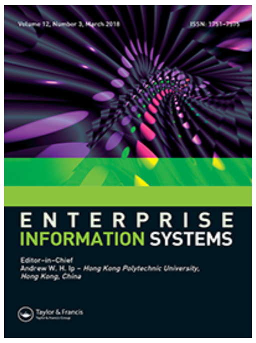


Enterprise Information Systems

ISSN: 1751-7575 (Print) 1751-7583 (Online) Journal homepage: http://www.tandfonline.com/loi/teis20

A web-based system architecture for ontology-based data integration in the domain of IT benchmarking

Matthias Pfaff & Helmut Krcmar

To cite this article: Matthias Pfaff & Helmut Krcmar (2018) A web-based system architecture for ontology-based data integration in the domain of IT benchmarking, Enterprise Information Systems, 12:3, 236-258, DOI: 10.1080/17517575.2017.1329552

To link to this article: https://doi.org/10.1080/17517575.2017.1329552

Published online: 29 May 2017.

ENTERPRISE INFORMATION SYSTEMS, 2018
VOL. 12, NO. 3, 236–258
https://doi.org/10.1080/17517575.2017.1329552

Taylor & Francis
Taylor & Francis Group

ARTICLE

OPEN ACCESS Check for updates

A web-based system architecture for ontology-based data integration in the domain of IT benchmarking

Matthias Pfaff [a,b] and Helmut Krcmar [b]

[a]Department of Business Model and Service Engineering, fortiss GmbH, An-Institut Technische Universität München (TUM), München, Germany; [b]Department of Informatics, Technische Universität München (TUM), München, Germany

ABSTRACT
In the domain of IT benchmarking (ITBM), a variety of data and information are collected. Although these data serve as the basis for business analyses, no unified semantic representation of such data yet exists. Consequently, data analysis across different distributed data sets and different benchmarks is almost impossible. This paper presents a system architecture and prototypical implementation for an integrated data management of distributed databases based on a domain-specific ontology. To preserve the semantic meaning of the data, the ITBM ontology is linked to data sources and functions as the central concept for database access. Thus, additional databases can be integrated by linking them to this domain-specific ontology and are directly available for further business analyses. Moreover, the web-based system supports the process of mapping ontology concepts to external databases by introducing a semi-automatic mapping recommender and by visualizing possible mapping candidates. The system also provides a natural language interface to easily query linked databases. The expected result of this ontology-based approach of knowledge representation and data access is an increase in knowledge and data sharing in this domain, which will enhance existing business analysis methods.

ARTICLE HISTORY
Received 14 October 2016
Accepted 8 May 2017

KEYWORDS
Web Ontology Language (OWL); Semantic Web; system architecture; IT benchmarking; data integration

1. Introduction

Benchmarking, as a systematic process for improving organizational performance, has considerably increased in popularity worldwide since the 1980s. This process is based on the insight that by observing organizations and analyzing their performance, an organization can transform the way that it conducts business. In the context of benchmarking, such a transformation is generally achieved by applying the lessons learned from benchmarking results to one's own organization (Camp 1989; Peters 1994). Moreover, such performance measurements (or benchmarking) can often assist in explaining value or cost aspects to stakeholders (Spendolini 1992). Thus, the analysis and evaluation of this type of performance measurement approach have been the subject of various studies (e.g., Smith and McKeen 1996; Gacenga et al. 2011).

In fact, research in the field of IT benchmarking (ITBM) is typically focused on the structuring, standardizing and generalizing of IT service catalogs and on their implementation within companies (e.g., Dattakumar and Jagadeesh 2003; Kütz 2006; Nissen et al. 2014) to model internally provided IT services in a standardized manner. Since IT service catalogs are commonly

CONTACT Matthias Pfaff pfaff@fortiss.org Department of Business Model and Service Engineering, fortiss GmbH, An-Institut Technische Universität München (TUM), München, Germany

designed for internal or individual purposes only, they are often not directly comparable, particularly across different organizations. The information collected in a benchmark exercise is generally obtained using questionnaires on a broad range of topics, such as employee costs, software licensing costs, quantities of hardware and so forth. All of these approaches have one commonality: a concept for a uniform data management is not considered although it is strongly recommended (Wollersheim, Pfaff, and Krcmar 2014; Pfaff and Krcmar 2015). Moreover, little work published to date in the IS literature addresses this challenge of data integration across different types of IT benchmarks. Thus, most literature sources omit facts related to data quality and data integration. The lack of a uniform description of any arbitrary parameter that is measured and the relationships between parameters, limit the comparability of different types of benchmarks. In general, a domain-specific ontology may be a solution to ensure that the collected data are meaningful and to overcome these limitations of data comparability (Horkoff et al. 2012; Pfaff and Krcmar 2014).

An ontology can either be constructed with assistance from domain experts or be discovered from domain-specific data. The first approach in ontology construction is performed manually and has high time and energy demands. If the ontology is to be developed for a more complex application area, then it tends to become increasingly subjective. An ontology may differ in numerous aspects depending on the recipient of the ontology, even when the ontology is constructed by domain experts. This is in contrast to the idea of a universal, common description of domain-specific knowledge. The second method of developing an ontology using the support of automated or semi-automated methods reduces the manual effort required for ontology construction and enhances the quality of the obtained ontology. Therefore, this paper is based on the results of the development of a domain-specific ontology in the ITBM domain supported by the use of natural language processing (NLP) techniques, as presented in (Pfaff and Krcmar 2014). This ontology was initially constructed based on already existing IT service descriptions and catalogs of numerous small- to medium-sized enterprises and on several questionnaires from different ITBM approaches. The data presented here were collected over the past seven years; they were supervised and evaluated within different benchmarking approaches and they encompass data from strategic and consortial IT benchmarks. Subsequently, as a result of the different acquisition channels through which the data were collected (i.e., on-line web platforms, Excel questionnaires and other sources), various different distributed data sources could be integrated using this domain ontology. In this paper, this ontology is used as the basis for a uniform data description in the domain of IT service management (ITSM) in general and ITBM in particular. To foster reuse of the benchmarking ontology the linkage to concepts provided by the DOLCE UltraLite (DUL) ontology (2010) is also implemented. The benchmarking ontology in version 1.1 is available at https://w3id.org/bmontology. In addition to this domain ontology, a system architecture for the integration of existing distributed data sources is presented in this paper. Thus, this work addresses the following questions: How can a system be designed to integrate existing distributed data sources using a domain-specific ontology? How can the administrator be supported to keep all the system components (mappings) up to date? To provide users with simple access to these distributed data sources NLP techniques are used to translate natural language requests into SPARQL (W3C 2008) queries. The system architecture follows a service-oriented design, encapsulating client (user)-side functionalities in a browser application and server-side funcionalities in replaceable (service) components. Because ontologies are not static entities but evolve over time, the system is able to handle version changes of the ontology to safeguard data accessibility to the attached data sources.

