Usage-dependent maintenance of structured Web data sets

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

am Institut für Informatik
des Fachbereichs Mathematik und Informatik
der Freien Unviersität Berlin

vorgelegt von

Dipl. Inform. Markus Luczak-Rösch

Berlin, August 2013

Referent:
Prof. Dr.-Ing. Robert Tolksdorf (Freie Universität Berlin)

Erste Korreferentin:
Natalya F. Noy, PhD (Stanford University)
Zweite Korreferentin:
Dr. rer. nat. Elena Simperl (University of Southampton)

Tag der Disputation: 13.01.2014

To Johanna.
To Levi, Yael and Mili.
Vielen Dank, dass ich durch Euch eine Lebenseinstellung lernen durfte,
"... die bereit ist, auf kritische Argumente zu hören und von der Erfahrung zu lernen. Es ist im Grunde eine Einstellung, die zugibt, daß ich mich irren kann, daß Du Recht haben kannst und daß wir zusammen vielleicht der Wahrheit auf die Spur kommen." – Karl Popper

Abstract

The Web of Data is the current shape of the Semantic Web that gained momentum outside of the research community and becomes publicly visible. It is a matter of fact that the Web of Data does not fully exploit the primarily intended technology stack. Instead, the so called Linked Data design issues [BL06], which are the basis for the Web of Data, rely on the much more lightweight technologies. Openly available **structured Web data sets** are at the beginning of being used in real-world applications. The Linked Data research community investigates the overall goal to approach the Web-scale data integration problem in a way that distributes efforts between three contributing stakeholders on the Web of Data – the data publishers, the data consumers, and third parties. This includes methods and tools to publish and consume Linked Data on the Web, to deal with data quality issues in such structured Web data sets, and to improve the linkage between data sets as well as the mappings between underlying vocabularies. Web ontologies are the Web-compatible and machine-processable schema for structured Web data. With reference to the Semantic Web standards one can claim that a structured Web data set altogether is nothing else than a specific form of an ontology that applies subsets of the conceptual knowledge of numerous ontologies to populate instance data. Research on ontology engineering advances the processes, guidelines, and tools for the creation and management of ontologies and has evolved from describing their scratch development towards an integrative life cycle support.

Contextualized at the interface between research on ontology engineering and research on structured Web data this thesis deals with the analysis of the usage of structured Web data sets with the goal to **support data publishers in managing**

maintenance activities as part of the data set life cycle. We focus data sets which follow the Linked Data design issues and offer a SPARQL endpoint to query the data. In this frame we design, develop, instantiate, and study approaches to answer the following three research questions: (1) What are the blind spots between ontology engineering and Linked Data? (2) How can classical Web usage mining methods be applied in the context of structured Web data sets and how does that affect managing data set maintenance? (3) To which extend can usage-dependent metrics help to assess the quality of a Web data set?

Since the scope of this thesis spans across the borders of more than one discrete research area its contributions do so as well. Furthermore, it is characteristic for research that deals with methodological innovation that not only process guidelines are designed in theory but also necessary tools when an approach cannot be fulfilled by state-of-the-art technology which also results in multiple contributions. This thesis contributes an analytical survey among a representative set of Linked Open Data providers to understand the role of ontology engineering methodologies in structured Web data publication and management. Along the requirements derived from this study we propose a methodology that describes the processes, activities, methods, and tools of a usage-dependent data set life cycle. The evaluation of the data set quality is a core component of our proposed methodology. We present an instantiation of this by performing Web usage mining on SPARQL query logs. For the preprocessing of classical server log files which contain SPARQL queries we introduce a preprocessing and enrichment algorithm which allows for an in-depth analysis of successful and failing queries as well as their atomic parts. A statistical framework on top of our enriched usage database features a plain and a hierarchical data quality score function.

The evaluation is done multi-perspectively capturing three pillars. First, the data set life cycle is set into context with the most recent and most established methodologies for ontology and data set development or management by application of the state-of-the-art framework for the qualitative evaluation of ontology engineering methodologies and the ONTOCOM cost model. Second, the representativeness of our data quality framework is critically discussed by comparing it to the state-of-the-art in data quality research which is based on empirical evidence about the importance of particular quality dimensions for the data consumer. Third, in a number of experiments we analyzed real-world log files of different Linked Open Data data sets to prove the applicability of the entire approach in practice and to gain an understanding how our usage-dependent data quality score functions perform in comparison to the state-of-the-art in data set maintenance as a baseline procedure.

Acknowledgments

Writing a doctoral thesis may be like a long run. In the end it is for no one else than yourself, but along the way there are times when you need the support of many people to stay the course.

First and foremost I would like to thank my supervisor, Prof. Dr.-Ing. Robert Tolksdorf, for his indispensable advice and unequaled patience. I think it is not usual to receive the professional and personal support I received while being your research assistant. I also thank my co-supervisors Natasha Noy and Elena Simperl for the stimulating discussions about my research and their relaxed attitude towards the unexpected dynamic the fatherhood of an apprentice may develop.

I thank my parents, Gerd Luczak and Elke Luczak, as well as my brother, Sascha Luczak, for encouraging me to follow my scheme of life. Special thanks go to Dr. phil. Tilo Naatz for philosophical discussions and a new view to life in any literal sense. Claudia Müller-Birn deserves gratitude for pitching a couple of unimagined insights into my research.

But in the end there are only four people who experienced the ups and downs of the last five years directly. My wife Johanna and our children Levi, Yael, and Mili became my key motivator. Johanna, thank you for what you are and for sharing a wonderful life with me. I owe you five years of hilarity. Levi and Mili, both of you remind me every day that "who can no longer pause to wonder and stand rapt in awe, is as good as dead" (Albert Einstein). But sometimes I think it is you, Yael, who suffered the most from my occasional absence. Be assured that I recognized the whole potential of your cordiality and also the seriousness behind it.

Markus Luczak-Rösch, Berlin, August 15, 2013

Contents

List of Figures

List of Tables

List of Formulae

CHAPTER 1

Introduction

The **Semantic Web** [BLHL01], as it was invented early in the first decade of this millennium, is an extension of the classical Web of documents so that machine processable metadata facilitates the interoperation and cooperation of humans and machines on the global Web scale. Since the 1980s the classical Web itself has evolved that far that it is commonly understood as one global information space, characterized by its openness and driven by simple and well-known principles: HTTP, URIs, and hypertext.

A comprehensive set of technologies has been designed and developed to enable the vision of the Semantic Web. The most commonly agreed ones have been subsumed within a Semantic Web technology stack. This stack walks through different layers of complexity: From the bottom of Unicode and URIs (or IRIs respectively), via the median levels of structured data, knowledge representation and ontology languages, up to the higher levels where security, provenance and trust play crucial roles for semantic applications that rely on heavy modelling, complex rules, and inference. Ontologies are a central component for the Semantic Web. They are the means to represent, share, and exchange knowledge on and through the Web, defined as an "explicit specification of a conceptualization" [Gru95]. Research on ontology engineering has evolved from describing the scratch development of ontologies towards an integrative life cycle support. The ontology engineering discipline, sometimes also referred to as ontological engineering, has changed from an expert-oriented art towards a collaborative and distributed process with disparate skilled users developing

consensual and distributed networks of ontologies.

The Web of Data is the current shape of the Semantic Web that gained momentum outside of the research community and becomes publicly visible. It is a matter of fact that the Web of Data does not fully exploit the primarily intended (and repeatedly revised) technology stack. Instead, the so called Linked Data design issues [BL06], which are the basis for the Web of Data, rely on the much more lightweight technologies. Promoting another set of four simple principles – namely HTTP, URIs, the RDF and SPARQL standards, and RDF links – this initiative was successfully advanced at the latest when, amongst others, the public sector, online retailers and the media started to publish Linked Data. Reverse, these open available **structured Web data sets** are also at the beginning of being used in real-world applications. Web ontologies are the Web-compatible and machine-processable schema for Linked Data. The Linked Data research community investigates the overall goal to approach the Web-scale data integration problem in a way that distributes efforts between three contributing stakeholders on the Web of Data – the data publishers, the data consumers, and third parties. This includes methods and tools to publish and consume Linked Data on the Web, to deal with data quality issues in such structured Web data sets, and to improve the linkage between data sets as well as the mappings between underlying vocabularies. Linked Data is typically applied to (1) provide unambiguous concept identifiers within Web applications, (2) enhance the experience of users by aggregation and integration of corresponding content within Web-based information systems , and (3) be browsed and mashed up in an individual way. Common access patterns to realize such application scenarios are (1) requests for atomic information resources via HTTP and potentially the augmentation of RDF links identified in the responded representation, (2) download of data set dumps for client-side processing, and (3) querying of data sets via server-side HTTP SPARQL interfaces.

This thesis deals with the analysis of the usage of structured data sets on the Web with the goal to support data publishers in **managing maintenance** activities as part of the data set life cycle. We focus data sets which follow the Linked Data design issues and offer a SPARQL endpoint to query the data. That means the data sets constitute of a set of RDF triples which describe Web resources represented by dereferencable HTTP URIs and contain triples which link to Web resources in other data sets which conform to the same principles. As a schema to describe such data the data publishers consult selected concepts and properties of multiple Web ontologies. The process of building and maintaining these ontologies is performed by (1) a party which wants to provide a standard vocabulary which can be used in a variety of data sets to describe specific parts of the data or (2) a data publisher who wants to share instance data about a specific domain of discourse or conforming to an individual viewpoint of a domain at which our focus is on the latter group. Data set

usage via publicly available SPARQL endpoints is the usage paradigm of interest in the context of this thesis. Such queries have the special character of accessing more than one atomic Web resource at a time due to the flexible use of SAPRQL basic graph patterns. Some of the approaches in and results of this thesis will also apply to other forms of structured Web data sets, other ontology or query languages and also different application scenarios. On the contrary the outcome will not apply to specific usage patterns such as some of the federated querying approaches which shift query processing on the client side. In turn, several issues and approaches discussed in earlier work on ontology engineering and Web usage mining cannot be transferred directly to structured Web data sets. With our research we focus on structured data sets which are created and maintained application-independent: the requirements of an application that uses the data set arise and evolve independently from the control of the data set publisher.

This chapter contains the introduction to the problem domain our work is addressing. This covers the overall motivation including representative use cases (Section 1.1), the hypothesis and research questions (Section 1.2) of the thesis, a brief overview of the contributions (Section 1.3), and a readers' guide (Section 1.5) for the course of the remainder of this thesis. It closes with a referential section on related publications by the author of this thesis (Section 1.6).

1.1 Motivation

Our research has its background in the ontology engineering area. ontology engineering is a mature research discipline which was comprehensively studied and evaluated but experienced limited adoption in real world scenarios except in some characteristic large or critical projects as well as specific domains. This was already evidenced in [ST06] and [SMB10] where the authors identified process-related gaps as a responsible factor for this limited adoption. Paslaru-Bontas and Tempich [ST06] found out that **ontology maintenance is one of the central process issues of ontology engineering** which hinders a broader success in practice. As an explicit part of the ontology life cycle **maintenance got limited regard in the literature most commonly treated as a post-development process in parallel but also interfering with ontology use** [GPFLC04, ST06].

A bit aside of research on ontology engineering the Linked Data paradigm came up and evolved towards the publicly most regarded shape of the Semantic Web today. **A Linked Data set is a set of structured data represented as RDF in a Web-compatible form and can be treated as an ontology** conforming to the Semantic Web ontology languages which do not necessarily divide terminological

knowledge from factual knowledge. Reviewing literature and examining the publicly available information about Linked Data projects reveals the intuitive assumption that **in this context methods and guidelines from the ontology engineering discipline that capture the life cycle of schemas (the terminological knowledge) as well as instance data (the factual knowledge) are even less regarded and in practice not relevant**. In a very recent approach Möller designs an abstract data life cycle for the "arguably largest ... [data-centric] ... system in existence" the Web of Data [MÏ12]. The author explicitly mentions the distinction of the ontology life cycle and the life cycle of instance data when a life cycle is dedicated to "a particular piece of data or metadata" and not "a complete system [including a schema and associated instance data] as a whole". This distinction also reflects in his process model where ontology development is treated as an isolated process apart from the data life cycle. Möllers argument holds if data set publishers on the Web of Data only use standard vocabularies being developed and maintained by other parties. But it fails, on the contrary, if data set publishers also apply self-developed ontologies as the schema of their data which need to be developed and maintained. As a matter of fact it is an open question which practice is commonly performed in projects implementing and managing data-centric systems in practice such as the numerous Linked Open Data sets.

This divergence between the ontology engineering discipline and research on Linked Data publication and management is generally worth to be investigated in order to design and study approaches that aim to overcome it. We decided to perform an empirical survey about the role of ontology engineering in Linked Data publication and management. This survey was dedicated and succeeded to reveal blind spots at the interface between the two research directions and is the fundament for focusing on a usage-dependent maintenance approach in this thesis.

To give a second more forward-looking motivation than the history of science perspective and the empirical study mentioned above, it is possible to consult the following two dimensions, each attended by at least one exemplary use case.

1.1.1 Open-ended requirements of data-centric applications

Data-centric applications are applications that apply the maximum amount of available and accessible data to provide statistical insights or decision support by a flexible orchestration and interlinking of data. Data in this context is not necessarily sourced from a single provider and the providers are not necessarily involved in the design and development of new applications on top of their asset so that the requirements of data-centric applications are mostly open-ended for the data provider. This decoupled evolution of data sets and applications calls for methods which allow tha data

provider to deduce requirements of the data consumer in order to provide convenient data of best quality possible.

The following use case illustrates how a usage-dependent maintenance approach aims to increase the adaptability of data in a scenario where applications try to use as much data as possible in an ad hoc fashion in order to make decisions.

> *In manufacturing and logistic environments product memories which monitor and log information about the manufacturing and delivery process chain are an example for data-collecting sensors which become state-of-the-art. It is also already possible that systems of the supplier for example communicate with such data in their environment. In the future the computational facility of each single entity in this context will further increase so that also such a product memory itself can process data of its environment in order to make decisions such as an ad hoc change of the delivery destination for example because a station in the supply chain offers data about delayed flights to the original destination so that a further processing of the product at an alternative location is beneficial.*

The scope of potential applications, especially when they aggregate data from various sources in a spatio-temporal environment, is open-ended and so are their requirements. In order to avoid that a huge number of manufacturers and application or service developers is required to perform a global-scale standardization effort for every single entity, device, application, or service it is a possibility to rely on the Web principles as a standardized access infrastructure and serve structured data. Tracking, reporting and analyzing the usage of this structured data helps to evolve the data continuously conforming to changing and a priori unknown application requirements.

1.1.2 Demands of the Web of Data

The literally largest data-centric system is the Web of Data which embraces all the open accessible raw structured data on the public Web and for which Heath and Bizer claim that the distribution of the data integration effort between data publishers, data consumers, and third parties is a key driver for its success.

In practice this theoretical demand of the Web of Data means that the three parties need to be supported by methods and tools for sustaining data publication and management and the following open data use case illustrates how this effects open data providers.

> *Consider the current world-wide trend – which is also a requirement of democ-*

racy pushed forward by many countries' citizens – that governments publish open data on the Web. Internally this helps to reduce the efforts for proprietary prepared data diagrams on Web pages and externally it opens new perspectives for third-party applications. In an "Open Traffic Data" scenario traffic lights collect data about the traffic but also the pollution at the respective place of a traffic light. Infotainment systems of cars and other vehicles, but also smartphones of pedestrians, are able to access the data of data-sharing devices in their environment and mobile applications use that urban data from the spatio-temporal environment of the user on demand. Two potential applications may be that (1) passing cars collect data on their track and use it for future route planning dynamically or (2) the car driver addicts to responsible behaviour and avoids to use her car at heavily polluted places because a smartphone app indicated that.

It is obvious that the possible use cases exploiting open government data are more than just wide. Applications can be beneficial for enterprises economically as well as a nation's citizens in terms of responsibility, transparency, and democracy. For the data publisher it is nearly impossible to anticipate any kind of applications on top of the data or any kind of new data being related to the current while for the public it is necessary to find data of high quality. Due to that the open data provider (or a third party offering fused and cleaned open data as a payed service) is required to maintain her data sets seriously which she needs to perform in a cost-sensitive fashion related to the users' needs.

1.1.3 Pay-as-you-go data integration in distributed dataspace scenarios

The classical relational database abstraction requires a unified schema when data from a set of different sources needs to be integrated. The massively increasing amount of data on the open Web but also in today's critical business applications – sometimes referred to as Big Data – motivated a new paradigm to look at information management and data integration at scale. Franklin et al. coined the term *dataspaces* [FHM05] for infrastructures consisting of huge numbers of structured data sources with heterogeneous schemas. An a priori schema consolidation in such environments is a literally impossible task, so Madhavan et al. introduced the pay-as-you-go data integration approach in the context of dataspaces which proposes "a combination of automated techniques that find and/or suggest relationships, and of techniques that involve feedback from users" [MJC+07] resulting an emergent integration of heterogeneous data sources over time.

While the Web itself is the largest and most prominent dataspace infrastructure

it is also possible to consider the following use case for a closed corporate dataspace setup.

> *A company operating a large network of hospitals and continuously expanding it through acquisition of further ones faces the challenge to integrate new IT infrastructures and data into the corporate wide standard continuously. One possibility to leverage the potential of dataspaces in this case would be to combine all corporate-wide data sources with the sources from a newly acquired hospital, model the requirements of the target infrastructure as a comprehensive set of queries and relationships in this new dataspace a priori and then image the new data into the standard infrastructure so the dataspace is no more needed afterwards. Another option would be to follow the way of dataspaces more rigorous and go further than just modeling and performing the integration of new data sources at a dedicated point of time. The company could also decide never to integrate any of the data sources from newly acquired hospitals a priori but set up all corporate-wide business applications to operate above a continuously growing dataspace architecture.*

One can see that the dataspace paradigm can be beneficial as an appropriate layer of abstraction for modeling a data integration scenario and for performing an imaging of data. But going further and applying the pay as you go data integration approach can make it obsolete to invest into data integration a priori. In the latter case the analysis of the usage of the data sources in the dataspace can be one "mechanism. . . to improve the semantic relationships over time" [MJC+07].

1.2 Research questions

Contextualized at the interface between research on ontology engineering and research on the Web of Data this thesis grounds on the following hypothesis:

1. Hypothesis. *Ontology engineering guidelines can be adapted for the maintenance of structured Web data sets.*

In this frame we design, develop, instantiate, and study approaches to answer the following three research questions:

1.2.1 What are the blind spots between ontology engineering and Linked Data?

This first research question is motivated by the gap between the two disciplines – ontology engineering and Linked Data – which is more or less commonly agreed based on intuitive observations and assumptions but never studied properly. This goes along with the terminological mismatch in literature for example if the schema of a Linked Data set is a vocabulary (conforming to the common wording in the Linked Data community) or an ontology (speaking with the words of ontology engineering researchers)[1]. It is an accepted research challenge within the Linked Data research community to understand the complete data set life cycle, but can ontology engineering research contribute adaptable results to this semantically rather lightweight approach? It is timely to study this on the way towards fully-fledged methodologies for Linked Data publication and management.

1.2.2 How can classical Web usage mining methods be applied in the context of structured Web data sets and how does that affect managing data set maintenance?

Linked Data is only one representative technology that allows publishing of structured data on the Web. A much more general understanding of the Web of Data would also involve structured data which is embedded into Web pages by application of Microformats, RDFa, or Microdata and in a further generalization step even the deep Web. But the Linked Data paradigm is the literally best example how to exploit the general architecture of the Web for resource identification, access, and interlinking. This makes it possible to apply methods from classical Web usage mining, which were originally developed to evaluate the usage and plan the maintenance of Web pages, to raw Web data now and study their applicability in this context. We think that this grants Linked Data a special position within the bunch of technologies to publish structured Web data and this motivates us to study how characteristic usage patterns of Linked Data can be utilized to advance emergent dataspaces.

1.2.3 To which extend can usage-dependent metrics help to assess the quality of a Web data set?

Assessing data quality is a task that combines a large number of data quality dimensions. Some can be measured objectively while others involve subjective notions of quality. It is accepted that the consumers perspective to data quality is of specific

[1] It is worth to mention at this point that we will introduce our understanding of these two terms in Chapter 2 in detail. By now it is sufficient to understand that, related to the schema level of data sets, we will use both terms synonymously.

significance. Web usage mining methods offer the possibility to analyze the most easy user feedback – the usage behaviour of the consumer of Web resources represented in log files. Applying this in the context of structured Web data brings Web usage mining methods together with data quality assessment and motivates us to study which data quality dimensions can be measured by Web usage analysis and to which extend this covers objective and subjective criteria.

1.3 Contributions

The contextualization of this thesis at the interface between ontology engineering and structured Web data causes that its contributions span across the borders of more than one discrete research area. Furthermore, it is characteristic for research that deals with methodological innovation that not only process guidelines are designed in theory but also necessary tools when an approach cannot be fulfilled by state-of-the-art technology which also results in multiple contributions.

1.3.1 Understanding the role of ontology engineering in the context of structured Web data publication and management (Chapter 5):

That ontology engineering methodologies are not very widely adopted in structured Web data publication and management is a common conception. However, studies that provide empirical evidence for this assumption and analyse the blind spots qualitatively are missing. This thesis contributes such an analytical survey among a representative set of Linked Open Data providers.

1.3.2 Life cycle management for structured Web data sets (Chapter 5):

The publication and consumption of structured Web data has been thoroughly studied from a perspective of standards and technology but guidelines for managing the data set life cycle gained rather limited regard. This work contributes a methodology that describes the processes, activities, methods, and tools of a usage-dependent data set life cycle.

1.3.3 Usage-dependent data quality assessment (Chapter 6):

It is widely accepted that assessing data quality is highly individual and sometimes rather subjective depending on the particular project it is contextualized with. Research on data quality in information systems has brought up a wide range of typical

data quality dimensions as well as methods and measures to calculate the data quality score. In this thesis we present a statistical framework that allows for quantifying selected dimensions of the quality of a structured Web data set with reference to its usage mined from server log files. This way of leveraging server logs as a sort of implicit user feedback about particular data quality dimensions is innovative.

1.3.4 Web of Data usage mining (Chapter 6):

The agenda for the emerging research area of Semantic Web usage mining has been published more than ten years ago. Since that date only few pieces of work have been published that address specific research questions raised by this trend-setting paper. Providing the necessary tool support for our usage-dependent life cycle inevitably touched open questions in this research area. We contribute a preprocessing algorithm for the in-depth analysis of SPARQL query logs and a high-level introduction into different types of transaction tables derived from these logs for the application of classical association rule mining algorithms on SPARQL queries.

1.3.5 Ontology engineering evaluation frameworks in the context of structured Web data set life cycles (Chapter 7):

The ontology engineering discipline provides rigorous frameworks for the qualitative evaluation of methodologies as well as the estimation of the costs of the ontology life cycle. In this thesis we will apply these frameworks in the context of structured Web data set life cycles which is a novelty affecting the state-of-the-art in both fields.

1.3.6 Analysis of real world data set usage (Chapter 7):

For the evaluation of our approach we exploit real-world usage data of a number of Linked Open Data data sets which has been made available to the public as a reference data set as of writing this thesis. On the one hand this selection of research data allows for a high reproducibility as well as representativeness of our findings. On the other hand our analysis contributes a very detailed insight into the actual usage of Linked Open Data from a quantitative (high-level statistics about the amount of queries performed) as well as qualitative (in-depth query pattern analysis) perspective back to the research community which actively discusses success factors of this technology which go "beyond triple counts"[2].

[2]Please refer to the blog post at `http://blog.ldodds.com/2011/09/28/beyond-the-triple-count/` where Leigh Dodds coins this terminology.

1.4 Research methodology

Our work is generally inspired by the reflection about the field of Computer Science described in [DC02]. The author surveys diverse viewpoints about Computer Science and consolidates the features that qualify it as a scientific discipline. In the context of this thesis we think the following two aspects emphasized in the article are noteworthy:

First, **the relation of technology and information in Computer Science**, which is also spotted in [MS95], led the author to conclude that the focus of investigation in Computer Science is not the theory-related physical object of a computer but information in the context of the computer as "a tool always capable of changing in order to accommodate even more powerful theoretical concepts"[DC02]. We deduce for our research methodology the importance of well-documented and repeatable experiments mentioned in this article – and further stressed by the work of Lukowicz et al. [TLPH95] – in order to provide reproducible findings about the shape of the investigated artifacts.

Second, the perspective of **information as a nucleus of Computer Science makes the discipline scientifically and sociologically relevant** gains an increasing momentum under the light of current research conducted in the areas of Web Science [BLHH+06] or Human Centered Computing for example which investigate originally computer-based artifacts in an interdisciplinary fashion. While the author of [DC02] refers former views which tried to distinguish characteristic research paradigms (e.g. empirical, mathematical, engineering-oriented) as kinds of sequential research epochs [Har79] we addict the view that many of today's challenges of Computer Science call for a multi-perspective view on research problems and adopt this principle in this thesis by combining empirical, engineering-oriented, and experimental means.

From a high level point of view we follow the seven guidelines for design science in information systems research introduced by Hevner et al. [HMPR04]. Practically we started our work by a comprehensive analysis of the state of the art in Ontology Engineering in order to identify open problems to be solved by conducting research. To our best knowledge we achieved an integrative overview of the different research directions and methodologies. We identified the adaptation of well-researched results of the Ontology Engineering discipline in the context of structured Web data publication and management as one open issue. To enrich this assumption which based on literature review with empirical evidence, we conducted an empirical survey amongst data publishers that contribute data sets to the well-regarded Linking Open Data Cloud. The survey proved the identified gap and helped to define the

related context of our proposed solution explicitly. That includes a characterization of the type of ontologies we are addressing, the deduction of requirements for data set maintenance, and the definition of a clear use case scenario. The single parts of our solution were designed and developed stepwise in the following, before we finally evaluated the theoretical saturation of our technologies in a multi-perspective way. In order to illustrate how our research process conforms to the guidelines for design science we adapt the design cycle of Takeda et al. [TVTY90] and model the seven guidelines orthogonal to the core process (cf. Figure 1.1).

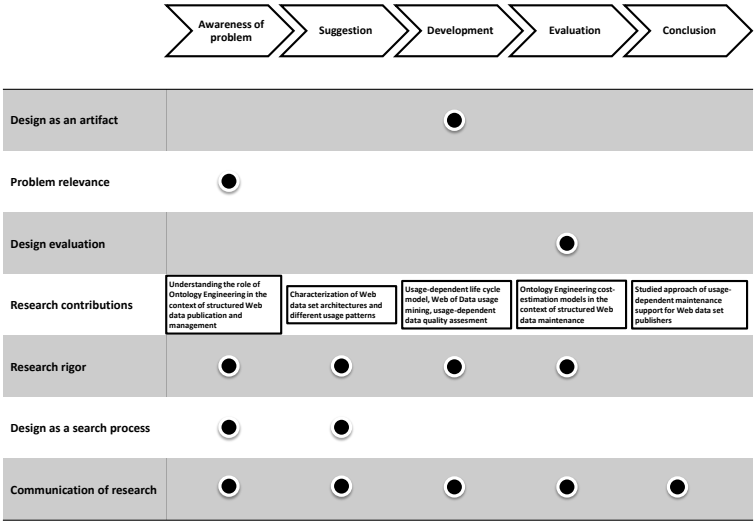

Figure 1.1: Visual representation of our research methodology adapted from [TVTY90] and [HMPR04]

1.5 Reader's guide

The main input of this thesis is divided into three parts preceded by an introductory chapter which should be referred to when a high level description of this thesis' motivation, research questions, and research methodology is needed or this reader's guide needs to be consulted.

Part I is entitled *Foundations and Theories* and contains on the one hand the fun-

damental terminology and definitions which are necessary to understand the complete work and on the other hand an empirical study which is intended to prove inductively that the thesis addresses a relevant problem.

Chapter2: The theoretical and terminological foundations for this thesis cover structured Web data, Web usage and usage mining, as well as a notion of quality, quality assessment and maintenance in information systems. Since all these fields are more than just wide for themselves, we structured this chapter in a fashion that the angle from which we look at them in the respective sections is shaped by the preceding ones. Overall this is routed by the first section introducing our notion of structured Web data being aligned with the Semantic Web standards and Linked Data. Given that, it is clear that that other perspectives on the various fundamental fields exist which we aware of but regard being outside of the scope of this thesis.

Chapter3: Subsequent to the foundations we contract our work with the most significant pieces of related research which are situated in the same problem space.

Part II envelops our *Approach* to provide usage-dependent maintenance of structured Web data sets. The four chapters sequentially introduce a life cycle methodology to manage the processes and activities aligned with data set maintenance, a method to collect data set usage feedback from log files, a framework that reflects the usage-dependent dimensions of data set quality, and an instantiation of the data quality framework by a number of metrics that leverage the data set log file analysis.

Chapter5: We described before that we follow the methodological guidelines of design science. This requires a clear understanding of the addressed problem and the approval of its relevancy which we conform to by presenting in this chapter the design and results of the *LOD Provider Survey*, an empirical study on ontology engineering practices in Linked Data projects.

Chapter5: In this chapter we introduce the methodology COLM for managing the maintenance of structured Web data sets. This is a guide for data set publishers to understand and plan the processes and activities which are part of the Web data set life cycle.

Chapter6: The usage-dependent evaluation of the data set quality is a core component of our proposed methodology. In this chapter we present an instantiation of this by performing Web usage mining on SPARQL query logs. For the preprocessing of classical server log files which contain SPARQL queries we introduce a preprocessing and enrichment algorithm

which allows for an in-depth analysis of successful and failing queries as well as their atomic parts. A statistical framework will be introduced featuring two different data quality score functions – a plain and a hierarchical one.

Part III, entitled *Evaluation and Conclusions*, rounds the thesis by a presentation of the evaluating studies for the different parts of our approach and a case study which brings all pieces together before we finally argue about the overall findings of this research.

Chapter7: The evaluation chapter consists of three parts. The first part is dedicated to compare our proposed methodology against related approaches qualitatively and to provide insight into the costs of data set maintenance conforming to the ONTOCOM model as an additional quantitative method. Afterwards we discuss our data quality dimension and category score functions as well as different weightings for these with reference to the empirically evidenced importance of particular data quality dimensions for the data consumer. A case study and further data set analysis based on real-world log data brings the individual pieces together and proves the applicability of the entire approach in practice.

Chapter8 In the final chapter of this thesis we summarize the contributions of our work and then consolidate the results of the individual parts of the evaluation into one picture of the overall findings from a practical but also epistemological perspective. We close with a critical review of the limitations of our research results and the issues which were left open for future work.

1.6 Related publications

We conducted our research step-by-step and continuously published parts of the work at international scientific workshops and conferences since the very beginning. This helped us to evaluate the quality of our research and assure early impact. Thus a number of publications of ourselves refer to this thesis. The most relevant ones will be introduced and contextualized within this thesis briefly in the following.

M. Luczak-Rösch, "How are people engineering linked data? – A survey snapshot about the engineering efforts spent by dataset publishers", Freie Universität Berlin, TR-B-11-09, 2011. The technical report

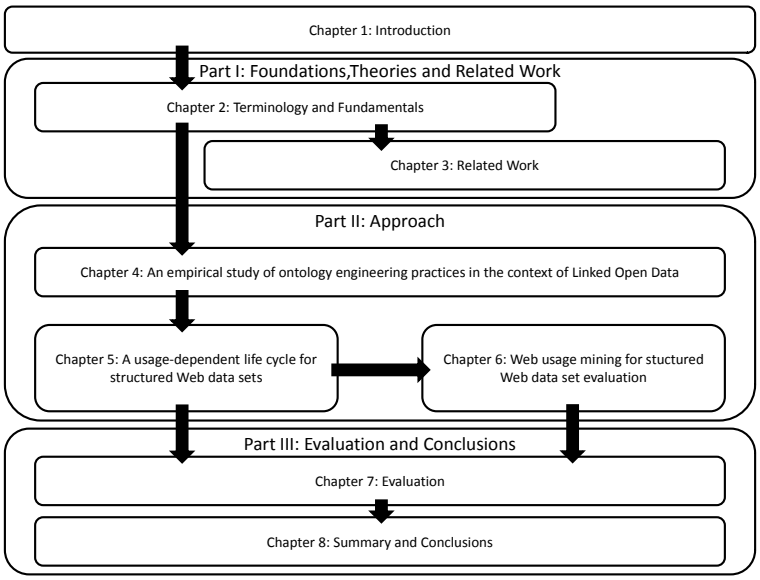

Figure 1.2: Structure of this thesis

presents a preliminary compilation and interpretation of the "LOD Provider Survey 2010", the survey which was dedicated to empirically assure the relevance of the central problem addressed by this thesis.

M. Luczak-Rösch and R. Heese, "Managing Ontology Lifecycles in Corporate Settings" Springer Berlin / Heidelberg, 2009, vol. 221, pp. 235-248. The paper presents the initial version of our life cycle in the context of corporate ontology engineering settings but not specified to the structured data use case as it is in scope of this thesis.

M. Luczak-Rösch and H. Mühleisen, "Log File Analysis for Web of Data Endpoints" in Proc. of the 8th Extended Semantic Web Conference (ESWC), Poster-Session, 2011. The short paper was accompanied by a poster introducing our preprocessing algorithm for an in-depth analysis of SPARQL query entries in log files of Web of Data endpoints.

M. Luczak-Rösch and M. Bischoff, "Statistical Analysis of Web of Data Usage" in Joint Workshop on Knowledge Evolution and Ontology Dynamics (EvoDyn2011), 2011. This paper presents a method to detect errors or weaknesses within ontologies used for Linked Data population based on statistics and network visualizations. We contribute a set of statistical measures

that help to visualize different usage aspects, and an exemplary analysis of one of the most prominent Linked Data set – DBpedia – aimed to show the feasibility and the potential of the approach.

B. Berendt, L. Hollink, V. Hollink, M. Luczak-Rösch, K. Möller, and D. Vallet, "Usage analysis and the web of data" SIGIR Forum, vol. 45, iss. 1, 2011. The article reports on one of our key initiatives during the work on this research, namely the collaborative pitch of a scientific workshop series on usage analysis in the context of the Web of Data together with colleagues from Belgium, Ireland, the Netherlands, and Spain. A core contribution of the workshop called "USEWOD – Usage Analysis and the Web of Data" is a data set of log files of Web of Data endpoints, which significantly consists of log files collected by us for the studies and evaluation in this thesis, and which are an important reference data set in this research area today (please refer to the papers which describe various studies based on this data, e.g. [KKL11, HJA12, lGBFMP11, EMC+11, EVS11, DFDM12]).

startatroot

Part I

Foundations

CHAPTER 2

Terminology and fundamentals

This chapter defines the terminology used and the fundamentals referred to in this thesis. In order to equip the reader properly and to keep a sharp focus this chapter follows a layered structure. Each of the successive sections respects the focus set by all the sections before. Starting from the definition of *structured Web data sets* we constitute *usage* at first in this context followed by *quality* as a key driver that influences *maintenance* which is introduced at last. It is a matter fact that most of the terms defined in this chapter also have a different or wider meaning outside the scope of this thesis. To allow a concentrated discourse we disrespect this meaning if no interference with our work exists. Terms which appear in bold typeset are the terms which are defined in the following. Words written italic are the primary occurrence of terms which will be defined afterwards.

2.1 Data, information, and knowledge

It is common in Computer Science but also beyond this discipline to differentiate between the terms *signals*, *symbols*, *data*, *information*, *knowledge*, and sometimes even *wisdom*. **Signals** are the physical occurrences conveyed in and between today's digital computer systems. According to Bellinger et al., who elaborate on Ackoff's characterization "From data to wisdom" [Ack89], **wisdom** is "an extrapolative and non-deterministic, non-probabilistic process"[BCM04]. Both, signals and wisdom, are situated on completely different layers of abstraction than we put emphasize on in this work – the former below on the physical layer, the latter above questioning fundamentals of today's understanding of machine computation. Thus, we treat them out of scope which is also in-line with the contextualization of this work as described in Section 1.4. We refer to **symbols** as the smallest unit above the physical layer of digital computer systems which is meant for human perception. A symbol is encoded by a number of signal states.

The terms data, information, and knowledge are part of a long-lasting discourse about their definition and differentiation as key terms in the context of information systems research as it is well summarized in [Zin07]. It is not the scope of this thesis to contribute fundamentally to this discourse but this work is concerned with structured data on the Web which can be regarded as the literally largest information system in existence. Moreover, the Web architecture builds the base for magnitudes of other scalable information systems. Hence, it is necessary to refer to a definition of data, information, and knowledge and contextualize these three fundamental concepts with this work.

As it is credited to Dragunalescu in [Zin07] "**data** are a set of symbols representing a perception of raw facts." On top of that I refer to what is credited to Burrell in [Zin07]: "**Information** is that which is conveyed [...] through data and the context in which the data are assembled. **Knowledge** is the general understanding and awareness garnered from accumulated information, tempered by experience [...]".

2.2 The Web

The **Web** (very often also referred to as the World Wide Web or WWW) is "an information space" which has been invented in the end of the 1980s by Sir Tim Berners-Lee. It was originally designed as an information management system for the European Organization for Nuclear Research (CERN)[1] which allows it to publish

[1] `http://www.cern.ch/`

and consume interlinked information on computers which are connected to the *Internet* in a platform independent fashion by utilization of *hypertext* or, more generic, *hypermedia* [BL89]. The core components of the first standardization effort concerning the Web are *Uniform Resource Locators (URL)*, the *Hypertext Transfer Protocol (HTTP)*, and the *Hypertext Markup Language (HTML)*.

Such efforts were and still are advanced by the **World Wide Web Consortium (W3C)**[2], a community organization founded for promoting, observing, and guiding the evolution of the Web as well as associated developments and trends. Standards of the W3C are called recommendations.

The Web is a service on – thus different from – the **Internet** which is a network of networks conforming to standards controlled by the Internet Engineering Task Force (IETF)[3], which are released as so called Requests for Comments (RFC). Most noteworthy with reference to the Internet are the transport protocol TCP [oSC81b] and the network protocol IP [oSC81a] (in the context of the Internet commonly referred to in combination as TCP/IP).

A **Uniform Resource Locator (URL)** is a unique identifier which refers "to objects accessed with existing protocols" on the Internet[BL]. Consequently an HTTP URL refers to an object – such as any hypermedia content – which can be accessed by requests conforming to the HTTP protocol via the Internet[4]. URLs are a specific form of the generic concept of Unique Resource Identifiers (URI), meant to "define the encoding of the access algorithm" into an address [BL] locating an object on which operations such as access or update can be performed. A **Uniform Resource Identifier (URI)** or with an extended character set a **Internationalized Resource Identifier (IRI)** [5] is a generic identifier which regards objects that are not necessarily real documents, like a hypertext or an image but may be completely abstract. The original definition says that a URI is "a way to encapsulate a name in any registered name space, and label it with the the name space, producing a member of the universal set" [BL]. This "member" is the above mentioned abstract notion of an addressed object generically called a *resource* in the context of the Web.

HTTP enables to transfer information between Web servers and clients in a request-response fashion. It is a text-based client-server protocol on the application layer that relies on TCP/IP connections (the HTTP 1.1 standard is specified in [FGM+99]). It is today common to send requests to URIs which are dissolved into

[2] `http://www.w3.org`

[3] `http://www.ietf.org/`

[4] In the remainder of this thesis I will use the generic terms URL and URI respectively when referring to HTTP URLs and HTTP URIs. Any other notion of such identifiers that use different protocols are not relevant for this work.

[5] Please note that we will simply say URI in the remainder of this thesis even though we are aware of IRIs as a complement to URIs that extends the character repertoire to Unicode/ISO 10646.

IP addresses for the requested host name by the DNS protocol [Moc87]. An HTTP message is self-descriptive in the sense that all information for the interpretation of the semantics of the message are encoded within it (e.g. resource name, status information, content types, content, etc.). This characterizes HTTP as a stateless protocol and is a foundation for caching of non-dynamic content. An HTTP request message contains the request method (one of GET, POST, PUT, DELETE, TRACE, CONNECT, OPTIONS), the requested resource name, the protocol version (HTTP 1.0 or 1.1[6]), the address of the requested host (mandatory since HTTP 1.1), a number of optional request header fields that specify characteristics and expectations of the client (e.g. the client's user agent or the expected type of the response content), and after a blank line an optional body of content to be transferred to the server. An HTTP response consists of the protocol version, the response code indicating the status of the request (e.g. code 200 for a successful request or 404 if the requested resource name could not be found on the server), a number of optional response header fields that specify the characteristics of the server (e.g. server software) and the responded content (e.g. type, length or language) or requirements to the client (e.g. to authenticate in order to access the requested resource or to request another resource in order to retrieve the requested information), and again after a blank line optional content. In the context of this thesis the important foundations of HTTP requests are the GET method and the request headers specifying the user agent and the requested content type. On response side I will put special attention on response codes, the content type header field and content serializations. In listings 2.1 and 2.2 I provide an example of a real HTTP request and response for the resource identified by the URI `http://dbpedia.org/page/Berlin`.

Listing 2.1: Example of a simple HTTP GET request for an HTML representation of the resource /page/Berlin on the Web server hosted at the domain name dbpedia.org.

```
GET /page/Berlin HTTP/1.1
Host      dbpedia.org
User-Agent        Mozilla/5.0 (Windows NT 6.1; WOW64; rv:19.0)
                  Gecko/20100101 Firefox/19.0
Accept    text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding gzip, deflate
Cookie    ...
```

Listing 2.2: Excerpt of the HTTP response of the Web server hosted at dbpedia.org for the above mentioned request example.

```
HTTP/1.1 200 OK
Date      Thu, 11 Apr 2013 09:08:02 GMT
```

[6] I will speak of HTTP genrically in the remainder of this thesis and the reader can assume that I refer to the HTTP 1.1 standard by that.

```
Content-Type     text/html; charset=UTF-8
Transfer-Encoding        chunked
Connection       keep-alive
Vary     Accept-Encoding
Server   Virtuoso/06.04.3134 (Linux) x86_64-generic-linux-glibc212-64  VDB
Expires Thu, 18 Apr 2013 09:08:01 GMT
Link ...
Content-Encoding         gzip

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
        "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:dbpprop="http://dbpedia.org/property/"
     xmlns:foaf="http://xmlns.com/foaf/0.1/"
     version="XHTML+RDFa 1.0" xml:lang="en">


<!-- header -->
<head profile="http://www.w3.org/1999/xhtml/vocab">
...
</head>
<body onload="init();" about="http://dbpedia.org/resource/Berlin">
    <div id="header">
...
 </body>
</html>
```

HTML originally was a specific document markup language for hypertext publishing on the Web. **Hypertext** is a concept of networked textual content where each text document contains its individual information and additional references – hyperlinks – to other related documents the consumer can browse to and has been invented by Ted Nelson in the 1960s [Nel65]. It is a matter of fact that Nelson's original vision of hypertext involves more than just interlinked documents – Nelson envisions an integral paradigm shift of content authoring and consumption – but this document-centric view is not of interest in the scope of this research, hence, I do not go into more details of hypertext here. The latest stable W3C recommendations of HTML are HTML 4 and XHTML 1 which is an *XML* serialization of HTML 4. HTML 5 evolves the original design of a markup language for human-readable document representation towards a language that aims at interoperation with Web content. Due to the focus of this thesis and the current status of the HTML 5 standardization effort I will refer to the document-centric markup features only when speaking of HTML. URLs, HTTP and HTML constitute a "hypertext Web", sometimes also referred to as the "Web of Documents". I will adopt the former and latter term to refer to this classical notion of a Web that provides human-readable information encoded in hypertext pages or multimedia content without focusing the provision of rich data and metadata.

2.2.1 The generic Web architecture

A generic **Web architecture** did not exist entirely when the Web was invented but one has evolved over time [Gro04]. It is mostly based on the work of Fielding who generalized the principles which were already inherent in the initial design of the Web by Berners-Lee but which were never explicitly compiled before as an integral architecture [Fie00, FT02]. Fielding coined the term **Representational State Transfer (REST)** which refers to an architectural style of "network-based application architectures" [Fie00]. The Web was the prominent example of such a system Fielding analyzed in order to design a set of six generic principles which allow to create highly flexible and scalable systems:

- The Web is a **client server** system.
- It is **stateless** in the sense that the transferred messages contain all information about the delivered content and the next state transition.
- This enables **caching** of resources which reduces the network load.
- HTTP and URIs constitute a **uniform interface** to retrieve and manipulate resource representations.
- The request-response interaction only involves two peers as client and server on the same protocol level which makes the Web a **layered system**.
- An optional feature is that it is possible to load **code on-demand**, e.g. in form of Java Applets.

From this list of six generic principles (derived from the characteristics of the Web) the most important one in the context of this thesis is the uniform interface consisting of HTTP and URIs. Specifically an HTTP GET request for a Web resource identified by a URI yields that the server delivers a **representation** of that resource in the requested content type[7]. The user of a hypertext Web browser may perceive that she always gets exactly the same single content available at a given URI which may be true for numerous URL-identified resources. Fielding explains this as follows: "Some resources are static in the sense that, when examined at any time after their creation, they always correspond to the same value set." [Fie00] But generally a **resource** is defined as follows:

[7]Please note that performing other request methods than GET (e.g. PUT or DELETE) may also mean that a representation of a resource is manipulated (meaning created, edited or deleted) which is out of scope of this work.

> "... a resource R is a temporally varying membership function $M_R(t)$, which for time t maps to a set of entities, or values, which are equivalent. The values in the set may be resource representations and/or resource identifiers." [Fie00]

This introduces the concept of the **information resource**, a resource which refers to one or more information objects like documents – named representations as mentioned above – which represent the equivalent information for the resource in a different serialization or format (the server will respond with code 200 "success" in this case) or to one or more other resource identifiers (the respective server response code is 303 "see other"). It is possible that a resource is not mapped to any representation, meaning that there is no information published describing it. References to the identifier of this **non-information resource** are albeit possible.

2.3 The Web of Data

Serving any kind of raw data on the Web in a device- and platform-independent fashion was a central idea of the Web already when it was invented. The **Extensible Markup Language (XML)** [W3C06] is a generic text-based markup language, a dedicated subset of the **Standard Generalized Markup Language (SGML)** [W3C04a], for this purpose. In contrast to HTML, which we introduced before as a specific markup language for human-readable hypertext, it allows the composition of user-defined storage structures that represent simple or complex **data objects**. Logically a data object – also called XML document – is a nested structure of one or more named elements. Elements may have a set of attributes of which each is named either and has a value. Correctly marked up XML documents are called well-formed. If an XML document refers to a schema defining a specific collection and arrangement of elements as well as attributes, it is possible to validate the structure and naming of elements. A schema, sometimes also referred to as a "markup vocabulary" **W3CXMLNS**, is either an XML document, which is valid with reference to the generic XML Schema specification, or a Document Type Definition (DTD).

XML namespaces[8] [W3C09] are a means to disambiguate identically spelled element names within one document that source from different markup vocabularies (e.g. a book's title and an author's title). A namespace is defined by a namespace prefix and an associated namespace name which is a URI. Element names that prepend a namespace prefix followed by colon belong to the namespace identified by

[8] We will refer to the concept of XML namespaces by simply calling it namespaces in the remainder of this thesis.

Namespace prefix	URI
xsd	http://www.w3.org/2001/XMLSchema
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns
rdfs	http://www.w3.org/2000/01/rdf-schema
owl	http://www.w3.org/2002/07/owl
skos	http://www.w3.org/2004/02/skos/core
dc	http://purl.org/dc/elements/1.1/
foaf	http://xmlns.com/foaf/0.1/
dbpedia	http://dbpedia.org/resource
swrc	http://swrc.ontoware.org/ontology

Table 2.1: Namespace declarations used in this thesis.

the URI and are called qualified names. This concept also allows the abbreviation of otherwise long URI references and is often applied in serialization formats that contain URIs. Table 2.3 lists examples of common namespace declarations and is at the same time the foundation for the abbreviation of URIs used in examples in the remainder of this thesis.

We will not go into further detail of the logical and physical XML markup specification here but refer to [W3C06] to be consulted for this purpose. From an abstract point of view we subsume the important characteristics of XML as follows: XML allows the representation of storage structures of any real or virtual *entities*[9] as trees. Names of entities are not necessarily URIs, thus, even when XML is shared on the Web, the only thing which is necessarily a Web resource is the entire XML document. Entities are either atomic or have further entities as parts or properties. If entities refer to other entities, the relations are unnamed and untyped. An XML processor can access the structure and the contents of an XML document in a unified way, but the elements themselves are only named and have no well-defined meaning. Consequently, we understand XML as a form of **semi-structured Web data** which is in-line with the argumentation in [Bun97].

While it is possible to share human-readable information as well as raw semi-structured data on the Web by application of HTML and XML, both forms of Web content still lack any fine-grained formal semantics that would allow machines to integrate and interpret them automatically and in a unified way on the Web. The Semantic Web has been introduced as an extension of the classical Web for this

[9]We are aware of the ambiguity of the term "entity". There are two types of entities in the context of the XML specification already – parsed and unparsed entities. As we will introduce later in the subsection about ontologies, we use this term when we refer to the real or virtual things that are or may be modeled as part of a data schema or knowledge representation.

purpose. In 2001 Berners-Lee, Hendler, and Lassila defined the **Semantic Web** as follows:

> "The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [BLHL01]

One interpretation of this initial definition of the Semantic Web is the vision of autonomous agents and services on the Web which make decisions based on heavyweight conceptual knowledge and *reasoning*. Another one, which is today much more common, is to formulate the goal of querying the Web as it was one giant database [HB11]. It is a matter of fact that it was always claimed that "the Semantic Web is a **Web of Data**" [BLF00, Con], and that the core components for the realization of it have evolved but never fundamentally changed. The latter can be reproduced very nicely by consulting the evolution of the well-known Semantic Web technology stack as depicted in Figure 2.1 [Koi01, Bra07]. Nevertheless, one can observe that it is more likely to use the term Semantic Web when a knowledge- and reasoning-centric perspective is represented while the term Web of Data is commonly used in a data-centric context as it is the scope of this work.

