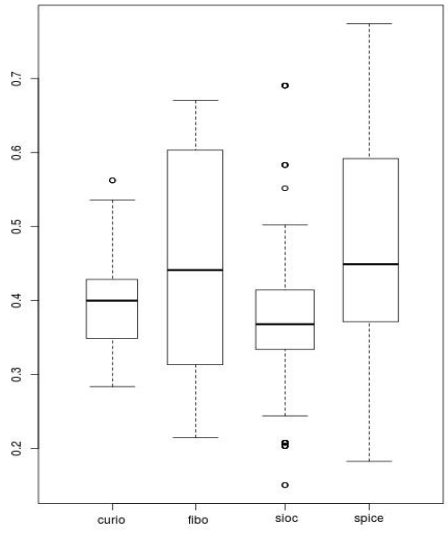


Figure 6-7: Distribution of the F-Measure values for the different ontologies. Each box of this box-and-whisker plot represents the range of the F-Measure values of the partitionings of one concrete ontology with 480 different configurations. The concrete ontology is named at the x-axis. (E.g. the first box in this figure shows that the F-Measure values for the different partitions created out of curio with 480 configurations.)

The overall average value is 0.424 and the average values for the four ontologies are quite similar between 0.35 and 0.45. In contrast, the spectrum differs stronger. For curioMerged and siocMerged the most results are within a narrow interval, namely between 0.35 and 0.43 and between 0.33 and 0.4 respectively. This indicates that the structure of both ontologies allow just a small number of different partitionings. Whereas, the most results for fiboMerged and spiceMerged are between a broader interval, 0.3 to 0.6 for fiboMerged and 0.37 to 0.6 for spice.

Since this analysis' main goal is to understand how the different parameters influence the performance of the proposed framework, it was analyzed to which results concrete values of the parameters lead. Figure 6-8 provides the results of this analysis.

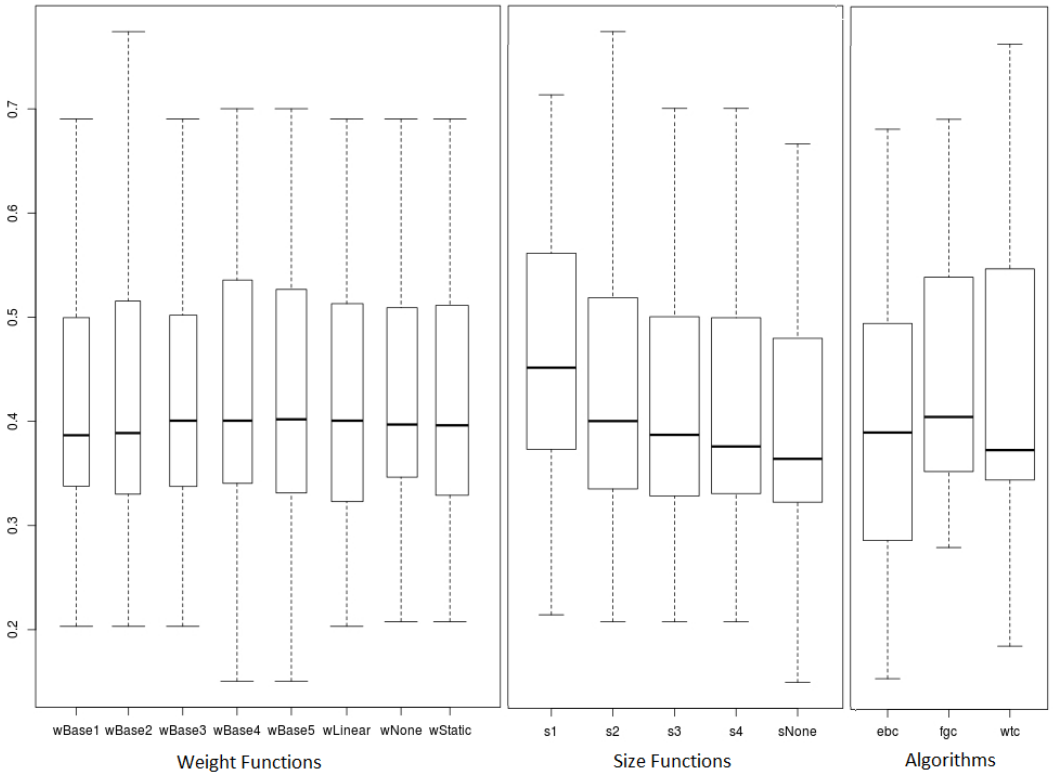


Figure 6-8: Distribution of the results for different fixed parameters. Each box of this box-and-whisker plot represents the range of the F-Measure values of the partitionings created with one fixed parameter for all ontologies. (E.g. the first box represents the range of the F-Measure values for the partitionings created with the weight function wBase1 whereas all other configurations have been changed and applied on all ontologies.)

The different parameters do not lead to similar results as in the previous analysis, but the difference is not significant. The only obvious observation is that the size function s1 leads to better results on average than the other size functions. In order to gain more insights about the influence of the parameters, the top ten results for each ontology have been analyzed regarding the fraction of the parameters. The results of this analysis are shown in Figure 6-9.

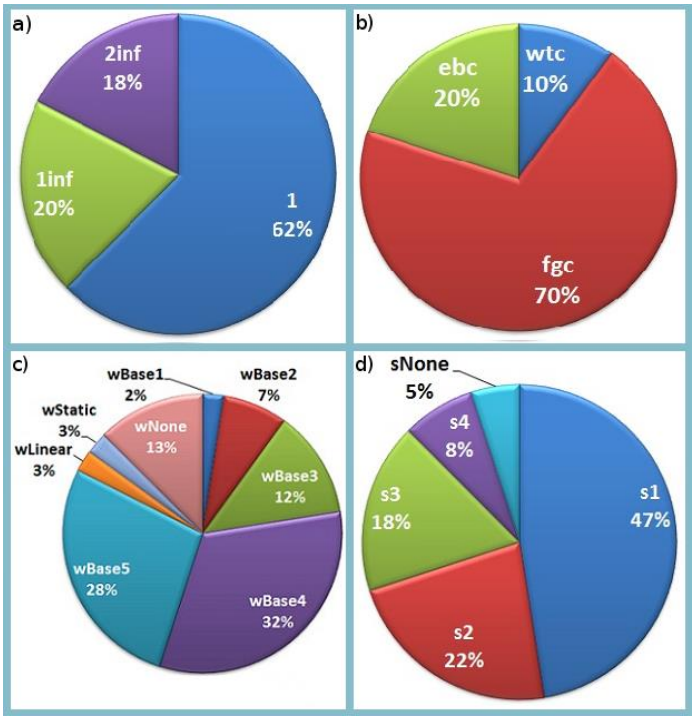


Figure 6-9 Share of the different properties on the top ten results of the partitioning processes

Using the plain RDF graph to represent the structure of the ontologies leads to 62% of the top ten results, as shown in Figure 6-9 a). The graph representations 2inf and 1inf lead to about one fifth of the top ten results, whereas the graph variant 2 has no share on the top ten results, at all. In Figure 6-9 b) the share of the different algorithms on the top ten results is shown. The fgc algorithm leads to 70% of all top ten results, while the ebc algorithm leads to one fifth and the wtc algorithm leads to one tenth of the top ten results. Regarding the weight function, as shown in Figure 6-9 c), the functions wBase4 and wBase5 have the highest share on the top ten results with 32% and 28%, respectively. Finally, the size function s1 leads to about the half of the top ten results, while the s2 and s3 function have a share about one fifth.

In case of a probability-based selection of the framework's properties the plain RDF representation of the ontology structure, the fgc algorithm, the wBase4 weighting function and the s1 size function should be selected. This configuration provides the highest probability to lead to good results. However, as in the previous analysis the highest difference is again between the different algorithms. Thus, Figure 6-10 provides one diagram for each ontology and allows comparing the behavior of each algorithm.