The remainder of this paper is organized as follows: Section 2 provides an overview of the relevant literature on the domain of ITBM, the ITBM ontology and on ontology-based applications. Section 3 addresses methods for data integration in ITBM and describes the proposed system architecture for the ontology-based data integration of various distributed data sources in this domain. Section 4 summarizes the results and metrics used for the data integration and presents the prototypical implementation of the proposed system architecture. Finally, Section 5 offers conclusions and perspectives for future work and extension possibilities of the proposed system.

2. Background

2.1. *The domain of IT benchmarking*

As a systematic process for improving organizational performance, benchmarks can be classified according to the type of study (e.g., processes, products, strategies or generic objects) (Carpinetti and Oiko 2008). Benchmarking partners may be units of the same organization, competitors in the same or different geographical markets, or organizations in related or unrelated industries. Thus, a distinction is drawn between internal and external comparisons of these performance measurements. Whereas an internal performance measurement focuses on the operation of a single company, an external performance measurement focuses on different companies. A benchmark can be subdivided into several process phases, beginning that the initial conception which describes the object of investigation and ending with optimizing and re-organizing internal (business) processes. In each of these phases of a benchmark, numerous data are collected in various data formats. These data consists of both qualitative and quantitative statements and are collected throughout the entire benchmarking cycle for every benchmark. Furthermore, these data are collected for every benchmarking participant. As previously stated by Ziaie et al. (2012) and described in a structural form by Riempp, Müller, and Ahlemann (2008), tool-based data collection is quite common in the ITBM domain.

The representation of business knowledge using ontologies has become popular in recent years, with a particular focus on the representation of business processes (Thomas and Fellmann 2009; Garcia-Crespo et al. 2011; Aldin and Cesare 2011; Jung et al. 2015; Hachicha et al. 2016). By nature, when an ontology is constructed with a focus on business processes, it lacks the information needed to shift the focus to financial aspects, which are of crucial importance in the ITBM domain. The same holds true for ontologies used for business modelling, system configuration and execution management systems, as presented by Cai et al. 2016), as well as for typologies in the context of business process management (BPM), as introduced by Müller et al. (2016). On the one hand, this also applies for ontologies in the context of ITSM (Freitas, Correia, and E Abreu 2008; Valiente, Garcia-Barriocanal, and Sicilia 2012), IT governance frameworks in the context of the Information Technology Infrastructure Library (ITIL) (Office 2011) and for related ontologies, such as the GoodRelations ontology (Hepp 2008) or the Financial Industry Business Ontology (FIBO) (Council 2016). On the other hand, ontologies such as the Business Model Ontology (BMO) (Osterwald, Pigneur, and Tucci 2005) and the e3-value ontology (Gordijn and Akkermans 2001) only focus on the conceptualization of economic aspects within a single enterprise or economic aspects within a network of enterprises. To the best of our knowledge, the only existing approach for measuring the impact of IT infrastructure changes on business processes and vice versa by an ontology was introduced by Vom Brocke et al. (2014). However, the focus of this study is in linking (inner) organizational processes to their corresponding IT resources. However, (semi) automatically compare IT-related and business-related performance indicators across company boundaries, a more fine-grained conceptualization of such information is needed, especially if linking external data sources (i.e., map ontology concepts to IT business-related KPIs) to concepts within an ontology.

2.2. *The IT benchmarking ontology*

The basis for the development of the ITBM ontology is IT service descriptions in the form of IT service catalogs from different (IT) companies. Moreover, ITBM questionnaires (based on Riempp, Müller, and Ahlemann (2008); Rudolph and Krcmar (2009)) are used to construct the ontology. The structural layout of an IT service catalog can be generalized to (i) basic organizational information (such as the number of employees and revenue), subsequently referred to as basic data service, and (ii) 19 additional IT services, describing more specific aspects of IT offerings (see Figure 1). These IT services provide some general information about the purpose of the service offering (for example providing a mailbox or a virtual machine/server) and detailed information about the performance and cost indicators that are used to measure the performance of this service. Note that calculations of indicators may be dependent on different services. For example, the storage service contains all costs associated with disk storage in a data center; however, some of these storage-specific costs are also required within a more general IT service, such as in the context of server costs (as disk storage is associated with servers in general). Additionally, costs originally related to the database service are based on both the general server costs as part of the infrastructure component and the more specific disk storage costs. Again, some cost indicators of the database service depend on the performance indicators of the server and data storage service. It is also possible that IT services inherit indicators or values from the basic organizational information (such as the total number of employees of an organization) to perform further calculations within a specific service based on such a basic indicator.

The structural layout of the IT service catalogs and IT service descriptions used to construct the ontology is presented in Figure 1. In short, IT services are mono-hierarchically structured. Each top-level service consists of a set of subordinated service segments and optionally additional indicator groups. As shown in Figure 1, the basic data service's segments correspond to general organizational information (i.e., organizational structure, IT costs, and so forth), and the remaining IT services are segmented based on whether they are cost or performance indicators and optionally grouped into smaller logical units (for example, the host or guest systems in the context of the virtual server service). Services may also include the costs of other services (e.g., a database service also includes the cost specified in a virtual server service). The core concepts of the benchmarking ontology are described in Section 3.1.2.