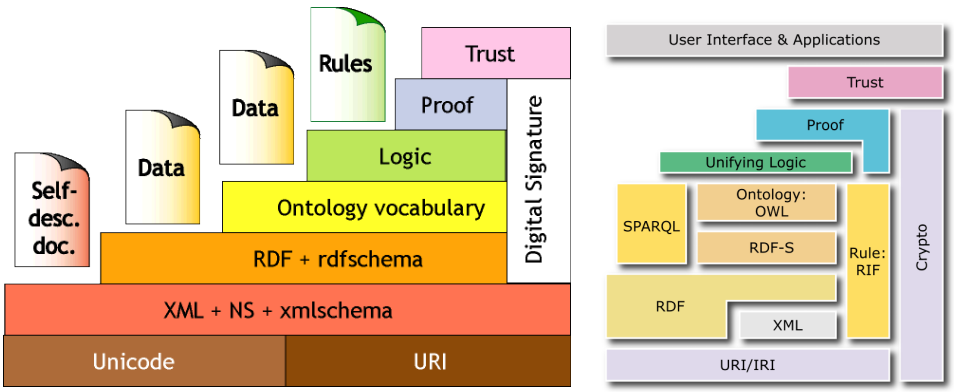

Figure 2.1: The initial technology stack of the Semantic Web (left, taken from [Koi01]) compared to the revision performed in 2007 (right, taken from [Bra07]).

2.3.1 Structured Web data

Even though we will apply and evaluate our approach in the context of a specific form of structured Web data which is widely adopted, it is necessary to provide a

generic definition of **structured Web data**. Based on the common terminology of the generic Web architecture introduced before, we derive the following definition:

1. Definition (Structured Web data). *Structured Web data are structured descriptions of Web resources. A structured description of a resource is its inter-relation with another resource or with a plain literal value by application of a resource as the link. Raw facts about any real or virtual entity or property emerge from the meaning assigned to resources by describing them with a set of resources representing a vocabulary of the terminology of a domain of interest.*

In the beginning of this section we introduced that we adopt the definition of data as "a set of symbols representing a perception of raw facts" [Zin07]. It is noteworthy that our definition of structured Web data conforms to this generic notion of data, since we regard URIs as the symbols to name and identify resources. And a set of inter-related resources – thus a set of symbols – represent raw facts.

XML is based on a tree structure and does not allow to create typed or named links, neither between complete data objects nor the atomic entities which are part of one or different data objects. Our definition of structured Web data results in a graph structure of entities which are proper Web resources identified by URIs. Relations between entities are in fact entities themselves. This makes not only resources named, referable, and retrievable but also the links between them. A feature of this is the schema-less extendability of entity structures by adding such typed links between entities of different graphs which has been highlighted as a feature of semi-structured data by Buneman in [Bun97] already. However, in our definition we explicitly mention the role of a vocabulary of a domain of interest – short, a schema – which makes the semi-structured data finally structured. This role of the schema is also acknowledged by Buneman [Bun97] who describes it as follows:

> "Schemas are useful for browsing and for providing partial answers to queries. They will also be needed for the passage back from semistructured to structured data for which a richer notion of schema is necessary."

The current standard, that allows to create data conforming to this definition, is the **Resource Description Framework (RDF)**. Basically, RDF is a language for a unified representation of metadata about Web resources. By making so-called statements about Web resources a graph is created that does not need to be connected. An RDF statement is a 3-tupel (triple) that consists of a subject, a predicate, and an object in exactly this order. The subject and predicate of a triple are commonly URI-identified resources[10]. The object may be a resource or a string called **RDF**

[10] We are aware of the general possibility that subjects are literals but this is out of scope of this thesis. We will refer to the discourse about this later again.

literal. Several features of RDF are recommended to be avoided in the context of structured Web data [HB11]. Hence, we pass on the introduction of blank nodes, reification and the language primitives for collections and containers.

Various formats allow the **serialization** of RDF such as RDF/XML (RDF as a well-formed XML document) or – more intuitively to read in case of rather small exemplary RDF snippets – N-Triples (a line-based plain text format). A serialization of an RDF graph is called an **RDF document**. Listing 2.3 shows an excerpt of five RDF statements from the data retrieved by requesting the URI `http://dbpedia.org/data/Berlin.ntriples` as N-Triples.

Listing 2.3: Excerpt of an RDF representation of the resource http://dbpedia.org/data/Berlin in N-Triples serialization.

```
<http://dbpedia.org/resource/Berlin>
        <http://www.w3.org/2000/01/rdf-schema#label>
        "Berlin"@en .
<http://dbpedia.org/resource/Berlin>
        <http://dbpedia.org/ontology/wikiPageExternalLink>
        <http://www.berlin.de/international/index.en.php/> .
<http://dbpedia.org/resource/Berlin>
        <http://dbpedia.org/property/elevation>
        "34"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://dbpedia.org/resource/Berlin>
        <http://dbpedia.org/ontology/thumbnail>
        <http://upload.wikimedia.org/wikipedia/.../Berlin_Montage_4.jpg> .
<http://dbpedia.org/resource/Berlin>
        <http://dbpedia.org/property/rulingParty>
        <http://dbpedia.org/resource/Social_Democratic_Party_of_Germany> .
...
```

RDF graphs can be stored as plain files or in an **RDF store** (also called triple store). Various different approaches exist how RDF stores reflect and exploit the triple structure in order to provide scalable retrieval and querying of even large amounts of data and in recent years one can observe a continuous improvement of triple store technologies. [BS09] provides the state of the art benchmarking approach for triple stores and is a starting point to investigate the performance of different storage architectures.

The **SPARQL Query Language (SPARQL)** is a standard language for querying and manipulating RDF data based on the principle of graph pattern matching. We call an instantiation of a SPARQL implementation a **SPARQL endpoint**. In the context of this thesis only the usage of data is in focus so we pass the introduction of the manipulation features of SPARQL which are available since version 1.1 released very recently in March 2013 [W3C13a]. It is furthermore a matter of fact that only few SPARQL endpoints already support the 1.1 recommendation extensively, while SPARQL 1.0 [W3C08b, W3C08a] compatibility can be regarded standard as of writing this thesis and has bee chosen as the reference for our approach.

Our approach exploits only a subset of the entire SPARQL query language as introduced in [W3C08b]. We do not regard (a) blank nodes, (b) literals as subjects, (c) solution modifiers different from the projection, and (d) multiple graphs. We do so because blank nodes are not recommended to be used in the context of structured Web data as they do not act as persistent identifiers, the use of literals as subjects is possible in the RDF data model but not allowed in any of the current serializations[11], and the remaining five solution modifiers as well as multiple graphs do not interfere with our approach that is mainly based on deconstruction of query patterns. In the following we will define the important components of the SPARQL query language as they will be referred to in the remainder of this thesis.

$$\begin{aligned} &D \text{ is a data set.} \\ &G \text{ is an RDF graph.} \\ &URIS \text{ is the set of all URIs.} \\ &LITERALS \text{ is the set of all RDF literals.} \\ &VARIABLES \text{ is the infinite set which is disjoint from URIS} \cap LITERALS. \end{aligned} \tag{2.1}$$

The basic form of a pattern is a **triple pattern (TP)** which contains a subject, a predicate, and an object in a similar fashion as they appear in RDF statements. The difference is that any of these three components also may be a variable.

$$\begin{aligned} TP = \{[u_1, u_2, u_3] \\ |u_i \in URIS \\ \vee\, u_i \in LITERALS \\ \vee\, u_i \in VARIABLES\} \end{aligned} \tag{2.2}$$

$$\begin{aligned} S(TP) &= u_1 \\ P(TP) &= u_2 \\ O(TP) &= u_3 \end{aligned} \tag{2.3}$$

A set of triples patterns is called a **basic graph pattern (BGP)**.

$$BGP = [TP_1, TP_2, \ldots, TP_i] \tag{2.4}$$

[11] Please refer to `http://www.w3.org/2000/03/rdf-tracking/#rdfms-literalsubjects` for a more detailed discourse on this issue.

A BGP "matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph"[W3C08b]. This is called a **solution mapping**.

$$\text{A solution mapping is a partial function } \mu : VARIABLES \rightarrow URIS \cap LITERALS. \tag{2.5}$$

All forms of basic graph patterns can be grouped (then also called **group graph patterns (GGP)**) and combined by a number of patterns such as UNION or OPTIONAL. It is possible to formulate result restricting filters that effect on the GGP they are defined in. Any form of grouping or combining basic graph patterns is also a BGP.

SPARQL allows the SELECT, CONSTRUCT, ASK and DESCRIBE query forms. CONSTRUCT queries return an RDF graph that is generated from the query result conforming to a specified graph template. The DESCRIBE query form returns a description of a single resource also as an RDF graph but the nature of this graph is unspecified and can vary from endpoint to endpoint due to the underlying implementation of SPARQL. Since CONSTRUCT queries are only sparsely applied as of writing this thesis and since DESCRIBE has an informative nature, our work focuses the SELECT and ASK query forms which are commonly in use for performing complex queries. A SPARQL **query** generally consists of a SPARQL evaluation expression of a BGP (which recursively can contain further BGP) and a query form.

$$\begin{aligned} Q = \{&(E, F) \\ &| E = eval(p) \text{ is a SPARQL evaluation semantics expression} \\ &\text{without declaration of any data set or active graph to match against,} \\ &p \text{ is a } BGP, \\ &F \text{ is a SPARQL query form}\} \end{aligned} \tag{2.6}$$

To express that a specific BGP p appears in the evaluation semantics expression E of a SPARQL query Q we simply write $p \in Q$.

For a complete and formal introduction into the SPARQL algebraic operators and the evaluation semantics for graph patterns we refer to the definition given in [W3C08b]. The operators which are relevant in the context of this thesis are Filter, Join, Diff, Left Join, Union, and Project.

When performed against a query endpoint SELECT and ASK queries return a **query result** which is a serialization of either **bindings** of variables in case of

SELECT queries or a Boolean value in case of ASK queries. Formally this means there is a multiset of solution mappings for a given query, performed against a data set.

$$R(Q, D) = [\mu_1, \mu_2, \ldots, \mu_i] \tag{2.7}$$

Query results can be serialized in the Query Results JSON Format [W3C13d], the Query Results CSV and TSV Format [W3C13b], or the Query Results XML Format [W3C13c].

Listing 2.4: Example of a SPARQL query.

```
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX dbprop: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?city ?name
        WHERE {
                ?city
                  dbprop:rulingParty dbp:Social_Democratic_Party_of_Germany .
                ?city
                  rdfs:label ?name .
        }
```

Listing 2.5: Excerpt of the query result of the above mentioned example query performed agains the endpoint at http://dbpedia.org/sparql.

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation=
          "http://www.w3.org/2001/sw/DataAccess/rf1/result2.xsd">
 <head>
  <variable name="city"/>
  <variable name="name"/>
 </head>
 <results distinct="false" ordered="true">
  <result>
   <binding name="city">
   <uri>http://dbpedia.org/resource/Borchen</uri>
  </binding>
   <binding name="name">
   <literal xml:lang="de">Borchen</literal>
  </binding>
  </result>
  <result>
   <binding name="city">
   <uri>http://dbpedia.org/resource/Borchen</uri>
  </binding>
   <binding name="name">
   <literal xml:lang="en">Borchen</literal>
  </binding>
  </result>
```

```
  ...
  <result>
   <binding name="city">
        <uri>http://dbpedia.org/resource/Berlin</uri>
   </binding>
   <binding name="name">
        <literal xml:lang="en">Berlin</literal>
   </binding>
  </result>
  <result>
   <binding name="city">
        <uri>http://dbpedia.org/resource/Berlin</uri>
   </binding>
   <binding name="name">
        <literal xml:lang="de">Berlin</literal>
   </binding>
  </result>
  ...
   </results>
</sparql>
```

It is as an important feature in the scope of this thesis that query languages for structured Web data, such as SPARQL, are self-descriptive in the same fashion as the queried data. Especially non-trivial queries carry information about the schema a user applied for creating a query which can be retrieved as structured Web data. We will describe how our approach exploits this feature later in this thesis.

The **Linked Data** principles are a consensual paradigm for publishing RDF data on the Web initiated by Berners-Lee in 2006. It is based on the following four principles[BL06]:

"Use URIs as names for things."

"Use HTTP URIs so that people can look up those names."

"When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)."

"Include links to other URIs, so that they can discover more things."

Linked Data does not extend the RDF standard theoretically but it is the guideline that promoted the application of the Web architecture as an architecture capable to represent abstract information. Hence, it is an implementation of structured Web data in the sense of our generic definition and will be the representative example we refer our approach and studies to in the remainder of this thesis.

2.3.2 Web ontologies

The foundation of Web data, which is meaningful to machines, is the availability of machine-readable definitions of entities, their properties and inter-relation which can be retrieved conforming to the Web architecture as well. In our definition of

structured Web data we called these definitions "vocabulary of the terminology of a domain of interest".

Ontologies have been introduced into the ICT landscape decades ago as a means for representing machine-readable knowledge in digital computer systems. The term ontology originates from philosophy where Ontology[12] is concerned with the conceptualization and classification of entities which "essence" and "existence" is subject to be cleared [GPFLC04].

Definition of ontologies in Computer Science

We regard two definitions of ontologies in the context of information systems as the most appropriate ones with reference to our work.

First, the definition of Gruber [Gru95] which is rather sharp and short:

> "An ontology is an explicit specification of a conceptualization."

Second, the more detailed definition by Ushold and Jasper [JU99]:

> "An ontology may take a variety of forms, but it will necessarily include a vocabulary of terms and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms."

The latter definition refers well to what is necessary for meaningful Web content ("definitions and an indication of how concepts are inter-related") and it allows a better insight in what Gruber means by the key terms of his definition – "specification" and "conceptualization".

Even though both definitions do not use the term "knowledge" we want to mention that it is common in Computer Science to speak of ontologies as an explicitly encoded knowledge representation for the purpose of sharing it. At this point we think it is noteworthy that the notion of knowledge we referred to at the beginning of this section situates this concept in relation to "understanding", "awareness", and "experience" which one can treat as human capabilities beyond the means of computers. Being strict would mean that ontologies fail to conform to this definition of knowledge consequently. However, we want to introduce an example that we regard as sufficient for explaining the equalization of the terminology in the scope of this research. One can regard ontologies as a positivistic model or image of a knowledge state that leaves out some aspects of the real human world in favor of creating a sharable artifact.

[12] Please note the capital "O" when we speak of Ontology in a philosophical context.

When people communicate and one person shares knowledge with another, then this person also shares a positivistic model of the very personal real world, because the person will only be able to share a limited amount of her intrinsic conditions, such as emotions or experiences which will never be completely transferable. Hence, there is also a non-technical notion of sharing a subset of the reality as knowledge. We avoid deeper discourse on this issue here but we would like to refer to Alan Turings reflection about the differentiation of human and machine intelligence in his 1950 paper "Computing Machinery and Intelligence" [Tur50].

Practically ontologies are a semantically rich form of **Knowledge Organization Systems** which are defined by Zeng and Chan in [ZC04] as follows:

> "Knowledge Organization System (KOS) is a general term referring to the tools that present the organized interpretation of knowledge structures."

A KOS can take any form that organizes knowledge in a way that is standardized in a particular context, such as simple lists of terms (e.g. controlled vocabularies), classification hierarchies (e.g. taxonomies), or relationship models (e.g. thesauri) [Hod00]. Ontologies are characteristic for the latter form of KOS, providing classification hierarchies of labeled terms and extend them by complex relations and properties of the terms. [Zen08]

Web ontology languages

Ontology languages specify a consensual base terminology that cannot be generalized further and which can be applied to develop more specific ontologies. A first generation of ontology languages based on frames or first-order logic addressed the classical Artificial Intelligence and not knowledge representations conforming to the architecture of the Web at that date. Examples are Ontolingua [FFR97] or F-Logic [KLW95]. Languages for creating **Web ontologies** have been developed generally since the majorly of this millennium. When we speak about Web ontologies we refer to those developed in *RDF Schema (RDF-S)*, the *Simple Knowledge Organization System (SKOS)*, or one of the different fashions of the second version of the *Web Ontology Language (OWL)* which are state of the art standards as of writing this thesis. **RDF-S** is a "vocabulary description language". It is specified as a generic RDF vocabulary for the description of more specific RDF vocabularies. RDF-S allows the description of terms that represent either classes of entities or properties. **SKOS** provides another "model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, folksonomies, and other similar types of controlled vocabulary" [IS09] as

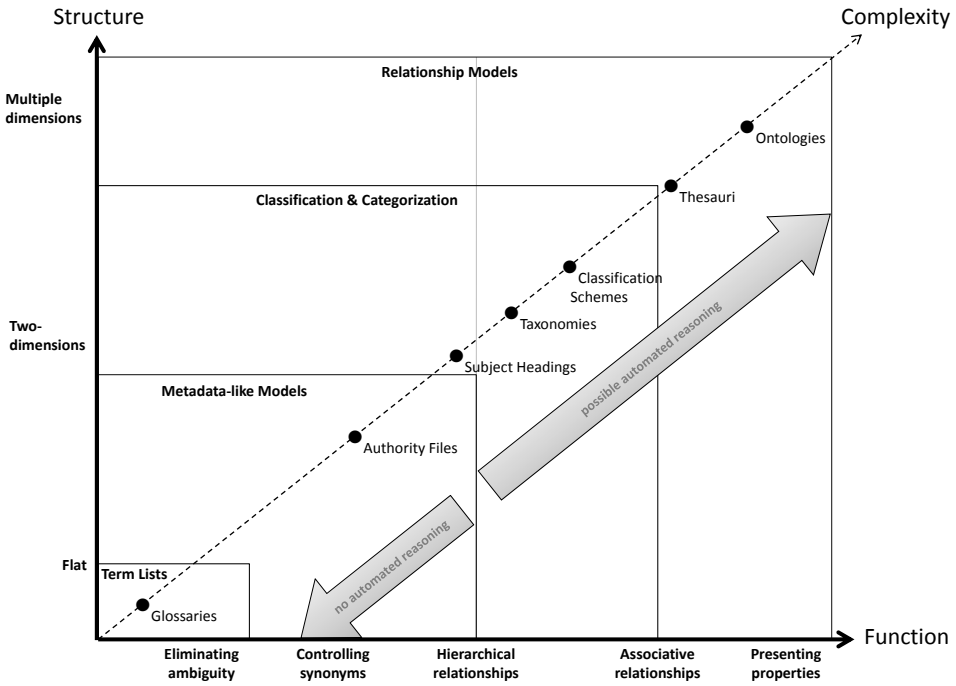

Figure 2.2: Ontologies as a specific type of Knowledge Organization Systems with reference to complexity and reasoning capabilities. Adapted from [Zen08] and [SW01].

RDF graphs. Concepts in SKOS represent terms which can be described using natural language labels, hierarchical and non-hierarchical relationships, and documentary notes. **OWL 2** is an extension to RDF-S in order to describe classes and properties in a more comprehensive way by providing vocabulary primitives for cardinalities, equality, and disjointness amongst others [W3C12]. There are two core language specifications – OWL2 Full and OWL 2 DL. OWL 2 DL has further three tractable profiles (OWL 2 EL, OWL 2 QL and OWL 2 RL). In the remainder of this thesis we will simply say OWL when we refer to the OWL 2 standard. All three, RDF-S, SKOS, and OWL use URIs for identification of classes and properties. RDF-S and SKOS can be serialized in the same fashion as RDF and for OWL various other formats are provided (e.g. OWL Abstract Syntax [PSHH04], OWL XML presentation syntax [HEPS03], OWL Functional Syntax [MPPS09], or OWL Manchester Syntax [HDG+06]). A serialized Web ontology is called an **ontology document**.

Conforming to the standards it is possible that Web ontologies contain terminological knowledge (sometimes also referred to as conceptual knowledge, T-Box

knowledge, vocabulary or schema) about **concepts** (abstract classes of things) and their **properties** as well as the factual knowledge (sometimes also called knowledge base, instance data or A-Box knowledge) about concrete **instances** of the concepts. In-line with the aforementioned definition of Ushold and Jasper, which says that "an ontology may take a variety of forms", we adopt this perspective in the remainder of this thesis and regard an ontology as the integration of a **vocabulary** and respective **instance data**. We admit that there are also other opinions on this aspect which reject instances as part of an ontology (e.g. in [OCM+07]).

Semantics

This thesis is not concerned with the theory of the semantic expressivity of different ontology languages. However, exploiting semantics in order to make implicit facts explicit based on a set of rules plays a role in the context of structured Web data sets anyway. Smith and Welty [SW01] give a general overview of the different degrees of semantics of ontologies in information systems as follows:

> "Information systems as simple as catalogs, in which each product type has a unique code (e.g. the item number), have been dubbed 'ontologies'. A catalog is, in a sense, the ontology of the things a company sells. A slightly more complex information system may provide simple natural language texts and allow string matching. Glossaries are information systems that provide natural language descriptions of terms, thus imposing some structure on the text (indexing by terms). Thesauri are standardized information systems that provide, in addition to descriptions of terms, also relations to other more general or more specific terms within a common hierarchy. The fields of knowledge representation, database development, and object-oriented software engineering all employ ontologies conceived as taxonomies in which properties of more general classes are **inherited** by the more specific ones. Frame-based systems provide, in addition to taxonomic structure, relations between objects and restrictions on what and how classes of objects can be related to each other. Finally, the most expressive and complex information system ontologies use the axioms of full first order, higher order, or modal logic. All these types of information systems satisfy Grubers definition, and all are now common bedfellows under the rubric of 'ontology'."

Mika and Akkermans coined the term **semantic dimension** in order to refer to "the different levels of formality of ontologies" [MA04]. They distinguish semantics by the "knowledge processes that ontologies support in applications". These are

communication, integration, and reasoning. An ontology supports communication when its formal semantics allow the sharing of common sets of terms and descriptions. Integration is facilitated if an ontology features relationships between these terms representing entities of a domain. Based on these two layers, that describe entities and their relationships in a domain of interest, **reasoning** exploits "the rules and principles behind a certain conceptualization" in order to check its consistency and to make implicit facts explicit [MA04].

The formal semantics of OWL allow automated reasoning with Description Logic (DL) [BCM+03]. OWL 2 DL is decidable and all three profiles of OWL 2 DL are even decidable in polynomial time. However, reasoning on this degree of semantic formality raises significant scalability issues and is not state of practice in RDF stores for example. This thesis is concerned with formal semantics up to the complexity of the **intensional semantics** of RDF-S [W3C04b, HKRS08]. We also take into account what is commonly referred to as RDF-S Plus [AH11] because it is commonly adopted for instance and vocabulary mappings in the context of Linked Data. However, we do not regard the effects those links have to complex reasoning because of the arising challenges of potential undecidability or at least too high complexity due to the amount and distribution of the data in regard. Table 2.2 lists the RDF-S Plus language primitives as well as their function and effect on the semantics of a structured Web data set.

Table 2.2: The RDF-S Plus basic concepts adapted from [AH11].

Category	Concept	Semantic function	Application level
Base concepts	rdfs:subClassOf	Members of subclass are also member of superclass.	Schema
	rdfs:subPropertyOf	Subproperty relations also hold for superproperty.	Schema
	rdfs:domain	Classifies subject into the domain of the predicate.	Schema
	rdfs:range	Classifies object into the range of the predicate.	Schema
Annotation properties	rdfs:label	Printable name.	Instance and schema
	rdfs:comment	Information for readers of the model.	Instance and schema
Equality features	equivalentClass	Members of each class are also members of the other.	Schema
	equivalentProperty	Relations that hold for each property also hold for the other.	Schema
	sameAs	Statements about one instance hold for the other.	Instance
Property characteristics	inverseOf	Exchange subject and object.	Schema
	TransitiveProperty	Chains of relationships collapse into a single relationship.	Schema
	SymmetricProperty	A property that is its own inverse.	Schema
	FunctionalProperty	Only one value allowed (as object).	Schema
	InverseFunctionalProperty	Only one value allowed (as subject).	Schema
	ObjectProperty	Property can have resource as object.	Schema
	DatatypeProperty	Property can have data value as object.	Schema

We subsume that in the context of this thesis we refer to RDF-S Plus semantics without regarding formal reasoning. A definition which also puts strong emphasize on the impact of the relations which describe *what* information and *how* information are related but leaves out the *why* – the aforementioned reasoning dimension – has been given by Berners-Lee in [BLF00] as follows:

> "In an extreme view, the world can be seen as only connections, nothing else. ... I liked the idea that a piece of information is really defined only by what its related to, and how its related. There really is little else to meaning. The structure is everything."

We adapted the layered visualization of Mika and Akkermans in order to clarify the position of structured Web data with reference to the degree of semantics, distribution, and overall complexity. The result is depicted in Figure 2.3.

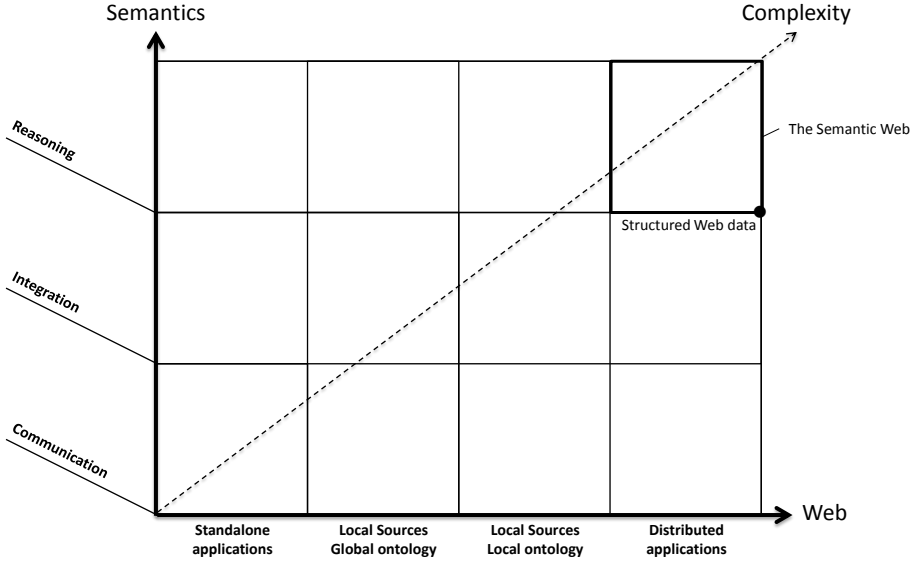

Figure 2.3: The semantic dimension of structured Web data. Adapted from [MA04].

2.4 Structured Web data sets

In 1999 the W3C started an effort towards a unified terminology of the core components of the Web [W3C99]. That involved terms of the hypertext Web such as Web

site, Web page, Web server or Web site provider amongst others. As a matter of fact, this effort was discontinued in working draft state most likely in favor of the Web architecture effort [Gro04]. In the context of this work we observed a lack of standard definitions of several important components of the Web of Data, for example – and most prominently – a data set.

The only publicly available definition of a notion of a data set is specifically focused to interlinked RDF data. The **vocabulary of interlinked datasets (VoID)** [ACHZ11] provides a socio-technical definition of a Linked Data set as "a set of RDF triples that are published, maintained or aggregated by a single provider" [ACHZ09].[13]

In 2012 the Web Data Commons (WDC) corpora were published which represent another important part of the Web of Data – structured data which is embedded in Web pages. The project is making this data available as a set of gzipped text files containing RDF-quads extracted from the publicly available Commons Crawl corpus. Analysis based on the WDC corpus reduced extracted resource identifiers to their respective pay level domains (PLD) interpreting each PLD as a data set[MB12, MP12].

It becomes obvious that there is no consistent notion of what a structured Web data set is. The VoID definition only applies for Linked Data sets which essentially publish VoID descriptions. We experienced that the best practices and guidelines around VoID are focused on what we will call *physical data sets* – data sets which are served by a dedicated Web application for data set hosting, such as Pubby[14]. There is a lack of guidelines for Web site providers to embed a VoID description together with the data of interest, which defines the namespace of what we will call *virtual data set* explicitly. Consequently this is not a best practice on the Web and the VoID definition of a data set is not applicable for most of the structured data in Web pages. The generalization of the WDC project to PLDs also has its shortcomings since it is too restrictive assuming that at one PLD only one discrete data set is served. A counter example are the Linked Open Drug Data project and the DBLP mirror which were hosted at the Freie Universität Berlin (FU Berlin) until 2012. All these physical data sets were served at sub-paths of the `www4.wiwiss.fu-berlin.de` server (e.g. `http://www4.wiwiss.fu-berlin.de/sider/`, `http://www4.wiwiss.fu-berlin.de/dblp/`) which is reduced to the PLD `fu-berlin.de`. Regarding personal blogs hosted at `[someuniquealias].blogspot.com` and service providers which differentiate user spaces by URI paths instead of subdomains, e.g.

[13] Please note that there is no difference in meaning between the different spellings of "data set" or "dataset" and that we adopted the former one consistently while the VoID definition adopted the latter one.

[14] `http://wifo5-03.informatik.uni-mannheim.de/pubby/`

`twitter.com/[uniqueusername]`, shows how the PLD approach is inconsistent with respect to virtual data sets. Each of these unique URI spaces can be treated as one of such if the maintainer of the account provides embedded structured data. From a perspective of stability of URIs the PLD granularity foments an adequate level of abstraction but from a perspective of effectively distinct data sets it fails to be representative.

To stay strictly in-line with the Web architecture, we regard it as an important characteristic that it is possible to describe a base URI identifying the data set namespace as in VoID[15], which makes a data set itself a Web resource. However, the existing VoID definition needs to be extended to reflect this general property of a structured Web data set. We define a **structured Web data set**[16] as follows:

2. Definition (Structured Web data set). *A structured Web data set is a collection of structured Web data that is published by a single provider. Publication can be either physical or virtual. Physical means to serve serializations of the structured descriptions of atomic resources within a dedicated namespace in control of the provider as representations of these resources or to offer a service that allows to query a structured Web data store directly on the Web in a standardized way. A virtual data set is a serialization of structured Web data served on the Web as raw data files or hypermedia Web content that embeds serialized structured data.*

It is a matter of fact that Web ontologies can be mapped to an RDF graph by nature as depicted in Figure 2.4 for the OWL 2 language specifically. Hence, such ontologies themselves are collections of structured Web data conforming to our generic definition.

In extension to this, there is no necessity by theory to regard structured Web data sets as something different than a specific form of Web ontologies. This form is characterized by the current best practices for the creation and management of structured Web data sets. Heath and Bizer recommend to reuse "suitable terms" which "can be found in existing vocabularies"[HB11]. We deduce from this recommendation that structured Web data sets very probably contain terminological knowledge from different namespaces. We call a collection of resources taken from one or more vocabularies the **vocabulary setup** of a data set.

It is noteworthy that Heath and Bizer generally define vocabularies as "collections of URIs that can be used to represent information about a certain domain" but that they do not distinguish between resources that represent concrete instances and those that represent abstract concepts in [HB11]. Both, instances and concepts, are simply

[15] http://www.w3.org/TR/void/dataset-uris

[16] Please note that we will use the terms data set and structured Web data set interchangeably in the remainder of this thesis.

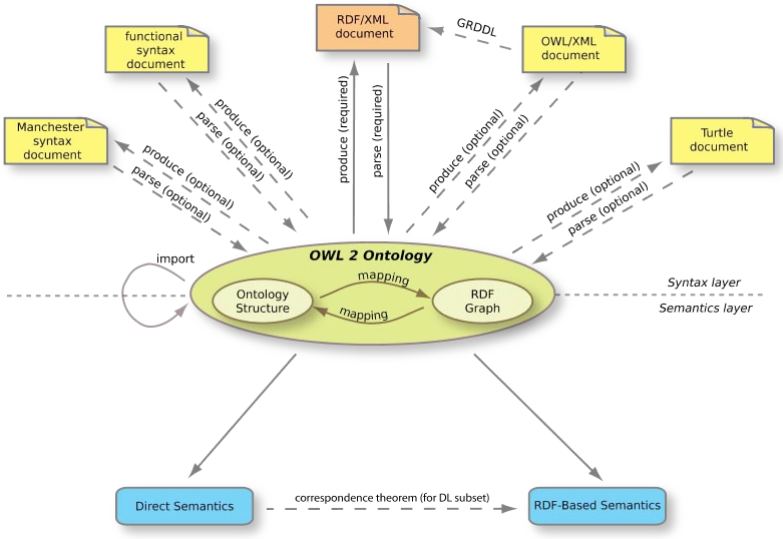

Figure 2.4: The structure of OWL 2. Taken from [W3C12]

treated as data and the only prominent position is accorded to predicate URIs in RDF triples which essentially "come from vocabularies". We still regard the differentiation between terminological knowledge and instance data as an essential foundation. The schema determines how things relate to each other and what they are [BLF00]. The meaning of data can be changed by changing the schema, because both, the *what* and the *how*, can change completely independent from the resources representing the related instances. Changing the schema can influence the possibility to integrate instance data from different data sets for example, hence, up- or downgrading the applicability of instance data.

We define the **schema level** of a data set as follows:

3. Definition (Schema level). *The schema level of a structured Web data set is the subgraph that only contains resources from the vocabulary setup.*

And reverse, the following definition borders the **instance level** of a data set:

4. Definition (Instance level). *The instance level of a structured Web data set is the subgraph that only contains descriptions of resources which are not part of the vocabulary setup.*

While the schema level definition results in a graph that only contains abstract concepts and properties from either vocabularies or ontology languages, the instance

level definition regards some resources from the schema level being part of the instance level as well, when they are explicitly part of the description of an instance. Consequently, the extend to which the instance level includes the schema level depends on if and how inferred descriptions are materialized in the data set.

2.5 Usage of structured Web data sets

Usage of the Web is characterized by two basic patterns today: *publication* and *consumption*. The **consumer** requests resources from a Web server operated by a **publisher**. These resources may either have any static contents as representations or dynamically generated contents from a service invoked on the server by the request. Practically Web usage is dominated by HTTP GET and POST requests by consumers which are navigating through static or dynamic Web pages or emit queries which are mostly in natural language or any semi-structured format. In the context of this thesis we define **usage** of structured Web data sets as follows:

5. Definition (Usage of structured Web data sets). *Usage of structured Web data sets is the retrieval of the descriptions of Web resources by directly accessing them or by performing queries against a structured data query endpoint over HTTP.*

2.5.1 Architectures of structured Web data sets

In [HB11] Heath and Bizer introduced a generic Linked Data publication architecture as well as crawling as the most representative usage pattern on the Web of Linked Open Data (cf. Figure 2.5). The authors introduced the data publisher, the data consumer, and third parties (e.g. publishers of pure link sets or data search engines) as the stakeholders which share the overall data integration effort among each other. It is noteworthy that Heath and Bizer focus on data which is openly available on the Web. Due to that fact they emphasize the crawling pattern because it is currently not a wide practice to let people query data directly on the Web in a completely unrestricted way (e.g. without limits on result sets).

Our approach regards structured Web data not necessarily in a completely open infrastructure but also as a general means for the instantiation of dataspaces that are integrated in an evolutionary fashion. We furthermore defined the notion of physical and virtual data sets and will now give an overview about characteristic architectures of these two forms. This is the foundation for the classification of *usage patterns of structured Web data sets* which we will introduce afterwards.

The difference between a physical data set architecture and a virtual one is the possibility for the data set publisher to monitor the access to atomic resources of the

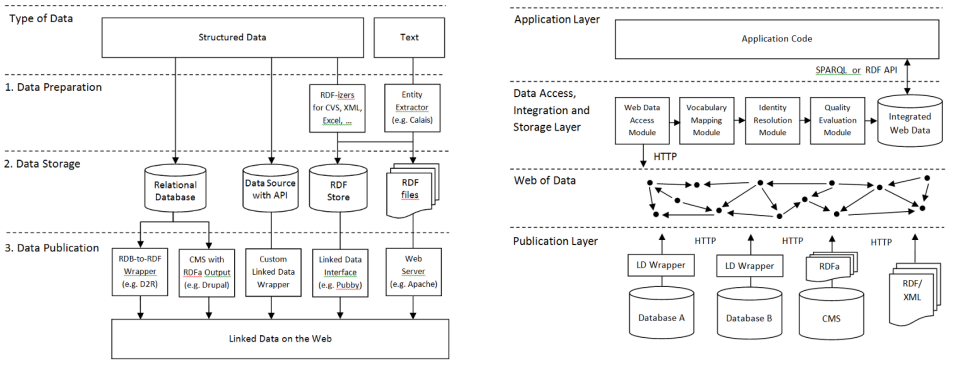

Figure 2.5: The generic Linked Data publication architecture (left) and the Linked Data crawling pattern as introduced by Heath and Bizer in [HB11].

data set. While this is easy when the data publisher provides a query endpoint or makes every single resource of a dedicated namespace retrievable via HTTP, it is not possible to assess any fine grained usage statistics when resources are never directly retrieved because they are embedded in other resources (Figure 2.6).

This is an important aspect since our notion of usage presumes that the resources of a structured data set are either directly accessible or can be retrieved by performing queries against a dedicated endpoint. Virtual data sets require the user to extract the structured data from the proxy resource which they were embedded in. Hence, the data publisher will never be able to analyze the usage of the exposed data which is an anti-pattern for the applicability of the approach in this thesis.

2.5.2 Data set usage patterns

Usage of Linked Data as the most representative technique for structured Web Data has been described in works such as [BHBL09],[Hau09] and [Hea08]. One can summarize that Linked Data typically is used (1) to provide unambiguous concept identifiers within Web applications, (2) to enhance the experience of Web users by aggregation and integration of corresponding content within CMS systems and Web applications, and (3) to be browsed and mashed up in a user-specific way. It becomes apparent that the classical browsing scenario plays a minor role and is outperformed by the access and use of Web resources through libraries or applications which are not or only indirectly connected with a human user's interaction plays an important role in

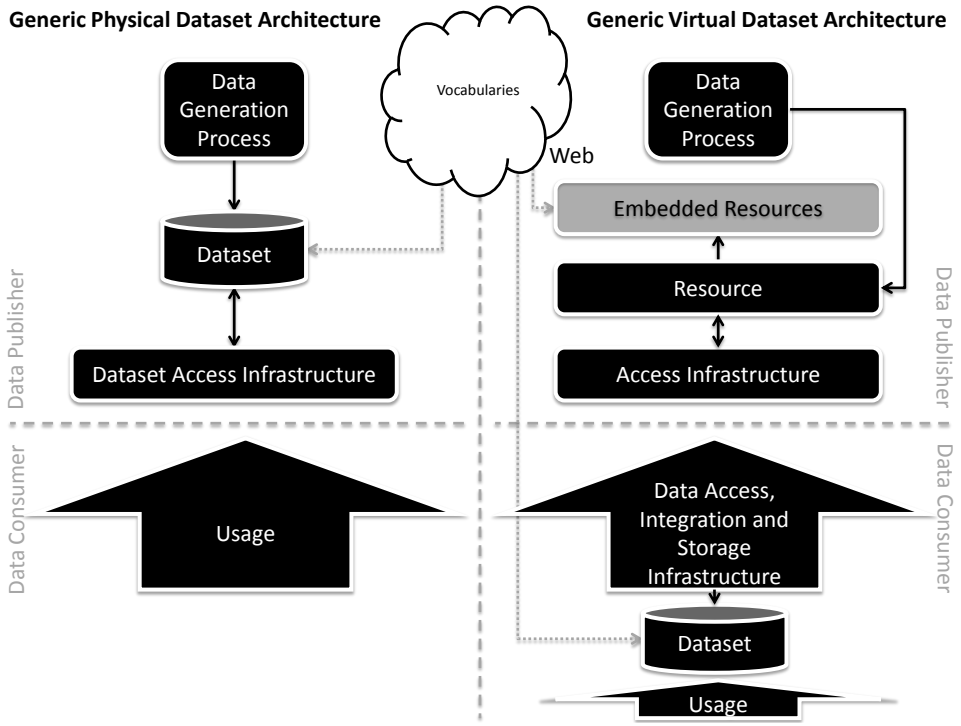

Figure 2.6: Generic architecture of physical and virtual data sets.

these scenarios.

In the scope of our research we adapt this view but refer to usage of structured Web data sets on a different layer of abstraction. Physical data sets typically have two characteristic usage patterns which are depicted in Figure 2.7 and described as follows:

Resource centered access: The consumer accesses the resources which are part of a data set directly by HTTP GET requests. The publisher's infrastructure aggregates a description of the requested resource as a graph and sends a serialization of the result back to the consumer. Any processing of this result, the selection of the relevant parts of the retrieved resource description, or the integration with other resources is outsourced to the data consumer.

Query pattern access: The consumer formulates queries and sends them as HTTP GET requests to a query endpoint provided by the publisher. The publisher's infrastructure processes the query and creates a result serialization which is

responded to the consumer. This allows the data publisher to decompose the performed query and retrieve information about which and how atomic resources are queried together.

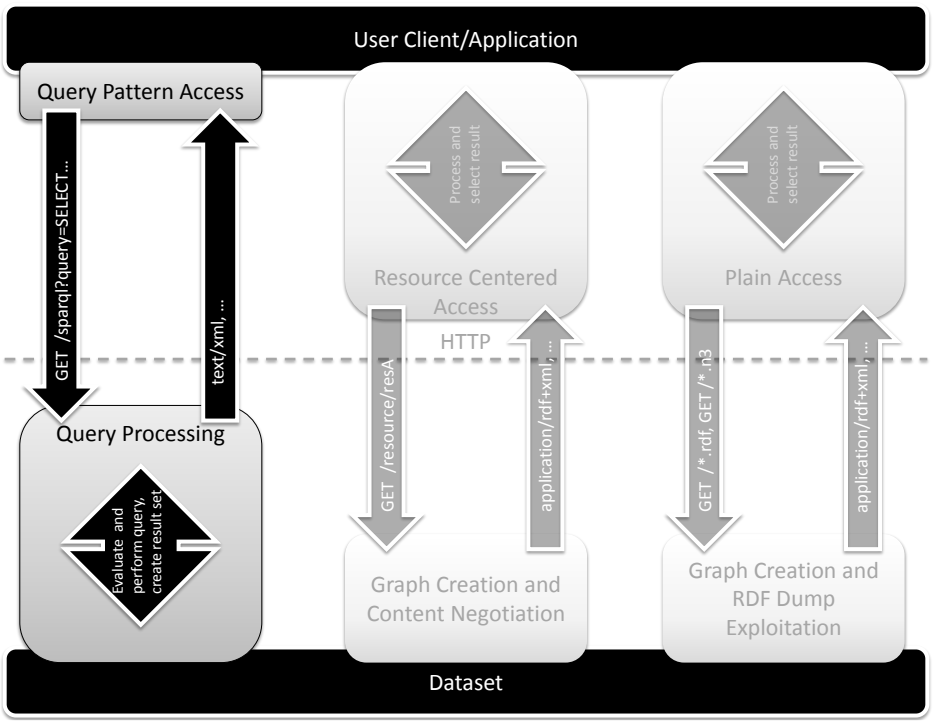

Figure 2.7: Classification of usage patterns of structured Web data sets

2.5.3 Web usage mining

Classical **Web usage mining** is concerned with the application of *data mining* techniques to the captured usage data of Web sites or services [CMS97]. **Data mining** in general addresses the problem of the algorithmic identification and extraction of patterns from data. When any kind of knowledge is the intended output of a data mining process it is common to use the term **Knowledge Discovery in Databases (KDD)**. The characteristic KDD process introduced in [FPSS96] is depicted in Figure 2.8

Essential parts of Web usage mining are the characteristic metrics and patterns one tries to identify, such as hits, page impressions, visits, time and navigation heuris-

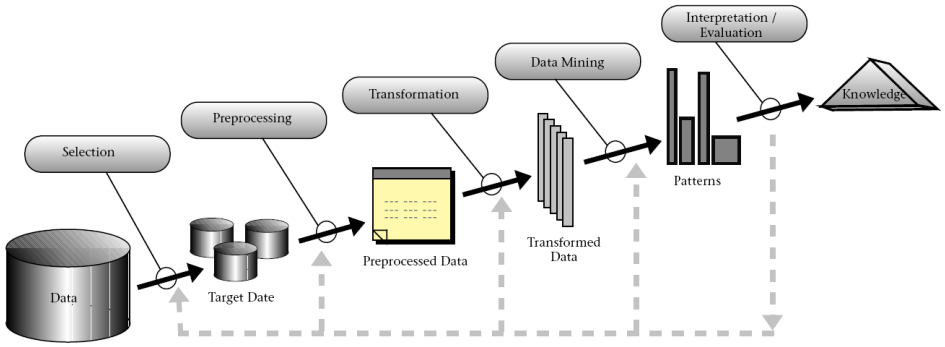

Figure 2.8: KDD process as introduced in [FPSS96].

tics, unique visitors, click-through, view time, sessions, path analysis, association rules, sequential patterns, classification rules or clusters [Spi00, SC00].

Already in 2002 and again in 2004 Berendt et al. introduced a new research area – the so called Semantic Web mining [BHS02, B$^+$04]. The authors describe how the two disciplines, namely the Semantic Web and Web mining, may converge. They present three perspectives which reflect this: First, the perspective how Web mining can help to extract real semantics from the Web. Second, the exploitation of semantics for Web mining. And third, the perspective of mining the Semantic Web. This third perspective is subdivided into Semantic Web structure and content mining as well as Semantic Web usage mining from which our work is related to the latter because it deals with the analysis of the usage of structured data on the Web.

In the context of our work we apply Web usage mining techniques for the analysis of **Web server log files** in order to facilitate an "adaptive behaviour" [B$^+$04] of structured Web data sets. We regard Web server log files which are most typically provided in the Common Log Format (CLF)[17] as shown in Listing 2.6.

Listing 2.6: Anonymized excerpt of a DBpedia log file showing different types of client requests and the responded HTTP status codes.