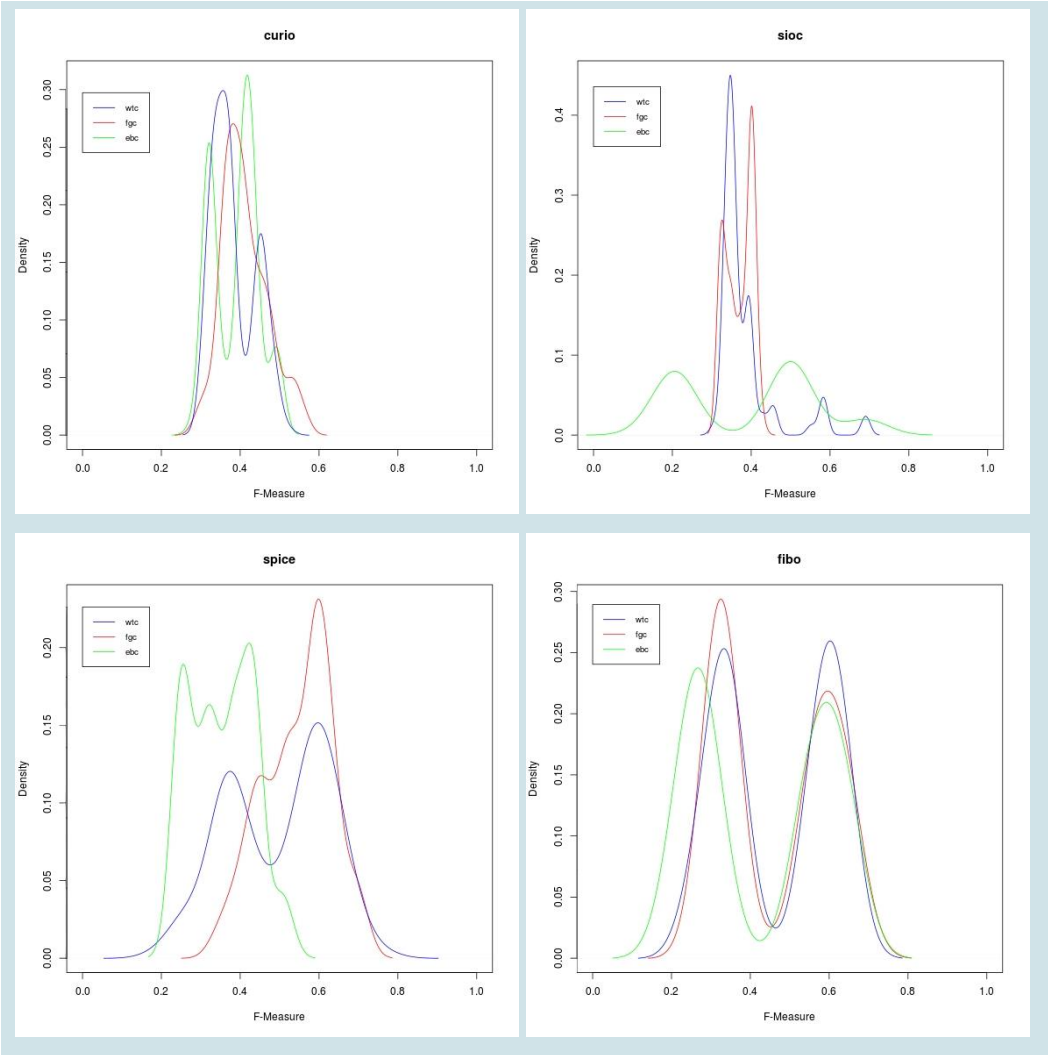


Figure 6-10: Performance of the algorithms for each ontology

It becomes clear that the performance in reconstructing modular ontologies depend on the ontologies as in the previous analysis. In case of curioMerged the spectrum of the different algorithm is similar while in case of fiboMerged the peaks are similar. In case of siocMerged and spiceMerged, on the contrary, neither the spectrum nor the peaks are simiar. For the selection of the framework's parameter the structure of the ontology should have the highest influence.

6.5.1 The Role of the Modularity Function for Reconstructing Modular Design

For understanding the role of the modularity function in context of reconstructing modular design the relation between its values and the values of F-Measure are compared as shown in Figure 6-11.

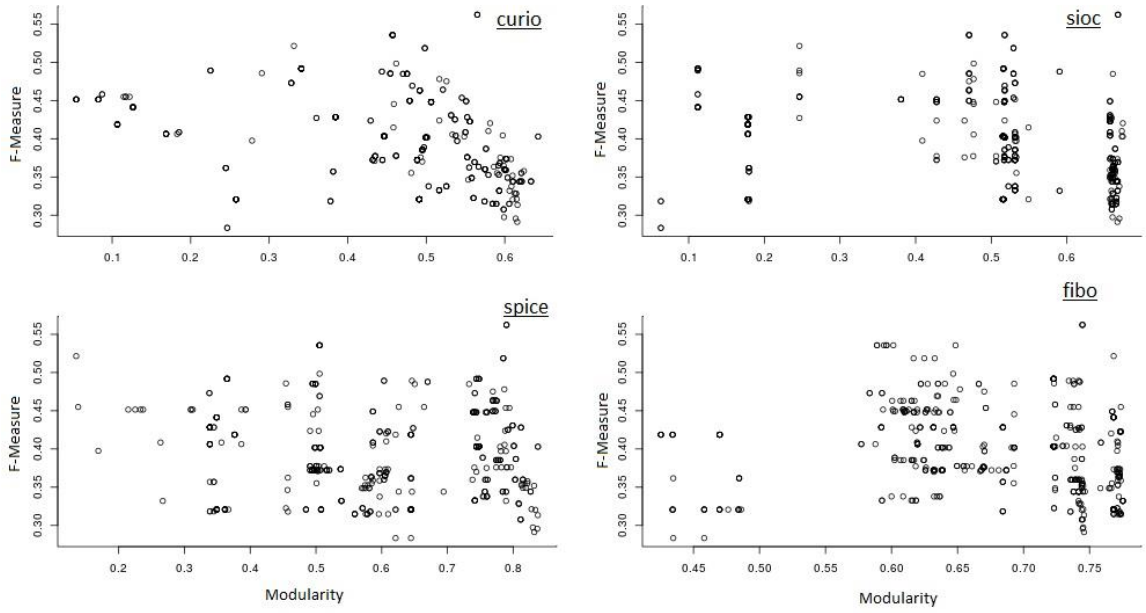


Figure 6-11: F-Measure values above the modularity values

The different values are quite equally distributed over the whole area of the diagram. Hence, it is obvious that there is no relation between the F-Measure values and the Modularity values. On the one hand, high Modularity values lead to high as well as low F-Measure values. On the other hand, low Modularity values lead again to high as well as low F-Measure values. This means that the assumption that community detection algorithms, which seek to optimize the Modularity function, are a good means to create modular ontologies, cannot be proofed.

6.6 Chapter Summary

In this chapter the parameters of the proposed framework have been analyzed. For that purpose each one of the ontologies that have been presented in Section 3.3 and Section 3.4 has been partitioned with 480 different configurations. The created partitions have been compared with the term chunks in the documentations and with the partitions of modular ontologies, respectively. The main insights gained by this analysis are twofold.

Firstly, from a probabilistic point of view, there are values for the parameters which provide a higher probability to lead to good results. If the goal of partitioning is to create term chunks, the best configuration is to use the graph variant 1inf, the weight function wNone, the wtc algorithm, and the size function s2. If the goal of partitioning is to create modular ontologies, the best configuration is to use the plain RDF representation of the ontology structure, the fgc algorithm, the wBase4 weighting function and the s1 size function.

Secondly, the analysis uncovers that the most important factor of the framework's performance is the ontology that is to be partitioned. Therefore, the structural properties of the ontology should be taken into consideration during the selection of the values for the different parameters. The metrics which have been described in Section 5.3 provide a good basis to compare the structure of two ontologies and to calculate a similarity value.

Another important insight of this chapter is that high modularity values for partitions do not indicate good score values in terms of F-Measure values. However, with respect to term chunks it is possible to say that partitionings with bad Modularity scores are probably not good in terms of similarity to the reference models.

7 EXPERIMENTAL PERFORMANCE ANALYSIS

Aiming at the creation of a support system that partitions ontologies for different purposes, this work proposes an adaptable ontology partitioning framework. This framework has been described in detail in Chapter 5. The different parameters that enable this frameworks adaptability has been analyzed exhaustively in Chapter 6. In this chapter it is now analyzed, how well this framework actually performs. An experimental evaluation is applied, which is based on the findings of the parameter analysis. The evaluation method that is presented in Section 7.1. This method is then used to measure the performance for creating term chunks which is presented in Section 7.2. In Section 7.3 the proposed framework's performance regarding the creation of modular ontologies is then presented, again by the application of the mentioned methodology. Finally, this Chapter is concluded in Section 7.4 with a chapter summary.