To allow ontologies to be machine processable, their modeling is often implemented in the Web Ontology Language (OWL) because it is part of the World Wide Web Consortium (W3C) languages (Calvanese, De Giacomo, and Lenzerini 2001; McGuinness and Van Harmelen 2004). Technically, OWL is an extension of the *Resource Description Framework* (RDF) and the *RDF Schema* (RDF-S), which are based on XML as an interchange syntax. As an extension of RDF and RDF-S, OWL ensures the smooth technical exchange of information among applications within the context of the Semantic Web and business

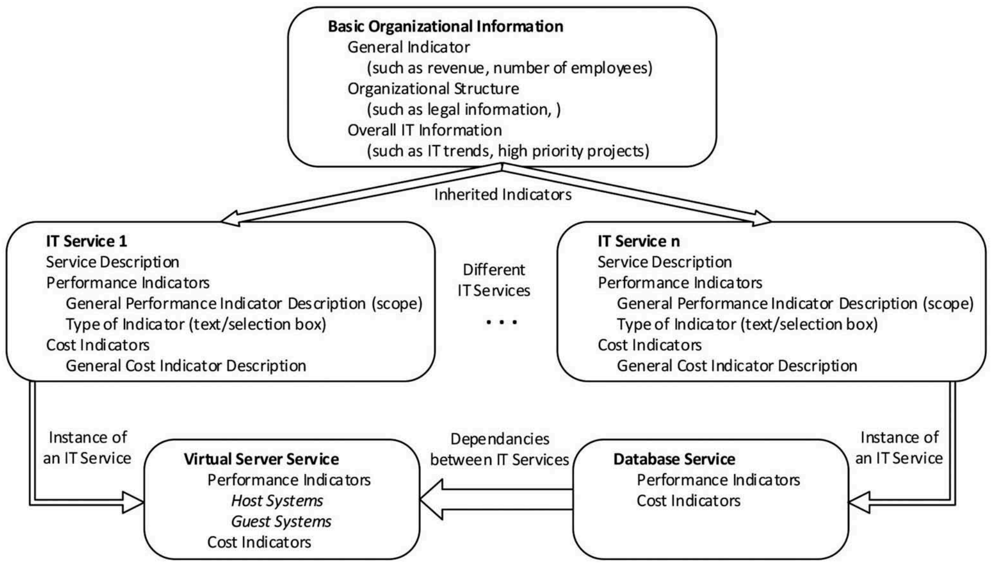

Basic Organizational Information
General Indicator
(such as revenue, number of employees)
Organizational Structure
(such as legal information,)
Overall IT Information
(such as IT trends, high priority projects)
Inherited Indicators
IT Service 1
Service Description
Performance Indicators
General Performance Indicator Description (scope)
Type of Indicator (text/selection box)
Cost Indicators
General Cost Indicator Description
Different IT Services
. . .
IT Service n
Service Description
Performance Indicators
General Performance Indicator Description (scope)
Type of Indicator (text/selection box)
Cost Indicators
General Cost Indicator Description
Instance of an IT Service
Virtual Server Service
Performance Indicators
Host Systems
Guest Systems
Cost Indicators
Dependancies between IT Services
Database Service
Performance Indicators
Cost Indicators
Instance of an IT Service

Figure 1. Structural overview of the IT service catalogs used to construct the ontology.

modeling frameworks (e.g., BPMN framework), which are also based on XML as their interchange syntax. An OWL ontology consists of: (i) classes as sets of individuals, (ii) individuals as instances of classes (i.e., real-world objects in the domain), and (iii) properties as binary relations between individuals. In addition to the implementation of domain knowledge, it is possible to define cardinality ranges and reasoning rules within an ontology. Several reasoning engines (e.g., Pellet 2015) exist that can be used to infer additional knowledge explicitly included in an OWL ontology (e.g., class equivalence checks). An OWL ontology can be modeled using open-source ontology editors such as Protégé (2014), which is one of the most common tools for ontology development (Khondoker and Mueller 2010).

To develop the ITBM ontology, we implemented a customized process based on the NeOn framework for ontology engineering Suárez-Figueroa, Gómez-Pérez, and Fernández-López (2012). The ITBM ontology is the result of a number of iterations of the overall ontology engineering process, which is based on an iterative-incremental life cycle model. Thus far, both the NeOn core scenario and the NeOn scenario for the reuse of ontological resources have been used. In addition, some of the NeOn activities were adapted to keep the engineering process as lightweight as possible. According to the NeOn specification for *knowledge acquisition* ontology learning was conducted to support the domain experts in performing the ontology elicitation activity; here, existing service catalogs and databases were analyzed using NLP techniques to extract the most important concepts, as described in detail in (Pfaff and Krcmar 2015). Following the NeOn guidelines for the specification activity, competency questions were formulated, categorized and prioritized (see Table 1). Moreover, the ITBM ontology is grounded in the upper ontology DUL to set the semantic foundation of the ITBM ontology (for details on the relevant concepts that are linked in DUL and the ITBM, see Section 3.1.2). The ITBM ontology was modeled using the open-source ontology editor Protégé.

2.3. *Ontology-based applications*

Storing information in ontology-based knowledge bases or systems is becoming increasingly popular across various areas of research. Lehmann et al. (2015) introduced an approach to extract knowledge from Wikipedia using the Semantic Web and linked data technologies, called DBpedia.

Table 1. Extract of competency questions created during the *specification* activity grouped by pre-established categories as suggested by NeOn: (i) indicator structure, (ii) individual benchmarks, and (iii) participants and values. The square brackets indicate lists of values (Pfaff, Neubig, and Krcmar 2017).

Group	Competency Question	Exemplary Answer
Indicator Structure	What performance indicators exist?	[NumberOfUsersIndicator]
	What performance indicators are contained in the BENCHMARK_NAME in YEAR?	[NumberOfUsersIndicator]
	Regarding BENCHMARK_NAME of YEAR, how many cost indicators exist?	NUMBER
	What IT services are of interest (i.e., values have been provided for) for the ORGANIZATION_NAME ?	[BasicDataIndicator]
	How many values have been provided for the revenue indicator of the SERVICE_NAME in total?	NUMBER
Individual Benchmarks	How many benchmarks exist?	NUMBER
	In which years was the BENCHMARK_NAME conducted?	[YEAR]
	Which indicators have been queried in at least two benchmarks?	[DesktopInstallCostIndicator]
Participants and Values	How many organizations exist?	NUMBER
	How many organizations have participated in at least one benchmark?	NUMBER
	Does ORGANIZATION_NAME participate in at least one benchmark called BENCHMARK_NAME ?	YES/NO
	What is the yearly revenue of ORGANIZATION_NAME ?	[(YEAR, NUMBER)]
	Regarding YEAR, what is the minimum number of employees of organizations having a revenue between $NUMBER and $NUMBER ?	NUMBER