```
xxx.xxx.xxx.xxx - - [21/Sep/2009:00:00:00 -0600]
  "GET /page/Jeroen_Simaeys HTTP/1.1"
  200 26777 ""
  "msnbot/2.0b (+http://search.msn.com/msnbot.htm)"
xxx.xxx.xxx.xxx - - [21/Sep/2009:00:00:00 -0600]
  "GET /resource/Guano_Apes HTTP/1.1"
  303 0 ""
  "Mozilla/5.0 (compatible; Googlebot/2.1;+http://www.google.com/bot.html)"
xxx.xxx.xxx.xxx - - [21/Sep/2009:00:00:01 -0600]
```

[17] http://www.w3.org/TR/WD-logfile.html

```
"GET /sparql?query=PREFIX+rdfs%3A+%3Chttp%3A%2F%2Fwww.w3.org..."
200 1844 ""
""
```

A problem on the Web of Data in its current shape is that the meaning of HTTP status codes[18] does not work out at all time. When accessing a URI which does not point to any resource on a Web server, the server responds the 404 code. The SPARQL protocol specifies that servers respond the HTTP status code 200 and a serialization of the SPARQL results format that contains no result bindings in the case that a SELECT query is performed correctly but yields an empty result set. The HTTP status code definitions would also allow the use of the 204 status code in this case. While this looks like a misuse of HTTP response codes at a first sight, it also may be a desired feature for developers which deal with empty result sets in an application-dependent fashion and detect this when the serialization of the result is processed. However, for an in depth analysis of requests that contain SPARQL queries this means that queries must be re-ran to find out whether they returned any result or not.

2.6 Data set maintenance

Maintenance in the context of information systems most commonly refers to software maintenance. The IEEE definition for software maintenance has been given in as follows [IEE90]:

> "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."

Since we defined structured Web data sets as a specific form of ontologies we also regard structured Web data set maintenance from a perspective that is inspired by *ontology engineering*.

2.6.1 Ontology engineering

Ontology engineering[19] has been best defined by Gomez-Perez et al. as follows [GPFLC04]:

[18] `http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`

[19] Ontology engineering is sometimes also referred to as ontological engineering. We adopt to the term "ontology engineering" in this thesis.

> "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."

Ontology engineering methodologies provide guidelines for developing, managing and maintaining ontologies. Such methodologies decompose the ontology engineering process in a number of steps, and recommend activities and tasks to be carried out for each step by the stakeholders typically involved in ontology engineering – **ontology engineers**, **domain experts**, **knowledge engineers**, and **ontology users**. The importance of a particular activity within a concrete ontology-related project depends on the characteristics of the environment in which the ontology is to be used, the complexity of the ontology to be developed, the availability of domain-relevant information sources, and the experience of the ontology engineering team, to name but a few of the relevant factors in this context. Orthogonally thereof, in [GPFLC04] the authors distinguish among three types of activities within an ontology engineering process - management, development and support activities. The first covers the organizational setting of the overall process, including a feasibility study that examines if an ontology-based application, or the use of an ontology in a given context is the right way to solve the problem at hand. The second type of activities refers to classical activities such as domain analysis, conceptualization and implementation, but also maintenance and the use, which are performed at post-development time. Ontology support activities such as knowledge acquisition, evaluation, reuse, and documentation are performed in parallel to the development activities.

Research has resulted in a wide range of ontology engineering methodologies which mainly differ in details referring to the composition of ontology engineering and application development, the range of users interacting on ontology engineering tasks, and the degree of life cycle support. Some methodologies assume their users to be ontology experts only or at least to be knowledge workers with little technical experience while others also address users with no experience in ontology engineering at all.

In summary three generations of ontology engineering were passed through since the very beginning of research on this topic:

1. Ontology engineering from scratch as the art of experts,
2. ontology engineering inspired by software engineering, and
3. collaborative ontology engineering.

The distinction into two categories depending on the application setting is also very common [SLR13]:

1. Centralized Ontology Engineering and
2. decentralized ontology engineering.

2.6.2 Maintenance of structured Web data sets

Within a structured Web data provisioning context the relevance and scope of the activities traditionally covered by the ontology life cycle needs to be adjusted in order to optimally meet the requirements of this new class of application scenarios from a process-oriented and a technical point of view. Structured Web data sets are generally developed application-independently, so they are settled in a decentralized setting. A key driver for the evolutionary nature of structured Web data sets is collaboration between the different stakeholders which are in the context of this work described as follows:

Data set architect: While a data set publisher (or provider) is the legal entity (a person or organization) in which name a data set is published, the data set architect is a person which is concerned with the creation, publication, and maintenance of a data set.

Ontology engineer: An ontology engineer is a trained developer of terminological knowledge artifacts in RDF-S, OWL, or SKOS and potentially further Web ontology languages.

Data set user: A user of a structured Web data set is a person or organization that uses a data set by direct access or through an application.

Domain expert: A domain expert is a person that has profound knowledge about the terminology and relations in a domain of interest addressed by a specific data set and has potentially been involved into the creation and management of legacy data sources in this domain before.

Technology expert: Technology experts are researchers and practitioners that have proven skills around the standards and technologies for structured Web data.

The definition of software maintenance also emphasises that maintenance is a post-development process. But is furthermore introduces three main goals which we refer to in the context of this thesis as follows:

Correction: Correction is the change of data that cause errors.

Improvement: Improvement is the incrementation of the *quality* along a particular set of criteria.

Adaptation: Adaptation are changes in response to evolving extrinsic conditions.

We regard errors in applications as a result of messy data as well as the conformance of data to the environmental conditions as specific quality criteria. Hence a generic definition of **structured Web data set maintenance** can be given as follows:

6. Definition (Structured Web data set maintenance). *Structured Web data set maintenance is the improvement or preservation of the data quality along a particular set of criteria.*

With reference to the characteristic patterns of structured Web data sets usage we address and the Web usage mining methods we apply, we can now define **usage-dependent maintenance of structured Web data sets**:

7. Definition (Usage-dependent maintenance of struct. Web data sets). *Usage-dependent maintenance of structured Web data sets is the adaptation of a data set to improve or preserve the data quality along a particular set of criteria that reflect evolving extrinsic conditions assessed by the analysis of tracked usage feedback.*

At this point we explicitly mention that we do not regard criteria concerned with the access infrastructure of a data set but only those that offer any connection to the quality of the schema and the instance level. We do so because research on the performance of semantic data stores, optimized indexing strategies, or data prefetching strategies are independent topics which are heavily studied. The same holds for security and access control. We are aware that the WebID approach for example offers to implement fine-grained authentication and authorization on the level of structured Web data without any other proprietary infrastructure components. However, as of writing this thesis the application of this technology is not a wide practice at least in the context of projects that aim for an evolutionary integration of data sets in control of different providers, such as the Linked Open Data community effort.

2.7 Data quality framework

In the last section we identified that **data quality (DQ)** is a central aspect of data set maintenance, which may (a) indicate the necessity of maintenance activities

in response to low data quality or (b) help to measure the effect of maintenance activities. It is widely accepted that the definition of data quality is highly individual for each application context and standards are missing [WS96, SLW97, PLW02, BS06, MWL09].

In [BS06] the authors report on three typical approaches to define data quality dimensions – the theoretical approach, the empirical approach, and the intuitive approach. While the theoretical and the intuitive approach aim at providing a standardized model or characterization for data quality dimensions, the empirical approach is based on interviews with data consumers. Due to our focus on the usage of data sets we refer to the definition of data quality from the empirical approach given as follows [WS96]:

> "... data that are fit for use by data consumers"

Based on this consumer-oriented viewpoint that adopts the popular definition of quality in general from [GJB74], Wang and Strong first derived four core categories of data quality (see Figure 2.9).

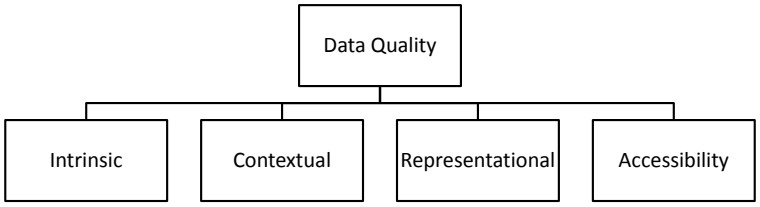

Figure 2.9: Data quality categories as defined by Strong et al. [SLW97].

These categories have been further refined into the following set of 16 **data quality dimensions** which is still state of the art as of writing this thesis [SLW97, PLW02].

2.7.1 Mapping data quality and ontology evaluation criteria

Since we introduced that any structured Web data set is a specific form of a Web ontology it is necessary that we also regard the state of the art in ontology evaluation to define the adequate quality framework for structured Web data set. We refer to the following set of eight evaluation criteria recently compiled by Vrandecic [Vra10] from the broad range of ontology evaluation approaches.

Now we create a mapping between the ontology evaluation criteria and the data quality dimensions introduced before in order to derive the final quality framework

Dimension	Definition
Accessibility	How well and quick is the data retrievable?
Appropriate amount of data	Is the volume of the data appropriate for the current task?
Believability	Is the data true and credible?
Completeness	Is data missing for the current tasks?
Concise representation	How compact is the data represented?
Consistent representation	Is the data represented uniformly?
Ease of manipulation	How far is it possible to adapt the data to apply to different maybe unanticipated tasks?
Free-of-Error	Is the data correct and reliable?
Interpretability	Are the languages, symbols, units, and definitions of the data appropriate for the current consumer?
Objectivity	Is the data impartial?
Relevancy	Is the data helpful for the current task?
Reputation	Is the data highly regarded?
Security	How appropriate are the access restrictions to the data?
Timeliness	How up-to-date is the data?
Understandability	Is it possible to comprehend the data easily?
Value-added	Does the data provide advantages from its use?

Table 2.3: Data quality dimensions as defined by Pipino et al. [PLW02].

which we refer to in this thesis. We do so in order to respect two possible perspectives to quality in the context of structured Web data sets – a data-centric perspective as well as an ontology-centric perspective.

Dimension	Definition
Accuracy	Is the ontology correct in the sense that the conceptualization and all possible inferences are true and compatible with the "gold standards" of the user?
Adaptability	How far is it possible to apply the ontology to different maybe unanticipated tasks?
Clarity	How well is it possible to retrace the intended meaning of the concepts and properties defined in the ontology?
Completeness	Is the domain of interest appropriately covered?
Computational efficiency	How efficiently can tools, e.g. reasoners, work with the ontology?
Conciseness	Does the ontology contain irrelevant conceptual knowledge about the domain of interest?
Consistency	Is the ontology free of intrinsic contradictions and faults?
Organizational Fitness	How well can the ontology be deployed within an organization's infrastructure?

Table 2.4: Ontology evaluation criteria compiled by Vrandečić [Vra10].

Accuracy ⇒ Free-of-error: Both, the ontology evaluation criterion accuracy and the data quality dimension free-of-error, refer to conceptual and factual truth not only on a technical level but also determined by human perception.

Adaptability ⇒ Ease of manipulation: Adaptability and easy of manipulation clearly address the same issue of adapting an ontology or data to different tasks.

Clarity ⇒ Interpretability + understandability: The interpretability and understandability dimensions regard traceability based on descriptions and encoding as well as if the proper natural languages are applied which is in combination the focus of the clarity criterion.

Completeness ⇒ Completeness: Both perspectives deal with the question if the domain of interest for the current task is appropriately covered.

Computational efficiency ⇒ Accessibility: We treat computational efficiency as a subdimension of accessibility because it depends on the ability of the accessed infrastructure to deal with complex modeled knowledge representations. This conforms to the fact that the accessibility dimension is concerned with how quick data is retrievable.

Conciseness ⇒ Concise representation: Both perspectives deal with the question if not more than necessary is represented.

Consistency ⇒ Consistent representation: Consistency and consistent representation are both concerned with the intrinsic consistency of ontologies or data respectively.

Organizational fitness ⇒ Accessibility + believability + objectivity + relevancy + reputation + security + timeliness + value-added: Organizational fitness subsumes a number of technical but also organizational aspects that influence the applicability of an ontology in an organization.

Since we mapped each ontology evaluation criterion to one or more corresponding data quality dimensions we will adopt the data quality terminology in the remainder of this thesis which is also the much more common terminology in the area of information systems. Table 2.5 shows the final data quality framework we refer to in the context of this thesis. It reflects (a) the mapping of the ontology evaluation criteria and (b) the restriction to focus dimensions that are concerned with the quality of the schema and the instance level of data sets.

Data quality category	Data quality dimension	Ontology evaluation criterion
Intrinsic	Believability	Organizational fitness
	Free-of-error	Accuracy
	Objectivity	Organizational fitness
	Reputation	Organizational fitness
Contextual	Value-added	Organizational fitness
	Relevancy	Organizational fitness
	Timeliness	Organizational fitness
	Completeness	Completeness
	Ease of manipulation	Adaptability
	Appropriate amount of data	
Representational	Interpretability	Clarity
	Understandability	Clarity
	Consistent representation	Consistency
	Concise representation	Conciseness
Accessibility	Accessibility	Computational efficiency, organizational fitness
	Security	Organizational fitness

Table 2.5: Mapping of ontology evaluation criteria to data quality dimensions

CHAPTER 3

Related work

In the previous chapter we introduced those pieces of related research to which we refer to as the theoretical fundament of this thesis. We will now survey about work which is relevant because it addresses similar research issues in order to contract our work with related research in the concrete problem space. We already explained that and why this thesis spans across the borders of one discrete research area. Hence, we will now survey three dimensions of related work – engineering methodologies and life cycles, Web of Data usage mining, and data quality in the context of the Web of Data – with an emphasis on engineering methodologies and life cycles which is the core pillar of this research.

3.1 Engineering methodologies and life cycles

To show how the ontology engineering discipline evolved we will now briefly introduce a number of well-regarded ontology engineering methodologies. [1] We also introduce the two most recent and most regarded life cycles that specifically address the Web of Data use case and as a third perspective a KOS life cycle which discloses an interesting overlap with our concept even though being differently inspired and situated in a completely different area.

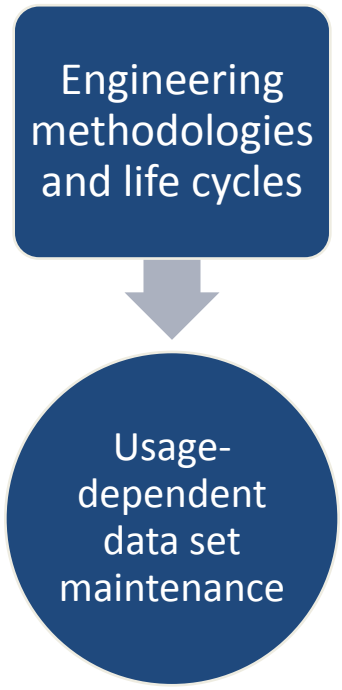

Figure 3.1: Engineering methodologies and life cycles – the first dimension of work which is related to our approach.

[1] We are aware that various other ontology engineering methodologies exist which we leave out here. We do so because we regard them to have lesser impact on the research discipline as a whole. Please refer to [FLGP02] for a more comprehensive overview and analysis.

3.1.1 METHONTOLOGY

Early approaches to unified ontology engineering, such as the methodology by Ushold and King [UK95], fundamentally defined the formal and informal aspects that make ontology engineering a specific engineering discipline instead of a hardly traceable art of experts. This has been further evolved by **METHONTOLOGY** [FGPJ97] which is an approach that was highly inspired by software engineering best-practices of that time. The approach aimed at improving the applicability of ontology engineering by assigning well-established software engineering activities to the ontology development process. Embraced by management activities and support activities the core development activities introduced by METHONTOLOGY are:

- Specification
- Conceptualization
- Formalization
- Implementation
- Maintenance

Figure 3.2 depicts how the METHONTOLOGY life cycle arranges these activities which were adopted from the IEEE 1074-1997 standard for developing software life cycle processes [S+97] and how they are influenced by the surrounding management and support activities.

3.1.2 On-To-Knowledge

The On-To-Knowledge methodology (**OTK**) developed by Sure and Studer [SS02, SS09] also has an application-dependent focus on ontology engineering emphasizing the integration of the knowledge meta process and the knowledge process which were most commonly regarded as orthogonal processes [SSSS01]. This brings together participants which are not very familiar with ontologies as well as technical experts especially in early phases of the process for the identification of use cases and competency questions. The OTK process arranges the following five phases:

- Feasibility study
- Ontology kickoff
- Refinement
- Evaluation

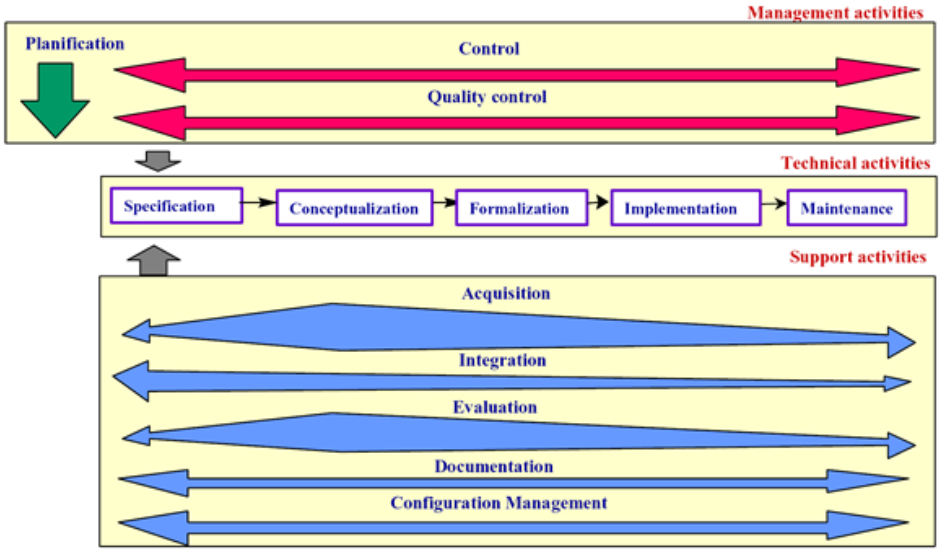

Figure 3.2: The METHONTOLOGY ontology development process taken from [FGPJ97].

- Maintenance

This methodology is related to our work since it promotes strategies for ontology maintenance. However, this part of the life cycle in OTK is "primarily an organizational process"[SSSS01]. Figure 3.3 shows that maintenance is introduced as the loop back to the refinement phase without any detailed description of internal and external or anticipated and unanticipated maintenance indicators for example.

3.1.3 DILIGENT and HCOME

The methodologies **DILIGENT** [PTSS06] and **HCOME** [KV06] address the problem of ontology engineering from the viewpoint that reaching an ontology consensus is highly dependent on enabling discourse of people with disparate skill level. Both methodologies assume a distributed setting. Every individual is free in adapting the central ontology consensus locally. The evolution of the consensual model is dependent on these local adoptions. Thus, HCOME and DILIGENT propose a human-centered approach to ontology engineering in a very similar fashion but different life cycle models. The core phases proposed by HCOME are:

- Specification

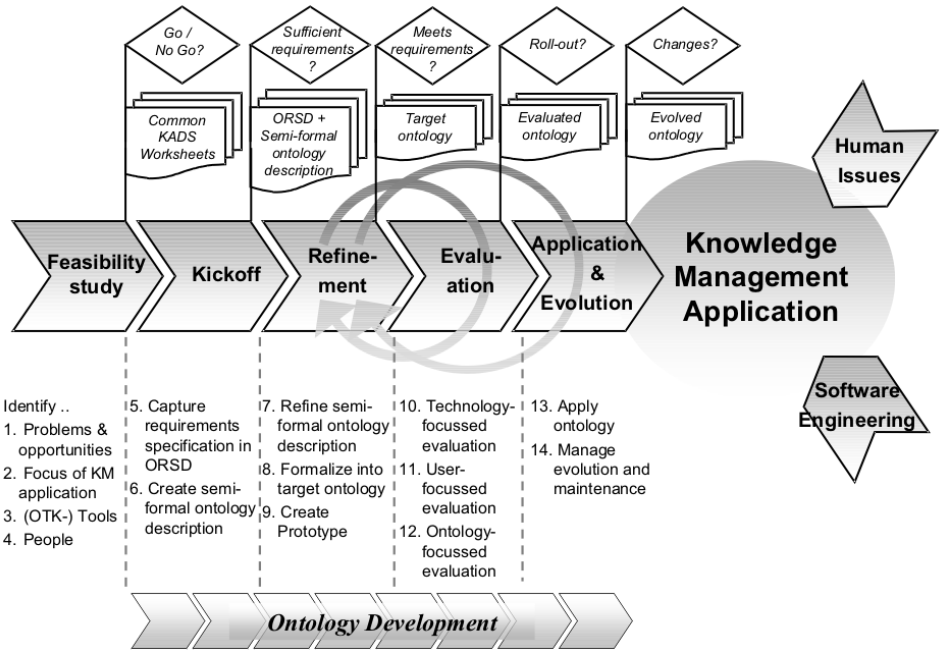

Figure 3.3: OTK life cycle as introduced in [SS02] and [SS09].

- Conceptualization
- Exploitation

And the DILIGENT methodology proposes the following life cycle phases:

- Build
- Local adaptation
- Analysis
- Revision
- Local update

Figure 3.4 allows to compare these two different life cycle models. One can see that HCOME is an approach to set the well-established ontology engineering activities into the context of a distributed community of development stakeholders with

different skills. DILIGENT on the other hand is much more innovative since it introduces completely new activities (e.g. local adaptation, local update, and analysis) emphasizing the flexible reuse of published ontologies which influences the further development of the original ontology by the feedback and individual changes of its users.

Both approaches share the characteristic that community discourse happens around the ontology aiming at developing the most representative ontology as possible. Our approach introduces the negotiation and communication of community best practices and the commonly applied vocabularies for characteristic domains as a form of community discourse. Participation of the data set publisher in this is a means to improve the capability of structured Web data sets of being integrable with other data sets and useful for applications accessing it.

3.1.4 RapidOWL

Following another trend in the software engineering discipline around 2006 agile ontology engineering came into focus of research. In [AH06] Auer introduces **RapidOWL**, a paradigm-based approach without any life cycle model. The methodology is designed to enable the creation of a knowledge base by domain experts even in absence of experienced knowledge engineers and Auer describes the following dimensions to achieve this:

- Values which outline the organizational long-term goals and philosophy.
- Principles as the characterization and guidance of the RapidOWL process.
- Practices as the enrollment of RapidOWL for the active development of application-dependent ontologies.

With reference to our work it is noteworthy that user feedback is introduced as a key principle. Auer highlights "the values of Communication and Feedback" [AH06, Aue07] but leaves it underspecified which forms of feedback can be collected and how it is evaluated.

3.1.5 Ontology maturing

Ontology maturing is a community-driven approach to ontology engineering [BSW+07] that puts emphasis on the role of the user community in steering a sustainable, long-term evolution of lightweight ontologies. The maturing process consists of four phases:

- Emergence of ideas

Ontology life-cycle phases	Goals	Tasks
Specification	Define aim / scope/ requirements/ teams	▪ discuss requirements (S) ▪ produce documents (S) ▪ identify collaborators (S) ▪ specify the scope, aim of the ontology (S)
Conceptualisation	Acquire knowledge	▪ import from ontology libraries (P) ▪ consult generic top ontology (P) ▪ consult domain experts by discussion (S)
	Develop & Maintain Ontology	▪ improvise (P) ▪ manage conceptualisations (P) ▪ merge versions (P) ▪ compare own versions (P) ▪ generalize/specialize versions (P) ▪ add documentation (P)
Exploitation	Use ontology	▪ browse ontology (P) ▪ exploit in applications
	Evaluate ontology	▪ initiate arguments and criticism (S) ▪ compare others' versions (S) ▪ browse/exploit agreed ontologies (S) ▪ manage the recorded discussions upon an ontology (S) ▪ propose new ontology versions by incorporating suggested changes (S)

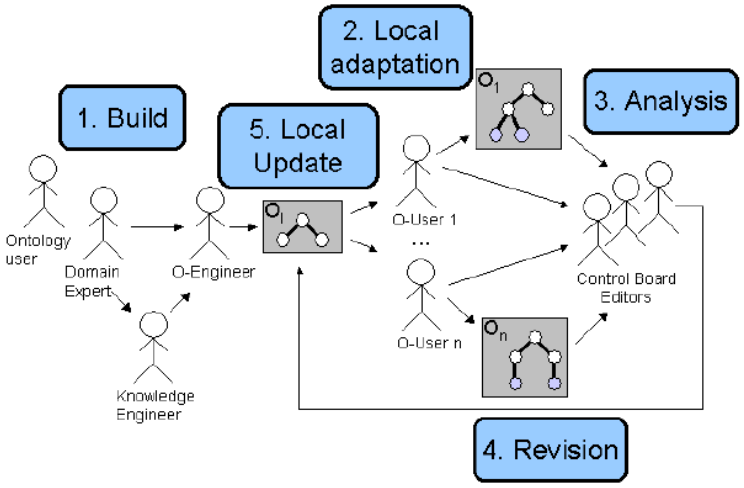

Figure 3.4: The HCOME (top, taken from [KV06]) and DILIGENT (bottom, taken from [Tem06]) ontology life cycles.

- Consolidation in communities
- Formalization
- Axiomatization

From the collection of new concept ideas in an ad hoc fashion by a simple tagging approach a common taxonomy is derived. The result are ontologies with an increasing degree of formality and expressivity (see Figure 3.5). A number of case studies that involve the collaborative ontology editing tool SOBOLEO have shown the feasibility of this approach [BZ10, BKS10].

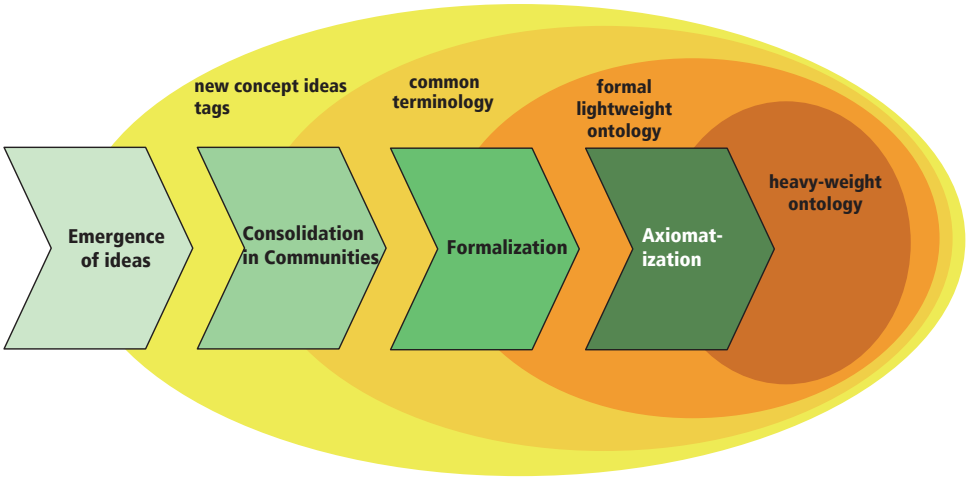

Figure 3.5: The ontology maturing life cycle as introduced in [BSW+07].

3.1.6 The NeOn methodology

As a result of the NeOn project the **NeOn methodology** for ontology engineering and the NeOn architecture for life cycle support in ontology-based systems have been developed [T+07, GPSF09, dCSdFB10]. NeOn is the most recent and in fact most comprehensive ontology engineering methodology. It brings together the results from three of the most representative ontology engineering methodologies developed before – METHONTOLOGY, OTK, and DILIGENT – and provides the current standard glossary of ontology engineering activities as well as a well-established set of methods and tools that support the ontology design and development process. Altogether the NeOn methodology introduces the following seven phases which are orchestrated in

different ways to instantiate life cycles that suit for different application scenarios (see Figure 3.6):

- Initiation phase
- Reuse phase
- Reengineering phase
- Merging phase
- Design phase
- Implementation phase
- Maintenance phase

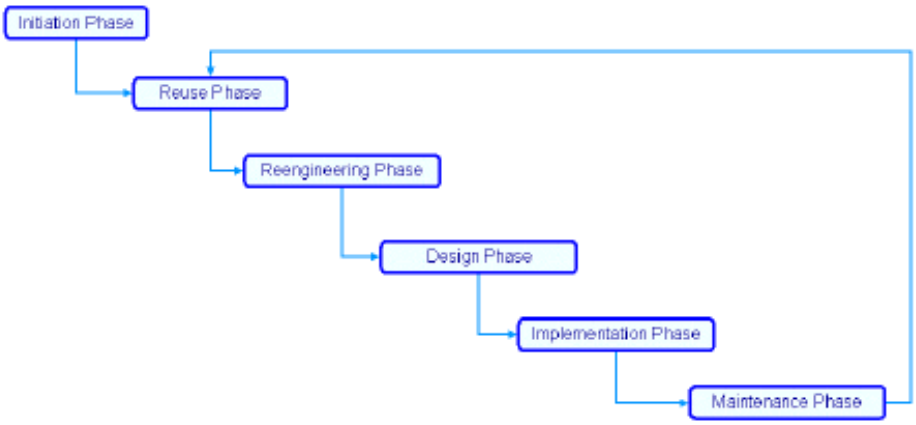

Figure 3.6: One NeOn life cycle model (taken from [dCSdFB10]).

The main goal of NeOn is to support the development of so-called ontology networks. Such an ontology network in the sense of the NeOn approach is a "collection of ontologies related to each other via [...] meta-relationships" [dCSdFB10]. These meta-relationships can express that an ontology consists of a number of modules which together capture the domain of interest, reuses, extends or maps to other ontologies, or simply is a successive version of an ontology. All these relationships show that the resulting network of ontologies is designed in an a priori fashion which is the first difference to the structured Web data scenario we are focusing on. It

is typical for structured Web data that any integrated network view to the ontologies which constitute the schema level dynamically emerges a posteriori from the set of vocabulary primitives which are in use at a certain point of time. Additionally, NeOn comes along with a comprehensive set of life cycle models allowing for different project setups which are depicted in Figure 3.6. One can see that again maintenance is treated as a loop back to the earlier steps in the process but that any detailed description of the sub-activities of maintenance is missing. This blind spot is furthermore emphasized since the glossary of activities in [dCSdFB10] also leaves out to define maintenance.

3.1.7 Eckert's KOS life cycle

Another piece of recent work by Eckert [EMS11, Eck12] proposes a maintenance approach for Knowledge Organization Systems (KOS), a specific sort of terminological knowledge, based on information about the usage of KOS concepts. Focusing on Web-compatible KOS developed in the SKOS language, Eckert's contribution is a statistical framework and a treemap visualization of the collected usage data. The author also introduces an own life cycle model consisting of the following five abstract phases:

- Selection
- Creation
- Use
- Evaluation
- Modification

Figure 3.7 depicts how Eckert designs the interdependence of usage and maintenance in a similar fashion as we do in our approach which was primarily introduced in [LRH09].

The difference between Eckert's approach and ours is the level of detail of the proposed life cycle, the situation of our approach on top of ontology engineering best practices, our focus on structured Web data, and the different notion of usage.

3.1.8 Möller's abstract data life cycle

In [MÏ12] Möller designs an abstract data life cycle for the "arguably largest ... [data-centric] ... system in existence" – the Web of Data – which consists of the following 10 phases:

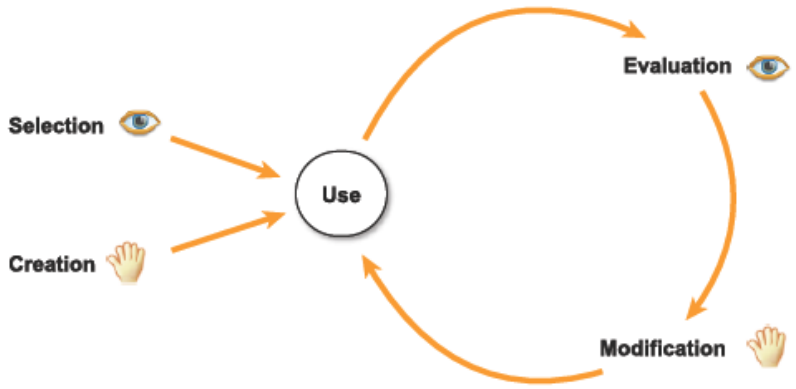

Figure 3.7: KOS life cycle as introduced by Eckert in [Eck12].

- Ontology development
- Planning
- Creation
- Refinement
- Archiving
- Publication
- Access
- External use
- Feedback
- Termination

It is noteworthy that Möller explicitly mentions the distinction of the ontology life cycle and the life cycle of instance data when a life cycle is dedicated to "a particular piece of data or metadata" and not "a complete system [including an ontology and associated instance data] as a whole". This distinction also reflects in the process model where ontology development is treated as an isolated process apart from the data life cycle as depicted in Figure 3.8. Möller's approach differs at this point completely from ours due to the fact that we regard the instance and the schema level of a data set integrative but still application-independent.

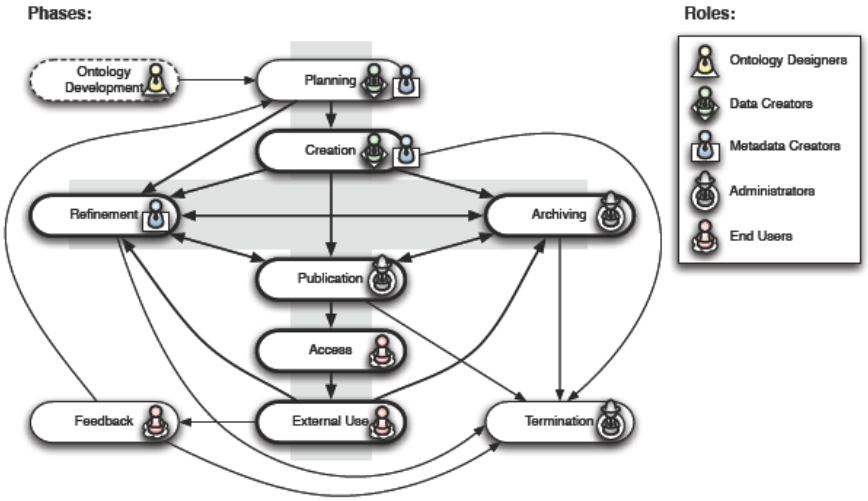

Figure 3.8: Abstract data life cycle as introduced by Möller in [MÏ2].

3.1.9 The LOD2 Linked Data life cycle

Resulting from the LOD2 European project the LOD2 technology stack aims at providing a fully-fledged tool framework to support the so-called Linked Data life cycle [ABL+12]. The Linked Data life cycle (depicted in Figure 3.9) is an iterative model of the following eight sequential phases which are aligned with ready-to-use tools typically applied in the Linked Open Data publication and consumption process:

- Storage/querying
- Manual revision/authoring
- Interlinking/fusing
- Classification/enrichment
- Quality analysis
- Evolution/Repair
- Search/Browsing/Exploration
- Extraction

The approach has an application-independent, developer-oriented focus being very detailed in describing the instantiation and orchestration the tools for the creation, interlinking and management of large amounts of instance data. Reverse, the schema level perspective is much less captured and the life cycle model does not reflect very detailed the triggers and effects of the individual process steps which are both motivating factors for our approach.

Figure 3.9: Linked Data life cycle model covered by the LOD2 technology stack from [ABL+12].

We acknowledge that several activities of the ontology life cyle which are concerned with maintenance, e.g. ontology evolution [NM00, NK04, FTN11], versioning [NM02], and tracking of changes [NKKM04] are very well researched. However, one can observe that the maintenance process and the composition of activities to be performed while an ontology is in use is much less described compared to other process steps in established ontology engineering methodologies and in data-centric life cycles. Reviewing literature even shows that the term ontology maintenance sometimes is used without defining it explicitly [ED07, GZT+11]. Most commonly maintenance is introduced as a post-development process step that interferes with usage in any unspecified way (see Figure 3.10).

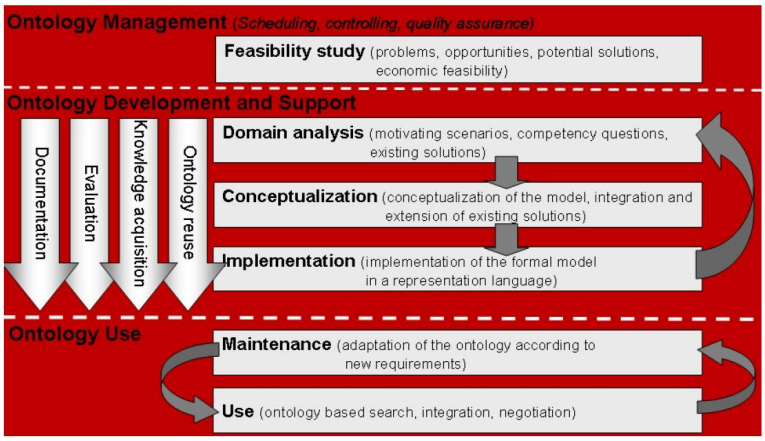

Figure 3.10: The general ontology development process of Simperl et al. [ST06].

These cost- and process-oriented problems which are responsible for the limited adoption of ontology engineering methodologies in real world projects were identified by Simperl et al. already in 2006 [ST06] and again in 2009 [SMB10]. They motivated us to conduct the survey which we will present in Chapter 5 and which confirms the limited adoption also in the specific context of structured Web data creation and provisioning. Hence, we decided to contribute an innovative arrangement of well-established activities, methods, and tools from ontology engineering with the requirements of structured Web data sets whenever possible instead of re-inventing the entire setup.

3.2 Web of Data usage mining

We already introduced that this thesis is concerned with Semantic Web usage mining in the sense as it has been introduced by Berendt et al. in [BHS02] and [B+04]. Early approaches which analyse the usage of semantic data in Web applications are [SAD+99] and [HMSS01]. However, consulting a recent survey in this field by Agosti et al. [ACDN12] shows that research on the mining of the usage of semantic data on the Web is rather sparse, while classical hypertext Web usage mining is continuously and heavily studied.

In 2010 Möller et al.[MHCG10] published a notable piece of work in the area of mining the usage of Linked Data sets. As a motivation the authors raise a set of challenges, namely *reliability*, *peak-load*, *performance*, *usefulness*, and *attacks*.

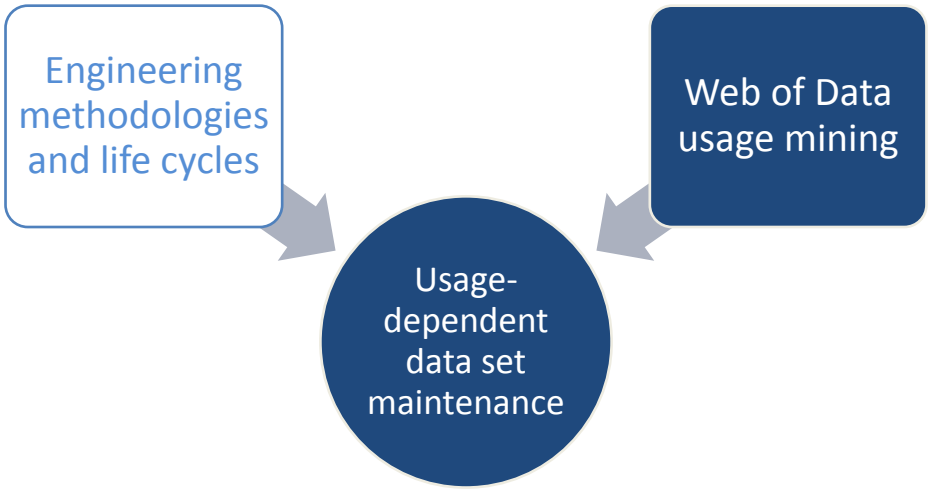

Figure 3.11: Web of Data usage mining – the second dimension of related work.

Möller et al. address these challenges by analyzing raw logs in order to learn about user clients, requested content types, and the structure of SPARQL queries but not in order to analyze the usage data on the level of basic graph patterns and the ontology primitives used in them. The authors aim at providing the data set provider with a framework for managing the quality of her data set alongside the above mentioned five dimensions. These dimensions are more or less an intuitive set lack any reference to the state of the art in data quality as it is the fundament of our approach.

Since 2011 the USEWOD workshop series pushed forward research on usage mining and analysis in the context of the Web of Data by providing a reference log data set from a number of well-established Linked Data sources which has been very actively used since its primary publication [B+11, BHH+11]. A large number of research papers originate from the USEWOD workshop series as well as the USEWOD data set covering those dimensions of the Semantic Web mining field which are out of scope of this thesis (e.g. Semantic Web content mining) [AFMPdlF11, EVS11].

Using an analysis of the syntactical and structural use of SPARQL in real-world scenarios to provide recommendations for index and store designers was introduced by Arias et al. [AFMPdlF11]. Such compact indexing of large data sets was also addressed in [lGBFMP11]. Related to these latter studies the optimization of data caching and prefetching based on real-world SPARQL queries from the USEWOD data set was presented in [LN13]. All papers deal with the optimization of the query performance which is part of the accessibility data quality category. We explicitly

excluded this category from our consideration in this thesis because this is highly depending on the low-level details of the data store.

The authors of [HJA12] develop a method to capture the cross-data-set browsing behaviour of human users. The goal of this is to enable a semantic description for accessed Web resources. SPARQL queries are out of scope of this study. This thesis is neither concerned with cross-data-set browsing nor do we put any regard on what we introduced as *resource centered access*.

As part of our usage mining approach we present a preprocessing algorithm for the in-depth analysis of SPARQL queries extracted from server log files in this thesis. We introduced this algorithm initially in [LRM11] and outlined the potential application areas for the resulting usage database. It is noteworthy that a very similar algorithm has also been designed by Elbedweihy et al. in [EMC+11]. The authors also propose to decompose SPARQL queries in order to derive insights into the atomic parts of queries and exploit this to draw network visualizations of data sets which are weighted depending on the usage of data set primitives such as resources and properties. As a matter of fact the approach in [EMC+11] differs from ours since it does not analyze the success of queries by re-executing them.

In [Rag12] the machine agent query behaviour on a single data set is studied with the goal to identify typical generic SPARQL query patterns applied. One dimension of the classification is the query result which means that, similar to what we do, queries are re-executed against the data set to which the source log file refers to. While Raghuveer re-executes extracted queries to analyze the result of successful ones, our special interest are failing queries and the atomic parts that caused the failure because those queries may disclose issues with a data set.

3.3 Data quality in the context of the Web of Data

Data quality in the context of the Web of Data is a very recent and heavily discussed topic. The data quality framework which we introduced before as the state of the art in data quality assessment conforming to user's requirements has inspired a number of works which we will outline in this section.

In a 2009 paper Hartig and Zhao propose to use provenance data to assess the quality of structured Web data [HZ09]. The authors declare that it is possible to adapt their approach to assess any other quality criteria generically. The feasibility is proved in an experiment assessing the timeliness data quality criterion.

In [Biz07] and [BC09] Bizer provided the fundamental work for this characteristic approach to data quality in the context of the Web of data which sets up on policies applied by the data consumer to filter out data of low quality. Figure 3.13 depicts

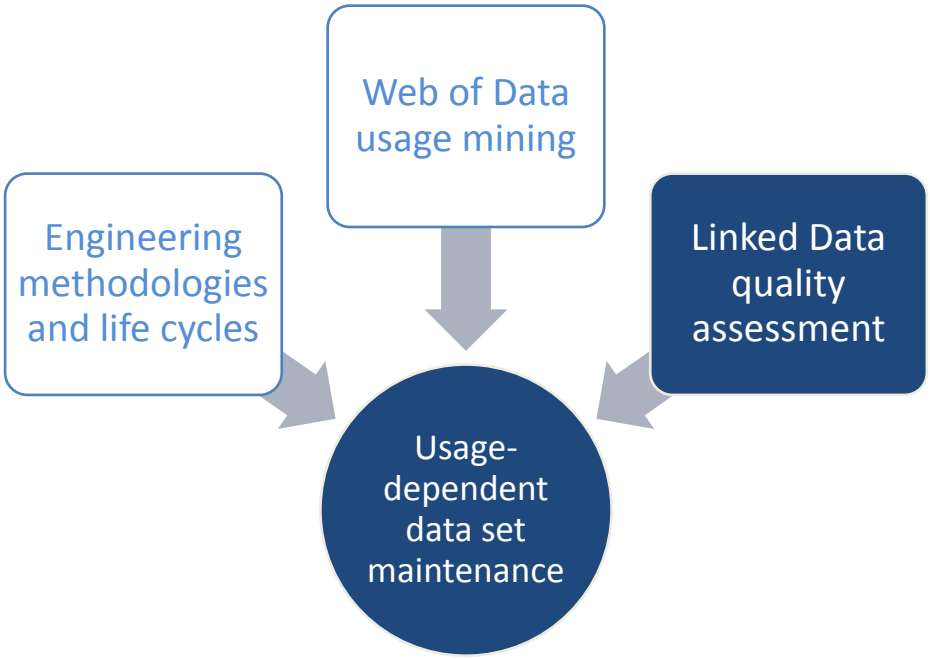

Figure 3.12: Data quality in the context of the Web of Data completes the three dimensions of related work.

the general process of this approach which we summarize as a *self-descriptive data quality assessment* because it heavily relies on the availability of quality-related meta information as are part of the data itself.

Figure 3.13: The WIQA filtering process as introduced in [BC09].

A more recent technical framework that provides Linked Data consumers with

an integrated data fusion and data quality assessment is SIEVE [MMB12]. The quality assessment module in SIEVE is a generic component which is configured by task-specific XML files which determine the metrics and respective scoring functions applied.

This kind of data quality assessment in the context of structured Web data is fundamentally different from our approach since it emphasizes the perspective of the user who wants to consume only data of high quality with reference to the intrinsic data quality dimensions especially which can be very well assessed by analyzing provenance information (e.g. information about the original source data or licensing information). In contrast we aim at supporting the data provider with a means to continuously maintain high quality data with reference to the evolving extrinsic conditions which are much more coupled with the contextual and representational data quality dimensions.

The work in this area has been further developed by Hoxha et al. in [HRE11] where a framework is presented which guides the provider of Linked Open Data towards publishing high quality data with reference to the community best practices. Again, this piece of work is bordered from ours as it can be seen as a community guideline for data publishers to provide retraceable self-descriptive data adequately but it lacks the contextual and representational data quality dimensions.

Most recently user-driven quality evaluation of DBpedia has been studied in [ZKS+13][2]. The authors describe and evaluate a tool which was set up in order to allow the user community to point out and fix errors in DBpedia data. While one has to critically regard the limited number of participants in this study (58 users evaluated 521 distinct resources), the crowdsourcing approach generally is an option for quality assessment and in fact it is one form of user feedback. However, it is questionable if it suits in the long tail and in large scale projects where the time the users spend with such a task has an economic impact which is why we decided to collect user feedback in the least invasive way by log file analysis. startatroot

[2] As of writing this thesis we were able to access a preprint version of this paper which was deleted once the paper has been accepted for publication at I-SEMANTICS 2013. The Website of the AKSW group links to `https://docs.google.com/spreadsheet/ccc?key=0AqePfpQUkw9xdEtnSkJlLXMwdTl4NzJxU04wQUpoU2c#gid=0` listing the results of this study.

Part II

Approach

CHAPTER 4

An empirical study of ontology engineering practices in the context of Linked Open Data

Adapting ontology engineering best practices in order to support data set publishers to maintain structured data sets based on usage analysis is the general problem statement of this thesis raised in Chapter 1. Literature review and observations of Linked Data projects significantly support this as an open research question indicating three issues: the missing adoption of well-researched results from the ontology engineering discipline in the context of Linked Data; the high dynamic and evolutionary fashion of Linked Data projects; less comprehensive schema engineering at the data set creation phase. In order to derive a research agenda for the development of methodologies for the creation and maintenance of structured Web data sets which conform to the requirements of data set publishers, we performed an analytical survey among a representative number of Linked Open Data publishers. In the remainder of this chapter we will describe the survey design, its results, and the conclusions with reference to the scope of our research.

4.1 Survey design

The *LOD Provider Survey* is an analytical survey about ontology engineering efforts in Linked Data publishing. It is concerned with the activities carried out by data set publishers related to the development and maintenance of ontologies for the data sets. The initial instance of the LOD Provider Survey refers to the data sets contained in the LOD Cloud release from fall 2010.

The survey was supported by a self-administered online questionnaire consisting of open-ended and close-ended questions. Open-ended questions do not impose any constraints on the form or the content of the responses, and are intended to capture general facts about the surveyed Linked Data publishing projects. For close-ended questions the answers of the respondents are limited to a pre-defined set. Typical examples of close-ended questions are dichotomous (yes/no) questions, multiple choice, as well as scaled (also called ranking) questions using various scale models. In our case we used domain-specific scales guided by the actual practice in ontology engineering. We did not allow multiple choices for all close-ended and scaled questions. We already mentioned some terminological mismatch between commonly used speech in ontology engineering literature and how the Linked Data research community refers to similar things. So we kept the number of open-ended questions as low as possible to avoid too much variance in the terminology of the interviewers which conforms to the best practices mentioned in [SB82].[1] A screenshot of the online version of the questionnaire, as it was presented to the survey participants, is shown in Appendix A.

The survey aimed to give a general overview of the usage of standard, widely accepted, as well as self-developed ontologies, and to document the activities performed by data providers from a methodological point of view. Accordingly, the questions can be classified into four distinct categories as follows:

Introductory meta questions about the data set and the provider 5 introductory questions were intended to capture the most important facts about the data set and its publisher: the name, affiliation, and contact information of the provider as well as the name of the data set and the address of a public SPARQL endpoint.

General questions about the ontologies used for the data set population The second group of 3 questions referred to the ontologies used to capture the structure of the data, and the extent to which they were reused or developed from scratch.

1. How many ontologies do you use to populate your data set altogether?

[1] Refer to [SB82] for a detailed account of questionnaire design principles.

2. Which are the ontologies you use? (give names or URIs)

3. How many of the used ontologies did you develop yourself?

Questions related to the self-developed ontologies A third group of 5 questions provided additional details with respect to the development process, the methodology followed, and the main characteristics of the ontologies in terms of size and knowledge representation language used in the implementation. Within the survey we adopted the term *vocabulary* for ontologies with no more than 150 concepts. We experienced this number as a good threshold differentiating between small but very generic ontologies characterized by flat hierarchies such as FOAF and much bigger data set specific ontologies featuring deeper taxonomies such as the DBpedia ontology. We are aware that vocabulary also is a common synonym for ontologies in general in some research communities.

1. How did you develop your ontology?
   - manually from scratch
   - ontology reuse and manual adaption
   - ontology learning
   - automatic generation from any semi-structured datasources
   - automatically derived from any relational database
2. Did you follow any methodology for engineering the ontology?
   - Yes
   - No
3. If yes, which one?
4. What is the size of this ontology in terms of the number of concepts?
   - Vocabulary (up to 150 concepts)
   - Small ontology (between 150 and 1000 concepts)
   - Mid-sized ontology (between 1000 and 5000 concepts)
   - Large ontology (more than 5000 concepts)
5. What is the complexity of these ontologies in terms of the usage of ontology language primitives?
   - RDF-S

- OWL-Lite
- OWL-DL
- OWL-Full

Three of the five questions in this category were closed-ended, two of these three even closed-scale. For the characterization of the development process we defined five different choices, corresponding to the main process models identified in ontology engineering methodologies such as NeOn [dCSdFB10] and which proved to satisfactory match the actual practice. To allow for a concise and effective analysis of the size of the self-developed ontologies, we allowed participants to choose between the four options mentioned above which rely on the understanding that an ontological entity corresponds to all classes, properties, fixed instances and restrictions defined manually through conceptual modeling in RDFS and OWL. These two classes of knowledge representation languages were also used as points in the scale defined for the question referring to the implementation of the ontology.

Question related to the evolution of the ontologies A last group of 2 questions described the evolution approach followed in that particular case, and allowed us to collect an informative description of the situations, which demand techniques and strategies to handle evolution at the conceptual and data levels.

1. Do you see any need to evolve these ontologies in the future?
   - Yes
   - No
2. Why?/Why not?

Feedback and free comments In the end of the questionnaire we provided a field for free text comments. The participants were able to give further information about their individual data publishing exercise which potentially were not covered by the prepared questions or to feedback about the survey design and methodology.

Table 4.1 summarizes the survey design and gives a clear overview of the distribution of open-ended, closed-ended and scaled questions as well as those questions which were asked for each of the self-developed ontologies repeatedly (marked with a *).

No.	Acronym	Topic	Response
Introductory meta questions			
1	PNAME	Name of the provider	open-ended
2	PAFFIL	Affiliation of the provider	open-ended
3	PMAIL	Mail address of the provider	open-ended
4	DNAME	Name of the data set	open-ended
5	DENDPOINT	public SPARQL endpoint	open-ended
Applied ontologies			
6	NONT	Number of distinct ontologies	open-ended
7	ONT	Name or URI of applied ontologies	open-ended
8	NSDONT	Number of self-developed ontologies among the applied ones	open-ended
Self-developed ontologies			
9	DEV*	How was the ontology developed	closed-ended
10	ANYMETHOD*	Any methodology followed	closed-ended
11	METHOD*	Which methodology followed, if any	open-ended
12	SIZEO*	Size of the ontology	scaled
13	KR*	Implementation language	scaled
Ontology evolution			
14	EVOLN	Need for an evolution strategy	closed-ended
15	EVOLR	Reasons for or against	open-ended
Feedback and comments			
16	COMMENTS	Free-text comments	open-ended

Table 4.1: Overview of the survey organization

4.2 Survey results

The initial edition of this survey was conducted from the beginning of October 2010 until mid January 2011. The validation of questionnaires as well as the data processing ended at the end of the first quarter of 2011. A search in well-known catalogues of Linked Data sets for the publishing organizations resulted in a list of 100 individuals, who were personally invited by email to participate in the survey. They represent 216 data sets, which covers the entire scope of the Linked Open Data Cloud diagram in October 2010. The response rate was 22%; 26 participants filled out the questionnaire, which after an initial curation of the responses resulted into 22 correctly filled

questionnaires. Among the interviewees one was representing 31, and a second one two data sets with the same properties (according to the questionnaire). This leads to an absolute number of 53 data sets covered by the survey, corresponding to 25% of all LOD Cloud data sets publicly available in October 2010. The distribution of the data sets across domains looks as follows: 40 life sciences;[2] three data sets containing bibliographic data; one data set each for music, media, library, geographic, education, and organizations; and 4 cross-domain data sets. DBpedia was initially represented twice in the survey by two different interviewees. As their answers were to a large extent compatible, we were able to keep this observation in the data collection by merging the two instantiations of the survey. It is worthwhile mentioning that the survey still maintains its relevance in the scope of the current version of the LOD Cloud diagram that was released in September 2011. This latest release contains 326 data sets, including all 53 data sets in our data collection, leading to a coverage of 16%.

Table 4.3 provides an overview of the proportion of reused and self-developed ontologies, according to the answers received for the second group of the questions in our survey. Almost half of the projects resorted to existing ontologies; this figure gives a significantly different picture than previous empirical surveys on the topic [ST06, SMB10] that attested the predominance of manual development and ontology learning in Semantic Web projects, and can be directly attributed to the application of Linked Data principles in data provisioning, including their emphasis on interlinking and the explicit recommendation to resort to existing vocabularies. Nevertheless, it is worthwhile mentioning that nearly every data set in the survey is described using more than one ontology, and that around 85% of the Linked Data providers developed at least one ontology as part of the publishing process.