7.1 Setup for the Experimental Evaluation

The proposed adaptable ontology partitioning framework is not the first artifact that is able to partition ontologies. Existing solutions have been discussed in Section 4.3. In this regard, this work can be considered as an extension of the outcomes in this field. In order to measure the progress achieved with this endeavor, the most appropriate approach is a direct comparison of the performances. For that purpose, the ontologies presented in Section 3.3.2 and in Section 3.4.1 have been partitioned with the aforementioned tools SWOOP, Pato and with the proposed framework as well.

SWOOP does not provide any configuration of the partitioning process. Pato, on the contrary, allows configuring the partitioning process with several parameters. For the creation of the network the inclusion of subclass relations, property links, and definition links have been activated. The weight for each has been set to one. The value for "Threshold" has been kept at the default value 0.5. The options "Assign unclustered vertices" and "Merge clusters" have been left activated as it is the default setting. According to the finding of the Section 3.3.3, the maximum island size is set to 14 in case of creating term chunks. For creating modular ontologies the island size is set to the concrete number of modules as shown in Table 3-3.

As seen in the previous chapter, there are 480 different possible configurations for the proposed framework. For a direct comparison with the other tools it necessary to define an approach for the selection of the parameters. Based on the insights from the last chapter, the following two approaches have been used.

Firstly, the parameters have been configured from a pure probabilistic view without considering the actual ontology to partition. Thus, this configuration is called "predefined". The concrete configuration is as follows: for creating term chunks the graph variant 1inf, the weight function wNone, the wtc algorithm, and the size function s2 have been used. Furthermore, for creating modular ontologies the plain RDF representation of the ontology structure, the fgc algorithm, the wBase4 weighting function and the s1 size function have been used.

Secondly, the parameters have been selected based on the performance of the framework for other ontologies. This can be considered as a training set for the framework. Thus, before an ontology is partitioned, its structural properties are analyzed with the metrics described in Section 5.3. Based on the complex metrics described in Section 5.3.3 the Euclidean distance between already partitioned ontologies and the ontology to partition is calculated. The configuration which led to the best results for the nearest ontology from the training set is then selected. The approach is called "distance-based" property determination.

7.2 Comparison for Reconstructing Term Chunks

The performance analysis with respect to the creation of term chunks has been done with the ontologies presented in Section 3.3.2. For each ontology in this set, the most similar ontology according to the Euclidean distance with the complex metrics from Section 5.3.3 has been chosen. From the exhaustive experiments presented in Chapter 6, the parameters leading to the best results are chosen. If this led to a set of configuration performing equally, the "predefined" approach has been applied to complete the parameter selection. Furthermore, if the most similar ontology is the inferred version of an ontology, than the ontology to partition was also loaded with activated inference. Table 6-1 provides an overview of the selected configurations for the performance analysis.

Ontology to partition	Nearest ontology	Graph representation	Algorithm	Weight function	Size function
aair	ecos	2	fgc	wNone	s2
bio	curio (inf)	2Inf	fgc	wBase5	s2
curio	swco	1	wtc	wBase2	s4
ecos	aair	2	wtc	wBase2	s2
foaf	aair	2	wtc	wBasw2	s2
gi2mo	bio (inf)	1Inf	wtc	wBase3	s2
music	premis	2	wtc	wNone	s2
opo	foaf	1	wtc	wBase2	s2
premis	pvc	2	ebc	wNone	s2
provo	ecos	2	fgc	wNone	s2
pvc	provo	1Inf	wtc	wNone	s2
rrdonto	provo (inf)	1Inf	wtc	wNone	s2
swco	curio	1	fgc	wNone	s2

Table 7-1: Overview of the selected configurations for term chunks based on the Euclidean distance.The column nearest ontologies shows the ontology that has the smallest Euclidean distance to the ontology in the first column and is therefore regarded as the most similar ontology.

Figure 7-1 presents the performance of the construction of term chunks with SWOOP, PATO and the mentioned two configuration approaches for the proposed framework, namely "distance-based" configuration and "predefined" configuration.

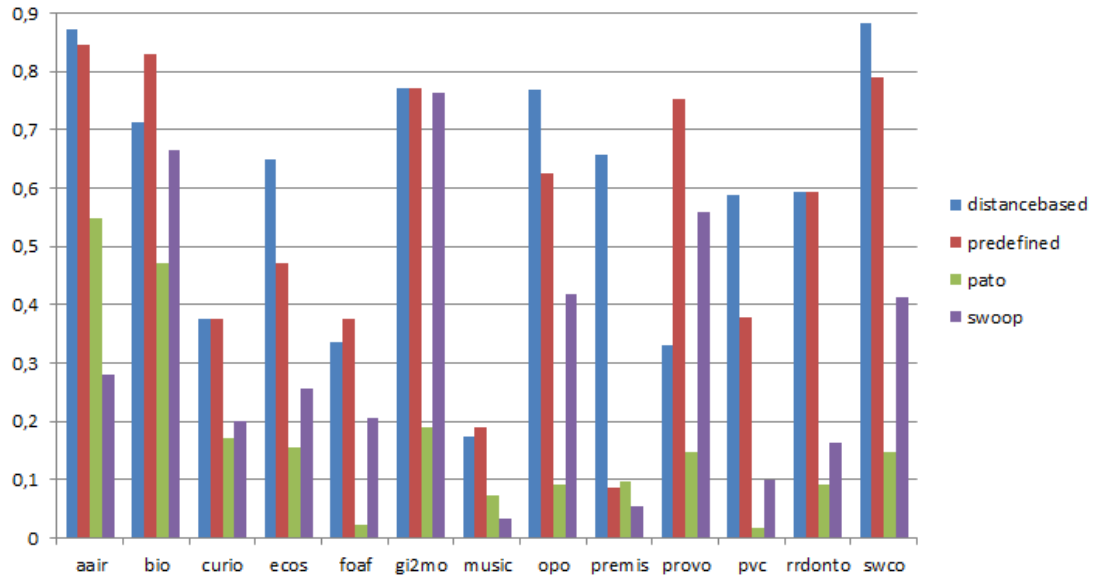


Figure 7-1: Performance for reconstructing term chunks with Pato, SWOOP, and the proposed framework with two different configurations: predefined parameter selection and distance-based parameter selection

In each case the proposed framework performs better than the other tools. The distance-based configuration leads to the best results for seven ontologies (aair with 0.871, curio with 0.377, ecos with 0.648, opo with 0.769, premis with 0.656, pvc with 0.587, and swco with 0.882), whereas the predefined configuration leads in four cases to the best results (bio with 0.829, foaf with 0.376, music with 0.191 and provo with 0.752). In case of gi2mo (0.77) and rrdonto (0.594) both approaches perform equally. Table 7-2 provides an overview of the average performance of the different partitioning techniques along with the standard deviations.

Partitioning technique	Average Score	Standard Deviation
Distance-based	0.593	0.178
Predefined	0.545	0.214
Pato	0.171	0.108
Swoop	0.317	0.189

Table 7-2: Average score values and standard deviation for the different partitioning techniques

The parameter analysis in Section 6.4 showed that the average value for all ontologies with all possible configurations led to an average value of 0.496. Comparing this value with the values from Table 7-2 makes clear that both configuration selection approaches perform better than a random selection would do. Furthermore, these values demonstrate that even a random selection of the configuration would perform better than SWOOP and Pato. Both are outperformed by the average value of all 480 configurations as well as with the two configuration selection approaches by far.

In order to evaluate the performance of the two configurations selection approaches "predefined" and "distance-based" with respect to the best possible configuration, their performances have been compared with the configurations leading to the best F-Measure values and the average score of the 480 different possible configurations. The result of this comparison is presented in Figure 7-2.