DBpedia serves as a linked data source on the Web since it covers RDF links pointing to various external data sources and vice versa. This linkage (mapping) is performed manually by the community. For DBpedia, Paredes-Valverde et al. (2015) developed ONLI (ontology-based natural language interface) for querying DBpedia using natural language techniques. Rodríguez-García et al. (2014) proposed a semantically enhanced platform based on an ontology for annotating cloud services to assist in the process of discovering the cloud services. This annotation for the cloud services semantic repository is generated automatically, but no further external data sources are directly attached by the semantic structure of an ontology. Ong et al. (2017) introduced Ontobee as a linked ontology data server that stores ontology information using RDF triple-store technology that supports the query, visualization and linkage of ontology terms in the biomedical and biological domains. Ontobee primarily used for ontology term querying and result visualization, and it allows the execution manually written SPARQL code. In the healthcare domain, Lasierra et al. (2014) introduced an ontology-based system to capture knowledge regarding item management and usage for hospitals and medical centers. The focus of this system is to align and unify dispersed health catalog modeling items and the structure of the organization related to their management rather than in data access of external sources by an ontology. Using Ontop, Calvanese et al. (2016) presented an open-source ontology-based data access (OBDA) system that is used for querying relational data sources in terms of executing manually written end-users SPARQL queries. The mapping is of mappings to an existing ontology and by executing end-users SPARQL queries. The mapping of ontology concepts to data sources is performed manually using traditional mapping languages, such as the W3C RDB2RDF mapping-language (R2RML) (Souripriya, Seema, and Cyganiak 2012). The advantages of an ontology-based data management approach were evaluated by Daraio et al. (2016). Keeping all components of the system up to date, particularly the ontology and the mapping, is still the responsibility of the administrators of the system and is performed manually. Tatu et al. (2016) presented an approach for converting users natural language questions into SPARQL for querying and retrieving answers from an RDF store. Because the focus of their research is in transforming semantic structures identified in unstructured data sources (documents) to an RDF store that is accessible via natural language questions, the mapping of ontological concepts to (external) data sources is beyond the scope of their proposed framework. The same constraint holds true for OntoNLQA (Asiaee et al. 2015), which was introduced to query RDF data annotated using ontologies to allow posing questions in natural language. In the clinical and clinical research contexts, Mate et al. (2015) introduced a system for linking information of different systems using declarative transformation rules for ontologies of the source system and the target system. Here, the mapping of the target ontology to the source ontology is also created manually. Focusing on specific technologies for the translation of RDB to RDF, Michel, Montagnat, and Faron-Zucker (2014) and Sahoo et al. (2009) provided a brief overview on the individual technologies. As a symmetrization of the work, at present, domain-specific mappings for data semantics that lies outside an RDB schema are commonly performed manually.

3. Benchmarking data and knowledge integration

A system for integrating various distributed data sources and documents must be able to not only handle various data sources but also integrate various data formats to serve as an effective tool for knowledge processing and knowledge representation (Nalepa 2010; Pfaff and Krcmar 2015). Therefore, this paper presents an ontology-based knowledge support system with a domain-specific ontology as a pivotal methodology for representing domain-specific conceptual knowledge, as proposed by Guo and Zhang (2009) and Pfaff and Krcmar (2014, 2015). Because ontologies offer certain advantages over regular database schema, for example, they are highly flexible and enable modifications and extensions (Zhang, Hu, and Xu 2010) in a straightforward manner, the presented system architecture addresses this unique capability through the use of a separate metadata repository. This repository is used to map the distributed data sources to the ontology (and its possible version changes over time) in a continuous update/integration interval.

3.1. *System architecture*

The basic service-oriented architecture of the web-based system for ontology-based data integration is illustrated in Figure 2. The web application is implemented using the Play Framework (Play 2016), offering stateless representational state transfer (REST) services (Fielding and Taylor 2000) for (client-side) interactions and encapsulating application logic in services with a uniformly defined interface (server-side). In this figure, *Client* represents the web browser-based user interface, allowing the user to interact with the server-side implementation. On the server side, the *Web*