63% of the self-developed ontologies were created manually from scratch, 32% were semi-automatically derived from a relational data base, and a remaining 5% were the result of applying ontology learning techniques. A large share of these ontologies (82%) contains no more than 150 concepts; 14% are of a size between 150 and 1000 concepts, while only 4% contain more than 5000 concepts. Most respondents (77%) noted that they did not explicitly follow a given methodology, while 23% of them affirmed that they are at least aware of the existence of such artifacts. In these five cases, three times the participants mentioned OntoClean, once the given methodology could not be explicitly named, and in the forth case participants considered the Linked Data principles themselves as methodological guidance for building ontologies.

73% of the respondents acknowledged the importance of ontology evolution, providing additional free-text explanations for the reasons they believe so. A compilation

[2]The 40 life sciences data sets are provided by 9 distinct publishers.

Data set	Domain
DBpedia	Cross-domain
Bio2RDF (representing 31 data sets in total as of October 2010)	Life sciences
MediCare	Life sciences
DailyMed	Life sciences
STITCH	Life sciences
SIDER	Life sciences
DrugBank	Life sciences
Diseasome	Life sciences
TaxonConcept, Geospecies (representing 2 data sets)	Life sciences
Ordnance Survey	Geographic
DBTune	Music
Freebase	Cross-domain
UB Mannheim	Publications
YAGO	Cross-domain
Pokedex	Media
data.dcs	Education
RAMEAU subject headings (STITCH)	Library
Semantic Web Dog Food	Publications
lobid-organisations	Organizations
lobid-resources	Publications
European Nature Information System (EUNIS)	Biodiversity
Lexvo	Cross-domain

Table 4.2: Data sets covered by the LOD Provider Survey

# of data sets covered	53
# of ontologies to populate data	67
average # of ontologies per data set	1,3
# of distinct ontologies to populate data	43
# self-developed ontologies to populate data	22
average # of self-developed ontologies per data set	0,42

Table 4.3: Ontologies used in the data sets

of the answers is given in Table 4.4.[3] To further yield meaning to these qualitative

[3] As a side comment, a participant considered evolution a non-issue due to the fact that the data set is using "standard ontologies".

answers towards devising a generic strategy for ontology evolution in the context of Linked Data, we classified the comments along two dimensions: (i) the factors that trigger evolution; and (ii) the phase or process step in the ontology life cycle which is effected by these factors. In the first dimension, we identified three classes of factors: (i) the need to be compatible with additional resources may lead to changes in the self-developed ontology in order to ensure it is a better fit to these resources and that links can be meaningfully defined (see answers 1 and 2 in Table 4.4); (ii) changes at the level of the data management infrastructure and technology components used in data provisioning (as reflected by answers 3 to 6); and (iii) evolution of the domain of the data set and of the requirements of the user community (as evidences by answers 7 to 15). Regarding the second dimension of our analysis, all factors are likely to affect the conceptualization of the underlying ontology, and the subsequent implementation. Additionally the last group of factors may lead to significant changes in the domain and scope of the ontology, which trigger a re-design of the overall ontology.

4.3 Survey discussion

The LOD Provider Survey yields empirical insight into the current ontology engineering practice as performed by the publishers of a representative share of the LOD Cloud, which amounts to a total of 53 data sets. As a core contribution, we were able to collect information about the proportion of reused and self-developed ontologies, and gain a better understanding of the usage of existing ontology engineering methodologies, and of the particulars of ontology reuse and maintenance, as two important aspects of the ontology life cycle that are essential in this new class of scenarios.

In the following we elaborate on the core findings of our analysis, which we organize into four categories. The first category *Vocabulary bootstrapping* refers to the actual ontology engineering practice on the Web of Data, which is characterized by a combination of frequent reuse of small ontologies and learning of individual vocabularies from existing (semi-)structured sources. The second category is concerned with the availability and impact of methodological guidelines from ontology engineering in the data publishing process. As the title of this category suggests – *Methodological underachievement* – existing ontology engineering methodologies require major revisions in order to ideally accommodate the requirements of Linked Data application scenarios from a technical, use case, and community perspective. A third category, titled *Shallow ontologies*, discusses the implications of the fact that most ontologies built and reused in Linked Data context are limited in size and con-

No.	Answer
1	"More vocabulary mappings"
2	"Mappings to more ontologies. Now only mappings to some others."
3	"Changes based on recommendations and discussions by various standards groups."
4	"User enhancement through wiki due to coverage of topics. The DBpedia ontology might have to be extended to cover additional information or amended when DBpedia changes."
5	"Because deriving predicate names from databases schema works, but actually modeling the real life concepts in each database would be more useful."
6	"The DBpedia ontology might have to be [...] amended when DBpedia changes."[4]
7	"These ontologies are used by other data sets and a wider community - they will need to adapt to its evolving needs."
8	"The DBpedia ontology might have to be extended to cover additional information..."[10]
9	"In the future I may need to add new terms into the ontologies. The ontologies were constructed to support the linked data being used and as that data evolves new predicates and classes made need to be added to the ontology."
10	"YAGO is constantly being improved and expanded."
11	"Move properties and classes will need to be included if more data from other sources is added."
12	"Simplification, discussions in community, change in scope, etc."
13	"We have more data we want to expose"
14	"Data-driven need for new properties and classes."
15	"User enhancement through wiki due to coverage of topics."

Table 4.4: Reasons for data set evolution in Linked Data provisioning.

ceptual complexity. While this state of affairs might be due to historical reasons, as frequently reused ontologies, which are de facto small and straightforward, tend to be reused even more, the question remains whether such conceptualizations will satisfy the needs of data providers across vertical domains, and, related to it, whether the study of Linked Data and associated processes might lead to new insights about the relationship between reusability and complexity in ontologies. The last category,

termed *Maintenance incertitude*, is related to ontology maintenance and evolution, in particular to the limited preparation of schema-level interlinking, and to the lack of awareness about the importance of mappings in general, which the majority of our survey respondents revealed.

Vocabulary bootstrapping *Vocabulary bootstrapping* is our characterization of the current practice in Linked Open Data projects to create and orchestrate the necessary subsets of ontologies which are later used for the instance data population. As mentioned earlier, ontology reuse has reached a higher significance in the context of Linked Data compared to former studies in ontology engineering. This holds since most data sets are populated applying more than only one ontology and 49% of these ontologies are reused. However, we also found out that 68% of LOD data set publishers also develop individual ontologies to populate their data. Reuse of such ontologies is actively performed by the developer if multiple data sets are populated using them. But ontologies developed by one data set publisher are not reused by other parties. This wide spread practice of individual ontology development is even more interesting in combination with the domain coverage of our survey. 77% of the data sets represented in the LOD Provider Survey serve data about the life sciences domain.[5] We conclude that data set publishers develop individual domain ontologies even though a significant number of data sets covers the same domain. The availability of data and ontologies in domains such as 'City', 'Life Sciences' and even 'Space-Time' offers an ideal environment for testing and trialing a new generation of alignment and interlinking techniques which are on their way; nevertheless, especially in general-purpose domains such as geospatial information or time a community-driven approach to standardize these representations, or at least to create high-quality mappings between the most important candidates will probably prove more effective than the essentially bottom-up approach that was followed in the Linked Data community so far. While this lead to the establishment of a few ontologies or vocabularies as de facto standards in their domains - take, for instance, FOAF - other areas seem to require different strategies in order to reach a similar level of consensus and, in the long run, facilitate the development of useful applications consuming Linked Data.

Methodological underachievement *Methodological underachievement* stands for the missing adoption of ontology engineering methodologies in the context of Linked Data publication and management and for the limited preparation of tasks related to structured creation and maintenance of data sets. 77% of the LOD vocabularies were developed without the application of a dedicated methodology. In addition a share

[5] This share summarizes data sets tagged with "life sciences" and "biodiversity" in CKAN. Please note again that these 77% are 40 data sets in total and that they are provided by 9 distinct publishers.

of 9% of publishers asses themselves to apply some methodology but none of those which are well-known from the ontology engineering research. This lets us conclude that altogether 86% of the dataset publishers are not yet fixed to any elaborated engineering process and do not know that ontology engineering methodologies could be applicable in the context of Linked Data. In order to close this gap from an ontology engineering perspective it is necessary to respect the fact that the Linked Data community is a rather young one, much more driven by practical Web development and real-world requirements than the ontology engineering discipline ever was. One of the consequences of this mismatch is the terminological gap addressed briefly earlier in this article, but also the reasonable expectation that heavyweight guidelines and comprehensive glossaries as provided by classical ontology engineering methodologies will hardly ever find adoption in the context of Linked Data. Lightweight executive summaries complemented with guidelines, best practices and examples referring to specific tools and programming environments are likely to match the expectations of this community to a greater extent. In terms of the activities which need to be covered by future methodologies, our analysis revealed that if a methodology is intended to be applicable in the context of Linked Data it should combine ontology learning, reuse, and manual adaptation of ontologies.

Shallow ontologies The term *shallow ontologies* refers to the limited size and complexity of ontologies underlying data sets in the LOD Cloud. Our survey results that 82% of the self-developed ontologies underlying Linked Open Data do not contain more than 150 ontology entities and are rather "shallow" - following the notion of "shallow ontologies" as *"relatively few unchanging terms that organize very large amounts of data"*[SBLH06]. This observation has been discussed before in, e.g., [AL10] and is now firstly confirmed based on empirical observations and qualitative interviews. In their paper Auer and Lehmann argue in favor of a semi-automatic semantic enrichment as a post-integration process step supported by ontology learning techniques to achieve a balance between acceptance by Linked Data publishers and support for advanced semantic capabilities such as reasoning.

Maintenance incertitude The *maintenance incertitude* reflects the multiple risks of a limited provision for tasks which need to be performed when an evolution step of a data set is indicated. The Linked Data community relies on the self-organized fashion of the Web which is expected to result in data convergence in the long-tail. However, our study counters that ontology evolution already in the context of a single data set effects effort-intense processes which are well-established in the ontology engineering discipline. If the Linked Data community misses to regard this it is logical that underestimated efforts and costs will disappoint adopters outside

academia and throw back the current promising trend.

4.4 Requirements for adapting ontology engineering in structured Web data set publication and management

Our survey empirically evidences that Linked Data providers develop and maintain individual ontologies for data set publication. Contributions of the ontology engineering discipline are sparsely applied in Linked Data projects. The evolution of data sets is majorly driven by external conditions, such as new remote data sets to link to or a new community guidelines with respect to the standard ontologies for data sets covering a specific domain. We derive the following three tipping points which should be in focus of methodologies for continuous structured Web data publication and management and which guide the design, development, and evaluation of our approach in the remainder of this thesis.

Vocabulary reuse and remixing: Structured Web sets are generally populated by **application of subsets of more than one ontology**. The data publisher assesses individually those parts of her data that fit to a domain view shared with publicly available vocabularies and those for which a small vocabulary needs to be built. A core requirement is supporting **criteria and methods for the identification of relevant ontologies** which are applied by other data sets in a domain of interest.

Evolutionary instance and schema level mapping: The most important evolution step within the life cycle of structured Web data sets is the **creation of links on the instance as well as the schema level**. This requires methods that allow the data set publisher to identify the relevant entities in other data sets to link to. Data consumers or third parties require **data set profiling and summarization** techniques which enable a quick overview of the domain of discourse of a data set and scalable algorithms to compute links between selected data sets.

Lightweight guidelines and ex-post creation of formal knowledge: In order to keep the **focus on publishing raw data** instead of creating a high degree of conceptual knowledge a priori, structured Web data publication and management methodologies need to be lightweight. **Involving user as well as community feedback** is a core requirement for establishing a reputable data set.

CHAPTER 5

A usage-dependent life cycle model for structured Web data sets

In Chapter we outlined a research agenda for ontology engineering in the context of Linked Data which we introduced as a representative technique for the publication of structured Web data. Furthermore we raised requirements which should be regarded when adapting engineering methodologies to suit to this specific use case of ontologies. The Corporate Ontology Life Cycle Methodology (COLM) is an ontology life cycle for evolving ontologies which are incorporated in an infrastructure of applications which are created, evolving, and retired in an agile fashion [LRH08, LRH09]. The model separates tasks to be performed for promoting and monitoring ontology usage from those which are related to the development of the conceptual model for the domain of interest. Feedback about the ontology usage supports the creator of an ontology by managing and performing maintenance activities aligned with user or application requirements and results in high transparency of maintenance efforts. In this chapter we describe the adaptation of this life cycle model in the context of structured Web data. We are going to describe the high-level view of the usage and engineering cycles in this context first before we give a detailed description of the key tasks and sub-activities, the input, methods and tools, as well as the output of each single process phase.

5.1 High-level introduction

As depicted in Figure 5.1, the two-parted cycle consists of seven phases which refer either to the outer cycle representing an environmental process which consists of engineering tasks familiar to experienced ontology developers from other fully-fledged ontology engineering methodologies (selection/development/integration, validation, and evaluation) or to the inner circle, which refers to the data set usage and which supports the creator or administrator of a data set by analyzing the necessity of changes (schema deployment, instance population, feedback tracking, and reporting).

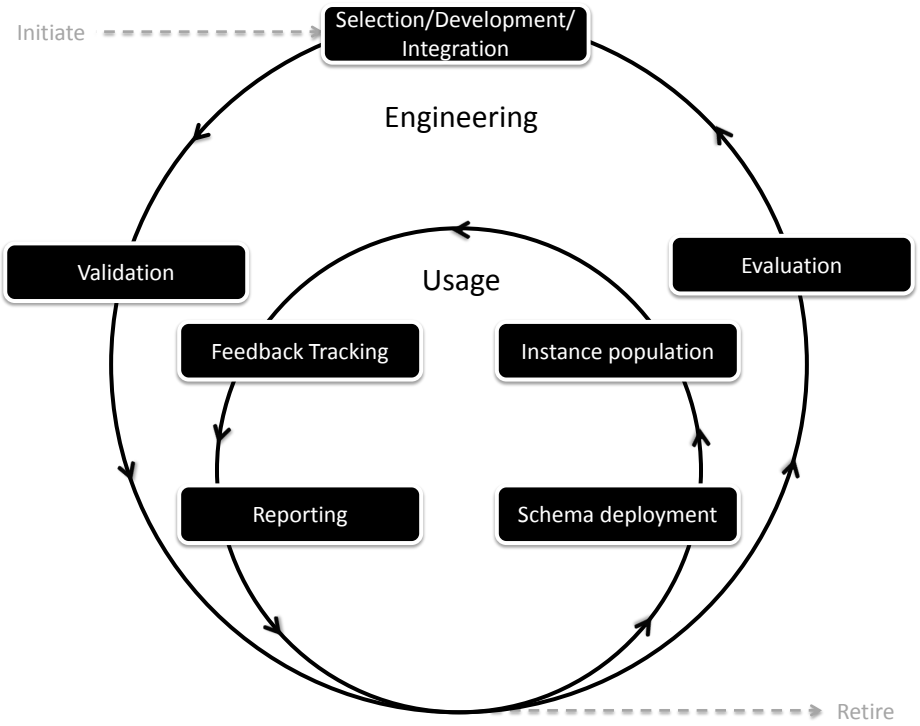

Figure 5.1: High-level visualization of the usage-dependent life cycle.

The life cycle starts when a data set publication process is initiated with the *selection/development/integration* phase. That means to start the knowledge acquisition and conceptualization, to re-use or re-engineer existing ontologies, or to commission a contractor to develop one or more ontologies which represent the conceptual knowledge needed to populate the data of choice. The result of this phase is a set

of ontologies, which is *validated* against internal requirements, such as the individual domain view, as well as external requirements, e.g. community guidelines for publishing structured data of adequate quality. At the intersection point between the engineering and the usage cycles the decision is made whether the orchestrated ontologies suite the requirements or not. If this is approved the ontologies are *deployed* to the structured data repository, e.g. a triple store. Then the instance data is *populated*, which means that a manual, semi-automatic, or automatic process for instance generation from structured, semi-structured or unstructured sources runs up. During the *feedback tracking* phase, feedback from and/or behavior of users is recorded. A *reporting* based on the collected feedback is performed at a point of time when the data set publisher is interested in a detailed analysis of the usage of her data set or when some realtime statistics on the tracking logs indicate this necessity. All feedback information, which was collected within a dedicated time period, is analyzed along a set of quality dimensions which may indicate to leave the usage cycle and *evaluate* the revealed weaknesses of the currently applied ontologies as well as change recommendations derived from the feedback. This point may also be reached, when the validation phase results that a set of ontologies is weak or improper with respect to the specification or requirements. The life cycle starts again with the implementation of the results of the evaluation. If either the reporting or the validation reveals that a data set is not used anymore or contradicts community consensus (from a technical, conceptual, or factual point of view) it is also possible to stop serving and maintaining it – the life cycle is over. The phases of this life cycle model only define the sequential order in which the individual steps need to run up and how the input of some phases depends on the output of others. Phases may overlap, which is perfectly shown by the example of providing a live version of a data set and tracking feedback continuously while the subsequent phases for the improvement of the next version are already in progress.

5.2 Detailed process description

In the last section a high-level introduction into COLM was presented which gives an overview about how this usage-dependent life cycle is intended to run up. In this section we present a detailed look at each phase. Figure 5.2 depicts the life cycle model as a flow diagram emphasizing the decisive points between the engineering and usage phase at which the data set publisher determines whether (a) the development process resulted in an approved vocabulary setup, so that the data set can be populated and used or (b) the analysis of the usage and user feedback does not indicate any necessary change on the vocabularies in use, so that the data set can

stay in use as it is or be extended by further instances conforming to the current vocabulary setup.

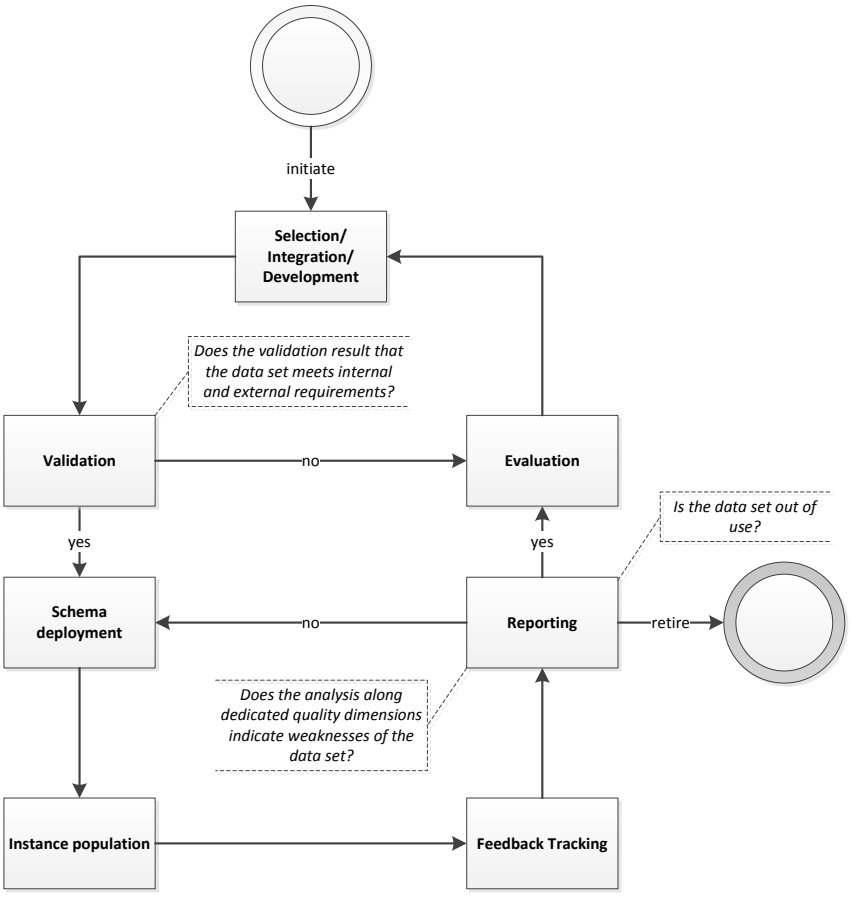

Figure 5.2: Sequence diagram of the usage-dependent life cycle model.

In the remainder of this section we will introduce a flow diagram for each of the seven phases accompanied by a general description of the phase and the sub-activities which are invoked, the input of the phase, supporting methods and tools, and the output.

5.2.1 Selection/Development/Integration

The first step a data set publisher is confronted with in each iteration of the data set life cycle is to decide about the instance data to expose, which roughly defines the do-

main of interest. Subsequently it is necessary to decide whether to perform all tasks for the development and selection of the necessary vocabularies for the domain of interest internally or to externalize this to a specialized contractor. The former results to be concerned with common ontology engineering activities such as specification of competency questions, domain knowledge acquisition, and conceptualization. Otherwise this tasks are provided by the service provider with explicit skills in ontology engineering for data set publication. In both cases the reuse and integration of existing vocabularies is an important aspect for efficient and cost-effective development but also a requirement for the publication of community-conform Web data.

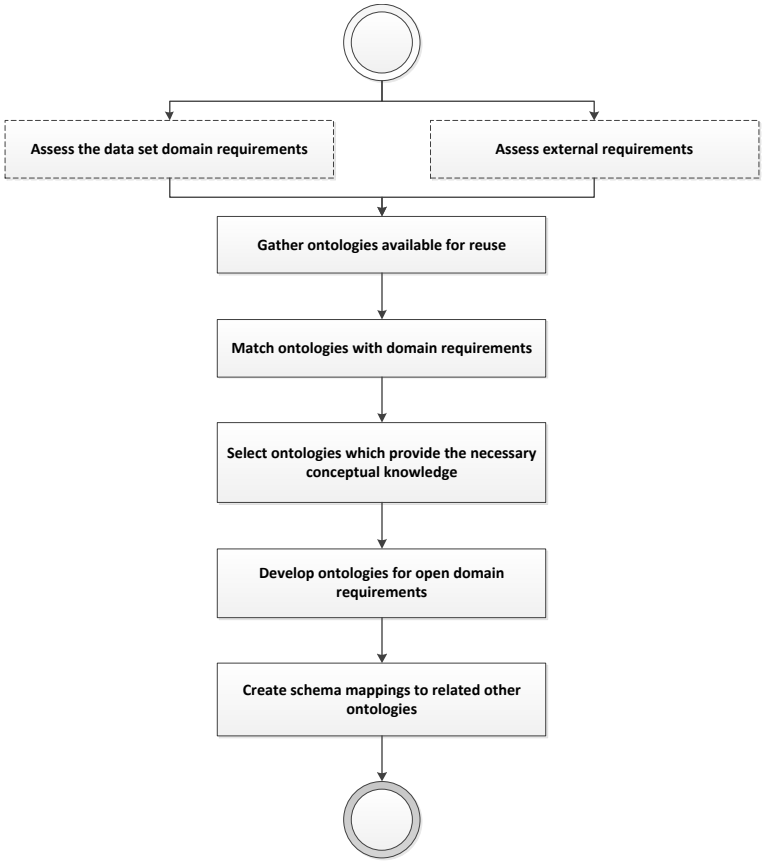

Figure 5.3: Life cycle phase I: Selection/Development/Integration

Input

In case of the initial development of a completely new data set, the input of this phase consists of the internal and external requirements. The internal requirements are

- data sources which do not conform to the principles of structured Web data but which contain important source data and
- requirements of applications which will work upon the final data set.

Most prominently, the latter may be a set of queries the applications will perform against the data set. External requirements are

- the standards and principles for structured Web data publication [BL06], but also
- lightweight guidelines for providing valuable structured Web data within an environment which involves a community of data set publishers.

An example for such lightweight external guidelines are the recommended standard vocabularies and the requirements for making Web data self-descriptive in order to be publicly accepted as a publisher of open Web data [HHP+10]. Both, internal and external requirements, are subject to change if standards or best practices evolve. Such changes are detected within the evaluation phase and effect the development in later iterations of the life cycle when the output of the evaluation is added as an input of the development.

Methods and tools

As a consequence of the importance of ontology reuse, the most important tool for this initial phase are tools that help to find, understand, and evaluate vocabularies which are commonly used by other data set providers. Additionally we propose two types of tools when it is necessary to develop an individual vocabulary or adapt a reused one: First, tools for rapid ontology prototyping by learning fundamental concepts from folksonomies, tag clouds, or wiki categories. Second, expert-oriented tools, such as Protégé or the NeOn Toolkit, which allow the manual adaptation of initially auto-generated or reused ontologies.

Output

The results of this phase are

- a documentation of the data set infrastructure containing information about URI schemes and access methods,
- a documentation of a dedicated set of primitives from one or more vocabularies (as mentioned before we will call this the vocabulary setup in the remainder of this thesis),
- a set of schema mappings, interweaving primitives from two different vocabularies through the owl:equivalentClass or owl:equivalentProperty properties respectively, and
- all sources of the vocabularies from which the primitives for the data population were taken and to which the schema mappings link.

5.2.2 Validation

The validation of the vocabulary setup is a decisive phase following the development or selection of a prototype version. The data set publisher is required to perform a final check of the consistency of the conceptual modelling and the conformance to the documented internal and external requirements. With respect to the consistency of the conceptual modelling it is necessary to check each of the selected vocabularies individually, the integrated set of vocabularies, and the integrated set of vocabularies including all further vocabularies linked by schema mappings. While the validation of the internal requirements may be a quick internal process it is recommended to perform an external evaluation of conformance towards the external requirements, e.g. by requesting a public review of the proposed setup from a community of experts. This allows to react to recently evolving community best practices and trends in an agile way before publishing data which gets rejected by the community due to formal or in-formal errors or misconception.

Input

The validation of an vocabulary setup requires

- all output artifacts of the selection/development/integration phase and
- a group of experts from the community which is involved in the environment to which the data set will be deployed.

Methods and tools

The validation of the different individual vocabularies and the integration of those requires a reasoning engine to be applied. The validation of external requirements

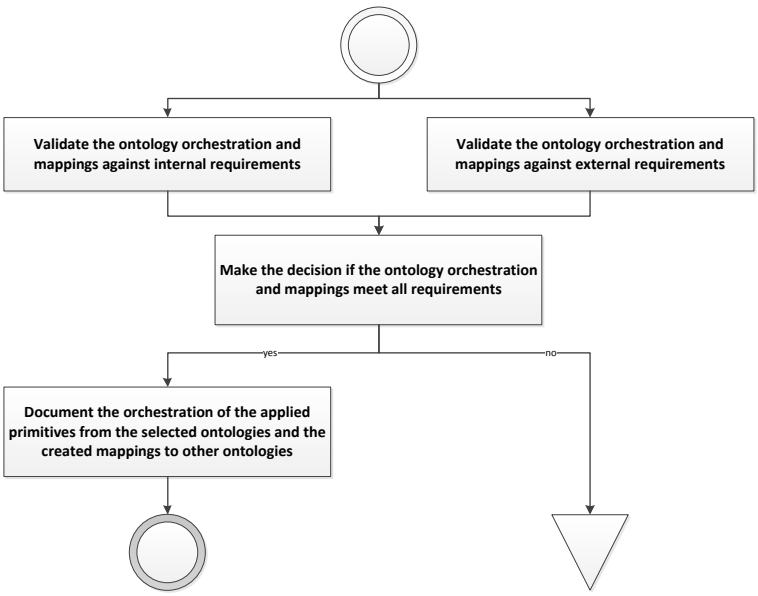

Figure 5.4: Life cycle phase II: Validation

may be either done by providing a publicly available documentation of the vocabulary setup for the data set and invite the community to comment on the approach via well-established mailing lists or by personal interviews.

Output

If the vocabulary setup and the data set infrastructure is validated successfully all output artifacts from the preceding phase are handed over to the schema deployment phase. In case the validation fails a documentation of the failures is added and everything is provided to the evaluation phase.

5.2.3 Schema deployment

Our life cycle model differentiates between the deployment of the vocabulary setup and the population of the instance data. This distinction and the order to deploy the schema first is due to the fact that it becomes more and more common to provide structured Web data sets with a live update facility which continuously updates the instance data from any legacy data source (e.g. DBpedia live update, Linked Geo Data). "Deploying" generally means to load data into a data repository which is accessible conforming to the principles of the Web architecture. Thus, schema

deployment means loading a Web-compatible serialization of the vocabulary setup into such a repository. Structured data sets are in use by various application which are rapidly created, evolve in an agile fashion, and may get retired immediately. The same applies for other structured data sets and vocabularies a data set may but does not need to refer to on the instance and the schema level. Beginning the deployment of a new data set version by exposing a new conceptual base can have various side effects outside of the control of the data set publisher. Changes need to be documented in a change log so that other data set publishers can react accordingly and adapt mappings and other data links or stop using the latest version of a shared vocabulary.

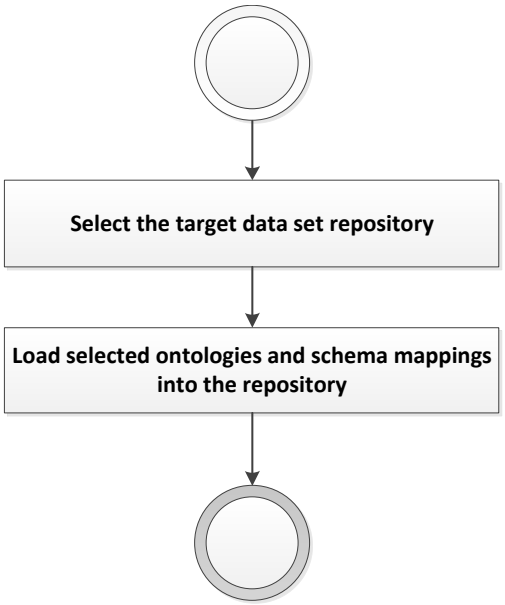

Figure 5.5: Life cycle phase III: Schema deployment

Input

The input artifacts for the schema deployment are

- all sources of the vocabularies from which the primitives for the data population were taken,
- all RDF triples mappings primitives from the selected vocabularies to other external ones,

- optionally all sources of the vocabularies to which the mappings link to, and
- a running data repository instance.

Methods and tools

During the selection/development/integration phase the decision was made how the data set infrastructure looks like. Independent from whether the data set is materialized at a dedicated point of time or if it is generated by a dynamic wrapper around a legacy data source, the most common tool setup for structured Web data is a data store (e.g. an RDF triple store) in the backend of a structured data endpoint that retrieves the data from this store and serves it on the Web in a standardized fashion.

Output

At the end of this phase the repository is running and serves the new schema of the data set.

5.2.4 Instance population

If an vocabulary setup has been deployed it is possible to populate the instance data of a data set. Existing instance data relying on an old vocabulary setup version has to be upgraded to suit to the new one. Conforming to the Web architecture it is only possible to serve one version of a data set for direct access to URI-identified resources. Thus it is recommended to serve the actual one and to provide dumps of old versions for download and local mirror instantiation. As mentioned before it is common to serve structured Web data by mapping or wrapping any conventional data source which are dynamically and continuously updated. This may be provided by an on the fly materialization and serialization of the instance data or by a short-cyclic live update mechanism. Consequently, the instance population phase does not need to be regarded as a finite process ending with a finite data set but as a continuous one. Data set providers need to know that the on the fly materialization and serialization allows the usage of the data set directly after the vocabulary setup is deployed to the mapper or wrapper infrastructure. In contrast a delay needs to be taken into account when instance data is first materialized offline and then loaded into a Web data repository.

Input

For instance population the following input is needed:

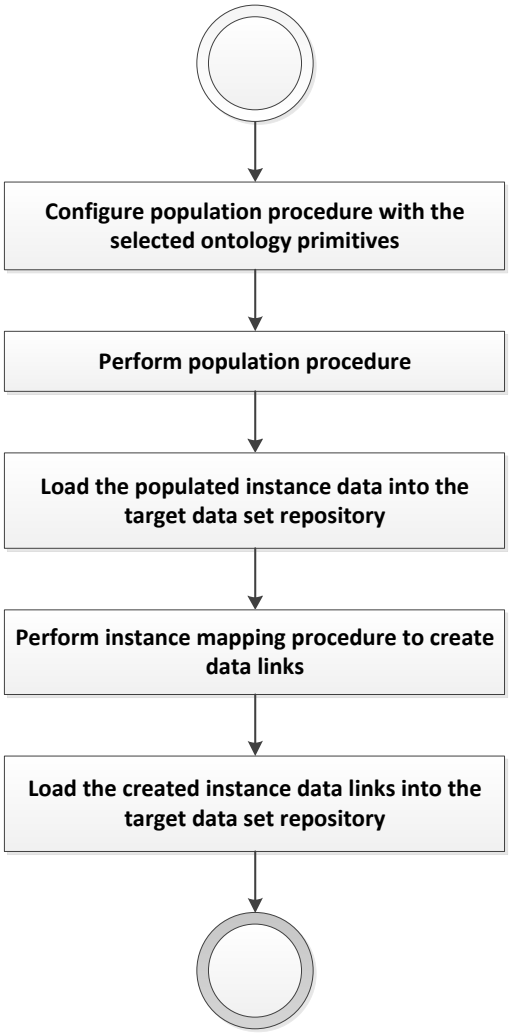

Figure 5.6: Life cycle phase IV: Instance population

- All sources of the vocabularies from which the primitives for the data population were taken and

- the data repository containing the loaded vocabulary setup.

Methods and tools

It is possible to extract structured Web data sets from semi- or unstructured data sources by extraction algorithms [SKW07, ABK+08] but it is also very common to build wrappers around relational databases. In both cases it is necessary to configure a mapping to conform to the vocabulary setup. Both approaches allow the cyclic generation of new versions of the data set based on actual source data. It is not very common but possible to create structured data by manually encoding it. Once a materialization of structured data in form of source files exists most data stores provide a batch import for such.

Output

The result is a running and accessible data repository instance serving the schema and the instances of the current data set version.

5.2.5 Feedback Tracking

During the feedback tracking phase the data set publisher collects user feedback with reference to the data set version which is publicly in use. This may include implicit (e.g. logs of client requests against the data set) and explicit (e.g. human user feedback statements) feedback. Intuitively the collection of implicit feedback is easier to achieve since no further action by a necessarily human user is required. This points to the noteworthy contrast between feedback tracking in the classical document-centric Web and the mining of the usage of structured Web data. It is possible to integrate scripts into Web documents which allows a detailed page visitor tracking or to provide feedback forms for open-ended or scaled questions that aim to understand the human user's perception of the offered content or service. While it is a promising feature that classical Web log analysis is feasible in the context of structured Web data, it is a matter of fact that it is impossible to embed tracking scripts into raw data requested or queried by and delivered to a Web client. Additionally, Web data is very often consumed by a machine or service and only seldom by a human user directly which makes it complex to ask for explicit feedback. If the evolution of the data set is highly dynamic and the availability crucial it is likely that further feedback is tracked while a snapshot has been taken for deeper inspection in the subsequent phases of the life cycle.

Input

The feedback tracking depends on a distinct data set version which is publicly in use.

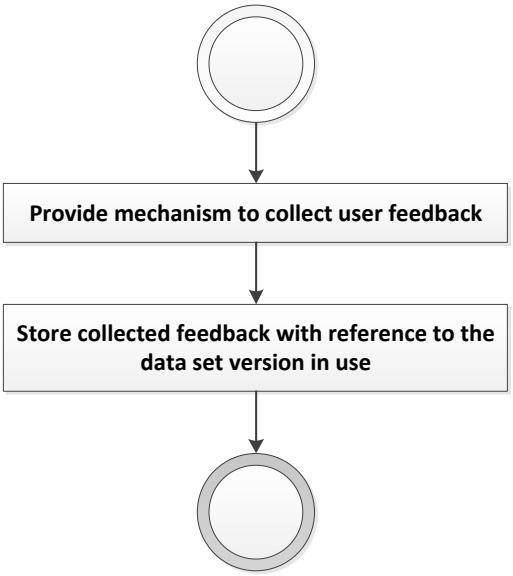

Figure 5.7: Life cycle phase V: Feedback Tracking

Methods and tools

As mentioned, the most lightweight way to capture user feedback is by consulting Web server access log files. This approach is not only least encroaching on the user but also does not require any additional software components beside the Web server which is already set up for serving the data. If not provided by default the only change is to configure the Web server for proper access logging. If the data set is regularly accessed via a human user interface provided by the data set publisher herself (e.g. [1]) it is possible to integrate more sophisticated implicit tracking mechanisms, such as scripts which report user input, navigation, and browsing behaviour to an additional tracking service (e.g. [2] which generates usage statistics and analysis in realtime. If a user interface is provided and frequently used it is also possible to integrate feedback forms and polls in order to ask the user for explicit feedback about the way the data set is provided and the data it contains.

Output

The feedback tracking outputs

[1] `https://github.com/kurtjx/SNORQL`
[2] `http://piwik.org/`

- a selected set of logs containing the collected feedback at a dedicated point or throughout a period of time and
- a dump of the data set state which was in use during the selected tracking period.

5.2.6 Reporting

In the reporting phase data set providers assess a set of quality criteria based on the collected feedback from the users. The overall goal of this process step is to find out whether maintenance activities are indicated. The indicators may refer to either the instance or the schema level of the data set. As it is state of the art in research on data quality in information systems, the orchestration of data quality dimensions and the weighting of each is highly individual. Nonetheless, the empirical approach of data quality research allows insight on the importance of specific dimensions for the data consumer who is an important entity in our usage-dependent life cycle. Thus, the data publisher should pay special attention to these quality dimensions.

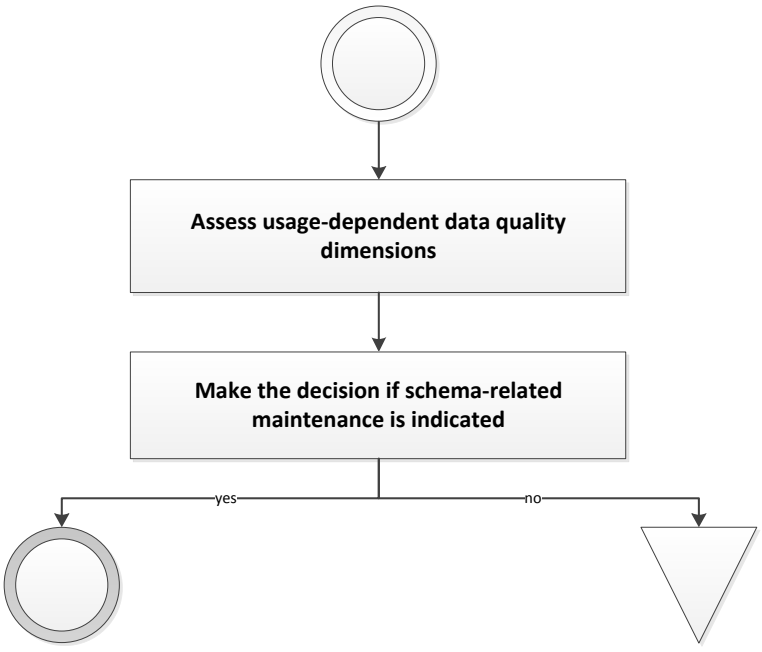

Figure 5.8: Life cycle phase VI: Reporting

Input

The input of the reporting phase are

- all output artifacts of the preceding phase and
- a set of data quality criteria to be assessed.

Methods and tools

The usage-dependent assessment of the selected data quality criteria requires tools which enable a preprocessing of log files in order to extract the relevant information, such as user queries. This yields an integrated usage feedback database which is the source for the computation of statistics by means of a statistical programming language for example.

Output

The reporting phase hands over to the subsequent phase

- all output artifacts of the preceding phase and
- a documentation of the assessed data quality criteria.

5.2.7 Evaluation

The result of the reporting phase is used to evaluate the current vocabulary setup and the respectively populated instance data. On the basis of the tracked information the data set provider can detect missing or obsolete facts, concepts, and relationships within the own data set but also links to other data sets on the schema as well as the instance level. By that it is possible to recommend changes based on the usage of the current data set version. However, the manual review of the data set as well as the internal and external requirements is also an important activity in order to generate the input for the development activities at the beginning of the next life cycle iteration.

Input

If the evaluation runs up succeeding the reporting phase the input consists of

- all output artifacts of the reporting phase and
- all output artifacts of the validation phase.

If this phase is reached after a failing validation the only input are the output artifacts of the validation phase.

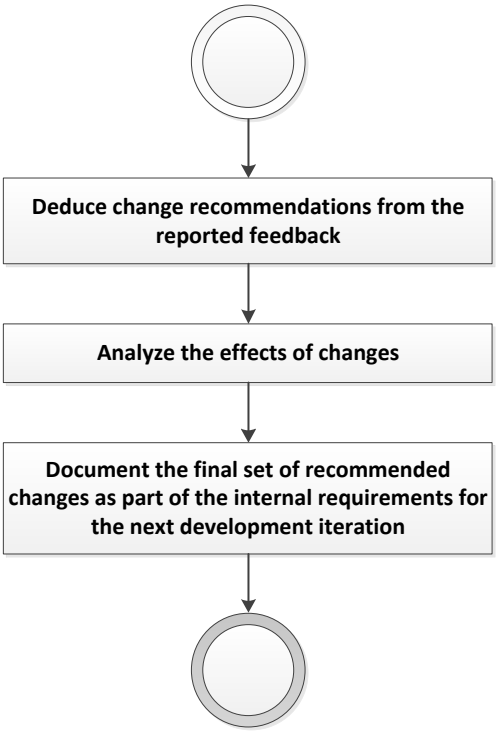

Figure 5.9: Life cycle phase VII: Evaluation

Methods and tools

The focus of the evaluation in our usage-dependent approach is on exploiting the collected feedback for recommending changes on the instance and the schema level of the data set. Consequently, this phase requires methods for identifying interesting patterns within the feedback which refer to blind spots within the data set. A number of well-researched and commonly applied data mining methods can be consulted for this purpose. Most prominently association rule mining or sequential pattern mining help to identify relations between resources which may not have any relation before. Furthermore, the assessed data quality criteria also allow a statistical insight in what maintenance activities should be performed in order to improve the data set.

Output

The results of the evaluation are

- a set of RDF resources and statements which are recommended to be added to

the data set

- a set of schema mappings which are recommended to be added to the data set
- a set of queries the data set should provide data for,
- a documentation of new or obsolete internal requirements, and
- a documentation of new or obsolete external requirements.

CHAPTER 6

Web usage mining for structured Web data set evaluation

As mentioned earlier in this thesis many process steps of our life cycle methodology are very common in established ontology engineering methodologies and well-captured with methods and tools except the feedback tracking and reporting phases. In this chapter we introduce an approach to track and report feedback about the usage of structured Web data sets which provide query pattern access. This completes the set of necessary tools to instantiate our methodology entirely. Concretely we developed an algorithm for the preprocessing of log files of Linked Data sets which provide a SPARQL query endpoint and a method to assess a data set's quality by usage-dependent measures.

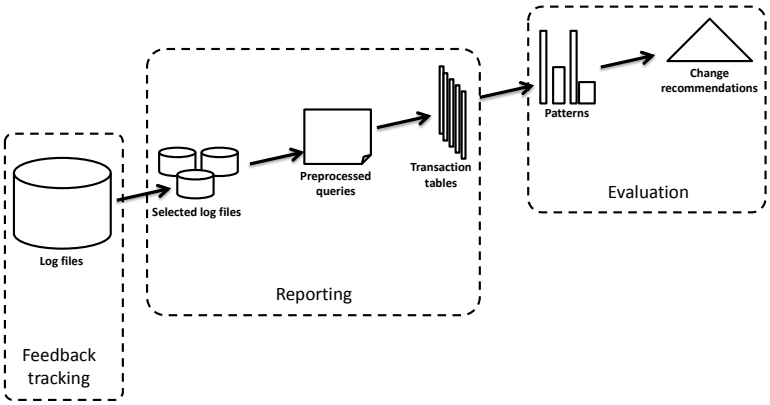

Figure 6.1: Instantiation of the KDD process of Fayyad et al. [FPSS96] in the context of usage-dependent data set maintenance.

6.1 General process

Our query log analysis approach is based on the established generic KDD process introduced in Chapter 2. Figure 6.1 depicts how specific phases of our life cycle refer to the KDD process.

The basic data source are Web server log files of Linked Data endpoints which conform to the the common log format [W3C] and which contain information about the direct access to served resources via their URIs as well as SPARQL queries. **Selecting** one or more log files that contain usage data of a single day or of a time period constitutes the target data for the reporting. Furthermore it is necessary to either have access to the data set the selected logs refer to or to set up a local mirror. The latter has to be preferred in order to avoid the pollution of the logs of productive data sets or to influence their performance.

In the **preprocessing** and **transformation** steps only the relevant parts are extracted from the logs and lifted into a structure and richness that is required for an in-depth statistical analysis and further processing.

When a statistical analysis indicates the necessity of maintenance activities later examination runs up. In the context of this work the goal of this examination is the detection of new or interesting patterns on the schema as well as the instance level of a structured Web data set. We introduced that the relations between resources are crucial for the meaning of structured Web data – short, the semantics. Thus, the **data mining** step aims at finding new or interesting associations between resources

requested in SPARQL queries.

Identified associations are presented to the data set provider for **evaluation** and may influence the next iteration of the data set life cycle.

6.2 Preparing the in-depth analysis of SPARQL query logs

We already pointed out that, amongst other issues, log files of structured Web data sets do not contain any information if a query had a non-empty result or not because the HTTP status code 200 is responded for every request that could be handled by the server properly. We designed and implemented an algorithm that combines the preprocessing and the transformation of the selected log data and resolves this issue by re-execution of every single extracted query as well the atomic basic graph patterns and triple patterns. Figure 6.2 depicts the sequential process of this algorithm (the complete pseudocode of our algorithm can be found in Appendix B).

In a first step the algorithm extracts only log entries that contain HTTP response codes < 400 and > 500. This excludes all requests with client failures. We keep requests which caused server failures because the failures may result from the server configuration that prohibits complex queries due to restrictive timeouts for example while a more generous configuration would allow their processing.

All requests against the query endpoint are extracted as they appear in the log and regular GET requests for resources are transferred into a respective SPARQL DESCRIBE query (e.g. `http://dbpedia.org/resource/Berlin` is transformed into DESCRIBE ¡`http://dbpedia.org/resource/Berlin`¿). This results into a normalized view to the usage of the data set on the level of SPARQL queries only.[1]

Afterwards all SELECT and ASK queries are iterated to extract and store all basic graph patterns and all triple patterns individually. The result is the extended query usage database as depicted in Figure 6.3.

This database is now filled with information about the success or failure of queries and all their respective atomic parts down to subjects, predicates, and objects in triple patterns by performing a number of auto-generated queries against a mirror of the original data set (or the original data set if a mirror is not available[2]). We rely on the following simple patterns for generating these queries:

Queries: All *LIMIT* and *OFFSET* parts of the original queries are replaced by a

[1] Please take note that the mapping to SPARQL DESCRIBE queries has been implemented in preparation for future analysis which are not in scope of this thesis.

[2] Using a data set mirror is highly recommended because otherwise the re-execution of queries from log files pollutes the log files of a live data set and interferes with future log analysis.

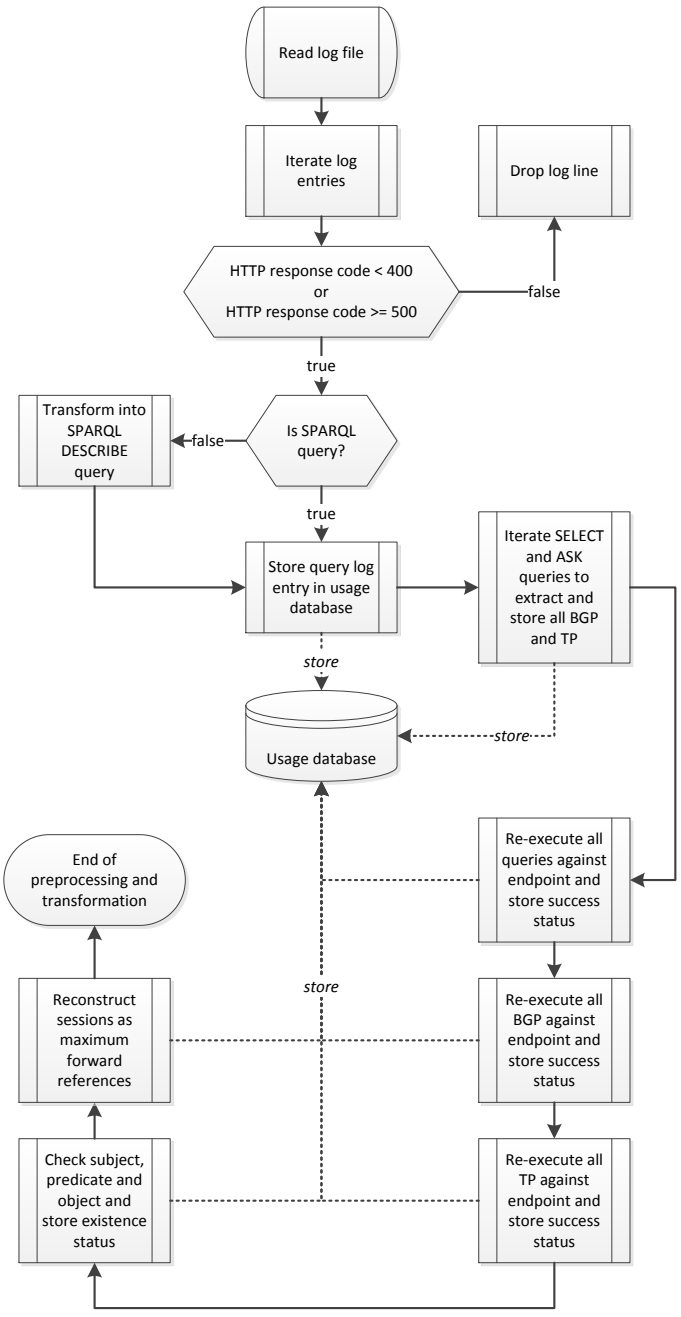

Figure 6.2: Sequential diagram of our preprocessing and transformation algorithm for the in-depth analysis of SPARQL query logs.

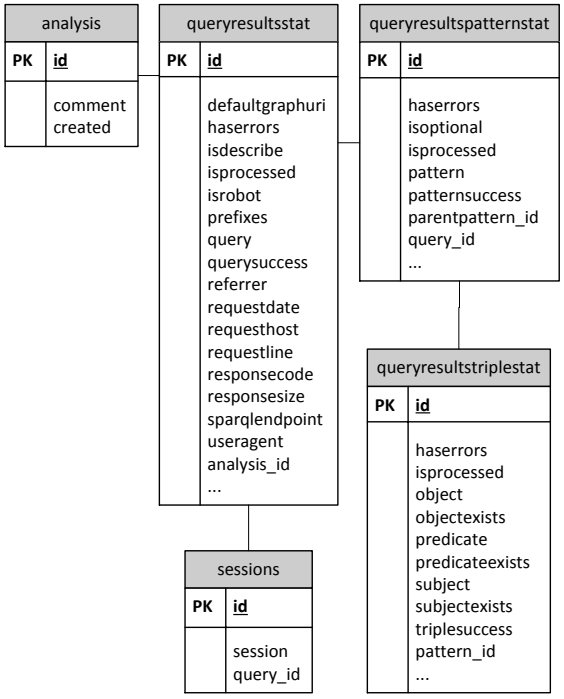

Figure 6.3: Schema of the resulting database of the log file preprocessing

LIMIT 1 because it is sufficient to find out whether queries have any result at all.

Basic graph patterns: The default query for analysing the success or failure of basic graph patterns is *SELECT * WHERE {[BasicGraphPattern]} LIMIT 1*. Optional patterns are flagged and in nested patterns the reference to the parent pattern is stored.

Triple patterns: Again the default query pattern is applied for analysing the success or failure of triple patterns (*SELECT * WHERE {[TriplePattern]} LIMIT 1*).

Subjects, predicates, objects: For the three atomic parts of all triple patterns it is stored whether it is a variable, a resource URI or a literal. If a URI is identified as subject or object the query *SELECT * WHERE {{[URI] ?property ?hasValue} UNION {?isValueOf ?property [URI]}} LIMIT 1* is performed to check whether this resource exists in the data set at all or not. In case of a

bound predicate URI the query for checking its existence in the data set is *SELECT * WHERE {?s [URI] ?o} LIMIT 1*.

The final step of our algorithm is to reconstruct user sessions for the decomposed SELECT and ASK queries by applying the simple maximal forward reference strategy [CPY96, CPY98]. We treat every single query as a footprint on the path of a user through the data set. A transaction is the set of queries from the first performed query until the user performs a previously performed query again (back reference). A user is identified by unique IP addresses and user client associations. When the user client is unknown the IP address acts as a fallback identifier. This approach heavily relies on the assumption that structured Web data sets are often used by applications which perform queries following a rather structured sequential pattern repeatedly. We are aware that this is only one rather trivial approach to reconstruct sessions from Web log files. However, this is not part of our contributions. A proper evaluation of different approaches is an open research issue but would go far beyond the scope of this thesis [SMBN03].

6.3 Usage-dependent data quality assessment

It is acknowledged that any assessment of data quality in a certain context is highly individual [BS06]. This is most intuitively depending on the type and structure of the imposed data to assess data quality but also on personal decision criteria. Instead of proposing an individual but hardly reproducible and hardly comparable heuristic, this section is dedicated to introduce a statistical framework that allows to assess the quality of a data set based on our usage analysis and conforming to the state of the art data quality framework introduced in Chapter 2. This proposal is shaped by the goal to facilitate an adaptive behaviour of structured Web data sets. Collecting usage feedback by feedback forms or user studies for example would generally change the imposed usage data and allow for a different statistical framework and selection of data quality dimensions than feasible with our usage database sourcing from preprocessed and transformed log data.

6.3.1 Query analysis framework

The usage database which results from our log file preprocessing algorithm enables a multitude of statistics based on three core levels of the SPARQL query language specification – the query level, the pattern level, and the triple level.

As the most fundamental entity we define a set of **queries** Q_A as:

$$Q_A = \{[Q_1, Q_2, \ldots, Q_n] | Q_i \text{ is a query of the generic form } Q = (E, F), i = 1, 2, \ldots, n\} \tag{6.1}$$

An important subset of $Q_A(D)$ are the **failing queries** Q_F, the set of queries which do not yield any results when they are performed against D.

$$\begin{aligned} Q_F(Q_A, D) = \{[Q_1, Q_2, \ldots, Q_n] | \forall Q_i : \\ R(Q_i, D) = \emptyset, \\ Q_i \in Q_A, i = 1, 2, \ldots, n\} \end{aligned} \tag{6.2}$$

Also depending on Q_A we now define the set of **all basic graph patterns** BGP_A as follows:

$$\begin{aligned} BGP_A(Q_A) = \{[BGP_1, BGP_2, \ldots, BGP_n] | \forall BGP_i \; \exists q \in Q_A : BGP_i \in q, \\ i = 1, 2, \ldots, n\} \end{aligned} \tag{6.3}$$

Analog to all failing queries we define those basic graph patterns of $Q_A(D)$ which fail when they are individually performed against a data set D as the set of **all failing basic graph patterns** BGP_F.

$$\begin{aligned} BGP_F(Q_A, D) = \{[BGP_1, BGP_2, \ldots, BGP_n] \\ | BGP_i \in BGP_A(Q_A) \; \wedge \; R((eval(BGP_i), SELECT), D) = \emptyset, \\ i = 1, 2, \ldots, n\} \end{aligned} \tag{6.4}$$

Self-evident we define the set of **all triple patterns** TP_A and the set of **all failing triple patterns** TP_F of $Q_A(D)$ as follows:

$$\begin{aligned} TP_A(Q_A) = \{[TP_1, TP_2, \ldots, TP_n] | \forall TP_i \; \exists p \in BGP_A(Q_A) : \\ TP_i \in p, i = 1, 2, \ldots, n\} \end{aligned} \tag{6.5}$$

$$\begin{aligned} TP_F(Q_A, D) = \{[TP_1, TP_2, \ldots, TP_n] \\ | TP_i \in TP_A(Q_A) \\ \wedge \; R((eval(TP_i), SELECT), D) = \emptyset, \\ i = 1, 2, \ldots, n\} \end{aligned} \tag{6.6}$$

Now we can also define the three complementary sets of **all successful queries** Q_S, **all successful basic graph patterns** BGP_S, and **all successful triple patterns** TP_S which have been performed against a data set D.

$$\begin{aligned} Q_S(Q_A, D) &= Q_A \setminus Q_F(Q_A, D) \\ BGP_S(Q_A, D) &= BGP_A(Q_A) \setminus BGP_F(Q_A, D) \\ TP_S(Q_A, D) &= TP_A(Q_A) \setminus TP_F(Q_A, D) \end{aligned} \tag{6.7}$$

Basic cross-level analysis

The way how triple patterns are combined to form more complex basic graph patterns as well as the grouping of basic graph patterns to complex queries can be interpreted as if each query comprises a virtual schema. The most trivial reason for failing queries when no solution mapping can be found is that requested resources do not exist in the data set at all. But it may also be that some data about one or more of the requested resources exists which simply does not match any of the queried graph patterns. While the former case means that a resource is completely unknown the latter means (a) that the requested data is not or only partly populated, or (b) the requested data is represented in a different structure. Figure 6.4 depicts the possible interdependence of atomic parts of a query which may be successful as individual queries but fail in combination or vice versa.

We define the set of **all failing queries which contain successful basic graph patterns**:

$$\begin{aligned} Q_{FPS}(Q_A, D) = \{[Q_1, Q_2, \ldots, Q_n] | \forall Q_i \; \exists p \in BGP_S(Q_A, D) : \\ p \in Q_i, Q_i \in Q_F(Q_A, D), i = 1, 2, \ldots, n\} \end{aligned} \tag{6.8}$$

To dive deeper into the query structure we also define the set of **all failing basic graph patterns which contain only successful triple patterns**.

$$\begin{aligned} BGP_{FTS}(Q_A, D) = \{[BGP_1, BGP_2, \ldots, BGP_n] | \forall BGP_i \; \exists t \in TP_S(Q_A, D) : \\ t \in BGP_i, BGP_i \in BGP_F(Q_A, D), i = 1, 2, \ldots, n\} \end{aligned} \tag{6.9}$$

In-depth instance and schema level analysis

The instance level of data sets is generally orders of magnitudes bigger and more dynamically updated than the schema level. Thus, we do not regard the instance level from an overall quantitative perspective (e.g. counting the overall number of triples in a data set). But, we define all queried resources as:

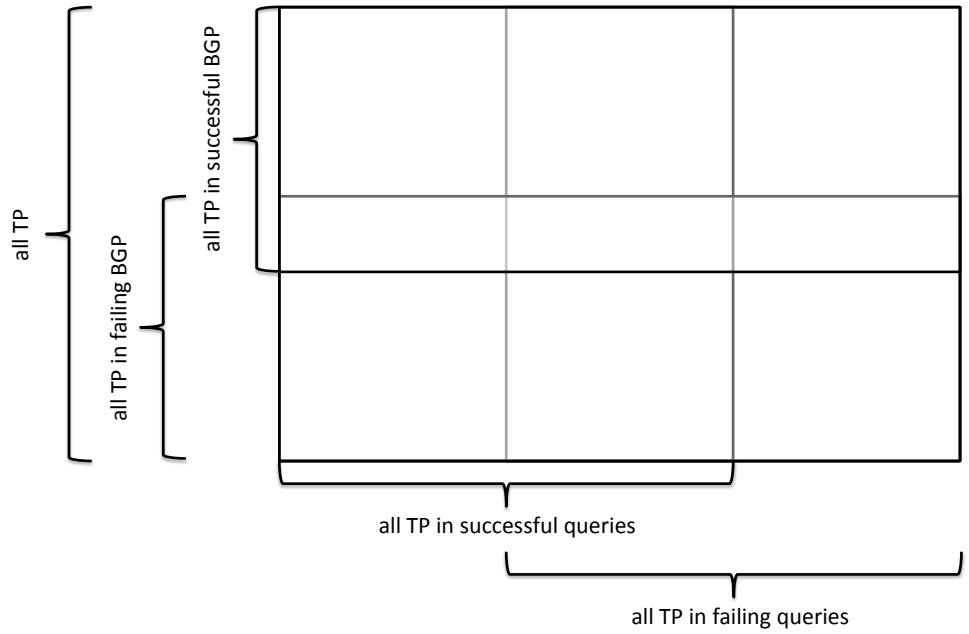

Figure 6.4: Interdependence of the atomic parts in successful and failing SPARQL queries.

$$
\begin{aligned}
R_A(Q_A) = &\{[u_1, u_2, \ldots, u_n] | \forall u_i \; \exists t \in TP_A(Q_A) : \\
&(S(t) = u_i \; \vee \; O(t) = u_i), \\
&u_i \in URIS, i = 1, 2, \ldots, n\}
\end{aligned}
\tag{6.10}
$$

Furthermore, we utilize the following SPARQL query which returns every triple that contains a specific resource as subject or object and nothing respectively when the resource does not exist in the data set at all.

Thus, the set of **all resources which appear as subject or object in triple patterns of a set of queries Q_A but which are not populated in a data set** D is defined as:

```
SELECT *
  WHERE {
   { ¡uri¿ ?p ?o }
   UNION
   { ?s ?p ¡uri¿ }
  }
```

$$
\begin{aligned}
R_M(Q_A, D) = \{[u_1, u_2, \ldots, u_n] | \forall u_i \; \exists t \in TP_A(Q_A) : \\
(S(t) = u_i \; \vee \; O(t) = u_i) \\
\wedge \; (R((eval(Union([u_i, v, w], [x, v, u_i])), SELECT), D) = \emptyset), \\
u_i \in URIS, v \neq w \neq x \in VARIABLES, \\
i = 1, 2, \ldots, n\}
\end{aligned} \tag{6.11}
$$

On the schema level we derive statistics about the distinct sets of classes and properties used for the population of the instance data by performing the following two SPARQL queries against the data set:

```
SELECT ?uri
  WHERE {
   { SELECT DISTINCT ?uri WHERE {
    ?s a ?uri
  } ORDER BY ASC(?uri) }
```

```
SELECT ?uri
  WHERE {
   { SELECT DISTINCT ?uri WHERE {
    ?s ?uri ?o
  } ORDER BY ASC(?uri) }
```

Thus, we define the distinct set of **all populated classes** C_P and the distinct set of **all populated properties** P_P of a data set D.

$$
\begin{aligned}
C_P(D) = \{[C_1, C_2, \ldots, C_n] | \forall C_i \exists u \in URIS : \\
R((eval([u, <rdf:type>, C_i]), SELECT), D) \neq \emptyset, \\
C_i \in URIS, i = 1, 2, \ldots, n\}
\end{aligned} \tag{6.12}
$$

$$\begin{aligned} P_P(D) = \{&[P_1, P_2, \ldots, P_n] | \forall P_i \exists (u_1, u_2) : \\ &R((eval([u_1, P_i, u_2]), SELECT), D) \neq \emptyset, \\ &P_i \in URIS, u_1 \in URIS, u_2 \in URIS \cup LITERALS, \\ &i = 1, 2, \ldots, n\} \end{aligned} \tag{6.13}$$

The analysis of how classes and properties are combined in complex queries and which classes and properties are used in queries which were not used to populate the data set allows for further analysis of the convergence of the requested and the offered vocabulary primitives and helps to adapt users' requirements to the schema level of the data set. We define the distinct sets of **all classes and properties which are used in queries** as follows:

$$\begin{aligned} C_Q(Q_A, D) = \{&[C_1, C_2, \ldots, C_n] | \forall C_i \exists t \in TP_A(Q_A) : \\ &(S(t) = C_i \wedge C_i \in C_P) \\ &\vee (P(t) = <rdf:type> \wedge O(TP_j) = C_i) \\ &\vee (P(t) \in [<owl:equivalentClass>, <rdfs:subClassOf>]) \\ &\wedge (S(t) = C_i \vee O(t) = C_i)), \\ &C_i \in URIS, i = 1, 2, \ldots, n\} \end{aligned} \tag{6.14}$$

$$\begin{aligned} P_Q(Q_A) = \{&[P_1, P_2, \ldots, P_n] | \forall P_i \; \exists t \in TP_A(Q_A) : \\ &P(t) = P_i, P_i \in URIS, i = 1, 2, \ldots, n\} \end{aligned} \tag{6.15}$$

As subsets of C_Q and P_Q we define the set of all **distinct queried classes populated in a data set** (used classes) C_U and the set of all **distinct queried properties populated in a data set** (used properties) P_U:

$$C_U(Q_A, D) = C_P(D) \cap C_Q(Q_A) \tag{6.16}$$

$$P_U(Q_A, D) = P_P(D) \cap P_Q(Q_A) \tag{6.17}$$

Depending on C_Q and C_P we can now simply define the set of all **queried classes not used in the data set** (missing classes) C_M as follows:

$$C_M(Q_A, D) = C_Q(Q_A, D) \setminus C_P(D) \tag{6.18}$$

Respectively the set of all **queried properties not used in the data set** (missing properties) P_M:

$$P_M(P_P(D), P_Q(Q_A)) = P_Q(Q_A) \setminus P_P(D) \tag{6.19}$$

A posteori solution restrictions and human-readable data analysis

As a matter of fact a query actually may also return an empty result because of a filter constraining the solution mapping. Filter expressions in queries allow for restricting the solution mappings for the basic graph pattern they appear in. In the context of this thesis we pass an in depth analysis of filters because we emphasize the importance of the graph pattern analysis in our approach. As a simple heuristic which allows for general statistics about filters we assume that a query or **a basic graph pattern contains a filter expression when it matches the regular expression** $.*FILTER$
$s.*$. This can be further extended to match language filters in order to analyze the requested natural languages in queries for example .

$$\begin{aligned} BGP_{Filter}(Q_A) = &\{[BGP_1, BGP_2, \ldots, BGP_n] \\ &|BGP_i \in BGP_A(Q_A) \\ &\wedge BGP_i \text{ contains the a string matching the regular expression} \\ &\quad \text{“.*FILTER.*”}, \\ &i = 1, 2, \ldots, n\} \end{aligned} \tag{6.20}$$

With reference to the RDF-S Plus concepts the annotation properties `rdfs:label` and `rdfs:comment` are dedicated to be used for describing resources human readable without any further inferential semantics by definition. This allows a more generic analysis of the requested human readable parts of a data set on graph pattern level than any filter analysis would allow. We define the set of **all triple patterns of a set of queries which contain an annotation property**:

$$\begin{aligned} TP_{HA}(Q_A) = &\{[TP_1, TP_2, \ldots, TP_n] | \forall TP_i : TP_i \in TP_A(Q_A) \\ &\wedge (P(TP_i) \in [<rdfs{:}label>, <rdfs{:}comment>] \\ &i = 1, 2, \ldots, n\} \end{aligned} \tag{6.21}$$

Intuitively the set of **all triple patterns of a set of queries which contain an annotation property but fail when they are performed against a data set** is defined as:

$$
\begin{aligned}
TP_{HF}(Q_A, D) = &\{[TP_1, TP_2, \ldots, TP_n] | \forall TP_i : \\
&TP_i \in TP_{HA}(Q_A) \wedge TP_i \in TP_F(Q_A, D) \\
&i = 1, 2, \ldots, n\}
\end{aligned} \tag{6.22}
$$

6.3.2 Data set quality score functions

Theoretically we aim at defining a data set quality score function $\Phi : \mathbb{R}^+ \rightarrow [0, 1]$. We propose two different approaches to calculate the integrated data quality score – the plain and the hierarchical data set quality score.

The **plain data set quality score** allows for assessing an individual score for each selected dimension and the overall score as the sum of the weighted scores of the dimensions. It is defined as:

$$
\begin{aligned}
\Phi_P(D, Q_A) = &\sum_{i=1}^{n} (w_i \cdot \delta_i(D, Q_A)) \\
&\text{where } \delta_i \text{ is the score of a data quality dimension} \\
&\text{and } w_i \text{ is the weight of this dimension} \\
&\sum_{i=1}^{n} w_i = 1, w_i \geq 0
\end{aligned} \tag{6.23}
$$

The **hierarchical data set quality score** allows for assessing an individual score for each selected dimension, the cumulated score of each data quality category from which dimensions were selected, and the overall score as the sum of the weighted scores of the categories. We define:

$$\Phi_H(D, Q_A) = \sum_{i=1}^{n} (w_i \cdot \gamma_i(D, Q_A))$$

where γ_i is the cumulative score of a data quality category

and w_i is the weight of this category within the set of n categories

$$\gamma_i(D, Q_A) = \sum_{j=1}^{m} (v_j \cdot \delta_j(D, Q_A))$$

where δ_j is the score of a data quality dimension

and v_i is the weight of this dimension

within the set of m dimensions in the same category

$$\sum_{i=1}^{n} w_i = 1, \sum_{j=1}^{m} v_j = 1$$

$$w_i, v_j \geq 0 \tag{6.24}$$

6.3.3 Usage-dependent data quality dimensions

The foundation for our specific selection of data quality dimensions and also their weighting is the data quality hierarchy of Wang et al. as depicted in Figure 6.5. The values next to the dimensions are their median rank resulting from the study in [WS96]. Lower values indicate a higher importance of the respective dimension for the data consumer.

We traverse the hierarchy in a top-down fashion for deciding which categories and which subsidiary data quality dimensions we take into account for usage dependent data quality assessment.

As a primary step we exclude all dimensions within the accessibility data quality category in the context of our consideration. We do so because: (a) structured Web data is generally characterized by a unified accessibility due to the exploitation of the Web architecture; (b) computational efficiency of structured Web data stores but also index generation or data prefetching strategies based on SPARQL query log analysis are discrete research challenges which exceed the problem space of this thesis; (c) security is an issue heavily discussed in the research community which is much more concerned with the legal aspects of open Web data than with structured Web data as a data integration paradigm.

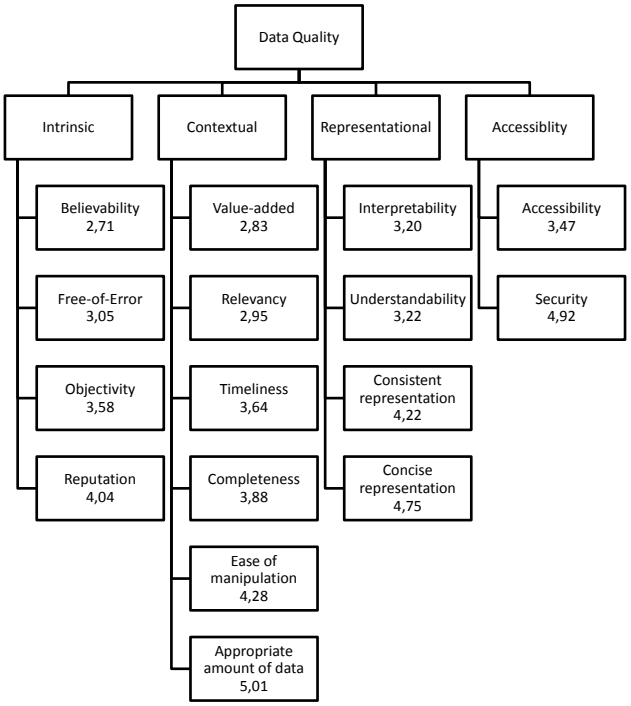

Figure 6.5: The hierarchical data quality framework of Wang et al. including the median rank of dimensions (lower values indicate a higher importance of the) [WS96, PLW02]

Three top level data quality categories are remaining – intrinsic data quality, contextual data quality, and representational data quality – which can be summarized as follows:

Intrinsic data quality is concerned with inherent properties of the data itself independent from certain application context.

Contextual data quality regards aspects which depend on the requirements of the application the data is used in.

Representational data quality reflects how well and suitable the schema and language of the data is.

The next step is to examine these three categories in order to decide about their relevance. A structured decision making strategy allowing for ranking alternatives

is pairwise comparison. Pairwise comparison is based on L. L. Thurstone's law of comparative judgement [Thu27] which relies on the fact that it is easier to decide between two alternatives with reference to one discrete decision criterion or goal than deciding between more than two alternatives. The most regarded but in fact also widely criticized application of pairwise comparisons is the Anlytic Hierachy Process (AHP). AHP is a rigorous decision making theory for multicriteria decision making (MCDM) developed by Thomas L. Saaty [Saa80, Saa90, Saa08] based on modelling decision problems as a hierarchy and ranking each distinct set of sibling nodes of the hierarchy by a pairwise comparison with reference to a single property. As a mathematical rigorous method Saaty proposed the eigenvector solution to evaluate pairwise comparisons. A pairwise comparison matrix A (also called an evaluation matrix) for a set of alternatives $a_1, a_2, \ldots a_n$ can be defined as follows:

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1i} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{i1} & \ldots & a_{ij} & \ldots & a_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nj} & \ldots & a_{nn} \end{pmatrix}$$

This matrix satisfies the conditions $i, j = 1, \ldots, n$, $a_{ij} > 0$, $\forall i = j : a_{ij} = 1$, and $a_{ij} = 1/a_{ji}$. The eingenvector W of A weights the alternatives a_i and is calculated as follows[3]:

Table 6.1: Calculating the eigenvector of an evaluation matrix.

	Evaluation matrix			**Normalization**				
	a_1	...	a_n	a_1	...	a_n	r_i	w_i
a_1	$a11$	...	a_{1n}	$\frac{a_{11}}{c_1}$	...	$\frac{a_{1n}}{c_n}$	$r_1 = \sum_{i=1}^{n} (\frac{a_{1i}}{c_i})$	$w_1 = \frac{r_1}{n}$
...	...	...	...	...	...	...	...	...
a_n	a_{n1}	...	a_{nn}	$\frac{a_{n1}}{c_1}$	...	$\frac{a_{nn}}{c_n}$	$r_n = \sum_{i=1}^{n} (\frac{a_{ni}}{c_i})$	$w_n = \frac{r_n}{n}$
c_i	$c_1 = \sum_{i=1}^{n} a_{i1}$	...	$c_n = \sum_{i=1}^{n} a_{in}$	1	...	1	n	1

[3]In order to be more precise W can also be calculated by application of power iteration which means to square the evaluation matrix before normalization and repeat this iteratively until the eigenvector of the squared normalized matrix converges.

Saaty also introduced the fundamental scale to quantify pairwise comparisons shown in Table 6.2 [Saa90, Saa08].

Table 6.2: The fundamental scale for pairwise comparisons by Saaty [Saa90, Saa08].

Importance	Explanation
1	two activities contribute equally to the objective
2	intermediate value
3	experience and judgment slightly favoring one activity over another
4	intermediate value
5	experience and judgment strongly favoring one activity over another
6	intermediate value
7	an activity is favored very strongly over another
8	intermediate value
9	evidence favoring one activity over another is of the highest possible order of affirmation
reciprocal	if activity i has one of the above numbers assigned to it when compared with activity j, then activity j has the reciprocal value when compared with i
rationals	if consistency is forced by obtaining n numerical values to span the matrix

In our concrete case we need to decide for each pair of the three aforementioned data quality categories *which category suits better to measure the fitting of a data set to a set of SPARQL queries*. We performed the pairwise comparison along this criterion by applying Saaty's fundamental scale and power iteration for the precise eigenvector calculation as shown in Table 6.3.

Table 6.3: Iterative pairwise comparison process for our decision.

	INT	CONT	REP		
	Evaluation matrix				
Intrinsic DQ (INT)	1.000000	**0.111111**	**0.111111**		
Context. DQ (CONT)	**9.000000**	1.000000	**0.333333**		
Repres. DQ (REP)	**9.000000**	**3.000000**	1.000000		
c_i	19.000000	4.111111	1.444444		
	Potentiation 1				
INT	1.000000	0.012346	0.012346		
CONT	81.000000	1.000000	0.111111		
REP	81.000000	9.000000	1.000000		
c_i	163.000000	10.012346	1.123457		
	Normalization 1			r_i	w_i
INT	0.00613497	0.00123305	0.01098901	0.01835703	0.00611901
CONT	0.49693252	0.0998767	0.0989011	0.69571031	0.23190344
REP	0.49693252	0.89889026	0.89010989	2.28593266	0.76197755
c_i	1.000000	1.000000	1.000000	3.000000	1.000000
	Potentiation 2				
INT	0.000038	0.000002	0.000121		
CONT	0.246942	0.009975	0.009781		
REP	0.246942	0.808004	0.792296		
c_i	0.493921	0.817981	0.802198		
	Normalization 2			r_i	w_i
INT	$7.6202 * 10^{-5}$	$1.8587 * 10^{-6}$	0.00015053	0.0002286	$7.6198 * 10^{-5}$
CONT	0.4999619	0.0121951	0.01219329	0.52435028	0.17478343
REP	0.4999619	0.98780304	0.98765618	2.47542112	0.82514037
c_i	1.000000	1.000000	1.000000	3.000000	1.000000
	Potentiation 3				
INT	0.000000	0.000000	0.000000		
CONT	0.249962	0.000149	0.000149		
REP	0.249962	0.975755	0.975465		
c_i	0.499924	0.975904	0.975613		
	Normalization 3			r_i	w_i
INT	$1.1615 * 10^{-8}$	$3.5402 * 10^{-12}$	$2.3227 * 10^{-8}$	$3.4846 * 10^{-8}$	$1.1615 * 10^{-8}$
CONT	0.49999999	0.00015239	0.00015239	0.50030478	0.16676826
REP	0.49999999	0.99984761	0.99984758	2.49969519	0.83323173
c_i	1.000000	1.000000	1.000000	3.000000	1.000000
	Potentiation 4				
INT	0.000000	0.000000	0.000000		
CONT	0.250000	0.000000	0.000000		
REP	0.250000	0.999695	0.999695		
c_i	0.500000	0.999695	0.999695		
	Normalization 4			r_i	w_i
INT	$2.6983 * 10^{-16}$	$1.2537 * 10^{-23}$	$5.3966 * 10^{-16}$	$8.0949 * 10^{-16}$	$2.6983 * 10^{-16}$
CONT	0.5	$2.3231 * 10^{-8}$	$2.3231 * 10^{-8}$	0.50000005	**0.16666668**
REP	0.5	0.99999998	0.99999998	2.49999995	**0.83333332**
c_i	1.000000	1.000000	1.000000	3.000000	1.000000

After four iterations the weights w_i converge. Since w_1 converges against 0 we eliminate the category intrinsic data quality from our further consideration. Contextual data quality has a final weighting of 0.17 and representational data quality has a weighting of 0.83. This means that our our approach does not suit to assess the data quality dimensions believability, objectivity, reputation, and free-of-error which one can treat as extremely subjective and data-value-oriented. Instead the approach allows for assessing dimensions which are related to the quality of the data representation as well as requirements resulting from the consumer's task. In the following we will enter each of the remaining two categories individually in order to derive a final selection of data quality dimensions which can be assessed in a usage-dependent fashion. We give a definition for each selected dimension with reference to our SPARQL query log analysis approach and a metric to calculate it.

Measuring contextual data set quality

Contextual data quality dimensions are (1) appropriate amount of data, (2) completeness, (3) ease of manipulation, (4) relevancy, (5) timeliness, and (6) value-added. Query log analysis provides no feedback about further processing of the requested and retrieved data on the client's side and no direct feedback of the user, which is anyway most likely an application outside of the control of the data set provider. It is hardly possible if not impossible to derive any qualitative statement about whether delivered data is actual or outdated by this means (e.g. find out whether the number of inhabitants of a country needs to be updated because an actual census has been performed). Hence, we do not regard the highly subjective dimensions relevancy, and timeliness of data. It is furthermore a fundamental characteristic of structured Web data to be easily manipulated for use by multiple applications and integration with multiple other data sets, so we exclude the ease of manipulation dimension as well.

Appropriate amount of data Appropriate amount of data measures to which extend a data set fulfills the queries performed against it. Based on the assumption that only syntactically correct queries are performed against a data set, the perfect amount of data is provided when all queries can be answered. A weaker score results from the proportion of the amount of failing queries by the amount of all queries and reflects that the data set does not conform to queries in any unspecified way on the schema or the instance level.

$$\rho(D, Q_A) = 1 - (\frac{|Q_F(Q_A, D)|}{|Q_A|}) \tag{6.26}$$

In open data scenarios this definition of appropriate amount of data is in fact complicated. It is not to be expected that only correct and reasonable queries will be performed. However, the open data use case is only one of many in which structured Web data is a promising technology as shown by our motivating use cases in Chapter 1. More controlled scenarios easier allow this strict assumption plus that it is also possible to apply preprocessing which cleans logs from evidently messy queries more sophisticated than our approach already does.

Completeness Completeness measures how many queried properties are missing in a data set. This is the typical attribute-oriented interpretation of completeness [NLF99, MMB12] and a central measure to evaluate the schema level of a data set. Incompleteness results from the proportion of the amount of distinct properties which are not populated in the data set with regard to the amount of all distinct properties which were queried.

$$\kappa(D, Q_A) = 1 - (\frac{|P_M(P_P(D), P_Q(Q_A))|}{|P_Q(Q_A)|}) \tag{6.27}$$

Value-added Value-added measures to which extend a data set contains the resources which are explicitly requested in queries. Independent from whether an entire requested graph pattern can be fulfilled or not this reflects if the data set provides any data at all for requested resources which appear as subject or object in triple patterns. The higher the proportion of not populated resources with reference to all queried resources, the lower the value provided by a data set.

$$\beta(D, Q_A) = 1 - (\frac{|R_M(Q_A, D)|}{|R_A(Q_A)|}) \tag{6.28}$$

Again, this is a complicated definition in scenarios with distributed data sets for various specific purposes, like the special case of open data. If a data set is not a cross-domain data set, there may be no reason why any value should be provided for a resource which has no relation to this domain. Hence, the value-added dimension will hardly ever achieve a score of 100%. But still, there may be situations – potentially unanticipated ones – where it can turn out beneficial to identify if (and subsequently which) entities are missing in a data set, be it just to recognize that there is another or much more common identifier for an entity which is identified differently in the own data set.

Measuring representational data set quality

Dimensions of the representational data quality category are (1) concise representation (conciseness), (2) ease of understanding, (3) interpretability, and (4) representational consistency. We refer to the fundamental characteristic of structured Web data to provide a unified interface and form so that we can exclude ease of understanding from our consideration.

Conciseness Conciseness measures to which extend the properties used to populate a data set are exploited by queries. It is defined as the proportion of the amount of queried populated properties with reference to the amount of all populated properties.

$$\pi(D, Q_A) = \frac{|P_U(Q_A, D)|}{|P_P(D)|} \tag{6.29}$$

Consistency Consistency measures to which extend a data set is consistent with the virtual domain view expressed by queries. The virtual domain view is constituted by how populated resources and properties are combined in complex basic graph patterns of queries. A basic graph pattern may fail because the combination of properties is generally not provided by the data set. Or it may fail because a specific type of resources used in the pattern results in untypical resources in subsequent triple patterns due to dynamic binding of variables. Hence, we put special attention on failing basic graph pattern that combine only triple patterns which are themselves satisfiable by the data set and define conciseness as the proportion of the amount of queries containing such patterns with reference to the amount of all queries.

$$\tau(D, Q_A) = 1 - (\frac{|BGP_{FTS}(Q_A, D)|}{|Q_A|}) \tag{6.30}$$

Intepretability Interpretability measures to which extend a data set provides requested data which is meant for human consumption. The interpretability of a data set is degraded by the proportion of failing triple pattern which contain an annotation property as predicate.

$$\iota(D, Q_A) = 1 - (\frac{|TP_{HF}(Q_A, D)|}{|TP_{HA}(Q_A)|}) \tag{6.31}$$

6.3.4 Calculating the usage-dependent data set quality score

In the preceding sections we introduced two generic data set quality score functions and a specific selection of data quality dimensions that facilitate data quality assessment in a usage-dependent fashion based on SPARQL query log files. Bringing both pieces together allows for defining two concrete data set quality score functions which exploit the selected data quality dimensions and categories.

We define the plain usage-dependent data set quality score function as:

$$\begin{aligned}\Phi_{Pusage}(D, Q_A) =&(w_\rho \cdot \rho(D, Q_A)) + (w_\kappa \cdot \kappa(D, Q_A)) + (w_\beta \cdot \beta(D, Q_A)) \\ &+ (w_\pi \cdot \pi(D, Q_A)) + (w_\tau \cdot \tau(D, Q_A)) + (w_\iota \cdot \iota(D, Q_A))\end{aligned} \tag{6.32}$$

Analog we define the hierarchical usage-dependent data set quality score function as:

$$\begin{aligned}\Phi_{Husage}(D, Q_A) =&v_{context} \cdot ((w_\rho \cdot \rho(D, Q_A)) + (w_\kappa \cdot \kappa(D, Q_A)) + (w_\beta \cdot \beta(D, Q_A))) \\ &+ v_{represent} \cdot ((w_\pi \cdot \pi(D, Q_A)) + (w_\tau \cdot \tau(D, Q_A)) + (w_\iota \cdot \iota(D, Q_A)))\end{aligned} \tag{6.33}$$

As part of our evaluation we will examine which of the two functions suits better for getting reasonable insight into particular issues of a data set with reference to its usage.

6.4 Pattern discovery for maintenance recommendation

When the aforementioned data set quality score indicates issues with a deployed data set version it is possible to exploit our enriched usage database again for the discovery of interesting patterns in queries. Such patterns can reflect the external view to the domain of interest covered by the data set at hand in a much more sophisticated fashion than each query already does because they can disclose cross-query relationships. They are a source of inspiration of concrete changes on the instance as well as the schema level.

6.4.1 Mining association rules

Association rule mining is a very mature technique to detect interesting implication patterns within various types of transactions stored in transaction databases by mining frequent itemsets. A very intuitive use case is the analysis of shopping baskets

of clients where association rule mining facilitates the identification of things which are commonly bought together (e.g. when customers buy bread it has certain likelihood that they also buy butter). The most widely applied algorithm is the Apriori algorithm [AS94, SA95].

The central entities for association rule mining are an *itemset* I and a *transaction database* T.

$$\begin{aligned} I =& \{i_1, i_2, \ldots, i_n\} \text{ is a set of } n \text{ items} \\ T =& (t_1, t_2, \ldots, t_m) \text{ is a list of } m \text{ transactions} \\ & \forall t_i : t_i \subseteq I \end{aligned} \tag{6.34}$$

Let X and Y be itemsets $X, Y \subseteq I$. An association rule is the implication of the form $X \Rightarrow Y$ where we call X the left-hand-side (LHS) and Y the right-hand-side (RHS). The number of transactions which contain an itemset define its *support*:

$$supp(X) = |\{t_i \in T : X \subseteq t_i\}| \tag{6.35}$$

To determine the probability of finding an association rule the *confidence* is defined as follows:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{6.36}$$

As the most common measure of interest of a rule *lift* is defined as the factor at which a rule appears more often than expected from the probability of the individual LHS and RHS itemsets. It is noteworthy that lift may be an indicator of interest but is also sensible towards itemset co-occurrences by chance which can harm the representativeness in small databases [BMUT97, TKS04].

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)} \tag{6.37}$$

In our query log analysis scenario items correspond to the atomic parts of triple patterns – namely URIs, variables, and literals – which appear together in queries. In the most intuitive case each query is a transaction. But it is also possible to regard every single triple pattern as an atomic transaction or to reconstruct sessions first and then regard all queries or triple patterns within a session as a transaction. In order to achieve a better representativeness we normalized all variables to the form *subj_variable*, *pred_variable* or *obj_variable* depending on their position within the triple pattern they appear in.

Table 6.4 shows an example of the schematic view of a transaction database with real world data from a DBpedia log file. Every 1 value indicates that the query

represented by that row contains the item represented by that column and every 0 indicates the inverse (with the exception that column 1 contains the incremental transaction id).

Table 6.4: Example of the schematic view of a transaction database for a set of SPARQL queries.

trans_id	?subj_variable	?pred_variable	ns1:res1	ns2:pr1	ns2:pr2	...
1	1	1	1	1	0	...
2	0	1	1	0	0	...
3	1	0	0	0	1	...
...	...	...	...	...	...	...

We fundamentally distinguish between *instance level associations* and *schema level associations*. **Instance level associations** consider those primitives as items which were originally queried. **Schema level associations** consider those primitives as items which can be retrieved by performing the query $SELECT?oWHERE\{rdf : typeo\}$ for every queried subject or object resource . Untyped resources as well as predicate resources are treated as items as they are.

Any kind of associations can be mined for all logged and preprocessed queries or only for those which conform to specific criteria. In the context of this work we focus only on the *failing query restriction* and the *unknown predicate restriction*. The **failing query restriction** selects only those queries from our usage database which fail. The **unknown predicate restriction** selects only those triple patterns which apply a predicate which is not used within the data at all.

For the evaluation of the mining results we simply leverage the capabilities of the arulesViz package [HC11] for the R statistical computing environment[4]. It is not a matter of this thesis to contribute anything to the field of visual analytics in general or to the visual analysis of association rules in special. We experienced that the arulesViz package provides a sufficient breadth of visualizations for the rules computed by the Apriori algorithm (as it is implemented in the arules package [HGHB09] for R) conforming to Schneidermann's Visual Information-Seeking Mantra: "overview first, zoom and filter, then details on demand"[Shn96]. startatroot

[4] `http://www.r-project.org/`

Part III

Evaluation and Conclusions

CHAPTER 7

Evaluation

We presented an approach for usage-dependent data set maintenance in the subsequent chapters, which was motivated by requirements resulting from an empirical study. Due to the fact that we decided not only to design an innovative methodological approach in theory but also to provide tool support for an explicit instantiation of it, the matter of this thesis ranges between more than one discrete research area. This chapter is dedicated to evaluate the individual contributions but also to bring the pieces together and answer our research questions: (1) What are the blind spots between ontology engineering and Linked Data? (2) How can classical Web usage mining methods be applied in the context of structured Web data sets and how does that affect managing data set maintenance? (3) To which extend can usage-dependent metrics help to assess the quality of a Web data set?

The chapter is structured as follows: We start with an analysis of our proposed methodology by two rigorous state of the art methods for the evaluation of ontology engineering methodologies. This sets our approach in a qualitative relation to the most significant pieces of related work in the problem space of ontology and data set life cycles. Afterwards we compare our selection and weighting of data quality dimensions with the empirically founded baseline in order to provide the reader with insight into how our approach fulfills the needs of data consumers and how this can be improved by collecting different feedback information than the ones available in raw log files. A number of real world data set analysis as well as a case study in

the context of the DBpedia data set prove the applicability of the entire approach in practice.

7.1 Evaluation of the data set life cycle methodology

This part of the evaluation is dedicated to our data set life cycle methodology COLM and has two goals: First, we describe and discuss the connection of our proposal to established ontology and data set creation and management methodologies along the state of the art practice for such evaluations. This sets our proposal in relation to the related work and equips the reader with the necessary knowledge to decide for the suitable approach in her specific application context based on a standardized set of criteria. Second, we apply the state of the art cost model for ontology development projects with a special focus on maintenance costs. This provides insight about how to generally estimate the costs of data set maintenance when our approach is chosen and furthermore sheds light on which tasks are most likely to turn out costly based on empirically evidenced cost drivers.

7.1.1 A comparative discussion of COLM

In order to evaluate our life cycle methodology we adopt the state of the art in analysing collaborative ontology engineering methodologies. It is based on the work in [Lóp99] and [FLGP02] proposed the following nine criteria for the analysis of ontology engineering methodologies:

C1. Inheritance from knowledge engineering : *How does traditional knowledge engineering influence the methodology?*

C2. Detail of the methodology : *What is the level of detail of the methodology?*

C3. Recommendations for knowledge formalization : *What is/are the knowledge formalism(s) the methodology is tailored to?*

C4. Strategy for building ontologies : *Is the strategy underlying the methodology application-dependent, application-semidependent, or application-independent?*

C5. Strategy for identifying concepts : *Is the strategy for identifying concepts top-down, bottom-up, or middle-out [UG96]?*

C6. Recommended life cycle : *Does the methodology propose a life cycle?*

C7. Differences between the methodology and IEEE 1074-1995 : *What are the differences between the methodology and the IEEE 1074-1995 standard?*

C8. Recommended techniques : *Does the methodology propose the use of specific techniques for carrying out activities?*

C9. Usage of the methodology : *Which ontologies have been developed following the methodology?*

In [SLR13] these criteria were revised since collaborative ontology engineering has become more and more common. The analysis in comparison traditional knowledge engineering as well as the IEEE 1074-1995 standard have been eliminated because in practice both disciplines, knowledge engineering and software engineering, have never become established practices that have a positive effect on adoption and impact of ontology engineering. Instead three criteria were added which respect that ontologies are most effectively built in an evolutionary fashion involving communities of users with disparate skills – roles, evolution, and collaboration. The roles criterion answers the question *which stakeholders are involved in the engineering process*. The evolution criterion is concerned with *how the methodology supports the publication and management of ontology versions. Which tools support collaboration and consensus finding* in the engineering process is the focus of the collaboration criterion. Our approach also involves different stakeholders in different phases and can be treated as a collaborative methodology. The final framework from [SLR13] proposes the following ten criteria:

C-I. Detail of the methodology : The level of detail of a methodology as well as empirically evidenced best practices are crucial for adoption.

C-II. Recommendations for knowledge formalization : If a methodology has dependencies towards a particular ontology language or the formal expressivity of a particular language the applicability of the methodology in settings which have constraints in this respect may be influenced.

C-III. Strategy for building ontologies : The degree of application dependence is important due to the trade-off between the level of assistance provided by a methodology in a specific environment and its applicability to other environments which might not exhibit the same characteristics.

C-IV. Strategy for identifying concepts : The ontology engineering scenario and the availability of domain-related documentation and requirements influence the identification of concepts.

C-V. Recommended life cycle : The life cycle needs to match the requirements of the scenario for which the ontology is being developed.

C-VI. Recommended techniques : The availability of dedicated methods and tools which support the ontology engineering team can have a positive effect on the usability of the methodology.

C-VII. Roles : The roles model of editors and contributors is crucial to ensure that the engineering process is performed in an efficient manner.

C-VIII. Evolution The evolution of an ontology might cause inconsistencies in conceptual modeling and discussions about how to optimally implement changes.

C-IX. Collaboration The support for consensus finding within the ontology community influences the effectiveness of the development process and the applicability of the ontology.

C-X. Usage of the methodology The number and type of real-world projects allows to gain insight into the overall adoption of a methodology as a good-practice and the scenarios to which it is well or less suited.

We will now analyse five methodologies alongside the criteria C-I to C-X. Beside our own life cycle methodology (COLM) these are (a) the LOD2 stack life cycle, (b) the NeOn methodology, (c) RapidOWL, and (d) ontology maturing. We will perform the analysis for our methodology, the LOD2 stack life cycle, and the NeOn methodology manually and refer to the results of the study in [SLR13] for RapidOWL and ontology maturing.

Table 7.1: Comparative analysis of ontology engineering methodologies (part I)

Criterion	**COLM**	**RapidOWL**	**Ontology maturing**	**LOD2 stack**	**NeOn**
C-I	Detailed	Not detailed	Not detailed	Detailed	Very detailed
C-II	Degree of formalization not prescribed, all knowledge representation based on RDF-triples	Degree of formalization not prescribed, all knowledge representation based on RDF-triples	Lightweight ontologies	Degree of formalization not prescribed, all knowledge representation based on RDF-triples	Predefined networks of ontologies of any complexity
C-III	Application-semidependent	Application-dependent	Application-semidependent	Application-independent	Application-dependent
C-IV	Focus on reuse and integration, bottom-up	Open community modeling and information integration, middle-out	Emergence of ideas, consolidation within the community and formalization, bottom-up	No strategy described	Introduces multiple strategies for bottom-up, middle-out, and top-down identification of concepts
C-V	Iterative model	Rapid prototyping (iterative)	No life cycle model proposed	Iterative model	Waterfall model and iterative model

Table 7.2: Comparative analysis of ontology engineering methodologies (part II)

Criterion	COLM	RapidOWL	Ontology maturing	LOD2 stack	NeOn
C-VI	Tracking and analysis of user feedback, classical ontology editing tools, access to ontology repositories	View-based editing for different roles, providing concrete techniques for performing different practices is stressed but not explicitly defined	Tagging, wikis	Techniques recommended or even provided for each phase with a special focus on semi-automation or automation of extraction, interlinking, and fusion	Activity-related filling cards, ontology requirements specification document, scheduling tools, competency questions transformations, classical ontology editing tools
C-VII	Domain experts, users	Domain experts, knowledge engineers, users	Domain experts	Domain experts	Domain experts, ontology developers, users
C-VIII	Frequent evolution based on the feedback of the users	Frequent evolution based on the feedback of the development community	Continuous process that results ontologies with an increasing level of formality	Continuous process that results into an increasing amount of schema and instance links	Planned evolution based on the ontology development schedule

Table 7.3: Comparative analysis of ontology engineering methodologies (part III)

Criterion	COLM	RapidOWL	Ontology maturing	LOD2 stack	NeOn
C-IX	Communication of design decisions via mailinglists and documentation in widely-accepted community portals	Communication as a general principle, most commonly wiki-based collaboration	Wiki-based collaboration	Communication not specified	Exploiting communication benefits by use of ontology design patterns and widely-accepted standard ontologies
C-X	DBpedia case study in this thesis	Catalogue of professors at University of Leipzig, semantic application for the Lutheran Church in Bavaria	Image-based navigation and bookmarking of digital cultural and scientific resources	LOD2 project case studies	Various case studies in the NeOn project (e.g. FAO use case), user studies with students

Summary

The comparison of methodologies shows a couple of interesting things. NeOn, as the commonly agreed state-of-the-art in conventional ontology engineering, proposes the most comprehensive process guidelines and tools for a knowledge-centered ontology development process. That means it suits to scenarios where the more heavyweight knowledge of a domain of interest should be captured in a maximum reusable fashion. But NeOn is not very much aligned with the dynamic nature and data-centric perspective of structured Web data publishing and management which is much more the focus of the three other methodologies as well as our approach. Amongst those one can see that Ontology Maturing suits well for the initial step of the data set life cycle when ideas about the important dimensions of a domain of interest as well as community best practices need to be collected. But the methodology lacks a more detailed description of the management and evolution process a data set underruns once it is created and published. We regard RapidOWL as the theoretical foundation of what the LOD2 stack brings into practice in a tool-oriented fashion. That means that RapidOWL provides the description of the complementary management and communication processes in a project while the LOD2 stack introduces a generic tool setup but completely lacks this perspective. COLM can be seen as a methodology that provides both the management perspective and the tool perspective integrative but focused on the specific project setup where usage can be analysed for maintenance management.

7.1.2 Applying the ONTOCOM cost estimation model to the COLM methodology

ONTOCOM is the state of the art cost estimation model for ontology development processes [MS06, STM07, SPB09, SBH+12]. It was inspired by the principles of the COCOMO II (Constructive Cost Model) for cost estimation in software development projects [BAB+00].

As a generic cost model for ontology development and maintenance ONTOCOM is based on a mostly generic sequential ontology life cycle which assumes ontology development projects to start with a domain and requirements analysis followed by the conceptualization, implementation and evaluation phases. The results of the evaluation indicate further iterations beginning in any of the phases before [SPB09]. In ONTOCOM costs are estimated as the sum of the costs arising in the development and maintenance of ontologies assessed by drivers which are weighted by an effort multiplier and have a rating level from very low to very high. The cost drivers reflect the required effort along the three dimensions (1) building, (2) reuse, and (3) maintenance. Table 7.4 provides an overview of these dimensions as well as the

respective drivers.

Table 7.4: Overview of the latest version of the ONTOCOM cost drivers according to [SBH+12].

Driver category	Driver	Concern
Product-related	DCPLX	domain complexity, requirements analysis, availability of auxiliary material
	CCPLX	applicability of modeling patterns, existence of naming and modeling constraints
	ICPLX	representation language
	DATA	degree of unambiguous semantics of the source data to be populated
	OE	testing against requirements
	DOCU	documentation
	REUSE	reusability depending on the level of generality (upper ontology, domain ontology, task ontology)
Personnel-related	OCAP/DECAP	team efficiency
	OEXP/DEEXP	general experience of ontology engineers (OEXP) and domain experts(DEEXP)
	LEXP/TEXP	team experience with a specific representation language (LEXP) and tools (TEXP)
	PCON	changes in the project team
Project-related	TOOL	degree of automation compared to manual labor required for a particular activity
	SITE	team distribution
	SCED	schedule constraints affecting refinement and evolution efforts

Applying ONTOCOM to a specific engineering methodology means to (1) identify the process stages, (2) identify activities and define mappings, (3) derive the cost formula, and (4) calibrate the start values of the drivers based on real world project data in order to improve the quality of the predictions. We perform only the steps (1) and (2) in order to show which ONTOCOM drivers apply to the different phases of the iterative COLM cycle. We leave out steps (3) and (4) due to the fact that our

COLM methodology currently lacks real-world project data. However, we think that the assignment of drivers to life cycle phases provides sufficient insight into COLM from a cost-oriented perspective. Tables 7.5, 7.6, and 7.7 show the result of the mapping.

Table 7.5: Mapping of COLM phases to ONTOCOM cost drivers (part I).

COLM phase	**Activities**	**ONTOCOM cost drivers**		
		product-related	personnel-related	project-related
Selection/ Integration/ Development	specification of int. and ext. requirements	DCPLX, DOCU	OCAP, DECAP, OEXP, DEEXP, PCON	SITE
	gathering and evaluation of reusable ontologies	DCPLX, CCPLX, ICPLX, OE	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	implementation of individual ontologies	DCPLX, CCPLX, ICPLX	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	implementation of schema mappings	DCPLX, CCPLX, ICPLX	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	documentation	DOCU	OCAP, OEXP, PCON	SCED
Validation	evaluation of the ontology orchestration with reference to int. and ext. requirements	DCPLX, CCPLX, ICPLX, OE	OCAP, DECAP, OEXP, DEEXP, LEXP, TEXP, PCON	SITE
	collection of feedback from expert communities		DECAP, DEEXP, PCON	TOOLS, SITE, SCED
	documentation	DOCU	OCAP, DECAP, OEXP, DEEXP, LEXP, PCON	SCED

Table 7.6: Mapping of COLM phases to ONTOCOM cost drivers (part II).

COLM phase	**Activities**	**ONTOCOM cost drivers**		
		product-related	personnel-related	project-related
Deployment	instantiation of a store and access infrastructure	ICPLX	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	import of ontology orchestration and schema mappings		OCAP, OEXP, TEXP, PCON	TOOL
Population	performing the instance data population	DATA	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	performing instance mapping procedure	DATA	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	import of instance data and instance links		OCAP, OEXP, TEXP, PCON	TOOL
Feedback Tracking	instantiate feedback tracking procedure or tool		DECAP, DEEXP, TEXP, PCON	TOOL

Table 7.7: Mapping of COLM phases to ONTOCOM cost drivers (part III).

COLM phase	**Activities**	**ONTOCOM cost drivers**		
		product-related	personnel-related	project-related
Reporting	preprocess and analyze collected feedback		DECAP, DEEXP, TEXP, PCON	TOOL, SCED
Evaluation	evaluation of quality assessment and change recommendations	OE	OCAP, DECAP, OEXP, DEEXP, PCON	TOOL, SITE
	evaluation of the effects of change	OE	OCAP, OEXP, PCON	TOOL
	documentation of scheduled maintenance activities	DOCU	OCAP, OEXP, PCON	SCED

Summary

The mapping of the particular ONTOCOM cost drivers to the COLM phases shows that the life cycle model successfully provides on key feature it was designed for: providing transparency about when in the maintenance process ontology engineers are needed. But in contrast to what could have been a potential expectation we cannot declare that ontology engineers are obsolete in most of the phases. Only the newly invented feedback tracking and reporting phases go without ontology engineering experts being present. It is furthermore noteworthy that our assignment exposes the personnel-related cost drivers as the core point of costs in the structured data life cycle. The documentation and scheduling cost drivers always appear in combination. Both, the reasonable usage as well as well-founded feedback from experts and user communities, depend on the time scheduled and spent for an appropriate documentation of a data set. This subsequently influences the efficient further development of a data set conforming to the COLM process model.

7.2 Evaluation of the data set quality framework

This part of the evaluation is dedicated to our usage-dependent data quality framework. We already mentioned that it is the commonly agreed consensus that data quality assessment always remains a subjective and application-dependent task [NLF99, Nau02, BS06]. Hence, any specific selection and measure of data quality dimensions will always remain subject to discussion. The aim of this evaluation is to self-critically acknowledge this aspect by providing a qualitative insight about the fitting of our specific selection to the quality needs of data consumers. This also suits to retrace how the fitting can be improved if other forms of usage data allow to measure additional or other dimensions.

From the empirical data quality assessment approach we selected a set of six dimensions which can be measured based on our usage analysis approach. We are going to discuss this selection with reference to the entire range of the state of the art data quality dimensions as described in [WS96] and [PLW02]. Wang et al. also provide the empirically grounded weighting of these data quality dimensions which reflects their importance for the data consumer [WS96] and was the tipping point for the weighting of our selected dimensions.

7.2.1 Plain score