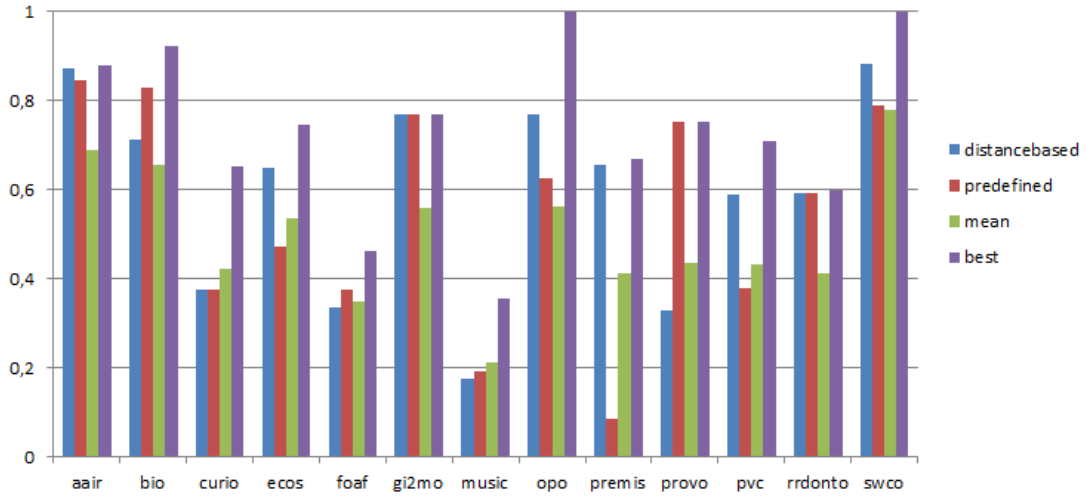


Figure 7-2: Performance of two possible configuration approaches of the proposed framework in comparison to the best possible result and the average performance

In eleven of thirteen ontologies (aair, bio, curio, ecos, foaf, music, opo, premis, pvc, rrdonto, and swco) both configuration approaches perform worse than the possible best solution. That means that both approaches are not able to identify the best possible configuration. In case of gi2mo the best possible configuration is found by both approaches, whereas in case of provo the predefined approach leads to the possible configuration. For six ontologies (aair, bio, gi2mo, opo, rrdonto, and swco) both configurations are leading to better scores than the average score for the 480 possible configurations. In two cases (curio and music) both configurations produce worse values

than the average value of the 480 configuration. That means that a random selection of the parameters would probably lead to better scores than both approaches.

On average, the distance-based approach leads to scores which are about 0.14 less than the highest possible values and 0.09 greater than the average score for all possible configurations. Similarly, the predefined approach leads on average to scores which are about 0.19 less than the highest possible value and 0.05 greater than the average score for all possible configurations.

7.3 Comparison for Reconstructing Modular Ontologies

The performance of the proposed framework with respect to the creation of modular ontologies out of monolithically created ontologies has been evaluated similar to the previous described evaluation. Modular built ontologies, which were presented in Section 3.4.1, have been merged and partitioned with SWOOP, Pato and the proposed framework with the mentioned two approaches to select the configuration. For the distance-based selection of the configuration the Euclidean distances between the ontologies have been calculated. Table 7-3 shows the result of this distance calculation and the selected configuration for each ontology.

Ontology to partition	Nearest ontology	Graph representation	Algorithm	Weight function	Size function
curioMerged	sioc	1	ebc	wBase4	s1
fiboMerged	curioMerged	1Inf	wtc	wLinear	s1
siocMerged	curioMerged	1	fgc	wNone	s1
spiceMerged	fiboMerged	1Inf	wtc	wBase4	sNone

Table 7-3: Overview of the selected configurations for modular ontologies based on the Euclidean distance.The column nearest ontologies shows the ontology that has the smallest Euclidean distance to the ontology in the first column and is therefore regarded as the most similar ontology.

Figure 7-3 shows how SWOOP, Pato and the proposed framework with the two configuration selection approached performed in creating modular ontologies out of monolithically created one.

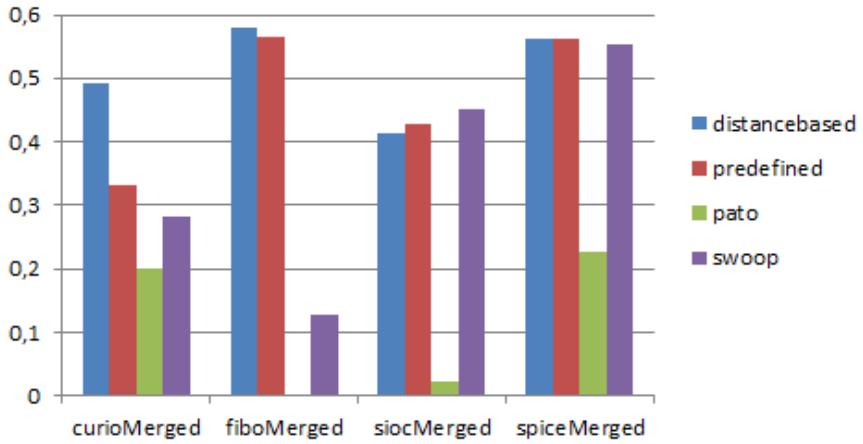


Figure 7-3: Performance for reconstructing modular ontologies with Pato, SWOOP, and the proposed framework with two different configurations

In three of four cases the distance-based configuration selection approach of the proposed framework leads to the best results (curioMerged with 0.491, fiboMerged with 0.578, and spiceMerged 0.563). In case of siocMerged the best result is produced by SWOOP with an F-Measure value of 0.452. Additionally, it is interesting to observe that in all cases the second best result is produced by the predefined configuration selection approach, whereas Pato produces the worst results for all ontologies. The overall performance is presented in Table 7-4.

Partitioning technique	Average Score	Standard Deviation
Distance-based	0.511	0.059
Predefined	0.472	0.092
Pato	0.113	0.101
Swoop	0.354	0.149

Table 7-4: Overall results for all merged ontologies with PATO, SWOOP and the proposed framework

Comparing these values with the average value of the 480 configuration as presented in Section 6.5 which is about 0.424 allows following conclusions: both configuration selection approaches perform better than a random selection of the configuration would do. Similar to the previous analysis, the best results on average are produced by the distance-based configuration selection approach. Even with a random configuration selection approach, the proposed framework outperforms the other tools by far.

The next analysis evaluates the performance of the configuration selection approaches. For that reason, both approaches are compared with configuration leading to the best results as observed in the parameter analysis in Section 6.5 and with the average performance of all possible 480 configurations. Figure 7-4 shows the results of this analysis.

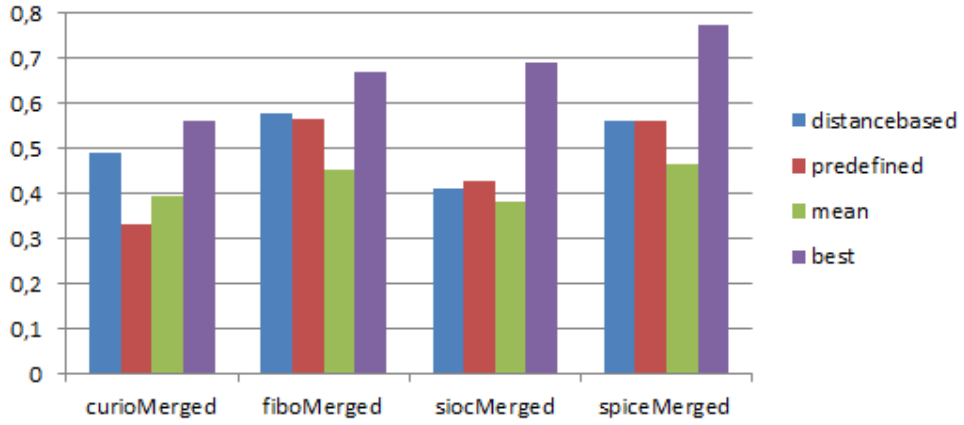


Figure 7-4: Performance of two possible configuration approaches of the proposed framework in comparison with the best possible result and the average performance

Obviously, both configuration selection approaches are not able to identify the best possible configuration in any case. However, apart from the predefined approach in case of curioMerged, they lead to configurations which perform better than the average value of the 480 possible configurations. That means that both approaches are on average better than a random selection approach of the configuration. The distance-based approach leads results which are 0.162 worse than the best possible results and 0.088 better than the average result. The predefined approach leads to results which are 0.203 worse than the best results and 0.048 better than the average result.

7.4 Chapter Summary

In this chapter the outcome of this thesis which is an adaptable ontology partitioning framework was evaluated experimentally. For that purpose its performance regarding partitioning ontologies to create term chunks and partitioning ontologies to create modular ontologies out of monolithic ontologies was compared with two existing solutions. The results of the experiments demonstrated that the proposed framework outperforms the other solutions by far. From a probability view, even a random selection of the configuration would lead to better results than the other proposed ontology partitioning solutions.

Furthermore, the experiments illustrated obviously that both configuration selection approaches perform better than a random selection would, whereas the distance-based approach has produced slightly better results. This is an indication that similarly structured ontologies should be partitioned similarly. Additionally, this indicates that a larger training set – more ontologies with reference partitions – would improve the performance of the distance-based configuration selection approach.