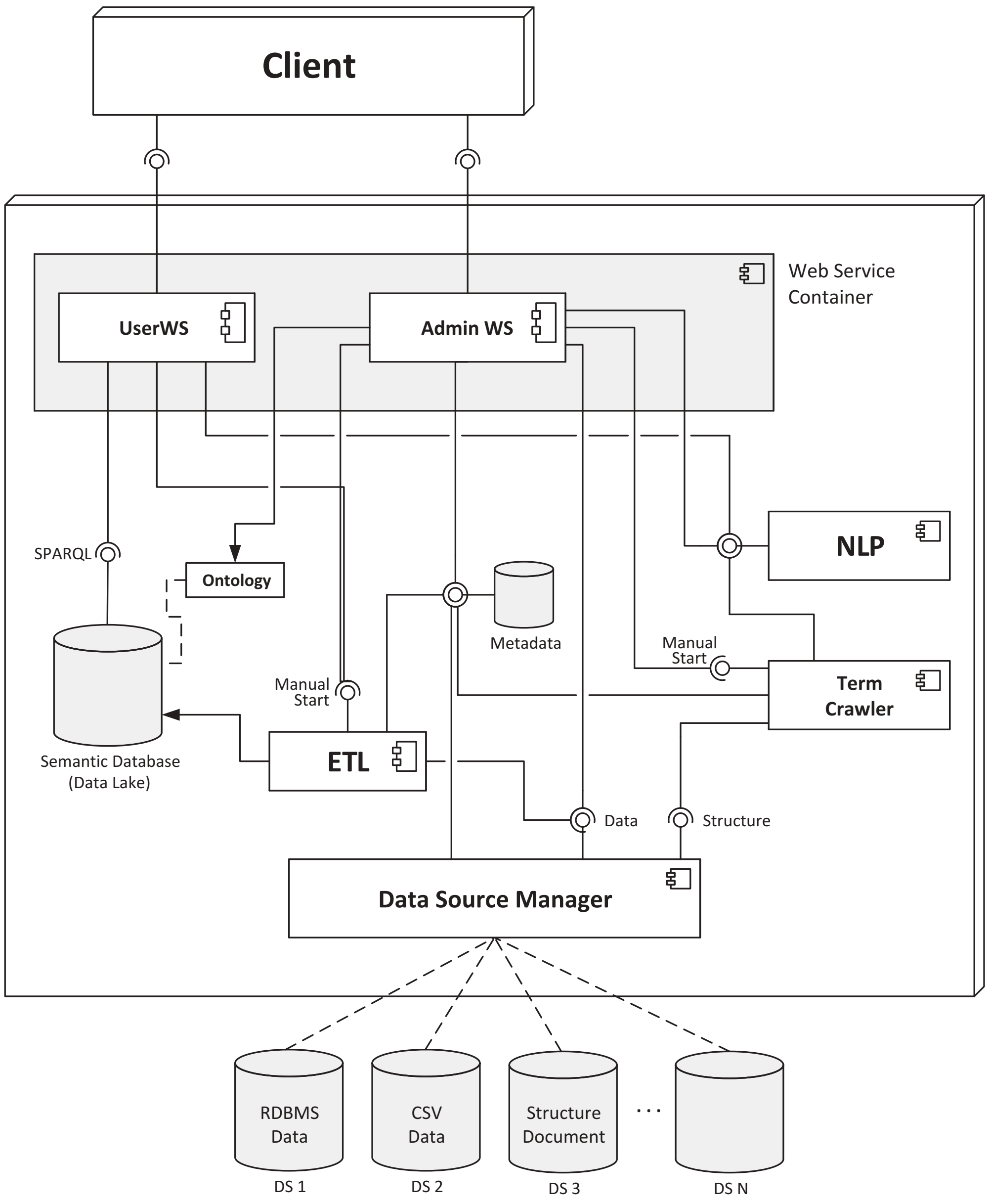


Figure 2. System architecture for ontology-based data integration.

Service Container encapsulates web services (WS) for both user roles: the general user (User WS) and the administration user (Admin WS). The general user has only limited rights to modify the links between the attached data sources and the ontology: thus he is only allowed to formulate natural language requests, which are automatically translated into SPARQL queries using NLP techniques and the extract, transform, load (ETL) module. Conversely, the administration user is allowed to reconfigure the complete system, including the mapping configuration. At present, this type of user and access management is sufficient because all individuals using the system have the right to access all data attached to the system. For future implementation possibilities in terms of more finde-graind user management and access controls, see Section 5.

3.1.1. *Web service container*

As previously mentioned, the system is implemented in the REST paradigm and is therefore accessible via the Web, and the web service container provides functionalities for two different user roles. The *user web service* (User WS) processes user requests in natural language form. These requests are analyzed using NLP techniques and are transformed and forwarded to the semantic database using SPARQL. By design, the NLP module, which can be executed by any user, focuses on a high rate of accuracy in its first iteration with the purpose of identifying as many domain-specific terms as possible within the data sets to be analyzed. In its second iteration, a high rate of *precision* is desired, identifying only results relevant to the (user or administration) queries (Pfaff and Krcmar 2015). In addition to these search requests, users may also trigger the ETL module to reload the linkage between the ontology and the attached data sources. Note that through the user role, only the existing linkage between the attached database and the ontology can be reloaded. It is not possible for the user to update or modify links between concepts of the ontology and database objects.

The *admin web service* (admin WS) performs the following operations:

- *Ontology Update*: Through this operation, it is possible to either upload a new ontology or update an existing one. This ontology is stored in the semantic database. At this point, the new ontology is versioned, and the metadata repository is flagged as no longer valid due to possible mismatches between the data sources and the new ontology (see Section 3.1.3 for details). Moreover, the dictionary that is part of the NLP module may be updated with new terms introduced by concepts or synonyms contained within the new version of the ontology.
- *Data Source Management*: The attached data sources can be configured using the data source manager. It is also possible to connect structured and unstructured data sources. All necessary configurations for access to the data sources, such as internal database names or source folders, are stored in the metadata repository. Moreover, all attached sources (ontology and databases) are versioned to ensure that later mapping activities are linked with the correct version (for details see Section 3.1.3).
- *Structure Mapping*: For a user with the administrator role, it is possible to specify the mapping of the attached databases to concepts contained within the ontology. Thus, this role possesses the right to read from the attached data sources and the right to write the mappings into the metadata repository. Using the NLP module, similar terms contained in the ontology and the attached data sources are first recommended as mapping candidates.
- *Term Crawler Configuration*: It is possible to configure the term crawler to run periodically in addition to its manual execution by a user with the administrator role. The term crawler, which uses NLP techniques was previously introduced Pfaff and Krcmar (2015).
- *Manual ETL Start*: In addition to the periodic execution of the ETL process, it is possible for this event to be triggered manually by a user or administrator.

All operations are performed through a graphical user interface (GUI) with which administrators and users are able to trigger the previously mentioned operations stepwise guided by an operation wizard.

3.1.2. *SemDB and ontology*

The semantic database (SemDB) is implemented with Virtuoso Universal Server as a triple store (Erling and Mikhailov 2010; OpenLink 2015). Because the database represents a SPARQL endpoint, it can be accessed through SPARQL queries. In addition to the semantically processed data provided by the attached external databases, SemDB also stores the ontology used for the mapping process.

The ontology can be divided into the three following sections: *individual benchmarks* (equivalent to one specific benchmark), *participants and values* and the *general indicator declaration*. Three concepts are used to describe the *individual benchmarks*, including a customizable structure of selectable indicators (measured within a benchmark), participants (viz., organizations) and the values that may be instantiated based on the concepts described in the *participants and values* section. The indicators themselves and their hierarchical and intermediate relationships are organized in the *general indicator declaration* section. An *indicator* itself is either a *PerformanceIndicator* or a *CostIndicator*. Indicators at the *PerformanceIndicator* level are non-cost indicators, such as quantity details or performance details. As indicated by the name, *CostIndicator* subsumes all indicators related to financial aspects that are compared in a benchmark. Because each indicator is included in at least one benchmark, this information is represented through by the indicator label. In this manner, it is possible to associate an indicator of one benchmark with an indicator of a different benchmark that has a different name but is identical from a semantic perspective (i.e., measure the same objective). A specific benchmark is specified by its label, represented by an arbitrary string and the year is represented by the standardized *gYear* literal type according to (Peterson et al. 2012) within the concept of *individual benchmarks*. Here, the type property refers to the set of benchmark types (such as a process, product, strategic or generic benchmark (cf. Carpinetti and Oiko 2008) and is limited to those values. For the connection to DOLCE, the *benchmark* class has been defined as a sub class of the *DUL:Event* class of the DUL ontology.