Table 7.8 documents how we calculated a normalized baseline weighting for the data quality dimensions adapted from [WS96]. As a result of their empirical study the

authors listed the "mean importance" for the data consumer of altogether 20 data quality dimensions which are a superset of the 16 dimensions which established as the commonly agreed state-of-the-art in this area. A lower importance value indicates a higher importance of the respective dimension for the data consumer.

For the plain data quality assessment approach we simply normalized the importance score of all 16 dimensions with reference to 1. This can be achieved by calculating the sum of the inverse value of all importance scores first. The normalized weight of a dimension then is the division of its inverse importance value by the aforementioned sum.

Table 7.8: Plain normalization of the mean importance of the data quality dimensions for the consumer based on the study in [WS96].

DQ dimension	**Mean importance for consumer**	**Inverse**	**Normalized**	**Rounded baseline weighting**
Believability	2.71	0.36900369	0.083038649	0.08
Value-added	2.83	0.35335689	0.079517576	0.08
Relevancy	2.95	0.338983051	0.076282962	0.08
Free-of-Error	3.05	0.327868852	0.073781882	0.07
Interpretability	3.20	0.3125	0.070323356	0.07
Understandability	3.22	0.310559006	0.069886565	0.07
Accessibility	3.47	0.288184438	0.06485151	0.06
Objectivity	3.58	0.279329609	0.062858866	0.06
Timeliness	3.64	0.274725275	0.06182273	0.06
Completeness	3.88	0.257731959	0.057998644	0.06
Reputation	4.04	0.247524752	0.055701668	0.06
Consistent representation	4.22	0.236966825	0.053325767	0.05
Ease of manipulation	4.28	0.23364486	0.05257821	0.05
Concise representation	4.75	0.210526316	0.047375734	0.05
Security	4.92	0.203252033	0.045738768	0.05
Appropriate amount of data	5.01	0.199600798	0.044917114	0.05

To provide a dimension weighting that reflects the baseline weighting presented in [WS96] but only for the dimensions covered by our approach we performed the same normalization method for the subset of six dimensions resulting in all other dimensions to be weighted 0. This weighting is shown in column *plain dimension weighting a* in Table 7.9. In column *plain dimension weighting b* we provide the balanced weighting as an alternative that does not highlight the importance of any specific dimensions.

Table 7.9: Comparison of our plain dimension weighting with the plain baseline weighting derived from [WS96].

DQ dimension	**Baseline dimension weighting**	**Plain dimension weighting a**	**Plain dimension weighting b**
Believability	0.08	0	0
Value-added	0.08	0.23	0.1667
Relevancy	0.08	0	0
Free-of-Error	0.07	0	0
Interpretability	0.07	0.2	0.1667
Understandability	0.07	0	0
Accessibility	0.06	0	0
Objectivity	0.06	0	0
Timeliness	0.06	0	0
Completeness	0.06	0.16	0.1667
Reputation	0.06	0	0
Consistent representation	0.05	0.15	0.1667
Ease of manipulation	0.05	0	0
Concise representation	0.05	0.13	0.1667
Security	0.05	0	0
Appropriate amount of data	0.05	0.13	0.1667

The most significant observation that can be made is that our selection of data quality dimensions covers one of the three dimensions which are of most importance for the data consumer. From the top five dimensions we still cover two but from the top ten only three. This shows that our approach for assessing the quality of a data

set based on log file analysis leaves the potential to be improved to better capture data quality dimensions that are the most important ones for the data consumer.

Furthermore one can see that our data quality dimension selection downgrades the importance of the concise representation and appropriate amount of data dimensions compared to the baseline weighting which regards all the dimensions. This results from the higher effect very small differences in the mean importance have in the normalization procedure when only selected dimensions are regarded. Compared to weighting a the plain weighting b upgrades four of six dimensions while downgrading the other two.

In the remainder of this thesis we will examine the performance of the aforementioned two weighting configurations for the plain data quality score function in more detail. We will refer to them as **plain score a** and **plain score b** according to the labelling in Table 7.9.

7.2.2 Hierarchical score

In the case of our hierarchical data quality assessment approach we performed the same normalization procedure as described above but with reference to a particular data quality category a dimension belongs to. That means that for every category the sum of the inverse importance values of the respectively assigned dimensions is calculated first. The normalized weight of a dimension then is the division of its inverse importance value by the aforementioned sum which relates to a category. The result of the normalization is shown in Table 7.10.

Table 7.10: Hierarchical normalization of the mean importance of the data quality dimensions for the consumer based on the study in [WS96].

DQ category	**DQ dimension**	**Mean importance for consumer**	**Inverse of mean imp.**	**Hierarchical dim. normalization**		**Hier. dim. weighting**
Intrinsic	Believability	2,71	0.36900369	1,223726904	0.301540882	0.30
	Free-of-Error	3,05	0.327868852		0.267926489	0.27
	Objectivity	3,58	0.279329609		0.228261394	0.23
	Reputation	4,04	0.247524752		0.202271235	0.20
Contextual	Value-added	2,83	0.35335689	1,658042833	0.213116865	0.21
	Relevancy	2,95	0.338983051		0.204447704	0.20
	Timeliness	3,64	0.274725275		0.165692508	0.17
	Completeness	3,88	0.257731959		0.155443487	0.16
	Ease of manipulation	4,28	0.23364486		0.140916058	0.14
	Appropriate amount of data	5,01	0.199600798		0.120383379	0.12
Representational	Interpretability	3,20	0.3125	1,070552147	0.291905444	0.29
	Understandability	3,22	0.310559006		0.290092367	0.29
	Consistent representation	4,22	0.236966825		0.2213501	0.22
	Concise representation	4,75	0.210526316		0.196652089	0.20
Accessibility	Accessibility	3,47	0.288184438	0.491436471	0.586412396	0.59
	Security	4,92	0.203252033		0.413587604	0.41

The original data quality approach in [WS96] did not provide a hierarchical weighting but we calculated this as the sum of the normalized dimension weights within a category. This results into the *baseline category weighting* presented in Table 7.11 next to our *category weighting 1* resulting from the decision making process by pairwise comparison presented in Chapter 6 and our *category weighting 2* which represents the inverse emphasis in the importance of data quality categories.

Analog to the procedure before we derived the *dimension weighting a* by normalizing the score of the selected dimensions with reference to all other selected ones but only within the same category in this case. This is again set in comparison to a balanced dimension weighting and the baseline dimension weighting for the hierarchical score.

Table 7.11: Comparison of our hierarchical dimension weighting with the hierarchical baseline weighting derived from [WS96].

DQ category	**DQ dimension**	**Baseline cat. weighting**	**Category weighting 1**	**Category weighting 2**	**Baseline dim. weighting**	**Dim. weighting a**	**Dim. weighting b**
Intrinsic	Believability	0.27	0	0	0.30	0	0
	Free-of-Error				0.27	0	0
	Objectivity				0.23	0	0
	Reputation				0.20	0	0
Contextual	Value-added	0.38	0.17	0.75	0.21	0.44	0.3333
	Relevancy				0.20	0	0
	Timeliness				0.17	0	0
	Completeness				0.16	0.32	0.3333
	Ease of manipulation				0.14	0	0
	Appropriate amount of data				0.12	0.24	0.3333
Representational	Interpretability	0.24	0.83	0.25	0.29	0.41	0.3333
	Understandability				0.29	0	0
	Consistent representation				0.22	0.31	0.3333
	Concise representation				0.20	0.28	0.3333
Accessibility	Accessibility	0.11	0	0	0.59	0	0
	Security				0.41	0	0

The baseline category weighting shows that from a data consumer's perspective the contextual data quality category is the by far most important one followed by the intrinsic and representational data quality categories which are rather similar weighted. We passed on regarding th accessibility category due to the discreteness of the research problems accessibility and security raise in the context of structured Web data. Our comparison shows that this category is of minor interest anyway. For the remaining three categories our decision making procedure resulted in a completely different weighting which caused our decision to pass on the intrinsic data quality category as well (hence weighting it 0) and to highlight the importance of the representational data quality category.

In the upcoming parts of this evaluation chapter we will also analyse the performance of the aforementioned weighting configurations for the hierarchical data quality score functions in more detail. We will refer to the four possible concrete hierarchical data quality functions as **hierarchical score a1**, **hierarchical score a2**, **hierarchical score b1**, and **hierarchical score b2**. In this case 1 and 2 refer to the different category weightings while a and b refer to the different dimension weightings we apply conforming to the labelling of the respective columns in Table 7.11. Together with the plain score functions a1 and a2 introduced before, we will analyze six different quality score functions.

7.3 Experimentation with real world data

This is the third and last part of our evaluation. It is dedicated to provide the reader with an insight into the application of our proposed approach in practice and in comparison to the current state-of-the-art in data set maintenance as a rigorous baseline.

7.3.1 Research data

During the time of our research we successfully acquired real world log files from a number of Linked Open Data providers. All these logs became part of the USEWOD data set [B+11] which we already introduced as the only publicly available reference data set in this research area which gained widely acknowledged momentum[1]. Hence, our research and especially this evaluation is highly reproducible and based on widely accepted reference data.

[1]The USEWOD workshop is held yearly since 2011 coming along with a new version of the log data set each year. The latest version can be obtained here `http://data.semanticweb.org/usewod/2013/challenge.html`

The breadth and depth of the logged information but also the format varies depending on the Linked Data source the log files were obtained from. We focused on three data sets which have proved to be rather actively used and which provide logs conforming to the common log format [W3C]. This makes the logs processable in a rather standardized fashion as supported by our log file preprocessing algorithm. These data sets are DBpedia (DBP), Semantic Web Dog Food (aka SWDF or Dogfood), and Linked Geo Data (LGD).

DBpedia is literally the most prominent Linked Data source containing structured data which is automatically extracted from a large number of language versions of Wikipedia[2] [ABK+08]. Hence, DBpedia is a cross-domain data set because its source is a general online encyclopedia. The project wiki of DBpedia[3] is well maintained and provides a comprehensive version history for download which allows for setting up mirrors of any particular DBpedia version and granularity (e.g. only specific language version or excluding particular link sets) easily. In a change log[4] the project team documents changes on the schema level of DBpedia as well as changes on the extraction and interlinkage framework. This outstanding transparency of the evolution process of DBpedia motivated us to choose DBpedia as the reference project for our case study.

The so-called Semantic Web Dog Food server[5] is a bibliographic data set containing publication records mainly for scientific events in the Semantic Web field but continuously expanding. The evolution of this data set is neither documented that well as DBpedia nor is it possible to download versioned data set dumps. However, SWDF is a data set which has been used in a number of Linked Data showcases providing a good quantity of log entries.

Linked Geo Data[6] publishes geographical structured Web data by wrapping the Open Street Map project and adding links to other Linked Open Data sources such as DBpedia [ALH09]. The project provides versioned data set dumps but no change log documenting the performed maintenance activities. It is worth to mention that LGD is a huge data set because it provides data down to the granularity of geographical points. This requires a massively scalable storage infrastructure and makes it hard to set up a representative mirror.

The log files of the three aforementioned data sets have actually not been available at the same point of time but were provided and further extended in an evolutionary fashion by the data set providers in charge. This process started in the beginning of 2011 and lasts till this day. We must admit that we had to make a selection for a

[2] http://wikipedia.org
[3] http://dbpedia.org
[4] http://wiki.dbpedia.org/Changelog
[5] http://data.semanticweb.org/
[6] http://linkedgeodata.org

particular set of log files to be taken into consideration within this thesis at a point of time which allowed us to perform the time-consuming preprocessing and analysis process. That means that it may be that the latest version of the USEWOD data set contains more recent log files which allow for more comprehensive and more actual analysis.

Resulting from the goals of this experimental evaluation part we derived the following requirements which constrain the random selection of log files from the available ones:

- We need DBpedia log files which correlate with different versions of this data set.
- It must be possible to analyze usage data over a period of time for all three data sets.

Table 7.12 lists the log files which are the foundation for our analysis and describes some general properties of each and altogether.

Table 7.12: Overview of the research data for this experimental evaluation.

Data set	Log file	Extracted queries per log	Analysed queries per log	Analysed queries per data set	Analysed queries
DBpedia	DBP 3.4 2010-01-29	746,090	238,826	3,839,496	6,578,156
	DBP 3.5.1 2010-05-28	360,706	360,351		
	DBP 3.5.1 2010-07-13	252,194	251,657		
	DBP 3.6 2011-02-02	1,894,471	1,835,218		
	DBP 3.6 2011-05-13	768,824	669,252		
	DBP 3.6 2011-06-10	486,507	484,192		
SWDF	Dogfood random period (2009-07-05 - 2010-03-10)	120,469	94,543	1,354,431	
	Dogfood 2011-04-08	1,274,942	1,259,888		
LGD	Linked Geo Data random period (2011-05-23 - 2011-11-25)	1,424,947	1,384,229	1,384,229	

Both, DBpedia and LGD, released nearly realtime live update versions of their data sets as publicly accessible SPARQL endpoints[7]. The SWDF data set is continuously but less frequently updated with bibliographic data of new events provided by the event hosts as RDF files or RDFa embedded in event Web sites. Public SPARQL endpoints of snapshot versions of all three data sets are available but sometimes offering limited resources to avoid overload[8].

For the analysis of older DBpedia logs we instantiated three virtuoso open source triple store instances serving the English version of the so-called DBpedia *core data set* in the 3.4[9], 3.5.1[10], and 3.6[11] versions. The core data set contains "an ontology to model the extracted information from Wikipedia, general facts about extracted

[7] As of writing this thesis both data sets released a live update endpoint at `http://live.dbpedia.org/` and `http://live.linkedgeodata.org/sparql` respectively. However, we observed on 2013-07-18 that the LGD live update service is no more available. We want to mention that we do not know if this is a temporary or permanent issue.

[8] Please refer to the discussion in `http://lists.w3.org/Archives/Public/public-lod/2013Apr/0198.html` on the W3C public-lod mailinglist to get an insight into the restrictions some data set providers apply and how this may effect on query execution against public SPARQL endpoints.

[9] `http://wiki.dbpedia.org/Downloads34?v=hmc#h99-3`

[10] `http://wiki.dbpedia.org/Downloads351?v=13ws#h115-3`

[11] `http://wiki.dbpedia.org/Downloads36?v=ebv#h141-1`

resources, as well as inter-language links"[12] and is described in detail at `http://wiki.dbpedia.org/Datasets?v=bli`. Using the core data set has the effect that our mirrors do not contain all the instance link sets provided by DBpedia as well as the WordNet and YAGO types as part of the schema level. This has the effect that queries which explicitly require resources from these subsets of the DBPedia data fail. But on the contrary this has the advantage that we are provided with another baseline to evaluate the change recommendations derived from our usage analysis.

Due to the problems in setting up representative mirrors for SWDF (dumps not available) and LGD (size of the data set exceeded the capacity of our server infrastructure) we performed all analysis on logs of these two data sets against the live endpoints available at `http://data.semanticweb.org/sparql` and `http://linkedgeodata.org/sparql`. All server-side query execution errors were recorded and it is traceable that such errors appeared that rare that the representativeness of our study is not harmed.

7.3.2 The DBpedia case study

We decided to examine the DBpedia project as an illustrative case study. This case study is dedicated to prove the feasibility of our proposed data set life cycle in the context of a cross-domain open data set which provides Linked Data. The focus is on deriving maintenance recommendations by application of our usage analysis and mining method and comparing the results with the manual effort performed by the DBpedia project team as the state-of-the-art baseline procedure.

We will present some general statistics about the data set and its usage first. Afterwards we demonstrate the application of our quality scoring functions with different weightings which reflect different data set provider foci. The real evolution of the DBpedia data set from its 3.4 version up to the current 3.8 version will be examined with special attention backward compatibility. Finally we present recommended changes resulting from association rule mining.

General statistics

In the context of the usage-dependent approach we presented before the amount of failing queries can be treated as an initial indicator whether any maintenance activity on a data set is necessary. Figure 7.1 depicts this high-level view to the DBpedia data set.

The trend of the curve shows that there are some days with a significant high load of overall queries and some days with a significant high proportion of failing queries.

[12] `http://dbpedia.org/Datasets/NLP#h172-2`

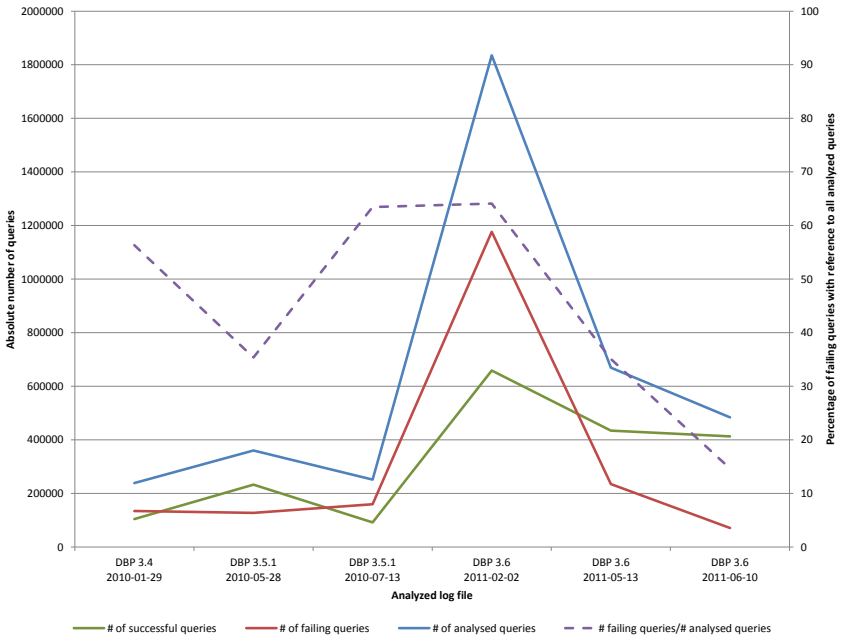

Figure 7.1: Evolution of the proportion of failing queries with reference to all analysed queries using the example of DBpedia.

Both phenomena do not necessarily appear together which means there is not dependence between them in evidence. At this point we rather face the question whether any other aspect causes this conspicuous feature be it either a specific application profile or a data set version step.

A more detailed analysis is necessary which will also involve the schema level of the data set. Figure 7.2 provides an initial insight into some basic properties of the schema level.

On the left we see the evolution of the amount of populated properties compared to the amount of properties which are used in queries against the respective data set version. One can see that (a) maintenance activities have been performed during the data set evolution which affected the schema level so that the amount of populated properties decreased and (b) the amount of populated properties still is circa to the factor 40 higher than the amount of requested properties. That means even though the schema level is more comprehensive than required there may be blind spots which are not sufficiently represented.

The graph on the right shows the relation between failing queries and failing

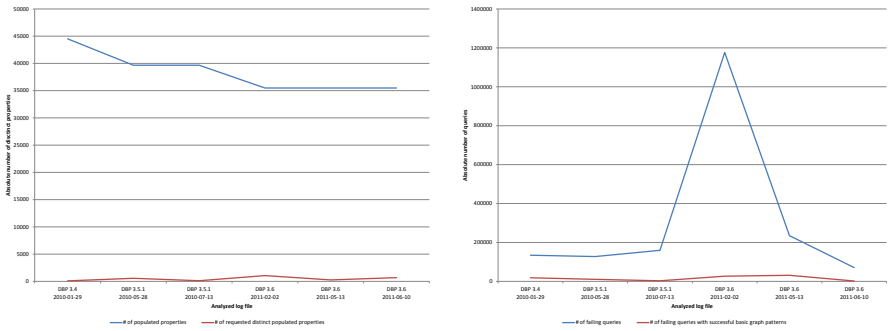

Figure 7.2: Basic schema level statistics: The amount of requested populated properties compared to the amount of all populated properties (left) and the amount of failing queries with at least one successful basic graph pattern compared to the amount of all failing queries (right).

queries which contain at least one graph pattern which succeeds when being executed as a standalone query. This latter form of failing queries can be exploited to find out how the user puts parts of the data set in relation to other data which is not represented. The curves allow for the interpretation that there continuously is an amount of such queries which relates to a niche in the data because the relation to the amount of all failing queries is not proportional.

The data quality score with reference to the evolution of a data set

In the previous section we presented and evaluated different weightings assigned to the generic plain and hierarchical data quality score functions presented in Chapter 6. These weightings reflect the data set provider's focus on particular quality dimensions or categories respectively. Figure 7.3 shows the evolutionary performance of the DBpedia data set with respect to these concrete plain and hierarchical quality score functions.

The most significant observation one can make is that the hierarchical score a1 and b1 show the least oscillation while the other four curves show a much broader range. They reflect much more sensitive the influence of particular category or dimension scores on the overall score. This feature is also stressed by the fact that, while the trend of all six curves is similar, the four more sensitive scores alternate around the hierarchical score a1 and a2.

The upcoming diagrams will examine this sensitivity in more detail. Figure 7.4

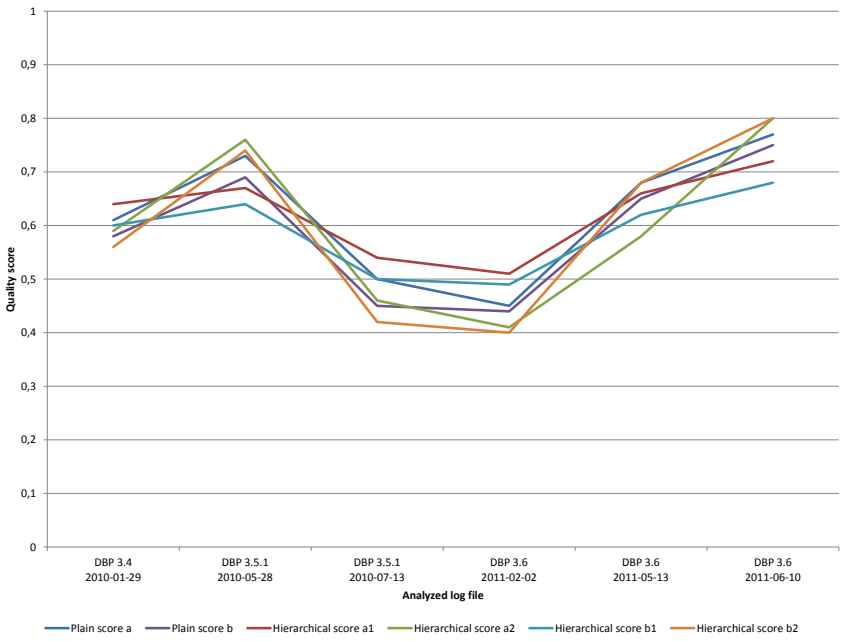

Figure 7.3: Results of the different data quality score functions in the context of the DBpedia data set evolution.

dives into the two covered data quality categories for the hierarchical scoring approach. We observe that the contextual data quality dimensions cause the oscillation of the overall score.

This observation is stressed by Figure 7.5 which shows the significantly more oscillating curves of the appropriate amount of data, value-added, and completeness scores which constitute the contextual data quality category.

From the representational data quality category only interpretability features a broader range. As expected the conciseness score is continuously low because as mentioned before the amount of requested properties was continuously to factors lower than the amount of populated properties.

Backward compatibility

Backward compatibility of data sets plays a key role in productive application environments. Maintenance activities performed on a data set can cause queries to fail which were successful before. As a consequence of this it may be that applications that use a data set get harmed in an unanticipated fashion.

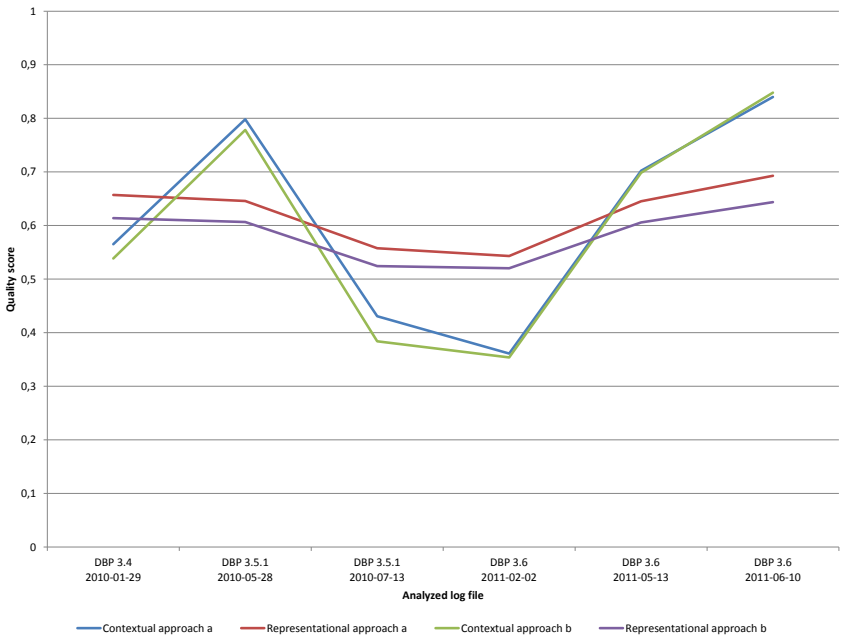

Figure 7.4: Comparison of the score of the contextual and representational data quality categories.

We already described that we instantiated data set mirrors for the DBpedia core data set version 3.4, 3.5.1, and 3.6. In order to analyze the evolution beyond version 3.6 we additionally consulted the public SPARQL endpoint at `http://dbpedia.org/sparql` which hosts the latest version of the DBpedia data set which was 3.8 as of writing this thesis. In contrast to our mirrors this data set contains not only the core data set but the entire data which is listed on `http://wiki.dbpedia.org/DatasetsLoaded?v=tbg`. The most obvious difference is that this includes all link sets, SKOS categories, and labels as well as abstracts in a large number of languages.

Figure 7.6 shows that the amount of populated properties in the DBpedia core data set was continuously decreasing over time. Furthermore one can see that the public endpoint (3.8 live[13]) features a larger amount of populated properties. This is most likely due to the extended data set served at this endpoint but it may also be that the endpoint is configured to materialize inferred triples.

[13]We use the term *3.8 live* in figures for this endpoint to highlight that this is the official, open accessible endpoint. The term *live* does not imply that we use the endpoint of the DBpedia live update server available at `http://live.dbpedia.org/sparql`.

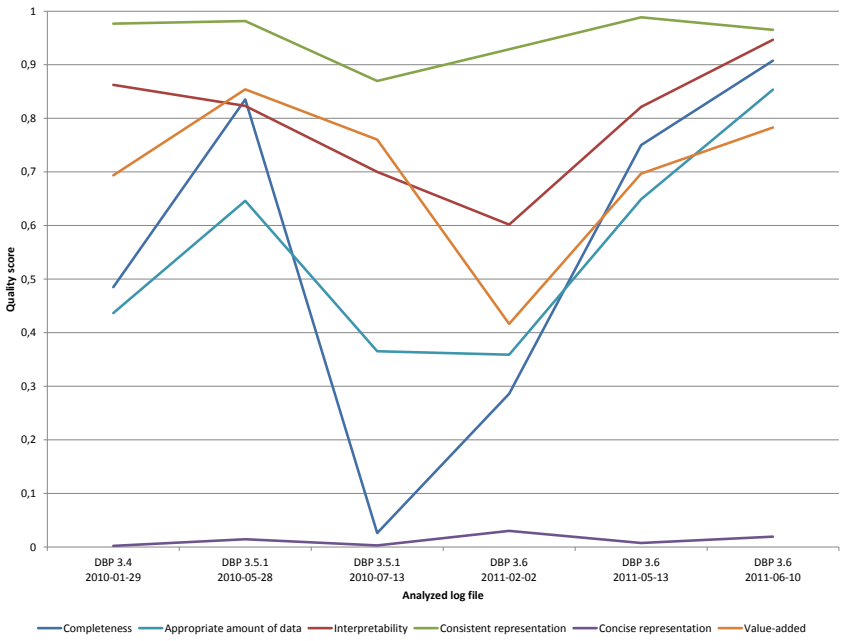

Figure 7.5: Comparison of the score of all six data quality dimensions.

This raises the question whether a more comprehensive schema level, which results in more served data, necessarily results in more queries that can be answered. We will analyze this in detail by comparing the evolution of some general properties as well as the quality score for the log files DBP 3.4 2010-01-29, DBP 3.5.1 2010-05-28, and DBP 3.6 2011-05-13. We preprocessed and analyzed each of theses logs against the data set version to which the logged time period refers to but also to succeeding data set versions. In the remainder of this section any three figures placed in parallel conform to the chronology of the three aforementioned log files: DBP 3.4 2010-01-29 (top left), DBP 3.5.1 2010-05-28 (top right), DBP 3.6 2011-05-13 (bottom).

Figure 7.7 presents the high-level statistics for the evolution of the performance of queries from the three logs. One can see that the version step to DBpedia 3.6 significantly harms successful queries of the past. This phenomenon can also be observed for the 3.8 version step with reference to the 3.5.1 and the 3.6 logs in our study. However, with reference to the DBpedia 3.4 log the 3.8 version step cures some of the aggravation caused by the 3.6 version step. The significant declension of the number of analyzed queries from the 3.5.1 log when being performed against the 3.8 live endpoint is due to internal server errors which were caused by complex

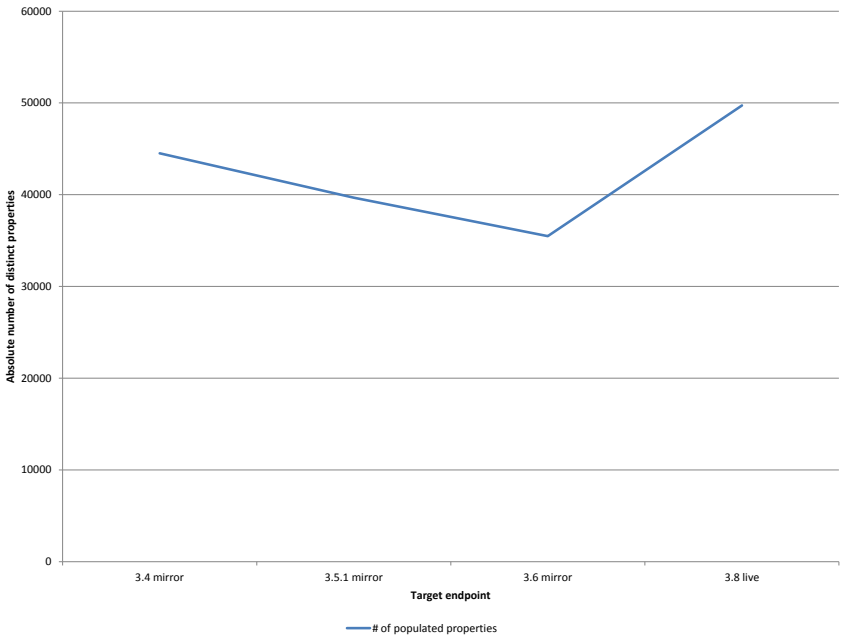

Figure 7.6: Evolution of the amount of populated properties served by the DBpedia data set endpoints of our study.

queries. But, the slight gain in the proportion of failing queries with reference to the amount of analysed queries still confirms the aggravation trend.

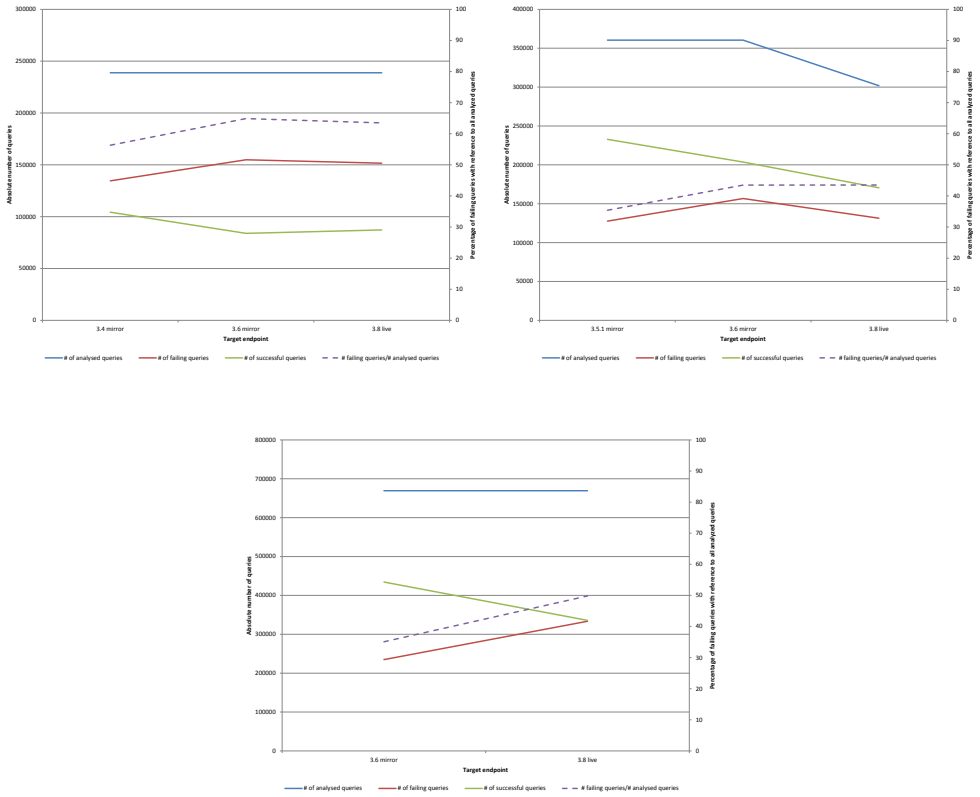

Figure 7.7: Evolution of the performance of queries from the three DBpedia logs.

This previous observations foment rather negative expectations about the evolution of the quality of the DBpedia data set with reference to its usage. Again we consult the specific form of failing queries which contain at least one successful graph pattern to get a more detailed insight if this trend can be generally approved on all levels of granularity of queries.

Figure 7.8 depicts that in case of the 3.4 log we can observe a continuous gain of failing queries which contain at least one successful graph pattern and also the 3.8 version step has this effect with reference to the 3.5.1 log.

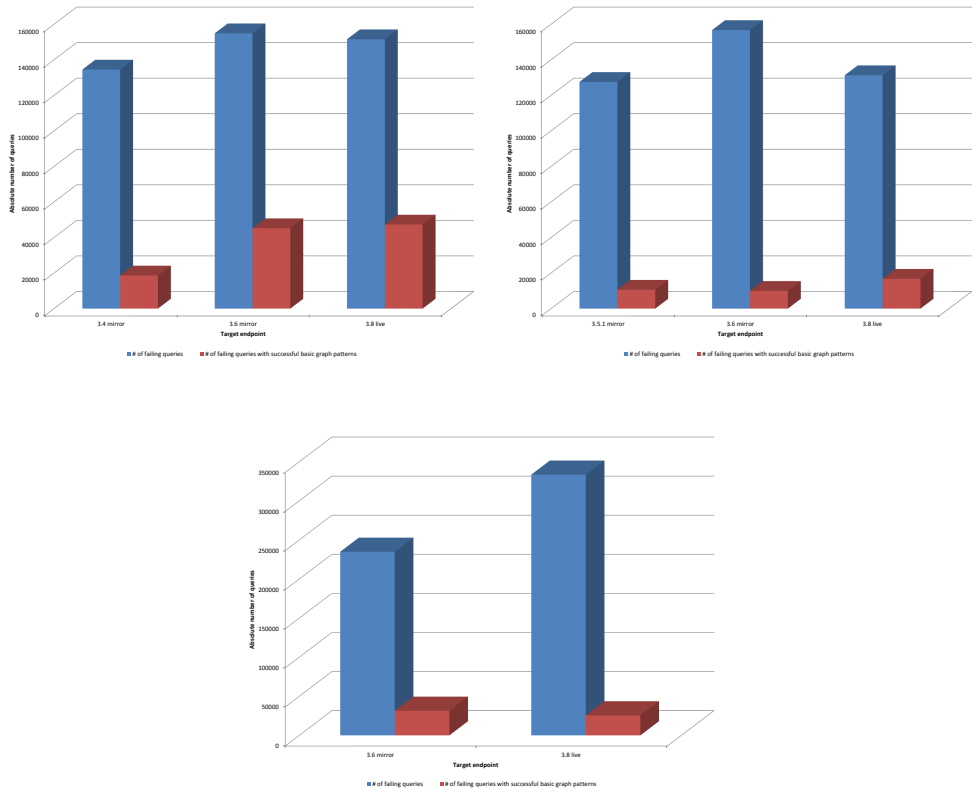

Figure 7.8: Evolution of the amount of failing queries which contain at least one successful graph pattern.

It turns out that the high-level statistics of the amount and proportion of failing queries suits as an indicator about the backward compatibility of changes only on the query level but that maintenance activities also may have effects on parts of queries. This raises the question if and how our data quality score functions bear this specific characteristic.

Figure 7.9 shows the evolution of our six quality score functions for the three analysed log files over the time of the data set evolution. Considering the 3.4 log the quality score follows very much the evolution of the amount of failing queries which contain at least one successful graph pattern. The better the proportion of these queries with reference to all analyzed queries the better the score. The hierarchical

score a1 for the 3.5.1 log also shows this characteristic but all other score functions for this log don't. The 3.6 log shows the contrary yet universally.

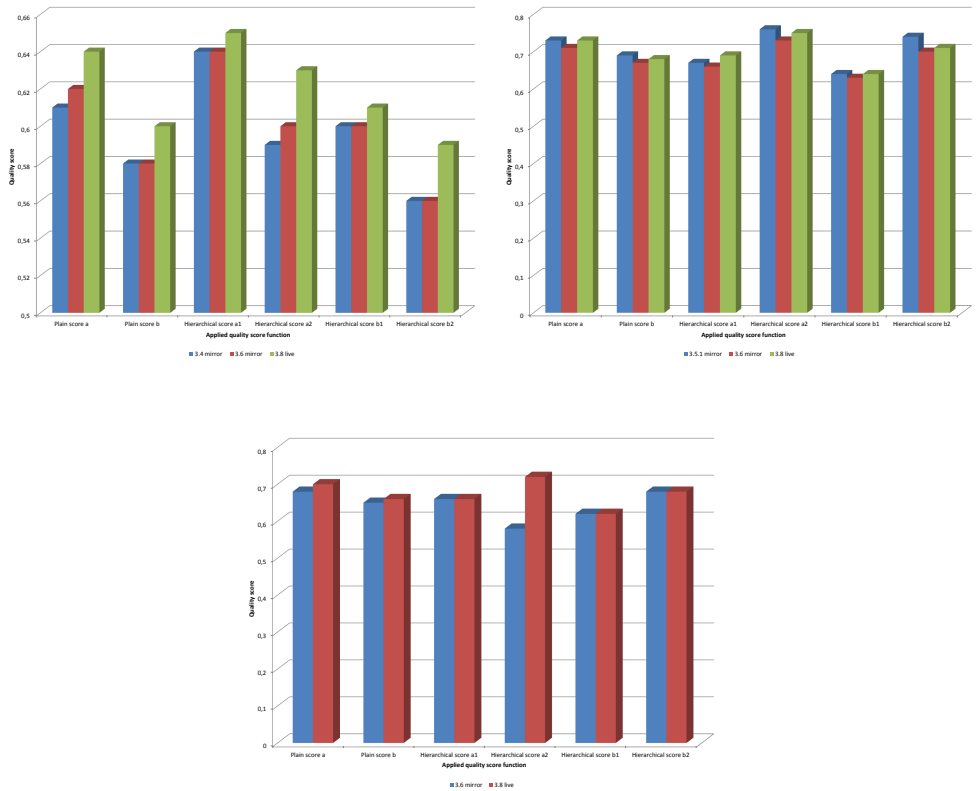

Figure 7.9: Evolution of the quality score functions for particular log files over the time of the DBpedia data set evolution.

We also observe that the score functions for the 3.4 log develop differently than for the other two logs. The most significant property is that hierarchical scores with category weighting 2 are lower than the respective score with category weighting 1 and that hierarchical scores with dimension weighting b are lower than those with dimension weighting a. This lets us conclude that the DBpedia data set has a better representational than contextual data quality with reference to the 3.4 log and that the reverse seems to hold for the 3.5.1 and 3.6 logs. The comparison shown in

Figure7.10 confirms this.

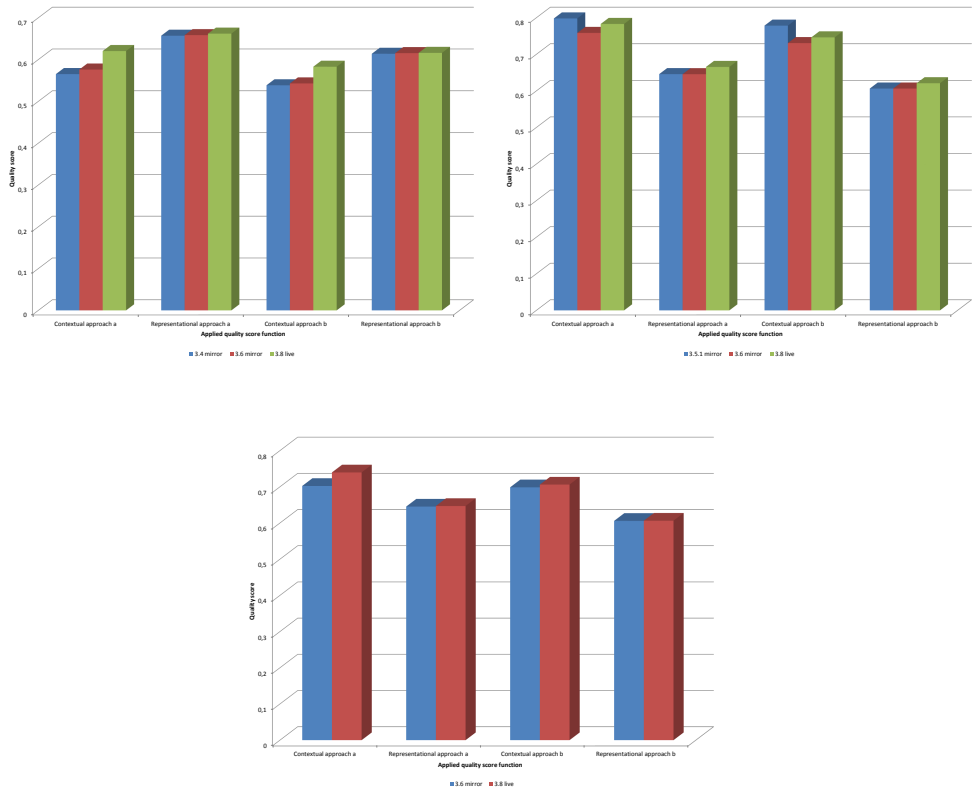

Figure 7.10: Evolution of the contextual and representational quality score for particular log files over the time of the DBpedia data set evolution.

At this point we touch the fundamental data quality dimensions again in order to analyse which dimension reflects which characteristics of a data set with reference to the usage captured in a particular log file (see Figure 7.11).

The two most intuitive observations are that the value-added and the interpretability scores increase with reference to the 3.8 live endpoint. This conforms to the specific character of the served data at this endpoint which contains more resources in general due to the integration of the link sets (affecting the value added-dimension) as well as much more labels and abstracts in many different languages

(affecting the interpretability dimension).

A second observation is that both, the appropriate amount of data and the completeness dimensions, are significantly low for the 3.4 log. Hence, the explanation for the weak contextual score for DBpedia with reference to this log is most likely the lack of requested schema level primitives. That the completeness dimension has this impact is stressed by the following observation which can be made consulting Figure 7.10 again: an increasing completeness score correlates with an increasing contextual score (category score evolution with reference to 3.4 log) and vice versa (category score evolution with reference to 3.5.1 log).

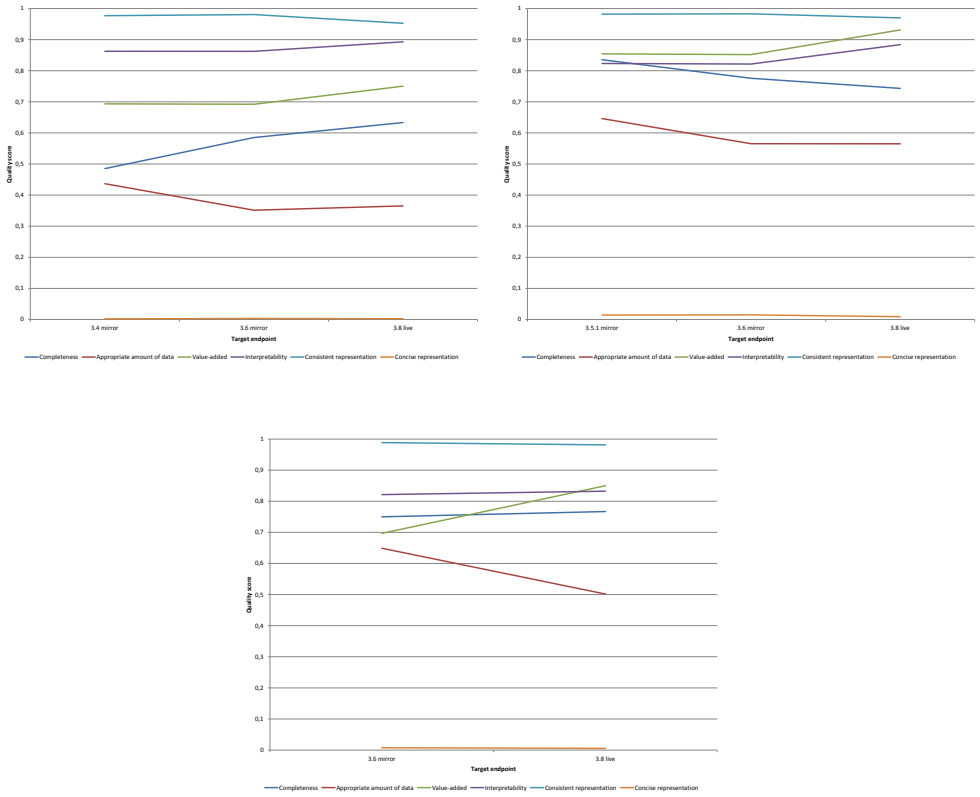

Figure 7.11: Evolution of the score of all six data quality dimensions for particular log files over the time of the DBpedia data set evolution.

Estimating the effects of change

In the preceding section we have shown a form of an a posteori evaluation of maintenance activities by analyzing the backward compatibility of data set versions. The study also emphasized the importance of the completeness dimension which is directly related to the schema level of a data set. In this part of the DBpedia case study we demonstrate how to measure and anticipate the effects of changes in an a priori fashion as a means for planning and managing data set maintenance.

We have shown that the overall amount and proportion of failing queries with reference to all analyzed queries suits as a high-level indicator of the quality of a data set. Especially in closed and productive scenarios this role is upgraded if only reasonable queries are performed and query success is the superordinate target to avoid application failures. We propose a very simple method to estimate the effects of changes based on the following assumptions:

- If a resource or property is added to a data set the best effect this can have is that a failing query that contains it becomes successful.
- If a resource or property is deleted from a data set the worst effect this can have that a successful query that contains it becomes a failing query.

Consequently we measure the effect of change as the difference based on the appropriate amount of data score function as follows:

$$\epsilon(D, Q_A, R_a, R_d) = (1 - (\frac{|Q_F(Q_A, D)| + x - y}{|Q_A|})) - (1 - (\frac{|Q_F(Q_A, D)|}{|Q_A|}))$$

where

R_a is a set of URIS to be added to a data set,
R_d is a set of URIS to be deleted from a data set,
x is the amount of queries from Q_S which contain a URI from R_d,
y is the amount of queries from Q_F which contain a URI from R_a (7.1)

The DBpedia project provides a comprehensive change log[14] which is an outstanding example for good data set documentation. However, most of the entries refer to changes performed on the data population framework in order to improve the extraction performance and quality. While it may be that there are implicit schema level changes resulting from the further development of the extraction framework only a limited number of schema level changes performed by the DBpedia team is explicitly

[14] http://wiki.dbpedia.org/Changelog?v=3ey

listed. Table 7.13 lists the retraceable schema level changes for the DBpedia 3.4 to 3.7 evolution and the estimated effect of the respective changes with reference to a particular log of our study. We regard these values which reflect the manually performed maintenance as a baseline to compare our change recommendations against.

Table 7.13: The effects of real changes performed by the DBpedia project team with reference to the tracked usage.

Evolution step	Change description	Effects of change	Reference log
3.4 to 3.5 (04/2010)	The URI for extended abstracts was changed from http://dbpedia.org/property/abstract to http://dbpedia.org/ontology/abstract.	-0.025797024	**DBP 3.4 2010-01-29**
3.5.1 to 3.6 (01/2011)	http://dbpedia.org/ontology/deathPlace replaces http://dbpedia.org/property/deathPlace	-0.000138754	**DBP 3.5.1 2010-05-28**
	http://dbpedia.org/ontology/deathDate replaces http://dbpedia.org/property/death	-0.000699318	
	http://dbpedia.org/ontology/birthPlace replaces http://dbpedia.org/property/birthPlace	-0.001029552	
	http://dbpedia.org/ontology/birthDate replaces http://dbpedia.org/property/birth	-0.002428188	
	http://xmlns.com/foaf/0.1/givenName replaces http://xmlns.com/foaf/0.1/givenname	-0.001082278	
	http://purl.org/dc/terms/subject replaces http://www.w3.org/2004/02/skos/core#subject	-0.047081873	
	http://www.w3.org/2004/02/skos/core#related is extracted as a relation between categories.	0	
	http://dbpedia.org/ontology/wikiPageID replaces http://dbpedia.org/property/pageId	0	
	http://dbpedia.org/ontology/wikiPageRevisionID replaces http://dbpedia.org/property/revisionId	0	
	http://dbpedia.org/ontology/wikiPageWikiLink replaces http://dbpedia.org/property/wikilink	0	
	http://dbpedia.org/ontology/wikiPageExternalLink replaces http://dbpedia.org/property/reference	-0.014524727	
	http://dbpedia.org/ontology/wikiPageRedirects replaces http://dbpedia.org/property/redirect	-0.000491188	
	http://dbpedia.org/ontology/wikiPageDisambiguates replaces http://dbpedia.org/property/disambiguates	0	
3.6 to DBpedia 3.7 (08/2011)	skos:related for category links starting with :" and having and anchor text	$8.71831 * 10^{-6}$	**DBP 3.6 2011-02-02**
	owl:equivalentClass and owl:equivalentProperty mappings to http://schema.org	0	

The analysis shows that most of the changes have a minimal negative or no estimated effect with reference to the particular log files in this study. The most significant change is the replacement of the `http://www.w3.org/2004/02/skos/core\#subject` property by `http://purl.org/dc/terms/subject` which yields a negative estimation of nearly 5% on the appropriate amount of data score. The only but very small positive change estimation value is calculated for the extension of the usage of `http://www.w3.org/2004/02/skos/core\#related` for specific category links.

We introduced different kinds of association rule mining approaches which are feasible based on our usage database. Resulting from the specific potential this case study assigned to the completeness dimension, we decided to present the results of computing *instance level associations* by application of the *failing query restriction* criterion. That means that we treat those queries as atomic transactions which fail. We use the high-level visualization (cf. Figure 7.12) based on the proposal in [HGHB09] as the entry point for a detailed inspection of changes that can be derived from the rules.

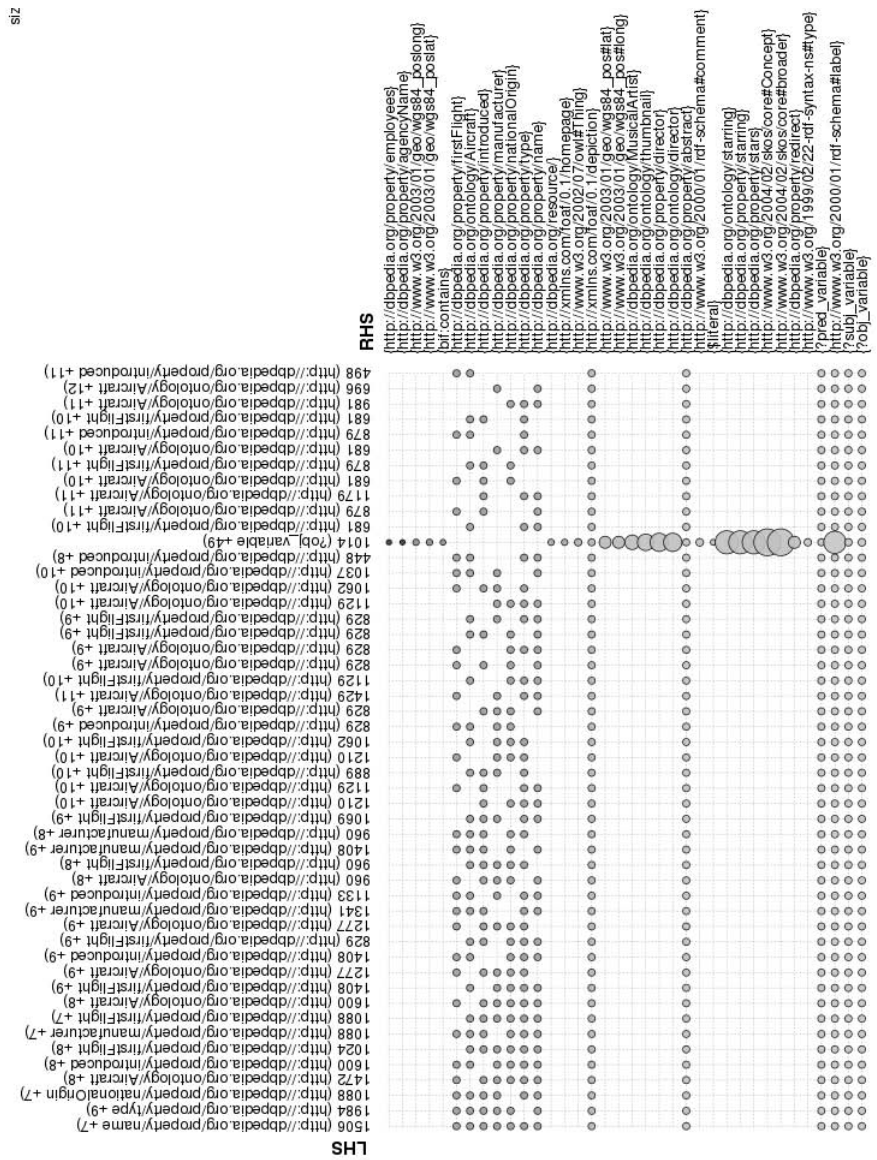

Figure 7.12: Visualization of instance level association rules computed on the DBP 3.4 2010-01-29 log by application of the failing query restriction criterion (size: support, color: lift).

The visualization shows how primitives on the left hand side (LHS) of a rule imply particular ones on the right hand side (RHS) and which likelihood such an association has. In our specific case this allows us to analyze which primitives are queried together frequently in failing queries. We spot two characteristic usage patterns: (1) the properties and classes queried in the context of `http://dbpedia.org/ontology/Aircraft`; (2) the properties and classes queried in the context of an object variable. These can be further analyzed by exporting the association rules to GraphML and visualizing the network by use of a network visualization and analysis tool like Gephi[15] for example. Figure 7.13 depicts one filtered network representation for our example case. Nodes with a degree lower than 5 are filtered out (k-core network with $k = 5$) to derive a well-arranged visualization of the most important primitives in failing queries. Nodes represent LHS and RHS of the computed rules. Edges point from the LHS to the RHS of the particular rules.

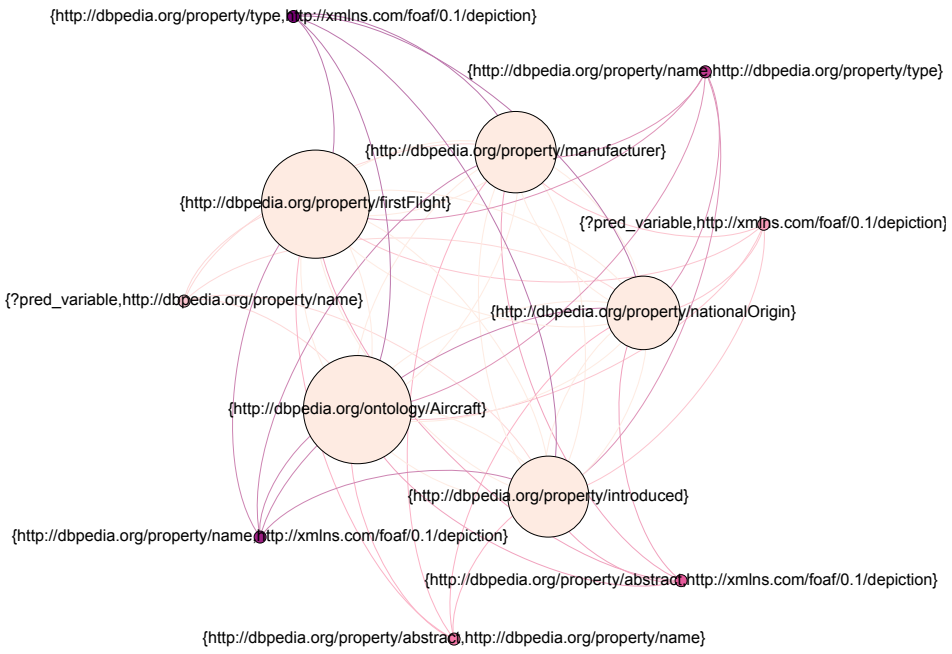

Figure 7.13: Filtered visualization of the association rule network (k-core 5 filter applied to reduce nodes with degree lower than 5).

Table 7.14 lists the an exemplary set of primitives which would be recommended

[15] `http://gephi.org/`

to be added to the DBpedia 3.4 data set conforming to our approach[16].

Table 7.14: Recommended predicates to be added to the data set and the estimated effects of change.

Primitive to add	Effects of change	Exists in data set
dbp:manufacturer	0.004505372	x
dbp:firstFlight	0.004505372	x
dbp:introduced	0.004505372	x
dbp:nationalOrigin	0.004505372	
dbo:thumbnail	0.021986718	x
dbo:director	0.025047524	
dbp:director	0.02503915	x
dbp:abstract	0.025797024	x
dbo:starring	0.034066643	
dbp:starring	0.034066643	x
dbp:stars	0.034066643	x
skos:Concept	0.040946128	x
skos:broader	0.04116386	x
dbp:redirect	0.066441677	x

Since this change recommendation is only additive it is clear that no negative effects are estimated. However, it is possible to estimate the positive potential of a change and consequently to prioritize the changes to be performed in case of conflicting or contradicting recommendations.

More complex and also subtractive change recommendations may emerge from additive ones. This is typified by the recommendation to add dbo:director and dbp:director for example to the data set which appear to be contradicting. Hence, they should be either matched to each other by an owl:equivalentProperty relation or one of the two should be eliminated.

7.3.3 Further data set analysis

Our case study has shown how the usage-dependent data set maintenance approach performs in the context of a cross-domain data set like DBpedia. We will now present results from our studies with SWDF and LGD as two different domain-specific data sets.

[16] To save space we apply the following namespace prefixes in addition to the ones defined before: dbo:`http://dbpedia.org/ontology/`, dbp:`http://dbpedia.org/property/`.

Figure 7.14 shows a comparison of the general statistics about successful and failing queries across all analyzed data sets. The most intuitive observation one can make is the significantly high percentage of failing queries in the context of the SWDF data set.

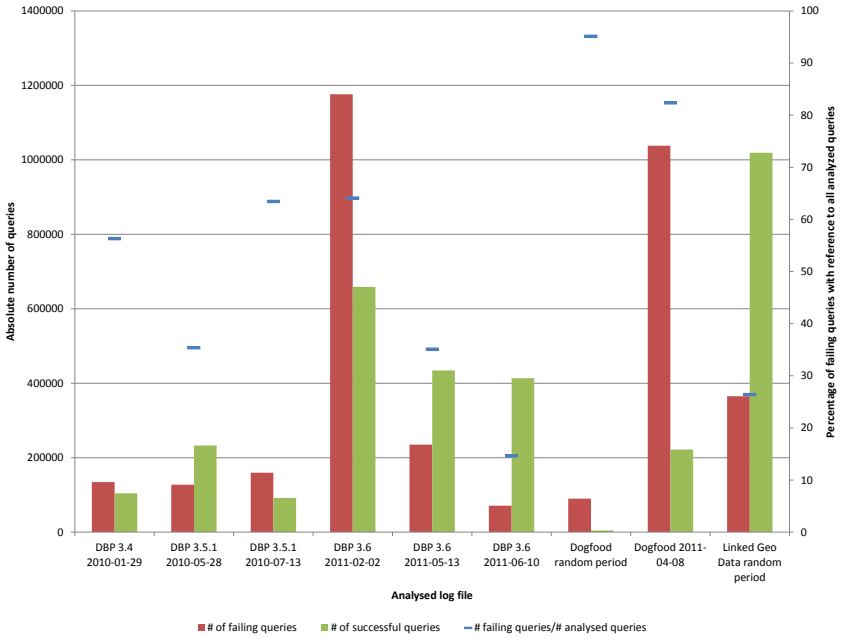

Figure 7.14: Comparison of the general statistics about successful and failing queries across all analyzed data sets.

That the usage of the SWDF data set seems to differ from the usage of DBpedia and LGD is further stressed by the fact that the amount of failing queries with at least one successful basic graph pattern amongst all failing queries is significantly low as well (cf. Figure 7.15).

Again we consult the different data quality score functions in order to see how the aforementioned observation affects them (cf. Figure 7.16). The gap between the hierarchical scores that exploit the category weighting 1 and those exploiting the category weighting 2 indicates that the SWDF data set reports a much better representational than contextual data quality with reference to the analysed logs because category weighting 2 upgrades the importance of the contextual data quality category.

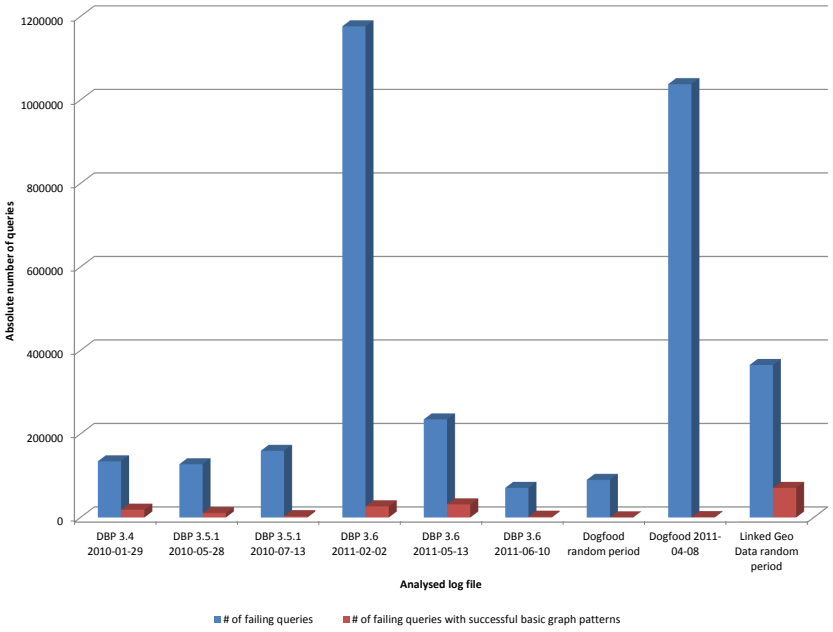

Figure 7.15: Amount of failing queries which contain at least one successful graph pattern compared to the overall amount of failing queries.

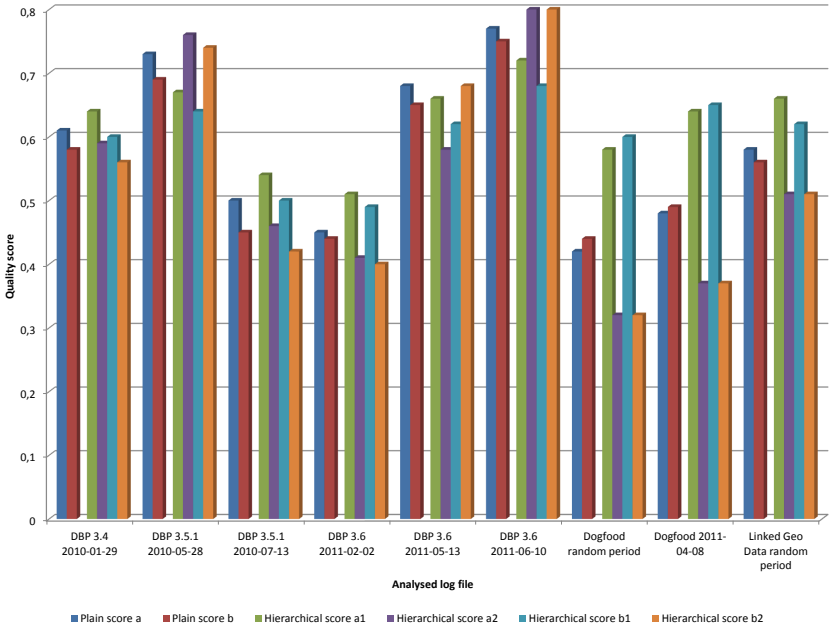

Figure 7.16: Comparison of the data quality score functions across all analysed data sets.

That this characteristic of the SWDF data set is outstanding is emphasised by Figure 7.17. The LGD and two DBpedia logs show a trend into this direction but the gap between the contextual and the representational scores of the SWDF data set remains unequaled. At this point it is reasonable to assume that this specific data set is shaped very tight to the covered domain on the conceptual level but lacks the required breadth on the instance level.

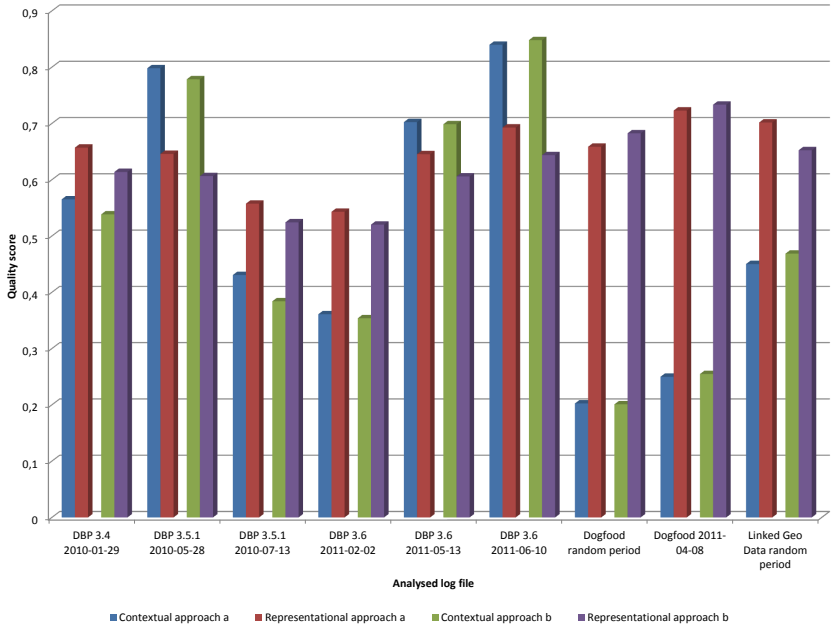

Figure 7.17: Comparison of the contextual and representational data quality scores across all analysed data sets.

We finally examine the data quality dimension scores for all analyzed logs as depicted in Figure 7.18. In conjunction with the aforementioned observations this statistic makes clear that the SWDF data set reports a very concise representation but on the contrary a limited appropriate amount of data and value-added score.

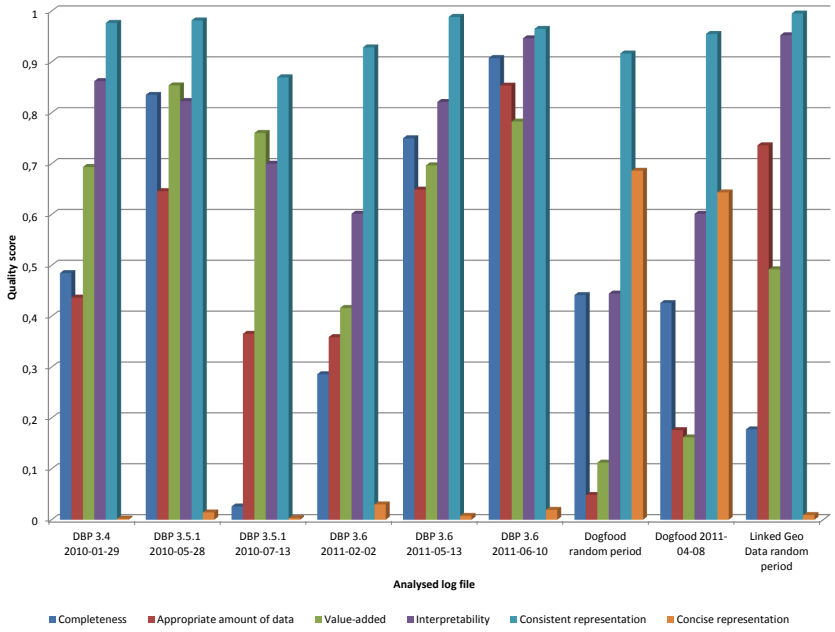

Figure 7.18: Comparison of the basic data quality dimension scores across all analysed data sets.

The analysis of the data quality score functions allows the conclusion that the SWDF data set either is used in a strange way or serves an insufficient amount of schema and instance level primitives. We computed the *instance level associations* without any restricting criterion on all queries because the amount of failing queries amongst all is large. The visualization in Figure 7.19 shows that it is not possible to identify characteristic usage patterns as in the DBpedia case study example. However, one can see that a significant amount of DBpedia resources is queried. Thus, the high proportion of failing queries most likely results from applications that perform queries against the SWDF data set which are originally meant for DBpedia.

That one has to be careful with interpreting this analysis result as a *misuse* of a data set is shown by Figure 7.20 which depicts the association rules computed by application of the *unknown predicates restriction* in the context of the analyzed LGD log file. Again a couple of DBpedia instances are queried (e.g. dbpedia:Berlin and dbpedia:Germany) for which can be assumed that they have a relation to the geographical data served in the LGD data set. And also a DBpedia schema level primitive is queried – http://dbpedia.org/ontology/country. Hence, potential links to another data set on the schema and the instance level are uncovered.

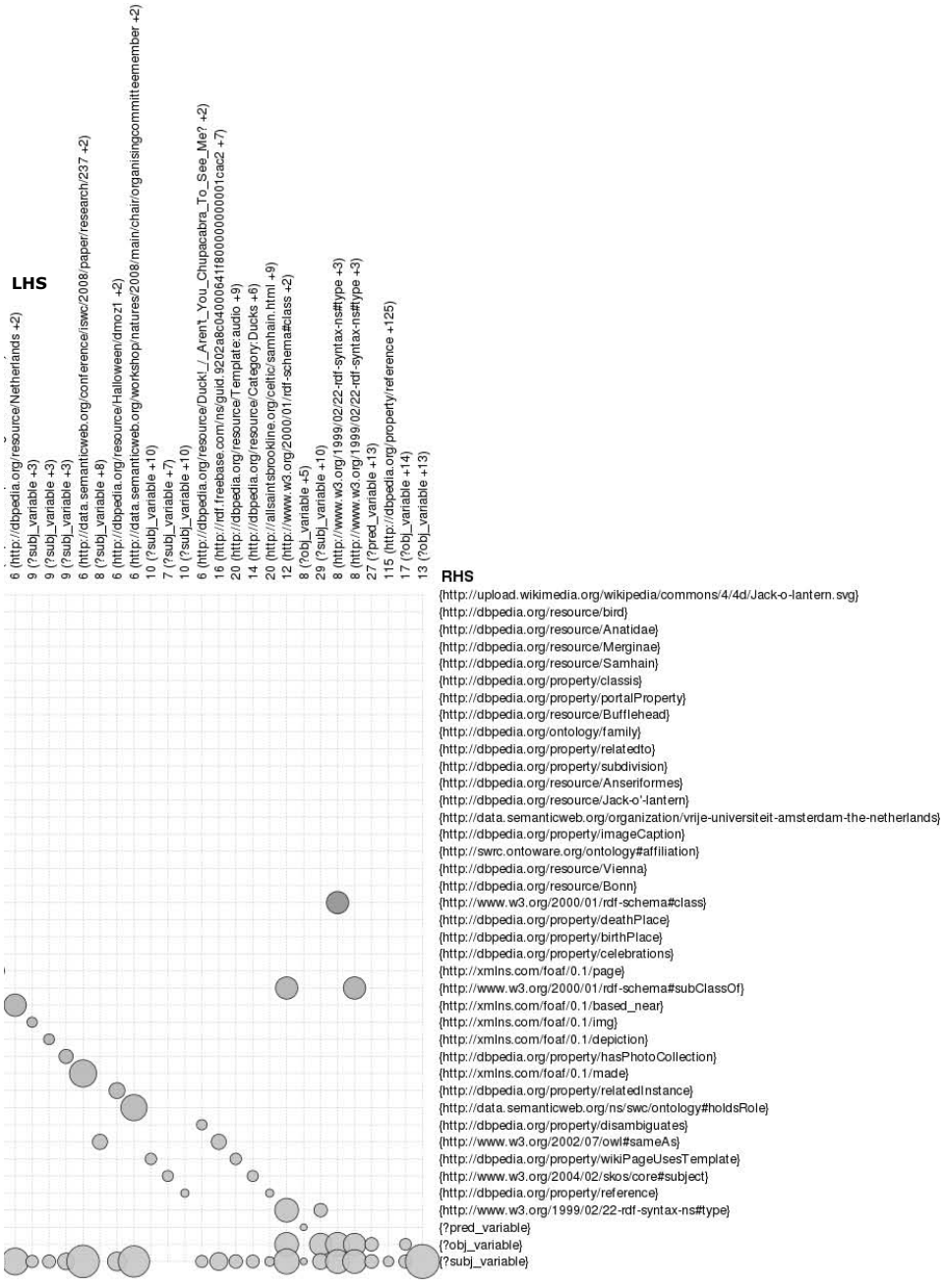

Figure 7.19: Excerpt of the visualization of association rules computed from all queries of the Dogfood 2011-04-08 log focusing the RHS (size: support, color: lift, LHS partially cropped).

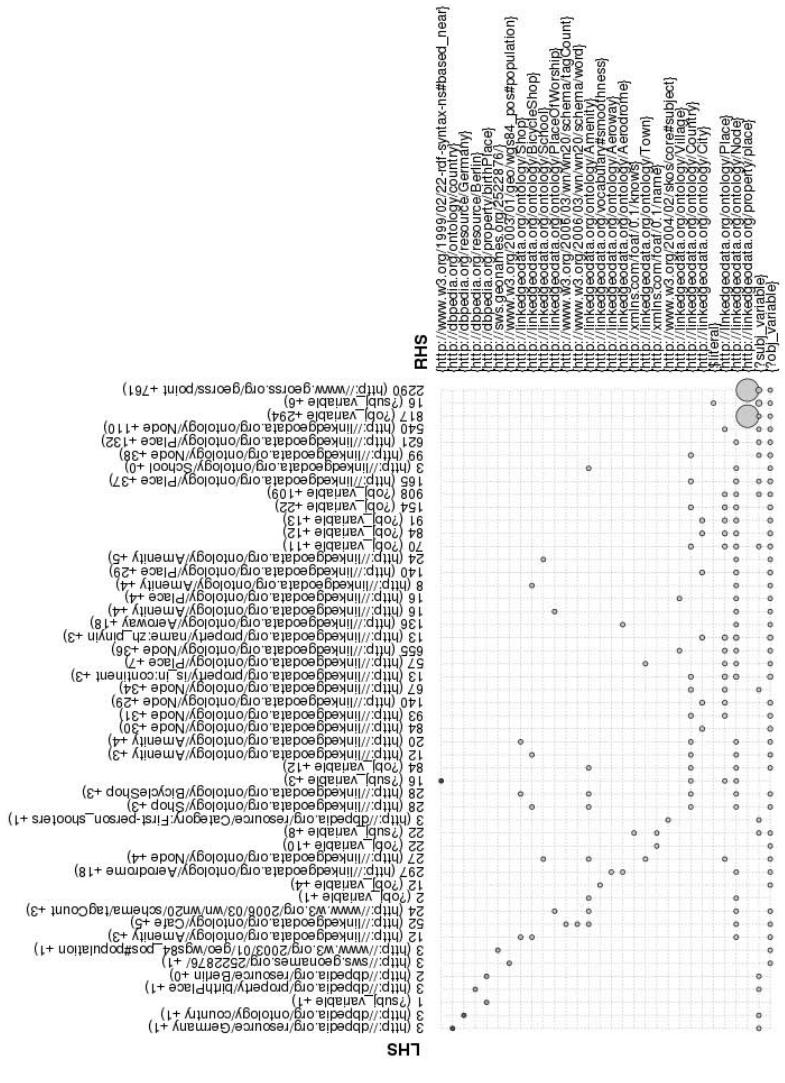

Figure 7.20: Visualization of association rules computed by application of the unknown predicates restriction in the context of the LGD log file (size: support, color: lift).

Picking up the association rules computed for the SWDF log again shows that also this log allows for the identification of meaningful cross-data-set links, e.g. by adding a triple which relates the resources http://data.semanticweb.org/organization/vrije-universiteit-amsterdam-the-netherlands and dbpedia:Netherlands to each other.

7.3.4 Statistical testing of the data quality dimension scores

We apply the Pearson product-moment correlation coefficient (PPMCC) in order to analyse the linear dependency of any two data quality dimension score functions. This helps to determine these score functions which are most likely to cover discrete aspects of data set usage.

The PPMCC between two variables takes a value between -1 and $+1$ with the following properties: -1 indicates that a linear equation exists which perfectly describes the dependency that if the one variable increases the other variable decreases; $+1$ indicates that a linear equation exists which perfectly describes the dependency that if the one variable increases the other variable increases as well; 0 indicates that the two variables do not show any linear dependency. Positive and negative intermediate values indicate that the determined linear equation only approximates the dependency because not all data points lie on the line. The closer the correlation coefficient is to 0 the more scattered are the data points.

Table 7.15 shows the correlation matrix for the calculated data quality dimension scores for the analysed DBpedia log files.

Table 7.15: PPMCC matrix for the analysed DBpedia log files.

	COMP	**AMOUNT**	**VALUE**	**INTER**	**CONSIST**	**CONCISE**
COMP	1	0.7363403	0.5377858	0.7379399	0.8564157	0.0847839
AMOUNT	0.736340307	1	0.433802	0.5059909	0.4392195	0.314677
VALUE	0.537785827	0.433802	1	0.645882	0.3024937	-0.4288847
INTER	0.737939911	0.5059909	0.645882	1	0.6084294	-0.430855
CONSIST	0.856415661	0.4392195	0.3024937	0.6084294	1	-0.0437975
CONCISE	0.08478391	0.314677	-0.4288847	-0.430855	-0.0437975	1

Drawing a bar chart for this correlation matrix results into the visualization shown in Figure 7.21 which makes it easier to identify the independent variables. In case of the analysed log files of the DBpedia data set one can see that the completeness and conciseness score functions are most likely to be linearly independent just as well as the consistency and conciseness score functions. Additionally the correlation coefficient of 0.86 for the completeness and consistency score functions is rather high, indicating that these two score functions are linearly dependent in a increasing way.

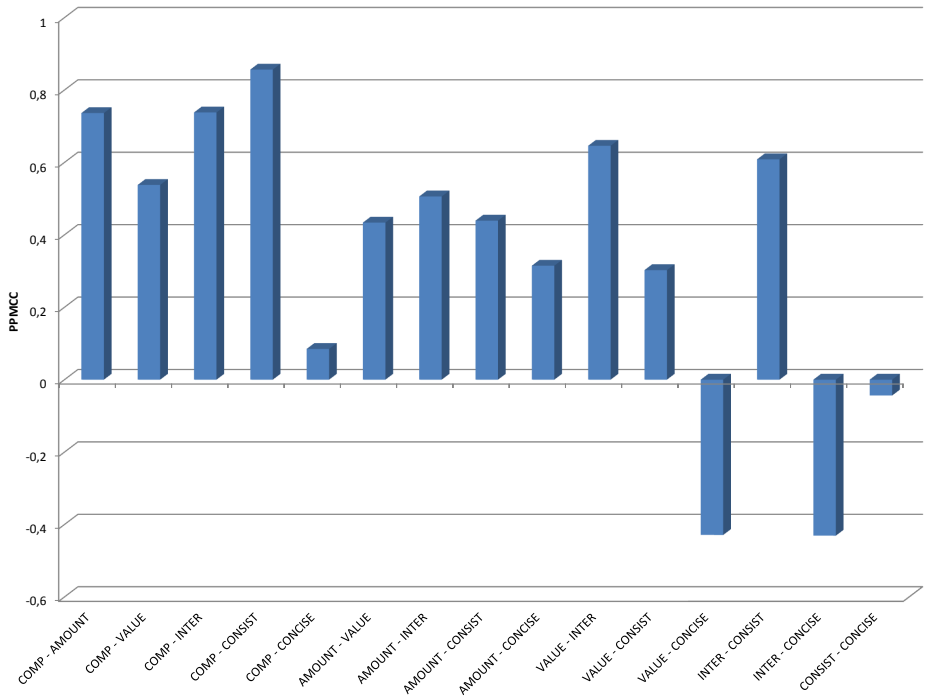

Figure 7.21: PPMCC for the analysed DBpedia log files as a bar chart visualization.

To examine the correlation of any two dimension score functions in a broader context than just the DBpedia data set we created a PPMCC correlation matrix for all analysed log files of all three data sets (cf. Table 7.16). The result can be visualized as a bar chart again which is depicted in Figure 7.22.

Table 7.16: PPMCC matrix for all analysed log files.

	COMP	AMOUNT	VALUE	INTER	CONSIST	CONCISE
COMP	1	0.4371944	0.5083479	0.4135023	0.6094031	-0.198844
AMOUNT	0.4371944	1	0.6537773	0.8088027	0.5606356	-0.6998659
VALUE	0.5083479	0.6537773	1	0.7571065	0.3168773	-0.8481727
INTER	0.4135023	0.8088027	0.7571065	1	0.6428217	-0.7769784
CONSIST	0.6094031	0.5606356	0.3168773	0.6428217	1	-0.309693
CONCISE	-0.198844	-0.6998659	-0.8481727	-0.7769784	-0.309693	1

The most significant change compared to the aforementioned results is that the completeness and consistency score functions do not show the high correlation coefficient anymore. Instead the correlation coefficient of the value-added and conciseness

dimensions (rounded −0.85) now indicates an opposing linear dependency. The linear independence of the completeness and conciseness as well as the consistency and conciseness score functions is not that clear as it was analyzing the DBpedia data in isolation. The value-added and consistency dimension score functions now have a similar probability to be linearly independent.