8 DISCUSSION AND OUTLOOK

This doctoral thesis pursued the goal of creating an adaptable and structure-based ontology partitioning framework. The main focus is on ontologies within the area of the Semantic Web and Linked Data. The strongly dynamic and highly distributed nature of the Web demands for sophisticated integration techniques and rapid adaptability to the changing environment. Thus, existing ontologies in this field are known to be created with agile methodological approaches of ontology engineering – if at all - and are of low expressivity. While the latter justifies the structure-based approach, the former emphasizes the need for partitioning tools supporting the ontology engineer.

Since the targeted outcome is an artifact, namely a piece of software, the Design Science approach has been chosen as the basic methodology to structure the overall process (Hevner et al., 2004). Accordingly, at the very beginning the broader context of the research area was captured (Chapter 2), the problem of ontology partitioning in general was investigated in depth (Chapter 3), and existing attempts to solve the problem have been discussed (Chapter 4). By this means, a knowledge base was built, which is essential to comprehend the dimensions of the research area as well as the state-of-the-art and to achieve the necessary awareness for the problem at hand. The main insights of this phase can be summarized as follows:

- Due to the challenges of the Web, ontology engineering methodologies tend to become more and more agile, lightweight and collaborative. Automatic and semi-automatic techniques supporting different phases of the overall ontology lifecycle gain in importance.
- Increasing size and complexity of ontologies exacerbate reusability and maintainability. This is a crucial issue, as these aspects of ontologies are especially in the context of Semantic Web and Linked Data of essential importance and critical success factors. For that reason, techniques for breaking down the size and the complexity to an appropriate level are necessary. It turned out that this level depends on the concrete context of the user.
- Ontologies are complex artifacts comprising different aspects. Defining size and complexity is not trivial. Thus, a considerable large number of existing proposals are present to measure different aspects of ontologies. However, each one is based on its specific application context and on its own notion of size and complexity.

- In order to improve the reusability, documentation pages for potential reusers are created by the authors of the ontologies. These pages provide information at different levels of detail. Grouping concepts is one application for partitioning ontologies, which is an appropriate technique to break down the complexity. It allows the user better readability, enables faster comprehension and accelerates making decisions about reusability. The analysis of existing term chunks unveil that the values regarding the number and size of groups correlates to some degree. This is reasonable, as these groups are all created for humans and take only cognition into consideration. Moreover, these values correlate with findings from cognitive science with respect to the capacity of the working memory.
- Ontologies, which are expected to be rather complex, are created in a modular way, if the time and expected quality constraints provide the necessary conditions. As mentioned before, this is rarely the case. The reason for this is twofold. On the one hand, the Web environment does not allow for heavyweight ontology creation processes. On the other hand, if an ontology is created with high time and financial investments, it becomes a valuable resource for the authors. This in turn is a counter-argument to share the ontology for free in the Web.
- The size and number of modules of modular created ontologies show significant differences. Correlations like with term chunks cannot be identified. This is reasonable, because ontology modularity does not primarily target the cognition of the author, but the inherent modularity of the domain of discourse.

With these insights an adaptable and structure-based ontology partitioning framework has been designed and implemented. During this endeavor research questions, which are listed in Section 1.1, have been tackled. The main contributions, which have been elaborated, are as follow:

- Besides the RDF graph of an ontology an additional class-centric representation has been proposed, that is similar to classic entity-relationship-diagrams.
- Weight functions for the edges in the structural representation have been developed. On the one hand, these functions take the mostly used RDFS and OWL properties into consideration. On the other hand, they make use of the so-called base level originated in linguistic.
- The Modularity function for communities in social networks has been extended to take the number as well as the average size of partitions into consideration, according to configurable weights.
- For the evaluation of the partitioning process, term chunks from documentation pages and modular created ontologies have been regarded as reference partitionings. The performance in the reconstruction of those partitions has been

accepted as a performance measure to assess the quality of ontology partitioning.

- In order to understand how the framework performs depending on the structure of the ontology to partition, a comprehensive set of metrics have been defined. Those metrics measure different size properties, various complexity dimensions and the hierarchical structure of an ontology.
- One basic assumption of this work is that similarly structured ontologies should be partitioned alike. To calculate this similarity, the set of complex metrics were used along with the Euclidean distance. Before starting to partition an ontology, it is compared with already partitioned ontologies, for which the best performing configuration is known. That means that already partitioned ontologies are taken as a training set. If the mentioned assumption is true, the proposed system will improve with each new ontology.
- An additional approach to find the potentially best configuration for the proposed framework was to partition all known ontologies with reference models. 480 different configuration have been defined, which are accepted to sufficiently cover the parameter space of the framework. Configurations, which lead to the best results on average, are then selected without taking the ontology to partition and its structure into account.

The assessment of the quality for the proposed ontology partitioning framework was done by an experimental evaluation. Thirteen ontologies with term chunks in the documentation pages and four modular built ontologies were identified. Those were partitioned with the proposed framework in competition with PATO and SWOOP. These experiments revealed that the proposed framework outperforms the competitors by far for both the creation of term chunks and the creation of modular ontologies. The distance-based configuration selection approach led to the best results on average with F-Measure values of 0.593 for creating term chunks and 0.511 for creating modular ontologies. Even though both values are at an acceptable level, they do not justify a fully automatic partitioning. Therefore, the proposed framework should be primarily used in a semi-automatic manner as a support system, whereas the outcome should be refined manually. However, the experimental results allow drawing the following conclusions for the structure-based approach with community detection algorithms and ontology partition in general.

- For the analyzed two motivations of ontology partitioning, the application of community detection algorithms with semantic-based weighting of properties and the modified Modularity function seems to be more appropriate than the logic based approach of SWOOP and the simple structure-based approach of Pato.

- The experiments with 480 different configurations for each ontology revealed that very high F-Measure values are possible in most cases. That indicates that the parameters of the proposed framework provide good adaptability and flexibility. It is able to create very good term chunks and modular ontologies with certain configurations. This in turn makes clear that the most important challenge is to find the best performing configuration.
- The fact that the distance-based configuration selection approach leads to the best results, indicates that the assumption that similarly structured ontologies should be partitioned alike, is correct.

For the assessment of this work's quality and to gain idea about its possible impact on the research area, it is important to shed light on some critical aspects. The major critical aspect is the number of ontologies which have been used for the parameter and the performance analysis. Thirteen ontologies for term chunks and four ontologies for modular ontologies are statistically not significant. Therefore, the results of this work depend strongly on the concrete set of ontologies. It is possible that exactly these ontologies are not representative and lead to a distortion. However, the rare existence of partitioned ontologies emphasizes the need for supporting ontology partitioning and justifies research in this area. Besides, the possibility to extend the training set provides flexibility and enables sustainability.

Furthermore, the proposed framework was analyzed and optimized with already partitioned ontologies in terms of term chunks and modules. This might raise the question, if exactly these ontologies were created by authors who have a special focus on modularity, which in turn led to certain structural properties. On the other hand, if this is the case, those ontologies can be further analyzed to derive some best practices and design patterns. This can be done by making use of the defined structural metrics.

In fact, this work leaves room for various extensions. Regarding the framework itself, the weight functions as well as the modified Modularity function can be extended for other properties. Additionally, it is possible to use other community detection algorithms. With respect to the analysis of the framework and its optimization, the most obvious future work would be to make more experiments with other ontologies. In this regard, authors of ontologies could be asked to create partitions for different purposes, which then can be used for further analyzing and optimizing the framework. Otherwise, partitions for different ontologies could be created and the authors could be asked with a questionnaire to evaluate the outcome. If a significant set of ontologies are available more general analysis of the different algorithms depending on the structural properties can be done.

BIBILIOGRAPHY

Alani, H., & Brewster, C. (2006). Metrics for Ranking Ontologies. In *Proceedings of the 4th International Workshop on Evaluation of Ontologies for the Web (EON2006) at the 15th International World Wide Web Conference (WWW 2006)* (pp. 24–30). Edinburgh, Scotland.

Auer, S., Lehmann, J., & Ngonga Ngomo, A. C. (2011). Introduction to linked data and its lifecycle on the web. In *Reasoning Web Semantic Technologies for the Web of Data* (Vol. 6848, pp. 1–75). Springer.

Baddeley, A. (2003). Working memory : looking back and looking forward. *Nature Reviews Neuroscience*, *4*(10), 829–839.

Barclay, J. R. (1973). The Role of Comprehension in Remembering. *Cognitive Psychology*, *4*(2), 229–254.

Basca, C., Corlosquet, S., Cyganiak, R., Fernández, S., & Schandl, T. (2008). Neologism: Easy Vocabulary Publishing. In *Proceedings of the Workshop on Scripting for the Semantic Web, in conjunction with ESWC 2008*.