The components property facilitates the assignment of multiple *BMComponents*. Each BMComponent is either an instance of an indicator or a collection *(BMCategory)* of indicators. Consequently, it is possible to instantiate any arbitrary hierarchical structure of *BMCategories* and indicators. A *participation* in a benchmark is represented for each participating *organization* and its associated responses to an indicator by the intermediate concept *IndicatorDeclaration*. Thus, it is possible to associate an organization with a benchmark even without the existence of any specific indicator values (e.g., no responses have yet been given but the organization is participating in the benchmark) using the concept of *participants and values*. To foster reuse, an *organization* refers to the *DUL:Organization* concept provided by the DUL ontology Gangemi (2010).

Figure 3 provides a conceptual overview of these three ontology sections and the relations in between. Grey nodes indicate DUL concepts and properties. The nodes of the graph illustrated in Figure 3 refer to *concepts* (i.e., classes) or *datatypes* (cf. Motik, Patel-Schneider, and Parsia 2012) of the ontology, whereas the edges refer to *properties* provided by the ontology. A class can also be considered as a set of instances, and a subclass can be considered as a subset of those instances (Motik, Patel-Schneider, and Parsia 2012). A property can either establish a direct link between instances of two classes or link an instance to a literal (i.e., a value of a certain data type).

3.1.3. *Extract, transform, load module and metadata repository*

The ETL module is implemented as an independent single thread with a continuous execution interval in addition to being a triggered event (executed on demand by the user). The main tasks of the ETL process are (i) loading the external data into the semantic database by generating a virtual table based on the database structure of the external data base, and (ii) resolving redundancies that may occur during the loading process by the entity resolution (ER) step (see Section 3.1.4 for details on ER).

Prior to the execution of the ETL module, the versions of the currently used ontology and its attached databases are identified. The versioning of the ontology is assured because an uploaded ontology is always annotated with its version number (using the OWL *versionInfo* tag). The data source manager is used to ensure the correct mapping of the relational structure of the attached databases to the

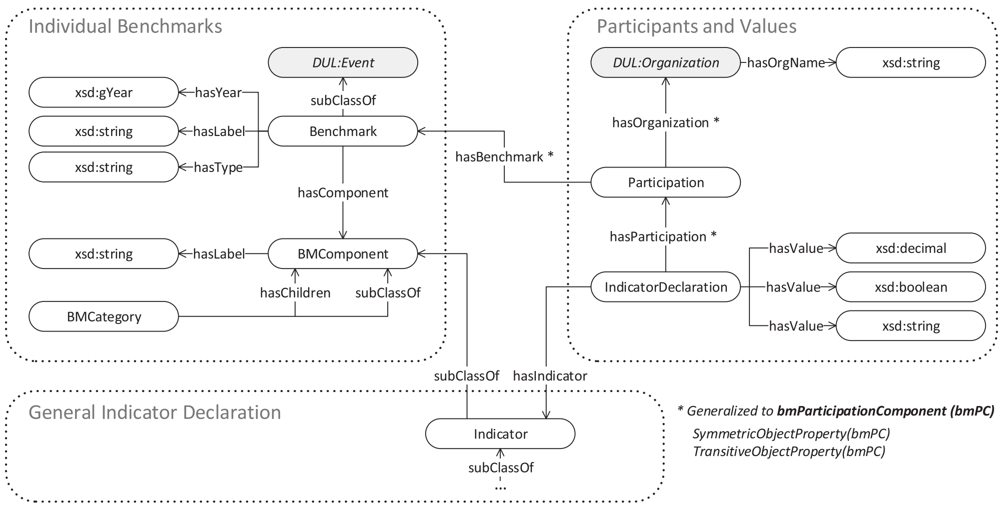


Figure 3. Benchmarking ontology. Source: Pfaff, Neubig, and Krcmar (2017).

corresponding ontology version. These steps are crucial to ensuring compatibility between the metadata and the ontology/databases and thus to guarantee that the mapping is performed on a sound basis.

The mapping of metadata that are stored in the metadata repository is two-fold. The first part specifies a set of transformation rules that transform the relational models of the connected databases into virtual models (i.e., nested SQL queries rather than physically transformed tables) and where each table of a connected database corresponds to a concept within the ontology. The second part specifies mappings from this virtual model to the target ontology itself. These mappings consist of the assignment between the entities and attributes from the data sources and their corresponding concepts and properties of the ontology. According to these mappings, the data integration process is performed stepwise as follows:

(i) Load the mapping entries from the metadata repository in accordance with the selected versions of both the ontology and the connected databases.
(ii) Apply transformation rules to the relational models of the connected databases to create an intermediate model with bidirectional links between tables; this is realized creating a set of SQL statements wrapped around the original tables.
(iii) Load data from the attached databases via the data source manager using the generated SQL statements.
(iv) According to the second part of the mapping specifications, map tables to concepts by converting their rows into instances of the ontology using the triple-store format.
(v) Use the Virtuoso bulk loader to load the data into a new graph within the semantic database; old data are retained in the old graph.
(vi) Check whether the new graph differs from the data loaded in previous ETL iterations and log changes.