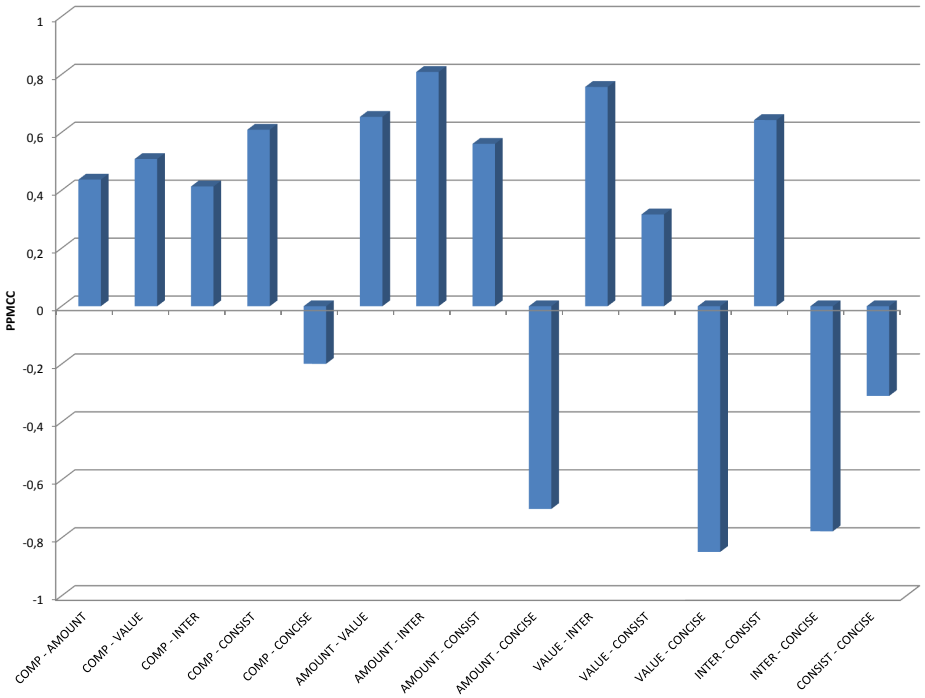

Figure 7.22: PPMCC for all analysed log files as a bar chart visualization.

The results of these two statistical tests is a bit ambivalent. While the DBpedia data in isolation indicates two linear independencies and one linear dependency the analysis across all logs and data sets is much more ambiguous. Based on the insight we derived from the entire research presented in this thesis we conclude that this is due to the different nature of cross-domain and domain-specific data sets.

For DBpedia as the most prominent cross-domain data set we found out that a higher completeness score appears in common with a higher consistency. Generically this means that the fitting of the schema applied in queries against a cross-domain data set has much bigger impact on the overall amount of successful queries than in the case of a domain-specific data set.

CHAPTER 8

Summary and Conclusions

When we started to perform this doctoral research we were driven by our very personal experience in ontology engineering research and practice that the ontology development and maintenance process is not very well aligned with the common practice of application developers or database administrators. From literature review but also from our personal experiences we noticed that existing methodologies majorly focus scenarios where the ontology itself is the project asset and not the application that is built on top of it. Furthermore, due to the success of the Linked Data community effort, the focus has shifted from emphasizing the conceptual dimension to emphasizing the instance dimension of Web ontologies. It is common practice to get structured data out fast and first and let a more complex schema level evolve in the long-tail. Meanwhile numerous applications emerged that aim at demonstrating the benefit of using the structured Web data paradigm for flexible data remixing and integration. The Linked Open Data Cloud is the open accessible example of a distributed dataspace that features both, a number of data sets that conform to standardized data publication and access principles as well as applications that use this data which is not entirely integrated a priori.

In this chapter we summarize our research which was situated at the aforementioned interface between ontology engineering and Linked Data. We present our conclusions, the limitations of our work as well as the directions for future research based on our work and findings.

8.1 Summary

We aimed to contribute one perspective how well-established ontology engineering practices can be applied in the context of structured Web data or how they need to be adapted. Structured Web data strongly requires methods and tools that support the data publisher, data consumer or third parties to fulfill their individual tasks in the evolutionary data integration process of an entire dataspace. By an empirical study amongst a representative set of Linked Open Data providers we approved this assumption and found out that data set evolution is very much driven by evolving external conditions such as community guidelines, standard vocabularies, and application requirements. Hence, we designed, developed, and studied an approach that provides the data publisher with a theory and tool to analyze the usage of a self-administered data set in order to manage maintenance activities that focus on the compatibility of a data set with the queries performed against it by applications.

As a core part of our approach we presented a data set life cycle which differentiates between engineering and usage phases and a framework for assessing the quality of a data set with reference to successful and failing SPARQL queries performed against it. The latter quality framework is the most important part of the evaluation component of the life cycle because any maintenance activity heavily relies on an evaluation result indicating it.

The evaluation is done multi-perspectively capturing three pillars. First, the data set life cycle is set into context with the most recent and most established methodologies for ontology and data set development or management by application of the state-of-the-art framework for the qualitative evaluation of ontology engineering methodologies. Additionally the ONTOCOM cost model for ontology engineering methodologies is applied to our life cycle. Second, the representativeness of our data quality framework is critically discussed by comparing it to the state-of-the-art in data quality research which is based on empirical evidence about the importance of particular quality dimensions for the data consumer. Third, in a number of experiments we analyzed real-world log files of three different Linked Open Data data sets. For the DBpedia data set we performed a case study which compares change recommendations derived by our approach with the manual maintenance effort of the DBpedia project team as a baseline.

8.2 Conclusions

In Chapter 1 we raised three central research questions for our work which were contextualized with the hypothesis that ontology engineering guidelines can be adapted for the maintenance of structured Web data sets. We will now come back to these questions and discuss the conclusions of our research with reference to them individually.

8.2.1 What are the blind spots between ontology engineering and Linked Data?

Our controlled interviews among the Linked Open Data providers has shown that established ontology engineering methodologies are not widely adopted in structured Web data publication and management. Furthermore, we derived a set of requirements which are very specific for the use case of distributed data sets which together form a kind of a dataspace and which shall be integrated in the long-tail. Especially the emphasize on external factors indicating maintenance is noteworthy.

Applying the evaluation criteria and the cost-estimation model for ontology engineering methodologies in the context of structured Web data life cycles has never been done before. This part of our work shows that and how rigorous results from the ontology engineering discipline can be applied in the context of structured Web data creation and management.

8.2.2 How can classical Web usage mining methods be applied in the context of structured Web data sets and how does that affect managing data set maintenance?

As an example instantiation of our usage-dependent life cycle we decided to present a Web usage mining approach which exploits the information about data set usage contained in SPARQL endpoint log files. We presented a preprocessing algorithm that dissolves the problem that classical Web server logs do not contain any information about the success or failure of SPARQL queries because they serialize no results proprietary. The re-execution of queries is time-consuming and makes it necessary to provide a mirror infrastructure for usage analysis in order to protect resources of the productive infrastructure.

Association rule mining has been presented as an exemplary application of classical Web usage mining methods on our data set usage database and we introduced different possibilities to map queries to transactions. The best results for the recommendation of data set changes were produced by the instance associations approach which simply regards every query as a single transaction. The schema level view to

queries we introduced in this context did not yield any digestible results which was most often caused by missing types for queried resources. The extend to which we studied association rule mining in this thesis was sufficient to proof its applicability for change recommendation by means of presenting limited anecdotal examples. However, we must admit that any more specific study of the performance of the different views to transactions and also the influence of restricting filters opened as an individual research challenge for future work.

8.2.3 To which extend can usage-dependent metrics help to assess the quality of a Web data set?

We developed a statistical framework which allows for assessing the quality of selected dimensions of a data set's quality based on our specific usage mining approach. The evaluation of our selection of six out of sixteen data quality dimensions which cover two out of four data quality categories has shown that we were able to cover only a limited amount of the most important dimensions for the data consumer. This is due to the fact that many of the data quality dimensions which are of most importance for the data consumer are part of the intrinsic data quality category which we excluded from our consideration due to the following reasons: (a) Intrinsic data quality dimensions measure only inherent properties of data. (b) Intrinsic data quality dimensions feature a special degree of subjectivity which is hard to be mined from raw log data.

The appropriate amount of data, value-added, and concise representation data quality dimensions as they were defined in this thesis turned out to provide the most generic insight into the quality of a data set while the other three dimensions (consistent representation, interpretability, completeness) are subject to strong dependence on the way how users formulate queries. If only reasonable queries can be expected the value of these three dimensions can be increased but in open data scenarios these dimensions should be devaluated or eliminated.

The general statistics about the proportion of failing queries suit well to provide the data set publisher with a high-level overview of data set issues and help to guide deeper analysis of those by consulting the hierarchical data quality scores first and then the individual dimension scores. Especially the contextual data quality score provides a high sensitivity for finding tipping points for maintenance activities that help to improve the data set quality. Data sets that hold a significant potential for better interlinkage with other data sets feature a low contextual score.

Our experimentation has shown that backward compatibility is generally not provided. Furthermore, the practice performed by the DBpedia project to provide different sub-versions of the data set (e.g. cleaned and uncleaned properties dump)

as well as to move schema level primitives from one namespace into another turned out to be problematic. It seems to confuse the users because many query failures are caused by using the wrong identifier for an existing concept. Providing additional mappings between these sub-versions as well as new and old identifiers of primitives could resolve this issue and improve the quality of the data set significantly.

8.3 Limitations of our research

We already mentioned that the complexity of the re-execution of queries to find out whether these were successful or not is a significant bottleneck which cannot be completely eliminated by parallelization because current stores for structured Web data hardly allow to perform many complex queries in parallel without any timeout failures. Consequently our analysis is time-consuming and currently does not allow for real-time usage analysis. A detailed logging of query success and failure directly in the query processor would be beneficial. Initial studies of heuristics for identifying failing queries based on log information only have been performed by us as another workaround but the preliminary results are not a matter of this thesis.

The analysis of SPARQL queries contained in log files actually is not possible if different usage patterns than the query access pattern via a dedicated query endpoint of a data set are applied. Most of the federated querying approaches – especially those which apply the link traversal approach for iteratively retrieving relevant data during query execution [HBF09] – would hinder the feasibility of our approach as long as no information about the performed query is logged on the server side.

We must admit that several aspects which were presented in this thesis cannot be regarded irrevocably. Our selection of data quality dimensions or the assignment of specific measures to them for example allows the question whether this is necessarily the optimal choice. Due to the highly individual and subjective character of data quality assessment, which is acknowledged in literature, it is not possible to answer this in general. The same holds for our proposed life cycle which would benefit from a user study for example to evaluate the fitting of the proposed processes and activities or its performance in reality. As part of our work we conducted one empirical survey to approve the relevancy of the addressed problem and to guide our research and development along rigorous requirements. We also tried to run further studies involving the maintainers of actively used and maintained data sets but failed because it is hard to motivate people to contribute beside their regular work. As a general principle we tried to set our approach in relation to commonly agreed baseline procedures whenever possible and to discuss how any adaptation would affect our findings.

8.4 Future work

Beside the aforementioned preliminary study on heuristics for query log analysis we already induced further research which sets up on the work presented in this thesis. Our usage database resulting from the log file preprocessing allows for a large number of statistics about the atomic parts of SPARQL queries. In a very recent experiment we started to compare the key concept extraction approach for ontology summarization [PMd08] with summarizations based on the usage frequency of schema level primitives of a data set.

One component of the evaluation of our data set maintenance methodology was the application of the ONTOCOM model. We presented a mapping of cost drivers to activities of all process stages. This allows to derive cost formulas for each single life cycle phase as it was introduced in [STM07]. However, without real world project data the start values for the cost drivers cannot be calibrated which is crucial for representative cost estimation formulas. Hence, the acquisition of this project data as part of a revised edition of our LOD Provider Survey is a necessary next step.

We presented a rather trivial measure to estimate the effects of changes in this thesis, in order to compare our change proposals to the ones derived from the baseline maintenance procedure. More complex measures are needed in order to become more precise with reference to the actual effect of data set changes. One question in this context is whether to measure the effects of changes with reference to the queries performed (to ensure backward compatibility) or with reference to the data set (to avoid data loss in general for example).

To bring the results of our work into practice we plan to develop a visual data set analytics tool. As an initial step this requires user studies with paper prototypes in order to find the adequate visualizations for the purposes of data set maintainers. It is also possible to plug our usage-dependent data quality assessment in an existing tool like Profiler [KPP+12].

In this thesis we presented the viewpoint that any structured Web data set is a specific form of an ontology. We did not formally define any logical framework that reflects the specific characteristics this kind of ontologies exposes and we also did not go into very much detail of the ad hoc schema carried by queries. This leaves space for further theoretical investigations which may yield new forms of reasoning on structured Web data for example.[1]

[1] We admit that we are aware of work addressing this research question explicitly. One paper was under publication as of writing this thesis. Please refer to the paper of Alessandro Adamou, Paolo Ciancarini, Aldo Gangemi and Valentina Presutti entitled "The foundations of virtual ontology networks" in the proceedings of the I-SEMANTICS 2013 (`http://i-semantics.tugraz.at/scientific-track/accepted-papers`).

startatroot

Bibliography

[ABK+08] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2008.

[ABL+12] Sören Auer, Lorenz Bühmann, Jens Lehmann, Michael Hausenblas, Sebastian Tramp, Bert van Nuffelen, Pablo Mendes, Christian Dirschl, Robert Isele, Hugh Williams, and Orri Erling. Managing the life-cycle of Linked Data with the LOD2 stack. In *Proceedings of International Semantic Web Conference (ISWC 2012)*, 2012.

[ACDN12] Maristella Agosti, Franco Crivellari, and Giorgio M. Di Nunzio. Web log analysis: a review of a decade of studies about information acquisition, inspection and interpretation of user interaction. *Data Mining and Knowledge Discovery*, 24(3):663–696, 2012.

[ACHZ09] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets - on the design and usage of VoID, the 'Vocabulary of Interlinked Datasets'. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009.

[ACHZ11] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets with the VoID vocabulary.

http://www.w3.org/TR/void, 2011. W3C Interest Group Note 2011-03-03, retrieved 2013-04-18.

[Ack89] R.L. Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 1989.

[AFMPdlF11] Mario Arias, Javier D. Fernández, Miguel A. Martínez-Prieto, and Pablo de la Fuente. An empirical study of real-world sparql queries. *CoRR*, abs/1103.5043, 2011. In proceedings of the 1st International Workshop on Usage Analysis and the Web of Data (USEWOD2011).

[AH06] Sören Auer and Heinrich Herre. RapidOWL - An agile knowledge engineering methodology. In Irina Virbitskaite and Andrei Voronkov, editors, *Ershov Memorial Conference*, volume 4378 of *Lecture Notes in Computer Science*, pages 424–430. Springer, 2006.

[AH11] Dean Allemang and James A. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, Waltham, MA, 2 edition, 2011.

[AL10] Sören Auer and Jens Lehmann. Making the Web a Data Washing Machine - Creating Knowledge out of Interlinked Data. *Semantic Web Journal*, pages 97–104, 2010.

[ALH09] Sören Auer, Jens Lehmann, and Sebastian Hellmann. LinkedGeoData - adding a spatial dimension to the Web of Data. In *Proc. of 8th International Semantic Web Conference (ISWC)*, 2009.

[AS94] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 487–499, Santiago de Chile, Chile, 1994. Morgan Kaufmann Publishers Inc.

[Aue07] Sören Auer. *Towards agile knowledge engineering: methodology, concepts and applications.* PhD thesis, University of Leipzig, 2007. http://d-nb.info/983577285.

[B+04] Bettina Berendt et al. A roadmap for web mining: From web to Semantic Web. In Bettina Berendt et al., editors, *Web Mining: From Web to Semantic Web*, volume 3209, pages 1–22, Heidelberg, 2004. Springer.

[B+11] Bettina Berendt et al. USEWOD2011: 1st international workshop on usage analysis and the Web of Data. In Sadagopan Srinivasan et al., editors, *WWW (Companion Volume)*, pages 305–306. ACM, 2011.

[BAB+00] Barry W. Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, and Bert Steece. *Software Cost Estimation with COCOMO II*. Prentice Hall, 2000.

[BC09] Christian Bizer and Richard Cyganiak. Quality-driven information filtering using the wiqa policy framework. *J. Web Sem.*, 7(1):1–10, 2009.

[BCM+03] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider. *The Description Logics Handbook: Theory, Implementations, and Applications*. Cambridge University Press, 2003.

[BCM04] Gene Bellinger, Durval Castro, and Anthony Mills. Data, information, knowledge, and wisdom. `http://courseweb.lis.illinois.edu/~katewill/spring2011-502/502%20and%20other%20readings/bellinger%20on%20ackoff%20data%20info%20know%20wisdom.pdf`, 2004. retrieved 2013-04-05.

[BHBL09] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[BHH+11] Bettina Berendt, Laura Hollink, Vera Hollink, Markus Luczak-Rösch, Knud Möller, and David Vallet. Usage analysis and the Web of Data. *SIGIR Forum*, 45(1):63–69, 2011.

[BHS02] Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards Semantic Web mining. In *In International Semantic Web Conference (ISWC*, pages 264–278. Springer, 2002.

[Biz07] Christian Bizer. *Quality Driven Information Filtering: In the Context of Web Based Information Systems*. VDM Publishing, 2007.

[BKS10] Simone Braun, Christine Kunzmann, and Andreas Schmidt. People tagging ontology maturing: Towards collaborative competence management. In David Randall and Pascal Salembier, editors, *From CSCW to Web2.0: European Developments in Collaborative Design Selected Papers from COOP08*, Computer Supported Cooperative Work. Springer, Berlin/Heidelberg, 2010.

[BL] Tim Berners-Lee. Universal Resource Identifiers in WWW. `http://www.ietf.org/rfc/rfc1630.txt`. IETF Request for Comments; retrieved 2013-04-05.

[BL89] Tim Berners-Lee. Information management: A proposal. `http://www.w3c.org/History/1989/proposal.html`, 1989. retrieved 2013-04-05.

[BL06] Tim Berners-Lee. Linked Data - Design Issues. `http://www.w3.org/DesignIssues/LinkedData.html`, 2006. retrieved 2013-04-05.

[BLF00] Tim Berners-Lee and Mark Fischetti. *Weaving the web - the original design and ultimate destiny of the World Wide Web by its inventor.* HarperBusiness, 2000.

[BLHH+06] Tim Berners-Lee, Wendy Hall, James Hendler, Nigel Shadbolt, and Daniel J. Weitzner. Creating a science of the web. *Science*, 313(5788):769–771, 2006.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In Joan Peckham, editor, *SIGMOD Conference*, pages 255–264. ACM Press, 1997.

[Bra07] Steve Bratt. Semantic Web, and other technologies to watch. `http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#%2824%29`, 2007. retrieved 2013-04-11.

[BS06] Carlo Batini and Monica Scannapieco. *Data quality. Concepts, methodologies and techniques.* Data-centric systems and applications. Springer, Berlin, 2006.

[BS09] C. Bizer and A. Schultz. The Berlin SPARQL benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(2):1–24, 2009.

[BSW+07] Simone Braun, Andreas P. Schmidt, Andreas Walter, Gábor Nagypál, and Valentin Zacharias. Ontology maturing: a collaborative web 2.0 approach to ontology engineering. In Natalya Fridman Noy, Harith Alani, Gerd Stumme, Peter Mika, York Sure, and Denny Vrandecic,

editors, *CKC*, volume 273 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[Bun97] Peter Buneman. Semistructured data. In Alberto O. Mendelzon and Z. Meral özsoyoglu, editors, *PODS*, pages 117–121. ACM Press, 1997.

[BZ10] Simone Braun and Valentin Zacharias. Soboleo editor and repository for living ontologies. In Mathieu d'Aquin, Alexander García Castro, Christoph Lange, and Kim Viljanen, editors, *Proceedings of the 1st International Workshop on Ontology Repository and Editors for the Semantic Work (ORES 2010) at the Extended Semantic Web Conference (ESWC 2010)*, 2010.

[CMS97] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *ICTAI '97: Proceedings of the 9th International Conference on Tools with Artificial Intelligence*, page 558, Washington, DC, USA, 1997. IEEE Computer Society.

[Con] World Wide Web Consortium. Linked Data. `http://www.w3.org/standards/semanticweb/data`. W3C standards introduction; retrieved 2013-04-05.

[CPY96] Ming-Syan Chen, Jong Soo Park, and P.S. Yu. Data mining for path traversal patterns in a web environment. In *Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS '96)*, ICDCS '96, pages 385–392. IEEE Computer Society, 1996.

[CPY98] Ming-Syan Chen, Jong Soo Park, and Philip S. Yu. Efficient data mining for path traversal patterns. *IEEE Trans. Knowl. Data Eng.*, 10(2):209–221, 1998.

[DC02] Gordana Dodig-Crnkovic. Scientific Methods in Computer Science. In *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, Skövde, April 2002. `http://www.mrtc.mdh.se/publications/0446.pdf` – last visited 7^{th} december 2007.

[dCSdFB10] María del Carmen Suárez de Figueroa Baonza. *NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. PhD thesis, Universidad Politécnica de Madrid, Madrid, Spain, June 2010.

[DFDM12] Lorand Dali, Blaž Fortuna, Thanh Tran Duc, and Dunja Mladenić. Query-independent learning to rank for rdf entity search. In *Proceedings of the 9th international conference on The Semantic Web: research and applications*, ESWC'12, pages 484–498, Berlin, Heidelberg, 2012. Springer-Verlag.

[Eck12] Kai Eckert. *Usage-driven Maintenance of Knowledge Organization Systems*. Doctoral dissertation, Universität Mannheim, 2012. retrieved 2013-05-21.

[ED07] Faezeh Ensan and Weichang Du. Towards domain-centric ontology development and maintenance frameworks. In *SEKE*, pages 622–627. Knowledge Systems Institute Graduate School, 2007.

[EMC+11] Khadija Elbedweihy, Suvodeep Mazumdar, Amparo Elizabeth Cano, Stuart N. Wrigley, and Fabio Ciravegna. Identifying information needs by modelling collective query patterns. In Olaf Hartig, Andreas Harth, and Juan Sequeda, editors, *COLD*, volume 782 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[EMS11] Kai Eckert, Robert Meusel, and Heiner Stuckenschmidt. User-centered Maintenance of Concept Hierarchies. In Wilson Wong, Wei Liu, and Mohammed Bennamoun, editors, *Ontology Learning and Knowledge Discovery Using the Web: Challenges and Recent Advances*. IGI Global, 2011.

[EVS11] Basil Ell, Denny Vrandečić, and Elena Simperl. Deriving human-readable labels from SPARQL queries. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 126–133, New York, NY, USA, 2011. ACM.

[FFR97] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, pages 707–727, 1997.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. `http://www.w3.org/Protocols/rfc2616/rfc2616.html`, 1999. IETF Request for Comments; retrieved 2013-04-11.

[FGPJ97] Mariano Fernandez, Asuncion Gomez-Perez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering.

In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, USA, March 1997.

[FHM05] Michael Franklin, Alon Halevy, and David Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34(4):27–33, December 2005.

[Fie00] Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. retrieved 2013-04-05.

[FLGP02] Mariano Fernández-López and Asunción Gómez-Pérez. Overview and analysis of methodologies for building ontologies. *Knowledge Eng. Review*, 17(2):129–156, 2002.

[FPSS96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.

[FT02] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150, 2002.

[FTN11] Sean M. Falconer, Tania Tudorache, and Natalya Fridman Noy. An analysis of collaborative patterns in large-scale ontology development projects. In Mark A. Musen and Óscar Corcho, editors, *K-CAP*, pages 25–32. ACM, 2011.

[GJB74] Frank M Gryna, J. M. Juran, and Richard S Bingham. *Quality control handbook. J. M. Juran, editor-in-chief, Frank M. Gryna, Jr., associate editor [and] R. S. Bingham, Jr., associate editor*. McGraw-Hill New York, 3d ed. edition, 1974.

[GPFLC04] A. Gómez-Pérez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering – with examples form the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Verlag, 2004.

[GPSF09] A. Gómez-Pérez and M. Suárez-Figueroa. Scenarios for building ontology networks within the NeOn methodology. In *Proceedings of the Fifth International Conference on Knowledge Capture*, K-CAP '09, pages 183–184, 2009.

[Gro04] W3C Technical Architecture Group. Architecture of the World Wide Web, volume one. `http://www.w3.org/TR/2004/REC-webarch-20041215/`, 2004. W3C Recommendation 15 December 2004, retrieved 2013-04-15.

[Gru95] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.

[GZT+11] Dragan Gasevic, Amal Zouaq, Carlo Torniai, Jelena Jovanovic, and Marek Hatala. An approach to folksonomy-based ontology maintenance for learning environments. *TLT*, 4(4):301–314, 2011.

[Har79] Juris Hartmanis. Observations about the development of theoretical computer science. In *FOCS*, pages 224–233. IEEE Computer Society, 1979.

[Hau09] Michael Hausenblas. Exploiting Linked Data For Building Web Applications. *IEEE Internet Computing*, 13(4):68–73, 2009.

[HB11] Tom. Heath and Christian. Bizer. *Linked Data: evolving the web into a global data space*. Morgan Claypool, [San Rafael, Calif.], 2011.

[HBF09] Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. Executing SPARQL queries over the web of Linked Data. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 293–309, Berlin, Heidelberg, 2009. Springer-Verlag.

[HC11] Michael Hahsler and Sudheer Chelluboina. Visualizing association rules: Introduction to the r-extension package arulesviz. *R project module*, 2011.

[HDG+06] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai Wang. The manchester owl syntax. In *OWLED2006 Second Workshop on OWL Experiences and Directions*, Athens, GA, USA, 2006.

[Hea08] Tom Heath. How will we interact with the Web of Data? *IEEE Internet Computing*, 12(5):88–91, 2008.

[HEPS03] Masahiro Hori, Jérôme Euzenat, and Peter F. Patel-Schneider. OWL Web Ontology Language XML presentation syntax, 2003. W3C Note 11 June 2003; retrieved 2013-04-11.

[HGHB09] Michael Hahsler, Bettina Grün, Kurt Hornik, and Christian Buchta. Introduction to arules–a computational environment for mining association rules and frequent item sets. *The Comprehensive R Archive Network*, 2009.

[HHP+10] Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres. Weaving the pedantic web. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *LDOW*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[HJA12] Julia Hoxha, Martin Junghans, and Sudhir Agarwal. Enabling semantic analysis of user browsing patterns in the Web of Data. *CoRR*, abs/1204.2713, 2012. In proceedings of the 2nd International Workshop on Usage Analysis and the Web of Data (USEWOD2012).

[HKRS08] P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure. *Semantic Web - Grundlagen*. eXamen.press. Springer, 2008.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1):75–106, 2004.

[HMSS01] A. Hotho, A. Maedche, S. Staab, and R. Studer. Seal-ii - the soft spot between richly structured and unstructured knowledge. *Journal of Universal Computer Science*, 7(7):566–590, 2001.

[Hod00] Gail M. Hodge. *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files*. Commission on Preservation , 2000.

[HRE11] Julia Hoxha, Anisa Rula, and Basil Ell. Towards green Linked Data. In Olaf Hartig, Andreas Harth, and Juan Sequeda, editors, *COLD*, volume 782 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[HZ09] Olaf Hartig and Jun Zhao. Using web data provenance for quality assessment. In Juliana Freire, Paolo Missier, and Satya Sanket Sahoo, editors, *SWPM*, volume 526 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[IEE90] IEEE. Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990. Technical report, IEEE Computer Society Press, 1990.

[IS09] Antoine Isaac and Ed Summers. Skos simple knowledge organization system primer. http://www.w3.org/TR/2009/NOTE-skos-primer-20090818/, 2009. W3C Working Group Note 18 August 2009; retrieved 2013-04-11.

[JU99] Robert Jasper and Mike Uschold. A framework for understanding and classifying ontology applications. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving*, pages 16–21, 1999.

[KKL11] Markus Kirchberg, Ryan K. L. Ko, and Bu-Sung Lee. From Linked Data to relevant data – time is the essence. *CoRR*, abs/1103.5046, 2011. In proceedings of the 1st International Workshop on Usage Analysis and the Web of Data (USEWOD2011).

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[Koi01] Marja-Riitta Koivunen. W3c Semantic Web activity. http://www.w3.org/Talks/2001/1102-semweb-fin/slide17-0.html, 2001. retrieved 2013-04-11.

[KPP+12] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Profiler: integrated statistical analysis and visualization for data quality assessment. In Genny Tortora, Stefano Levialdi, and Maurizio Tucci, editors, *International Working Conference on Advanced Visual Interfaces, AVI '12, Capri Island, Naples, Italy, May 22-25, 2012, Proceedings*, pages 547–554. ACM, 2012.

[KV06] Konstantinos Kotis and George A. Vouros. Human-centered ontology engineering: The hcome methodology. *Knowl. Inf. Syst.*, 10(1):109–131, 2006.

[L99] M. Mariano Fernández López. Overview of Methodologies for Building Ontologies. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem Solving Methods (KRR5) Stockholm, Sweden, August 2, 1999*, 1999.

[lGBFMP11] Sandra Álvarez García, Nieves R. Brisaboa, Javier D. Fernández, and Miguel A. Martínez-Prieto. Compressed k2-triples for full-in-memory rdf engines. In Vallabh Sambamurthy and Mohan Tanniru, editors, *AMCIS*. Association for Information Systems, 2011.

[LN13] Johannes Lorey and Felix Naumann. Caching and prefetching strategies for SPARQL queries. In *Proceedings of the 3rd International Workshop on Usage Analysis and the Web of Data (USEWOD)*, Montpellier, France, 0 2013.

[LRH08] Markus Luczak-Rösch and Ralf Heese. A generic corporate ontology lifecycle. In Christoph Lange 0002, Sebastian Schaffert, Hala Skaf-Molli, and Max Völkel, editors, *SemWiki*, volume 360 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[LRH09] Markus Luczak-Rösch and Ralf Heese. *Managing Ontology Lifecycles in Corporate Settings*, volume 221 of *Studies in Computational Intelligence*, pages 235–248. Springer Berlin / Heidelberg, 2009.

[LRM11] Markus Luczak-Rösch and Hannes Mühleisen. Log File Analysis for Web of Data Endpoints . In *Proc. of the 8th Extended Semantic Web Conference (ESWC), Poster-Session*. Springer LNCS 6643, 2011.

[MÏ2] Knud Möller. Lifecycle models of data-centric systems and domains. *Semantic Web journal*, 2012.

[MA04] Peter Mika and Hans Akkermans. Towards a new synthesis of ontology technology and knowledge management. *Knowledge Eng. Review*, 19(4):317–345, 2004.

[MB12] H. Mühleisen and C. Bizer. Web Data Commons - Extracting Structured Data from Two Large Web Corpora. In *WWW 2012 Workshop: Linked Data on the Web (LDOW2012)*, Lyon, France, 2012.

[MHCG10] Knud Möller, Michael Hausenblas, Richard Cyganiak, and Gunnar Aastrand Grimnes. Learning from linked open data usage: Patterns & metrics. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line*, 2010.

[MJC+07] J. Madhavan, S.R. Jeffery, S. Cohen, X. Dong, D. Ko, G. Yu, and Alon Halevy. Web-scale data integration: You can only afford to pay as you go. In *Proceedings of the Third Biennial Conference on Innovative Data Systems Research*, 2007.

[MMB12] Pablo N. Mendes, Hannes Mühleisen, and Christian Bizer. Sieve: Linked Data quality assessment and fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 116–123, 2012.

[Moc87] P. Mockapetris. Domain Names - Implementations and Specifications. `http://www.ietf.org/rfc/rfc1035.txt`, 1987. IETF Request for Comments; retrieved 2013-04-1.

[MP12] Peter Mika and Tim Potter. Metadata statistics for a large web corpus. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *LDOW*, volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[MPPS09] Boris Motik, Bijan Parsia, and Peter F. Patel-Schneider. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. `http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/`, October 2009. W3C Recommendation; retrieved 2013-04-11.

[MS95] Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266, 1995.

[MS06] Malgorzata Mochol and Elena Paslaru Bontas Simperl. Cost estimation for ontology development. In Witold Abramowicz and Heinrich C. Mayr, editors, *BIS*, volume 85 of *LNI*, pages 415–428. GI, 2006.

[MWL09] Stuart E. Madnick, Richard Y. Wang, Yang W. Lee, and Hongwei Zhu 0002. Overview and framework for data and information quality research. *J. Data and Information Quality*, 1(1), 2009.

[Nau02] Felix Naumann. *Quality-driven query answering for integrated information systems*. Springer-Verlag, Berlin, Heidelberg, 2002.

[Nel65] Ted Nelson. The hypertext. In *Proceedings of the International Federation of Documentation (FID) Congress*, number 31, Washington, DC, 1965. abstract, reprinted in Nelsons Possiplex, p. 154.

[NK04] Natalya F. Noy and Michel Klein. Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.*, 6(4):428–440, July 2004.

[NKKM04] Natalya Fridman Noy, Sandhya Kunnatur, Michel C. A. Klein, and Mark A. Musen. Tracking changes during ontology evolution. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 259–273. Springer, 2004.

[NLF99] Felix Naumann, Ulf Leser, and Johann Christoph Freytag. Quality-driven integration of heterogenous information systems. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 447–458, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[NM00] Natalya Fridman Noy and Mark A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In Henry A. Kautz and Bruce W. Porter, editors, *AAAI/IAAI*, pages 450–455. AAAI Press / The MIT Press, 2000.

[NM02] Natalya Fridman Noy and Mark A. Musen. Promptdiff: A fixed-point algorithm for comparing ontology versions. In Rina Dechter and Richard S. Sutton, editors, *AAAI/IAAI*, pages 744–750. AAAI Press / The MIT Press, 2002.

[OCM+07] Leo Obrst, Werner Ceusters, Inderjeet Mani, Steve Ray, and Barry Smith. The evaluation of ontologies. In Christopher J.O. Baker and Kei-Hoi Cheung, editors, *Revolutionizing Knowledge Discovery in the Life Sciences*, chapter 7, pages 139–158. Springer, 2007.

[oSC81a] Information Sciences Institute. University of Southern California. Internet Protocol. `http://www.ietf.org/rfc/rfc791.txt`, 1981. IETF Request for Comments; retrieved 2013-04-05.

[oSC81b] Information Sciences Institute. University of Southern California. Transmission Control Protocol. `http://www.ietf.org/rfc/rfc793.txt`, 1981. IETF Request for Comments; retrieved 2013-04-05.

[PLW02] Leo L Pipino, Yang W Lee, and Richard Y Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[PMd08] Silvio Peroni, Enrico Motta, and Mathieu d'Aquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In John Domingue and Chutiporn Anutariya, editors, *ASWC*, volume 5367 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2008.

[PSHH04] Peter Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Abstract Syntax and Semantics, 2004. W3C Recommendation 10 February 2004; retrieved 2013-04-11.

[PTSS06] Helena S. Pinto, Christoph Tempich, Steffen Staab, and York Sure. *Semantic Web and Peer-to-Peer*, chapter Distributed Engineering of Ontologies (DILIGENT), pages 301–320. Springer Verlag, 2006.

[Rag12] Aravindan Raghuveer. Characterizing machine agent behavior through SPARQL query mining. In *Proceedings of the International Workshop on Usage Analysis and the Web of Data, Lyon, France*, 2012.

[S+97] David J. Schultz et al. Ieee standard for developing software life cycle processes. Ieee std, The Institute of Electrical and Electronics Engineers, New York, USA, 1997.

[SA95] Ramakrishnan Srikant and Rakesh Agrawal. *Mining generalized association rules*. IBM Research Division, 1995.

[Saa80] Thomas L. Saaty. *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. McGraw-Hill, New york, 1980.

[Saa90] T.L. Saaty. How to make a decision: the analytic hierarchy process. *European journal of operational research*, 48(1):9–26, 1990.

[Saa08] Thomas L Saaty. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, 1(1):83–98, 2008.

[SAD+99] Steffen Staab, Jürgen Angele, Stefan Decker, Michael Erdmann, Andreas Hotho, Alexander Mädche, Hans-Peter Schnurr, Rudi Studer, and York Sure. Semantic community web portals. *Computer Networks*, 33(1-6):473–491, 1999.

[SB82] Seymour Sudman and Norman M. Bradburn. *Asking Questions : A Practical Guide to Questionnaire Design*. Jossey-Bass, 1982.

[SBH+12] Elena Simperl, Tobias Bürger, Simon Hangl, Stephan Wörgl, and Igor O. Popov. Ontocom: A reliable cost estimation method for ontology development projects. *Journal of Web Semantics*, 16:1–16, 2012.

[SBLH06] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[SC00] Jaideep Srivastava and Robert Cooley. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1:12–23, 2000.

[Shn96] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Visual Languages*, number UMCP-CSD CS-TR-3665, pages 336–343, College Park, Maryland 20742, U.S.A., 1996.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[SLR13] Elena Simperl and Markus Luczak-Rösch. Collaborative ontology engineering: a survey. *The Knowledge Engineering Review*, FirstView:1–31, 8 2013.

[SLW97] Diane M. Strong, Yang W. Lee, and Richard Y. Wang. Data quality in context. *Commun. ACM*, 40(5):103–110, 1997.

[SMB10] E. Simperl, M. Mochol, and T. Bürger. Achieving Maturity: the State of Practice in Ontology Engineering in 2009. *International Journal of Computer Science and Applications*, 7(1):45–65, 2010.

[SMBN03] Myra Spiliopoulou, Bamshad Mobasher, Bettina Berendt, and Miki Nakagawa. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. *INFORMS Journal on Computing*, 15(2):171–190, 2003.

[SPB09] Elena Paslaru Bontas Simperl, Igor Popov, and Tobias Buerger. Ontocom revisited: Towards accurate cost predictions for ontology development projects. In *6th Annual European Semantic Web Conference (ESWC2009)*, pages 248–262, 2009.

[Spi00] Myra Spiliopoulou. Web Usage Mining for Web Site Evaluation. *Communications of the ACM*, 43(8):127–134, August 2000.

[SS02] York Sure and Rudi Studer. On-to-knowledge methodology — expanded version. On-To-Knowledge deliverable 17, Institute AIFB, University of Karlsruhe, 2002.

[SS09] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2nd edition edition, 2009.

[SSSS01] S. Staab, R. Studer, H.-P. Schnurr, and Y. Sure. Knowledge processes and ontologies. *Intelligent Systems, IEEE*, 16(1):26–34, 2001.

[ST06] Elena Paslaru Bontas Simperl and Christoph Tempich. Ontology engineering: A reality check. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 836–854. Springer, 2006.

[STM07] E. Simperl, C. Tempich, and M. Mochol. *Technologies for Business Information Systems*, chapter Cost estimation for ontology development: applying the ONTOCOM model, pages 327–339. Springer, 2007.

[SW01] Barry Smith and Christopher Welty. Fois introduction: Ontology—towards a new synthesis. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, New York, NY, USA, 2001. ACM Press.

[T+07] Thanh Tran et al. Lifecycle-support in architectures for ontology-based information systems. In Karl Aberer et al., editor, *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, volume 4825 of *LNCS*, pages 505–518. Springer Verlag, November 2007.

[Tem06] Christoph Tempich. *Ontology Engineering and Routing in Distributed Knowledge Management Applications.* PhD thesis, Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften, Karlsruhe, Germany, August 2006.

[Thu27] Louis Leon Thurstone. A law of comparative judgement. *Psychological Review*, 34:278–286, 1927.

[TKS04] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right objective measure for association analysis. *Inf. Syst.*, 29(4):293–313, 2004.

[TLPH95] Walter F. Tichy, Paul Lukowicz, Lutz Prechelt, and Ernst A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software*, 28(1):9–18, 1995.

[Tur50] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[TVTY90] H. Takeda, P. Veerkamp, T. Tomiyama, and H. Yoshikawam. Modeling design processes. *AI Magazine*, 11(4):37–48, 1990.

[UG96] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–155, June 1996.

[UK95] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI)*, 1995.

[Vra10] Denny Vrandecic. *Ontology Evaluation*. Phdthesis, KIT, Fakultät für Wirtschaftswissenschaften, Karlsruhe, 2010.

[W3C] Apache HTTP server version 2.4 – log files. `http://httpd.apache.org/docs/current/logs.html`. Apache Foundation specification; retrieved 2013-07-18.

[W3C99] Web characterization terminology definitions sheet. `http://www.w3.org/1999/05/WCA-terms/`, 1999. W3C Working Draft 1999-05-24; retrieved 2013-04-18.

[W3C04a] Overview of SGML Resources. `http://www.w3.org/MarkUp/SGML/`, 2004. W3C markup overview, retrieved 2013-04-15.

[W3C04b] RDF Semantics. `http://www.w3.org/TR/rdf-mt/`, 2004. W3C Recommendation 10 February 2004; retrieved 2013-04-11.

[W3C06] Extensible markup language (XML) 1.1 (second edition). `http://www.w3.org/TR/2006/REC-xml11-20060816/`, 2006. W3C Recommendation 16 August 2006, edited in place 29 September 2006, retrieved 2013-04-15.

[W3C08a] SPARQL protocol for rdf. `http://www.w3.org/TR/rdf-sparql-protocol/`, 2008. W3C Recommendation 15 January 2008, retrieved 2013-04-15.

[W3C08b] SPARQL Query Language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`, 2008. W3C Recommendation 15 January 2008, retrieved 2013-04-15.

[W3C09] Namespaces in XML 1.0 (third edition). `http://www.w3.org/TR/REC-xml-names/`, 2009. W3C Recommendation 8 December 2009, retrieved 2013-04-15.

[W3C12] OWL 2 Web Ontology Language Document Overview (second edition). `http://www.w3.org/TR/owl2-overview/`, 2012. W3C Recommendation 11 December 2012, retrieved 2013-04-15.

[W3C13a] SPARQL 1.1 overview. http://www.w3.org/TR/sparql11-overview/, 2013. W3C Recommendation 21 March 2013, retrieved 2013-04-15.

[W3C13b] SPARQL 1.1 query results CSV and TSV formats. http://www.w3.org/TR/2013/REC-sparql11-results-csv-tsv-20130321/, 2013. W3C Recommendation 21 March 2013, retrieved 2013-04-15.

[W3C13c] SPARQL query results XML format (second edition). http://www.w3.org/TR/2013/REC-rdf-sparql-XMLres-20130321/, 2013. W3C Recommendation 21 March 2013, retrieved 2013-04-15.

[W3C13d] XML 1.1 query results JSON format. http://www.w3.org/TR/2013/REC-sparql11-results-json-20130321, 2013. W3C Recommendation 21 March 2013, retrieved 2013-04-15.

[WS96] Richard Y. Wang and Diane M. Strong. Beyond accuracy: what data quality means to data consumers. *J. Manage. Inf. Syst.*, 12:5–33, March 1996.

[ZC04] Marcia Lei Zeng and Lois Mai Chan. Trends and issues in establishing interoperability among knowledge organization systems. *J. Am. Soc. Inf. Sci. Technol.*, 55(5):377–395, March 2004.

[Zen08] Marcia Lei Zeng. Knowledge organization systems (KOS). *Knowledge organization*, 35(2-3):160–182, 2008.

[Zin07] Chaim Zins. Conceptual approaches for defining data, information, and knowledge: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 58(4):479–493, February 2007.

[ZKS+13] Amrapali Zaveri, Dimitris Kontokostas, Mohamed A. Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of DBpedia. In *To appear in Proceedings of 9th International Conference on Semantic Systems, I-SEMANTICS '13, Graz, Austria, September 4-6, 2013*. ACM, 2013.

Appendices

APPENDIX A

Online questionnaire of the LOD Provider Survey

LOD Provider Survey 2010

Your name

Your affiliation

Your mailaddress

LOD dataset you are representing

SPARQL endpoint URI of this dataset

General questions

How many ontologies do you use to populate your dataset altogether?

Which are the ontologies you use? (give names or URIs)

How many of the used ontologies did you develop yourself?

The next questions are related to the ontologies you developed yourself.

Details about first ontology

How did you develop your ontology?

manually from scratch

ontology reuse and manual adaption

ontology learning

automatic generation from any semi-structured datasources

automatically derived from any relational database

Did you follow any methodology for engineering the ontology?

Yes

No

If yes, which one?

What is the size of this ontology in terms of the number of concepts?

Vocabulary (up to 150 concepts)

Small ontology (between 150 and 1000 concepts)

Mid-sized ontology (between 1000 and 5000 concepts)

Large ontology (more than 5000 concepts)

What is the complexity of these ontologies in terms of the usage of ontology language primitives?

RDF-S

OWL-Lite

Figure A.1: Screenshot of the questionnaire opened in a browser window (part I)

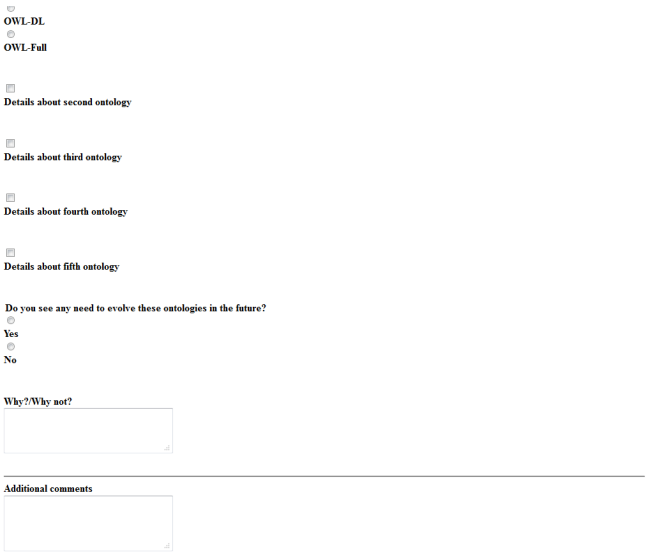

Figure A.2: Screenshot of the questionnaire opened in a browser window (part II)

APPENDIX B

The log file preprocessing algorithm

Algorithm 1 Full pseudocode of the preprocessing algorithm showing the degeneration of SPARQL queries found in the log into their atomic parts (part I).