Beckett, D., & Berners-Lee, T. (2008). Turtle - Terse RDF Triple Language. *W3C Team Submission*. Chapman and Hall.

Berners-Lee, T. (2005). Notation3 (N3) A readable RDF syntax. *Design Issues*. W3C. Accessed March 11, 2014, from http://www.w3.org/DesignIssues/Notation3

Berners-Lee, T. (2006). Linked Data. *Design Issues. W3C*. ACM Press. Accessed March 11, 2014, from http://www.w3.org/DesignIssues/LinkedData.html

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *284*(5), 34–43.

Bontas, E. P., & Mochol, M. (2005). Towards a reuse-oriented methodology for ontology engineering. In *Proceedings of the 7th International Conference on Terminology and Knowledge Engineering (TKE 2005)* (pp. 1–12).

Bontas, E. P., Mochol, M., & Tolksdorf, R. (2005). Case Studies on Ontology Reuse. In *Proceedings of the 5th International Conference on Knowledge Management (I-Know'05)* (pp. 345–353). Graz, Austria.

Brachman, R. J., & Schmolze, J. G. (1985). An Overview of the KL-ONE knowledge representation system. *Cognitive Science*, *9*(2), 171–216.

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2007). On Finding Graph Clusterings with Maximum Modularity. In *Proceedings of the 33rd international conference on Graph-theoretic concepts in computer science* (pp. 121–132). Berlin, Heidelberg: Springer-Verlag.

Brickley, D., & Miller, L. (2010). FOAF Vocabulary Specification 0.97. Accessed January 01, 2014, from http://xmlns.com/foaf/spec/20100101.html

Burel, G. (2011). CURIO Core Vocabulary Specification v0.5. Accessed August 22, 2012, from http://socsem.open.ac.uk/ontologies/curio/

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems*, *14*(1), 20–26.

Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, *70*(6), 661111–661116.

Corcho, Ó., Fernández-López, M., Gómez-Pérez, A., & López-Cima, A. (2003). Building Legal Ontologies with METHONTOLOGY and WebODE. In *Law and the Semantic Web* (Vol. 3369, pp. 142–157).

d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., & Motta, E. (2007). Characterizing Knowledge on the Semantic Web with Watson. In *Evaluation of Ontologies and Ontology-Based Tools: 5th International EON Workshop* (Vol. 329, pp. 1–10). CEUR-WS.org.

D'Aquin, M., & Noy, N. F. (2012). Where to Publish and Find Ontologies? A Survey of Ontology Libraries. *Web semantics: Science, Services and Agents on the World Wide Web*, *11*(August), 96–111.

d'Aquin, M., Sabou, M., Dzbor, M., Baldassarre, C., Angeletou, S., & Motta, E. (2007). WATSON : A Gateway for the Semantic Web. In *The 4th Annual European Semantic Web Conference (ESWC 2007)*. Innsbruck, Austria.

d'Aquin, M., Sabou, M., & Motta, E. (2006). Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components. In P. Haase, V. Honavar, O. Kutz, Y. Sure, & A. Tamilin (Eds.), *WoMO* (Vol. 232). CEUR-WS.org.

d'Aquin, M., Schlicht, A., Stuckenschmidt, H., & Sabou, M. (2009). Criteria and Evaluation for Ontology Modularization Techniques. In H. Stuckenschmidt, C. Parent, & S. Spaccapietra (Eds.), *Modular Ontologies* (pp. 59–79). Berlin, Heidelberg: Springer-Verlag.

Davis, I., & Galbraith, D. (2010). BIO: A vocabulary for biographical information. Accessed August 22, 2013, from http://vocab.org/bio/0.1/.html

Di Maio, P. (2009). Toward Just Enough Ontology Engineeering. *Business Intelligence*, *Vol. 9, No*, 1–18.

Di Maio, P. (2011). "Just enough" ontology engineering. *Proceedings of the International Conference on Web Intelligence, Mining and Semantics - WIMS '11*, 1.

Ding, L., & Finin, T. (2006). Characterizing the Semantic Web on the web. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)* (pp. 242–257). Athens, GA, USA: Springer.

Doran, P., Tamma, V., & Iannone, L. (2007). Ontology Module Extraction for Ontology Reuse : An Ontology Engineering Perspective. In *CIKM '07: Proceedings of the Sixteenth ACM conference on Conference on information and knowledge management* (pp. 61–70). New York, NY, USA: ACM.

Duque-Ramos, A., Fernández-Breis, J. T., Stevens, R., & Aussenac-Gilles, N. (2011). OQuaRE : A SQuaRE-based Approach for Evaluating the Quality of Ontologies. *Journal of Research and Practice in Information Technology*, *43*(2), 159–176.

Dzbor, M., & Motta, E. (2008). Engineering and Customizing Ontologies. In *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications* (Vol. 7, pp. 25–57). Springer US.

Fernandez, M., Gomez-Perez, A., & Juristo, N. (1997). METHONTOLOGY: from Ontological Art towards Ontological Engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering* (pp. 33–40). Stanford, USA.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3-5), 75–174.

Gasevic, D., Zouaq, A., Torniai, C., Jovanovic, J., Hatala, M., Ga, D., & Jovanovi, J. (2011). An Approach to Folksonomy-Based Ontology Maintenance for Learning Environments. *IEEE Transactions on Learning Technologies*, *4*(4), 301–314.

Gauvin, M., Delgado, J., & Rodríguez, V. (2007). Represent Rights Data Ontology (RRDOnto). Accessed August 22, 2013, from http://dmag.ac.upc.edu/ontologies/rrdonto/index.html

Gomez-Perez, A., Fernandez-Lopez, M., & Corcho, O. (2004). *Ontological Engineering*. *Young* (p. 403). London: Springer Verlag.

Grau, B. C., Horrocks, I., Kazakov, Y., & Sattler, U. (2008). Modular Reuse of Ontologies: Theory and Practice. *J. of Artificial Intelligence Research (JAIR)*, *31*, 273–318.

Grau, B. C., Parsia, B., & Sirin, E. (2004). Working with Multiple Ontologies on the Semantic Web. In S. A. McIlraith, D. Plexousakis, & F. van Harmelen (Eds.), *International Semantic Web Conference* (Vol. 3298, pp. 620–634). Springer.

Grau, B. C., Parsia, B., Sirin, E., & Kalyanpur, A. (2005a). Automatic Partitioning of OWL Ontologies Using E-Connections. In *Proceedings of the 2005 International Workshop on Description Logics* (Vol. 147). Edinburgh, Scotland, UK: CEUR-WS.org.

Grau, B. C., Parsia, B., Sirin, E., & Kalyanpur, A. (2005b). Modularizing OWL Ontologies. In *Proceedings of the KCAP 2005 Workshop on Ontology Management. Banff, Canada, October 2005* (pp. 1–15). Banff, Canada.

Graves, A., Adalı, S., & Hendler, J. (2008). A Method to Rank Nodes in an RDF Graph. In *The 7th International Semantic Web Conference (Poster Session)* (Vol. 401). Karlsruhe, Germany: CEUR-WS.org.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Guenther, R., Brama, Y., Bredenberg, K., Caplan, P., Dappert, A., Di Iorio, A., … Zwaard, K. (2011). *PREMIS Data Dictionary for Preservation Metadata v.2.1* (p. 223).

Hartig, O., & Zhao, J. (2012). Provenance Vocabulary Core Ontology Specification. Accessed August 22, 2013, from http://trdf.sourceforge.net/provenance/ns.html

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2009). OWL 2 Web Ontology Language: Primer. *OWL 2 Web Ontology Language Primer*. World Wide Web Consortium (W3C). Accessed March 11, 2014, from http://www.w3.org/TR/owl2-primer/

Hoser, B., Hotho, A., Jäschke, R., Schmitz, C., & Stumme, G. (2006). Semantic Network Analysis of Ontologies. In *Proceedings of the 3rd European Semantic Web Conference* (pp. 514–529). Budva, Montenegro: Springer.

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology/IEEE Std 610.12-1990. *IEEE Std 610121990*. Institute of Electrical & Electronics Enginee.

Internation Organization for Standardization. (2011). Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models. *ISO*. International Organization for Standardization.

Jannink, J., Mitra, P., Neuhold, E., Pichai, S., Studer, R., & Wiederhold, G. (1999). An algebra for semantic interoperation of semistructured data. In *Workshop on Knowledge and Data Engineering Exchange KDEX99* (pp. 77–84). Chicago, Illinois, USA: IEEE Computer Society.