The following example illustrates the result of the ETL process (i.e., the *mapping* between the ontology) based on Figure 3 and two external data sources. The name space used for the uniform resource identifiers (URIs) for the concepts and properties of the ontology is represented in shortened form by the prefix *bm*. The instances of benchmarking values depending on which data source is mapped are indicated by the prefixes *v* and (i). In this example, two indicators (Indicator1a and Indicator2a) of a data set *a* from the first data source *v* and one indicator (Indicator 2b) of a different data set *b* from the second data source *v* are linked to each other using the benchmarking ontology. As previously noted and shown in Figure 3, a data set is always linked to an organization that is a participant in a specific benchmark. Thus, these three indicators are associated with two organizations (organizations A and B, where organization A is a participant in two benchmarks). The linkage between these three indicators and the ontology is shown below. In this example, *OrganizationA* is a participant in *benchmarkA*, providing *indicator1* and *indicator2*, and it is also a participant in *benchmarkB*, providing only *indicator2*. *OrganizationB* is a participant only in *benchmarkB*, providing *indicator1*.

```
# Instances of indicators with labels for each benchmark.
v:indicator1             rdf:type bm:Iindicator1;
                         bm:label "Indicator␣1a"^^xsd:string.
v:indicator2             rdf:type bm:Indicator2;
                         bm:label "Indicator␣2a"^^xsd:string.
i:indicator1             rdf:type bm:Indicator2;
                         bm:label "Indicator␣2b"^^xsd:string.
```

```
# Definitions of benchmarks.
v:benchmarkA             rdf:type bm:Benchmark;
                         bm:year 2015;
                         bm:label "Benchmark␣A"^^xsd:string;
                         bm:components v:indicator1;
                         bm:components v:indicator2.
i:benchmarkB             rdf:type bm:Benchmark;
                         bm:year 2015;
                         bm:label "Benchmark␣B"^^xsd:string;
                         bm:components i:indicator2.
```

```
# Definitions of the organizations for each benchmark.
v:OrganizationA          rdf:type bm:Organization;
                         bm:organizationName "Name␣of␣Org␣A"^^xsd:string.
v:OrganizationB          rdf:type bm:Organization;
                         bm:organizationName "Name␣of␣Org␣B"^^xsd:string.
i:OrganizationA          rdf:type bm:Organization;
                         bm:organizationName "Name␣of␣Org␣A"^^xsd:string.
```

```
# Definitions of participation.
v:OrganizationA_part     rdf:type bm:Participation;
                         bm:benchmark v:benchmarkA;
                         bm:organization v:OrganizationA
v:OrganizationB_part     rdf:type bm:Participation;
                         bm:benchmark v:benchmarkA;
                         bm:organization v:OrganizationB
i:OrganizationA_part     rdf:type bm:Participation;
                         bm:benchmark i:benchmarkB;
                         bm:organization i:OrganizationA
```

```
# Values of indicators.
v:OrganizationA_ind1     rdf:type bm:IndicatorDeclaration;
                         bm:indicator v:indicator1;
                         bm:participation v:OrganizationA_part;
                         bm:indicatorValue 100
v:OrganizationA_ind2     rdf:type bm:IndicatorDeclaration;
                         bm:indicator v:indicator2;
                         bm:participation v:OrganizationA_part;
                         bm:indicatorValue 200.
i:OrganizationA_ind1     rdf:type bm:IndicatorDeclaration;
                         bm:indicator i:indicator1;
                         bm:participation i:OrganizationA_part;
                         bm:indicatorValue 100.
v:OrganizationB_ind1     rdf:type bm:IndicatorDeclaration;
                         bm:indicator v:indicator1;
                         bm:participation v:OrganizationB_part;
                         bm:indicatorValue 500.
```

3.1.4. *Entity resolution*

After data from multiple databases have been loaded using the ETL module, multiple instances resolved from different data sources may exist that actually refer to the same thing; in the above example, *organization A* exists in both connected databases (i.e., *v* and *i*). Thus, from the SemDB's point of view, they are considered as two distinct instances; consequently, associated properties are not considered as belonging to the same organization (e.g., organization *v:OrganizationA* participates in benchmark A, and a different organization *i:OrganizationA* with the same name participates in benchmark B).

To consider both instances equally and thus integrate all associated data sets, ER has to be performed. In contrast to the mapping metadata, the ER metadata are only bound to the ontology's version. For all concepts with instances to be resolved, the ER metadata specify criteria on how to compare such instances, i.e., (i) transformations to be conducted to ease comparison and (ii) criteria about the comparison itself. Considering organizations, transformations involve crossing out common suffixes (e.g., *Inc*), and comparison criteria may include the calculation of string distance metrics (e.g., Levenshtein distance). If two instances are considered equal with respect to the specified comparison criteria, then they are resolved by adding an *owl:sameAs* definition. In the current version of the system, only organizations are considered for ER. Data contributions within a benchmark are not integrated, even if the same indicator is requested within the scope of two different benchmarks running at the same time period. This is because each contribution refers to a distinct benchmark instance and we want to keep that knowledge.

3.2. *Semi-automatic mapping recommender*

To support the mapping of database contents to ontology concepts, a semi-automatic mapping recommender is developed. Here, 'semi-automatic' refers to the fact that mappings are recommended in the first place and not applied automatically; thus, human interaction is needed to confirm recommended mappings for the purpose of quality assurance. The system supports two different types of mapping recommendations. The first type assumes that an entire database table corresponds to an existing ontology concept, and the second type assumes that each database table record is mapped to a different ontology concept. In both cases, mappings are only recommended if a certain level of confidence is reached (see also Section 4.2).

Mapping (virtual) tables to ontology concepts: Often, a (physical) table from the original database schema directly corresponds to a concept defined in the ontology. In this case, all records of this table are converted into instances of this concept. Note that if concepts in the ontology are specified on a more fine- or coarse-grained level of abstraction, such a table may still be constructed virtually using appropriate SQL statements (e.g., JOINs); within the scope of the system, these types of tables have been referred to as *generators*. For this type of mapping, the implementation in pseudocode is shown in Listing 2.

Listing 1 Type-1-Generator-Mapping in pseudo-code

```
generateMappingsFromGeneratorLayer():

    // Create concept list and generator list
    conceptList = getConceptNamesUsingSparql()
    generatorList = getGeneratorNamesFromMetadata()

    // Clean generators by deleting unnecessary prefixes
    for (i, name) in generatorList:
            generatorList[i] = clean(name)

    // Execute bipartit matching
    matchings = bipartiteMatching(getLevenshteinMetric(),
        threshold = 0.6, conceptList, generatorList)

    // Create empty set of mapping meta data
    // and add identified matchings
    mappingMetadata = createEmptyMappingMetadata()
    for (concept, generator) in matchings:
        mappingMetadata.push(createMappingMetadata(
            from = generator, to = concept))

return mappingMetadata
```

Mapping (virtual) table records to ontology concepts: Occasionally records are not meant to be converted to instances of the same concept but are rather partitioned to different concepts. In this case, a specific table is chosen, and each of its records is converted into one instance of a specific concept of the ontology. For this second type of mapping, the implementation in pseudocode is shown in Listing 2.