```
currentLog ←< path to log file >
endpointURI ←< URI of SPARQL endoint mirror >
for all line in currentLog do
    requestString ← getRequestString(line)
    responseCode ← getResponseCode(line)
    if responseCode400 responseCode ≥ 500 then
        if isSPARQLQuery(requestString) then
            query ← extractQuery(requestString)
            querySuccess ← hasResult(query, endpointURI)
            for all graphPattern in query do
                patternQuery ← SELECT * WHERE {graphPattern}
                patternSuccess ← hasResult(patternQuery, endpointURI)
                if isSubPattern(graphPattern) then
                    storeParentPattern(graphPattern)
                end if
                if isOptionalPattern(graphPattern) then
                    flagAsOptionalPattern(graphPattern)
                end if
                if hasFilter(graphPattern) then
                    resultWithoutFilter(graphPattern)
                end if
```

Algorithm 2 Full pseudocode of the preprocessing algorithm showing the degeneration of SPARQL queries found in the log into their atomic parts (part II).

```
for all triple in graphPattern do
    tripleQuery ← SELECT * WHERE {triple}
    tripleSuccess ← hasResult(tripleQuery, endpointURI)
    subject ← getSubject(triple)
    predicate ← getPredicate(triple)
    object ← getObject(triple)
    if !isVariable(subject) then
        subjectQuery ← SELECT * WHERE
         {{ subject ?property ?hasValue }
        UNION { ?isValueOf ?property subject }}
        subjectExists ← hasResult(subjectQuery, endpointURI)
    end if
    if !isVariable(object)  !isLiteral(object) then
        objectQuery ← SELECT * WHERE
         {{ object ?property ?hasValue }
        UNION { ?isValueOf ?property object }}
        objectExists ← hasResult(objectQuery, endpointURI)
    end if
    if !isVariable(predicate) then
        predicateQuery ← SELECT * WHERE { ?s predicate ?o }
        predicateExists ← hasResult(predicateQuery, endpointURI)
    end if
end for
```

=0

APPENDIX C

SQL queries for the statistical framework

Table C lists the SQL queries which need to be performed against our usage database in order to derive the particular values for our statistical framework.

# of analysed queries	SELECT queryresultsstat.id FROM queryresultsstat WHERE queryresultsstat.haserrors = false AND queryresultsstat.isdescribe = false AND (queryresultsstat.id IN (SELECT queryresultspatternstat.query_id FROM queryresultspatternstat WHERE (queryresultspatternstat.query_id IN (SELECT p.query_id FROM queryresultspatternstat p GROUP BY p.query_id HAVING bool_or(p.haserrors) = false)) AND (queryresultspatternstat.id IN (SELECT queryresultstriplestat.pattern_id FROM queryresultstriplestat WHERE (queryresultstriplestat.pattern_id IN (SELECT t.pattern_id FROM queryresultstriplestat t GROUP BY t.pattern_id HAVING bool_or(t.haserrors) = false))))));
# of analyzed basic graph patterns	SELECT COUNT(*) FROM queryresultspatternstat as p WHERE p.query_id IN (SELECT id FROM analyzed_queries)
# of analyzed triple patterns	SELECT COUNT(*) FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE p.query_id IN (SELECT id FROM analyzed_queries)
# of requested distinct properties	SELECT COUNT(*) as cnt, t.predicate FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE t.predicateexists > 1 AND t.predicate <> 'bif:contains' AND p.query_id IN (SELECT id FROM analyzed_queries) GROUP BY t.predicate ORDER BY cnt DESC
# of requested distinct populated properties	SELECT COUNT(*) as cnt, t.predicate FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE t.predicateexists = 10000 AND t.predicate <> 'bif:contains' AND p.query_id IN (SELECT id FROM analyzed_queries) GROUP BY t.predicate ORDER BY cnt DESC

# of failing queries	SELECT COUNT(*) FROM queryresultsstat as q WHERE q.querysuccess = false AND q.id IN (SELECT id FROM analyzed_queries)
# of queries with failing basic graph patterns where all triple patterns are successful	SELECT COUNT(*) FROM queryresultsstat WHERE id IN (SELECT p.query_id FROM queryresultspatternstat as p INNER JOIN queryresultstriplestat as t ON t.pattern_id = p.id WHERE p.query_id IN (SELECT id FROM analyzed_queries) AND p.patternsuccess = false GROUP BY p.id HAVING bool_and(t.triplesuccess) = true)
# of queried resources in TP	(SELECT t.subject as res FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE (t.subjectexists = 100 OR t.subjectexists = 10000) AND p.query_id IN (SELECT id FROM analyzed_queries) GROUP BY t.subject) UNION (SELECT t.object as res FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE (t.objectexists = 100 OR t.objectexists = 10000) AND p.query_id IN (SELECT id FROM analyzed_queries) GROUP BY t.object)

# of queried resources in TP which do not exist in data set	(SELECT t.subject as res FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE t.subjectexists = 100 AND p.query_id IN (SELECT id FROM analyzed_queries) GROUP BY t.subject) UNION (SELECT t.object as res FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE t.objectexists = 100 AND p.query_id IN (SELECT id FROM analyzed_queries) GROUP BY t.object)
# TP that use annotation properties	SELECT COUNT(*) FROM queryresultstriplestat WHERE id IN (SELECT t.id FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE p.query_id IN (SELECT id FROM analyzed_queries) AND (t.predicate LIKE '%http://www.w3.org/2000/01/rdf-schema#comment%' OR t.predicate LIKE '%http://www.w3.org/2000/01/rdf-schema#label%'))
# of failing TP that use annotation properties	SELECT COUNT(*) FROM queryresultstriplestat WHERE id IN (SELECT t.id FROM queryresultstriplestat as t INNER JOIN queryresultspatternstat as p ON t.pattern_id = p.id WHERE p.query_id IN (SELECT id FROM analyzed_queries) AND (t.predicate LIKE '%http://www.w3.org/2000/01/rdf-schema#comment%' OR t.predicate LIKE '%http://www.w3.org/2000/01/rdf-schema#label%') AND t.triplesuccess = false)
# of failing queries with successful basic graph patterns	SELECT COUNT(*) FROM queryresultsstat WHERE id IN (SELECT p.query_id FROM queryresultspatternstat as p WHERE p.query_id IN (SELECT id FROM analyzed_queries) AND p.patternsuccess = true) AND querysuccess = false

APPENDIX D

Erklärung/Declaration

Hiermit versichere ich, Markus Luczak-Rösch, an Eides statt, dass ich die vorliegende Dissertationsschrift mit dem Titel "Usage-dependent maintenance of structured Web data sets" selbständig verfasst habe. Als Hilfsmittel bei der Durchführung der Arbeit und Verfassung der Schrift dienten mir nur die darin angegebenen Quellen.

I hereby affirm that I, Markus Luczak-Rösch, alone composed this thesis with the title "Usage-dependent maintenance of structured Web data sets". My research was carried out and this thesis was composed solely with the aid of the sources referenced herein.

Markus Luczak-Rösch, Berlin, August 15, 2013

APPENDIX E

Curriculum vitae

(Lebenslauf aus datenschutzrechtlichen Gründen entfernt)

APPENDIX F

Zusammenfassung

Mit den Linked-Data-Prinzipien hat sich seit einigen Jahren ein Paradigma etabliert, das beschreibt, wie man, in vollständiger Konformität zur Web-Architektur, strukturierte Daten in sogenannten Datensets veröffentlicht und Web-Ressourcen über die Grenzen einzelner Datensets hinweg in Beziehungen setzt. Als strukturiert kann man diese Web-Daten bezeichnen, weil alle atomaren Teile dieser als Tripel ausgedrückten Daten mit URIs als global eindeutigen Bezeichnern versehen sind. Durch Auflösung der URIs auf entsprechende Basisvokabulare stellt dies eine vollständige Typisierung von Instanz- und Schemaebene dar, die über die Web-Architektur abrufbar ist. Berücksichtigt man die Semantic-Web-Standards, so kann man zu dem Schluss kommen, dass ein Datenset nichts anderes ist als eine spezielle Form einer Web-Ontologie. Die Forschungsdisziplin des Ontology Engineering beschäftigt sich seit mehreren Jahrzehnten mit der Entwicklung und Evaluierung von standardisierten Prozess- und Lebenszyklusmodellen für die Entwicklung von Ontologien. Ontologien sind in der Domäne der Informationssysteme ein Mittel, um Wissen über Interessensbereiche in einer maschinenverarbeitbaren Form strukturiert zu repräsentieren.

Im Rahmen dieser Arbeit wird an der Schnittstelle zwischen Ontology Engineering und Linked Data geforscht und ich gehe der übergeordneten Hypothese nach, dass sich Prinzipien des Ontology Engineering auf Linked Data übertragen lassen. Um den Problembereich als relevant nachzuweisen, wurde zunächst eine Studie zur Anwendung etablierter Ontology Engineering Methoden im Kontext von Linked Data in Form einer Onlinebefragung von Linked-Data-Anbietern durchgeführt. Aus den

Ergebnissen der Studie wurden Anforderungen an Datenset-Lebenszyklen abgeleitet, anhand derer ein abstrakter Lebenszyklus entworfen, detailliert beschrieben und instanziiert wurde, um dessen Anwendung experimentell zu untersuchen. Hierbei spielt insbesondere die Evaluation von Datensets in Relation zur Nutzung eine Rolle. Feedback über die Nutzung eines Datensets wird aus Server-Log-Files gewonnen, welche gegen das Datenset ausgeführte SPARQL-Anfragen enthalten. Es wird der Bereich des Web Usage Mining im Kontext strukturierter Web Daten berührt und es stellen sich Fragen zur Ermittlung der Datenqualität auf Basis der Nutzung eines Datensets. Hier schlägt die Arbeit ein statistisches Rahmenwerk zur Bestimmung der Datenqualität auf Basis von Nutzungsdaten vor, das ausgewählte Daten-Qualitäts-Dimensionen des Stands der Wissenschaft abdeckt.

Die Evaluation des entworfenen Lebenszyklusses erfolgt durch Anwendung rigoroser Methoden aus dem Ontology Engineering Forschungsbereich, welche es erlauben Methoden qualitativ zu vergleichen, sowie Kostenfunktionen für die einzelnen Phasen des Ontologie-Lebenszyklusses zu bestimmen. Überdies wird eine experimentelle Untersuchung präsentiert, in der echte Nutzungsdaten unterschiedlicher Datensets aus der Linked-Open-Data-Cloud analysiert werden. Als Messbasis zum Vergleich des eigenen Ansatzes dient die derzeitige Praxis zur Wartung von Datensets, wie sie zum Beispiel im Rahmen des DBpedia-Projekts durchgeführt und dokumentiert wird [1].

[1] `http://wiki.dbpedia.org/Changelog`

APPENDIX G

Contents of the DVD

1. Research data
   (a) Collected data of the LOD provider survey (/data/survey)
   (b) Raw log files (aus Datenschutzgründen entfernt/removed due to privacy protection issues*)
   (c) sed scripts for unification of obfuscated logs (/data/log_preparation)
   (d) Preprocessed usage databases (aus Datenschutzgründen entfernt/removed due to privacy protection issues*)
   (e) Evaluation material (/data/evaluation)
2. Source code of the log preprocessing and analysis prototype (/src)
3. Digital copy of this thesis as a PDF document

* The research data, on which this work is based upon, has been made available as part of the anonymized and openly accessible USEWOD data set. It can be retrieved via http://usewod.org/data-sets.html.