Khilwani, N., & Harding, J. (2009). Enterprise Competence Organization Schema. Accessed August 22, 2013, from http://kmm.lboro.ac.uk/ontologies/ecos/

Klinker, G., Bhola, C., Dallemagne, G., Marques, D., & McDermott, J. (1991). Usable and reusable programming constructs. *Knowledge Acquisition*, *3*(2), 117–135.

Kutz, O., Lutz, C., Wolter, F., & Zakharyaschev, M. (2004). E-connections of abstract description systems. *Artificial Intelligence*, *156*(1), 1–73.

Lebo, T., Sahoo, S., & McGuinness, D. (2013). PROV-O: The PROV Ontology. Accessed August 22, 2013, from http://www.w3.org/TR/prov-o/

Leicht, E. A., & Newman, M. E. J. (2008). Community structure in directed networks. *PHYSICAL REVIEW LETTERS*, *100*, 118703.

Li, N., Motta, E., & D'Aquin, M. (2010). Ontology summarization: an analysis and an evaluation. In *The International Workshop on Evaluation of Semantic Technologies (IWEST 2010)* (pp. 1439–1446).

Luczak-Rösch, M. (2011). *How are People Engineering Linked Data ? A Survey Snapshot about the Engineering Efforts Spent by Dataset Publishers* (p. 9). Berlin: Freie Universität Berlin.

Luczak-Rösch, M., & Heese, R. (2008). Managing Ontology Lifecycles in Corporate Settings. In *International Conference on Semantic Systems (I-SEMANTICS)* (pp. 150–157). Graz, Austria.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266.

Miller, G. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, *63*(2), 81–97.

Minno, M., & Palmisano, D. (2010). Atom Activity Streams RDF mapping. Accessed March 19, 2012, from http://xmlns.notu.be/aair/

Möller, K., Bechhofer, S., & Heath, T. (2009). Semantic Web Conference Ontology. Accessed March 19, 2012, from http://data.semanticweb.org/ns/swc/swc_2009-05-09.html

Motik, B., Patel-Schneider, P. F., & Horrocks, I. (2008). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. *W3C*. Accessed May 15, 2013, from http://www.w3.org/TR/2008/WD-owl2-syntax-20080411/

Newman, M. E. J. (2003). The Structure and Function of Complex Networks. *SIAM Review*, *45*(2), 167.

Newman, M. E. J. (2004a). Analysis of weighted networks. *Physical Review E*, *70*(5), 0561311–0561319.

Newman, M. E. J. (2004b). Fast algorithm for detecting community structure in networks. *Physical Review E*, *69*(6), 0661331–0661335.

Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, *69*(2), 0261131–02611315.

Ning, H., & Shihan, D. (2006). Structure-Based Ontology Evaluation. In *e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on* (pp. 132–137).

Nottingham, M., & Sayre, R. (2005). The Atom Syndication Format. *Request for Comments 4287*. IETF.

Noy, N. F., & Musen, M. A. (2004). Specifying Ontology Views by Traversal. In *International Semantic Web Conference* (Vol. 3298, pp. 713–725). Springer.

Oh, S., Yeom, H. Y., & Ahn, J. (2010). Evaluating ontology modularization approaches. In *Proceedings of the 8th International Conference on Frontiers of Information Technology* (pp. 1–6). New York, NY, USA: ACM.

Orme, A. M., Yao, H., & Etzkorn, L. (2006). Coupling Metrics for Ontology-Based Systems. *IEEE Software*, *23*(2), 102–108.

Paslaru-Bontas, E. (2007). *A contextual approach to ontology reuse: methodology, methods and tools for the Semantic Web*. Freie Universität Berlin, Berlin.

Patel-Schneider, P. F., & Horrocks, I. (2004). OWL Web Ontology Language Semantics and Abstract Syntax Section 4. Mapping to RDF Graphs. *W3C*. Accessed April 04, 2013, from http://www.w3.org/TR/owl-semantics/mapping.html

Pinto, H. S., Staab, S., & Tempich, C. (2004). DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)* (pp. 393–397). Valencia, Spain: IOS Press.

Poli, R., & Obrst, L. (2010). The Interplay Between Ontology as Categorial Analysis and Ontology as Technology. In *Theory and Applications of Ontology: Computer Applications* (pp. 1–26). Springer Netherlands.

Pons, P., & Latapy, M. (2006). Computing Communities in Large Networks Using Random Walks. *Journal of Graph Algorithms and Applications*, *10*(2), 191–218.

Raimond, Y., Giasson, F., Jacobson, K., Fazekas, G., Gängler, T., Reinhardt, S., & Passant, A. (2012). Music Ontology Specification. Accessed March 19, 2012, from http://musicontology.com/

Reichardt, J., & Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E*, *74*(1), 0161101–01611014.

Roberta, C., Alexandre, D., Vincent, L., & Carlo, R. (2007). *The Technology Roadmap of the Semantic Web*. *Business* (p. 51). Knowledge Web.

Rosch, E. (1978). Principles of Categorization. In *Cognition and Categorization* (pp. 27–48). Lawrence Erlbaum Associates.

Sametinger, J. (1997). *Software Engineering with Reusable Components* (p. 275). Springer-Verlag.

Schlicht, A., & Stuckenschmidt, H. (2006). Towards Structural Criteria for Ontology Modularization. In *WoMO* (Vol. 232). CEUR-WS.org.

Schlicht, A., & Stuckenschmidt, H. (2007). Criteria-based partitioning of large ontologies. In *Proceedings of the 4th international conference on Knowledge capture - K-CAP '07* (pp. 171–172). New York, New York, USA: ACM Press.

Schlicht, A., & Stuckenschmidt, H. (2008a). A Flexible Partitioning Tool for Large Ontologies. *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, *1*, 482–488.

Schlicht, A., & Stuckenschmidt, H. (2008b). Distributed Resolution for ALC. In *Proceedings of the 21st International Workshop on Description Logics (DL2008)* (pp. 1–11). http://ceur-ws.org.

Seidenberg, J., & Rector, A. (2006). Web ontology segmentation: analysis, classification and use. In *WWW '06: Proceedings of the 15th international conference on World Wide Web* (pp. 13–22). New York, NY, USA: ACM Press.

Shneiderman, B. (1977). Measuring computer program quality and comprehension. *Information Systems Journal*, *9*(4), 465–478.

Simperl, E. (2009). Reusing ontologies on the Semantic Web: A feasibility study. *Data & Knowledge Engineering*, *68*(10), 905–925.

Staab, S., Studer, R., Schnurr, H.-P., & Sure, Y. (2001). Knowledge Processes and Ontologies. *IEEE Intelligent Systems*, *16*(1), 26–34.

Stankovic, M. (2010). OPO - Online Presence Ontology Specification. Accessed August 22, 2013, from http://online-presence.net/opo/spec/

Stuckenschmidt, H. (2006). Network Analysis as a Basis for Partitioning Class Hierarchies. In *Workshop on Semantic Network Analysis, ISWC* (pp. 43–54).

Stuckenschmidt, H., & Klein, M. (2004). Structure-based partitioning of large concept hierarchies. In *Proceedings of the 3rd International Semantic Web Conference* (pp. 289–303).

Suárez-Figueroa, M. C., de Cea, G. A., Buil, C., Dellschaft, K., Fernández-López, M., García, A., … Yufei, Z. (2008). *D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks* (p. 150).

Sure, Y., Staab, S., & Studer, R. (2009). Ontology Engineering Methodology. In *Handbook on Ontologies* (pp. 135–152). Springer Berlin Heidelberg.

Takeda, H., Veerkamp, P., Tomiyama, T., & Yoshikawam, H. (1990). Modeling Design Processes. *AI Magazine*, *11*(4), 37–48.

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., & Aleman-meza, B. (2005). OntoQA : Metric-Based Ontology Quality Analysis. In *IEEE ICDM 2005 Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*.

Theoharis, Y., Tzitzikas, Y., Kotzinos, D., & Christophides, V. (2008). On Graph Features of Semantic Web Schemas. *IEEE Transactions on Knowledge and Data Engineering*, *20*(5), 692–702.

Volz, J., Bizer, C., Gaedke, M., & Kobilarov, G. (2009). Silk – A Link Discovery Framework for the Web of Data. In *Proceedings of the 2nd Workshop on Linked Data on the Web (LODW2009)*.

Vrandecic, D. (2010). *Ontology Evaluation*. Karlsruher Institut für Technologies.

Westerski, A. (2011). Gi2MO Types Taxonomy Specification. Accessed August 22, 2013, from http://www.gi2mo.org/taxonomy/

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, *16*(3), 645–678.

Yang, Z., Zhang, D., & Ye, C. (2006). Evaluation Metrics for Ontology Complexity and Evolution Analysis. In *E-Business Engineering, IEEE International Conference on* (Vol. 0, pp. 162–170). Los Alamitos, CA, USA: IEEE Computer Society.

Yao, H., Orme, A. M., & Etzkorn, L. (2005). Cohesion Metrics for Ontology Design and Application. *Journal of Computer Science*, *1*(1), 107–113.

Yourdon, E. (2006). *Just Enough Structured Anylsis*. This is an update, condensation, and pragmatic revision of Yourdon's book \textit{Modern Structured Analysis}(Prentince Hall 1989).

Zhang, H., Li, Y.-F., & Tan, H. B. K. (2010). Measuring design complexity of semantic web ontologies. *Journal of Systems and Software*, *83*(5), 803–814.

TERMINOLOGY

Cognition: Cognition is the process of knowing and understanding.

Cognitive: Cognitive means relating to the mental process involved in knowing, learning, and understanding things.

Complexity: Complexity is the degree to which a system or component has a design or implementation that is difficult to understand and verify.

Conceptualization: Conceptualization is the creation of an ontological model describing a domain of interest independently of any particular implementation language. When referring to a general "model" the meaning of the term is similar to that of "conceptualization" or "conceptual model". These will be used interchangeably within this thesis.

Comprehension: Comprehension is the ability to understand the meaning or importance of something (or the knowledge acquired as a result);

Cognitive Capacity / Cognitive Limit: Cognitive capacity or the cognitive limit is the total amount of information the brain is capable of retaining at any particular moment.

Evaluation: Evaluation is the process of assessing the general-purpose or application-oriented quality.

Modular: Modular describes the property of being composed of discrete parts.

Modular Decomposition / Modularization: Modular decomposition is the process of breaking a system into components to facilitate design and development.

Module: (1) A module is a program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine. (2) A module is a logically separable part of a program. Note: The terms "module," "component," and "unit" are often used interchangeably or defined to be sub-elements of one another in

different ways depending upon the context. The relationship of these terms is not yet standardized.

Modularity: Modularity is the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Maintainability: (1) Maintainability is the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. (2) Maintainability is the ease with which a hardware system or component can be retained in, or restored to, a state in which it can perform its required functions.

Maintenance: (1) Maintenance is the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment. (2) Maintenance is the process of retaining a hardware system or component in, or restoring it to, a state in which it can perform its required functions.

Measurement Standard: A measurement standard is a standard that describes the characteristics of evaluating a process of product.

Method: A method is a codified series of steps taken to develop a product or to perform a service. In computer science its meaning is similar to that of the terms "technique", "algorithm", "function" or "procedure", which will be used alternatively in this thesis.

Methodology: Methodology is defined as "a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed."

Metric: A metric is a quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Ontology: An ontology is a problem-relevant, shared, and formal specification of a conceptualization.

Ontology engineering: Ontology Engineering or sometime referred to as "Ontological Engineering refers to the set of activities that concern the ontology development

process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them" (Gomez-Perez et al., 2004).

Ontology Reuse: Ontology reuse is the process in which existing ontological knowledge is used as input to generate new ontologies (Paslaru-Bontas, 2007)

Process: (1) A process is a sequence of steps performed for a given purpose; for example, the software development process. (2) A process is an executable unit managed by an operating system scheduler.

Partitioning: Partitioning is the act or process of dividing something into parts. These parts can be mutually exclusive or overlapping.

Quality: Quality is the degree to which a system, component, or process meets specified requirements.

Reusability: Reusability is "the degree to which a software module or other work product can be used in more than one computing program or software system". In a specific context the term is similar to "usability" or "utility".

LIST OF FIGURES

Number *Page*

LIST OF TABLES

ABBREVIATIONS

aair: Atom Activity Streams Ontology

ABox: Assertional Box

AI: Artificial Intelligence

bio: Biographical Ontology

COLM: Corporate Ontology Lifecycle Methodology

curio: Collaborative User Resource Interaction Ontology

DL: Description Logics

ebc: Edge Betweenness Community

ecos: Enterprise Competence Organization Schema

EDM: Enterprise Data Management

fgc: Fast Greedy Community

fibo: Financial Industry Business Ontology

foaf: Friend of a Friend Vocabulary

gi2mo: GI2MO Types Taxonomy

JEOE: Just Enough Ontology Engineering

LOD: Linked Open Data

music: Music Ontology

OMG: Object Management Group

opo: Online Presence Ontology

OTKM: On-To-Knowledge Methodology

OWL: Web Ontology Language

premis: Premis

provo: W3C Prov Ontology

pvc: Provenance Vocabulary Core

RDF: Resource Description Framework

RDFS: RDF Schema

rrdonto: Represent Rights Data Ontology

SemNA: Semantic Network Analysis

sioc: Semantically-Interlinked Online Communities

spice: SPICE Mobile Ontology

SQuaRE: Software Quality Requirements and Evaluation

SUMO: Suggested Upper Merged Ontology

swco: Semantic Web Conference Ontology

TBox: Terminological Box

URI: Uniform Resource Identifie

W3C: Word Wide Web Consortium

wtc: Walktrap Community

XML: Extensible Markup Language

ZUSAMMENFASSUNG

Komponentenbasierte Entwicklung von komplexen Softwaresystemen verbessert die Wartbarkeit und führt zu wiederverwendbaren Softwaremodulen. Ausgehend von dieser Erfahrung wird angenommen, dass die komponentenbasierte Entwicklung von Ontologien ähnliche Vorteile bringt. Allerdings sind die meisten Ontologien monolithisch aufgebaut, so dass mit der steigenden Anzahl online verfügbarere Ontologien auch die Größe und Komplexität mit angestiegen ist. Für die effiziente Nutzung, die einfache Wartbarkeit und die Möglichkeit wiederverwendbarer Komponenten bedarf es daher geeigneter Partitionierungstechniken. Insbesondere im Kontext von Semantic Web ist die Wiederverwendung von Ontologien von essentieller Bedeutung, da diese die webübergreifende Datenintegration und Interoperabilität heterogener Systeme ermöglichen.

In dieser Arbeit wird ein strukturbasierter Ansatz zu Partitionierung von Ontologien verfolgt, in dem Ontologien als Netzwerke repräsentiert werden. Diesen wird eine Kantengewichtung hinzugefügt, welches die semantischen Beziehungen innerhalb der Ontologien berücksichtig. Darauf aufbauend wird ein konfigurierbarer Ansatz zur Partitionierung von Ontologien mit Hilfe von Community Detection Algorithmen aus dem Bereich der sozialen Netzwerke erarbeitet. Dabei liegt das Hauptaugenmerk auf zwei konkreten Anwendungsfällen für die Partitionierung, nämlich der Modularisierung von existierenden komplexen Ontologien zur Vereinfachung der Wartbarkeit und der Erzeugung von Begriffsgruppierungen für die Dokumentationsseiten zur Unterstützung der Wiederverwendbarkeit. Anforderungen für beide Fälle werden aus existierenden Lösungen extrahiert, welche im späteren Prozess in einem Goldstandardansatz als Referenzmodell auch zur Evaluation verwendet werden.

In experimentellen Analysen des vorgeschlagenen Ansatzes werden die besten Parameterwerte für die jeweiligen Anwendungsfälle ermittelt. Mit diesen wird das System dann mit den bereits existierenden Werkzeugen zur Ontologiepartitionierung SWOOP und Pato verglichen. In diesem direkten Vergleich kann gezeigt werden, dass der hier erarbeitete Ansatz signifikant bessere Ergebnisse als die beiden Konkurrenten liefern kann. Allerdings sind die Ergebnisse nicht so gut, dass davon ausgegangen werden kann, dass ein vollständisch automatischer Prozess für die Partitionierung möglich ist. Der strukturbasierte Ansatz zur Partitionierung kann nur für eine semiautomatische Partitionierung verwendet werden, so dass die Nutzer die Ergebnisse manuell nachbessern müssen.

Curriculum Vitae

(Der Lebenslauf ist in der Online-Version aus Datenschutzgründen nicht enthalten.)

Erklärung

Hiermit erkläre ich, dass alle Hilfsmittel und Hilfen angegeben sind und versichere, auf dieser Grundlage die vorliegende Arbeit selbstständig verfasst zu haben. Diese Arbeit wurde bisher noch nicht in einem früheren Promotionsverfahren eingereicht.

Berlin, 18.03.2014 Gökhan Coskun