Listing 2 = Type-2-Generator-Mapping in pseudo-code

```
// Parameters are (i) the name of the generator,
// which instances shall be mapped to concepts
// and (ii) the pivotal columne name pivotal used fot the mapping,
generateMappingsFromGeneratorInstances(generator, column):

    // Create empty concept list and an empty list of instances
    conceptList = getConceptNamesUsingSparql()
    instanceList = []

    // Load instances (single row) of the generator form the external
    // data source and add the corresponding value to the list of instances
    result = executeSQL(generatorManager[generator].sql)
    for row in result:
            instanceList.push(row[column])

    // Execute bipartit matching
    matchings = bipartiteMatching(getFuzzyJaccardJaroWinklerMetric(),
        threshold = 0.2, conceptList, generatorList)

    // Create empty set of meta data for the mappings
    // and populate this set by the calculated best matches
    // of the FuzzyJaccardJaroWinklerMetric
    mappingMetadata = createEmptyMappingMetadata()
    for (concept, instance) in matchings:

        // A row of the generator (from) and a concept (to)
        // is only mapped if the generator row is a match
            mappingMetadata.push(createMappingMetadata(
                from = generator, to = concept,
                require = (column, instance)))

return mappingMetadata
```

Both of these mapping cases are implemented using the same underlying bipartite matching algorithm (based on (Kuhn and Yaw 1955)) while differing in terms of its run-time configuration. In the first case (i.e., mapping (virtual) tables to ontology concepts), the total set of virtual and physical table names and the names of the ontology concepts are used as the input configuration. In the second case (i.e., mapping (virtual) table records to ontology concepts), the total set of rows of a specified table and the names of ontology concepts are used as the input configuration for the mapping algorithm. The respective configurations of the algorithms are described in the following.

3.2.1. *Bipartite matching algorithm*

Both of the scenarios explained above are based on a highly configurable bipartite matching algorithm. Starting with two sets of items, this algorithm assigns each item of the first set to an item of the other set such that the total difference of pairwise matched items is as minimal as possible. Moreover, items are only matched if a certain confidence threshold of confidence is exceeded, meaning that the resulting set of matched items is not necessarily complete. As input, the bipartite matching algorithm requires two parameters, namely, a metric to be used to calculate the distance between two items and a minimum confidence threshold.

The implementation of the bipartite matching algorithm is based on an execution of the Hungarian method (Kuhn and Yaw 1955). In the first step, a cost metric is calculated by assigning each pair of items from the two different sets a specific distance, which is expressed as a floating point number between 0 and 1. Here, 0 refers to the equality of items, and 1 refers to a maximum difference. The derived cost matrix is passed to the Hungarian method, which assigns each item of the first set an item of the second set. After the Hungarian method has completed, the similarity of the items within each matched item pair is derived by subtracting the beforehand calculated cost from 1. If the resulting similarity is below the specified minimum (i.e., the passed confidence threshold), then this match is removed from the result set.

Two different groups of metrics are used within the mapping recommendation system based on the metric class of the SimMetrics[1] Java library. The first group of metrics compares strings and consists of the Levenshtein distance (Levenshtein 1966), and the Jaro-Winkler distance (Winkler 1990) is used to compare single words. The second is more coarse grained and compares complete groups of words. It is based on the Jaccard index (Jaccard 1901) (i.e., comparing two sets by dividing the number of common items by the number of (distinct) total items), which additionally makes use of the previously calculated distances of the first group of metrics. Assuming equality between items, even if they slightly differ, these metrics are denoted as fuzzy Jaccard metrics. Thus, in our case, this *FuzzyJaccardJaroWinkler metric* calculates the Jaccard index while assuming equality between two items if their Jaro-Winkler similarity is greater than 0.94. For further details see Section 4.2 .

4. Results and evaluation

4.1. *Ontology*

At present, the ITBM ontology (version 1.1) consists of a number of statements which are summarized in Table 2.

The number of classes corresponds to the concepts described in the previous sections, including the 20 top-level service classes (one of which is the basic data service), corresponding to IT services that are commonly measured within an IT benchmark. The 1,250 indicator classes correspond to key performance indicators that are measured during an IT benchmark. Entities of the indicator taxonomy do not have their own properties defined because they only inherit them from the BMComponent class. Therefore, only a small set of object and data properties need to be additionally defined, and they are shown in Figure 3. Currently, the majority of axioms refer to

Table 2. Number of classes, properties, axioms and annotations in the ITBM ontology.

Ontology Metric	#	Ontology Metric	#
Classes	1,250	Logical Axioms	2,927
Object Properties	113	Annotations	5,362
Data Properties	10		

the number of *SubClassOf* definitions. However, axioms on the domain and range of object properties and statements relevant to the characterization of disjoint classes also exist. The number of annotations includes bilingual (viz., English and German) *rdfs:label* for all classes. The description logic expressiveness for the benchmarking ontology itself is $\mathcal{SHI}(\mathcal{D})$, and in combination with the DUL ontology the logic expressiveness is $\mathcal{SHIN}(\mathcal{D})$.

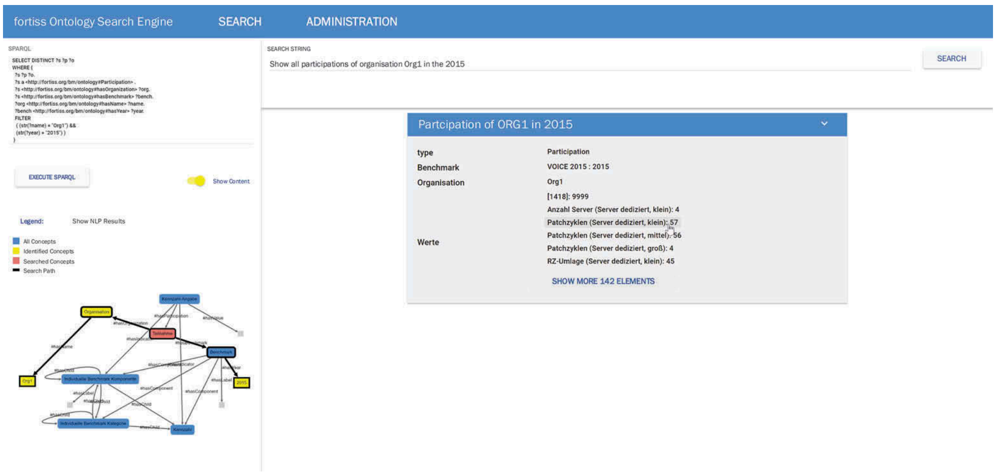
4.2. *Metrics and minimum confidences of the mapping recommender*

Both the previously described metrics (see Section 3.2.1) and the best minimum matching confidences have been derived and proven in various experiments. Regarding the mapping within the *virtual* table layer (case one in Section 3.2), a simple Levenshtein metric with a minimum confidence of 0.6 is applied; in the instance mapping scenario (case two in Section 3.2), a fuzzy Jaccard metric using the Jaro-Winkler metric is used. The internal threshold of equality has been set to 0.94 as already mentioned; the minimum confidence threshold necessary for accepting a match resulting from the Jaccard index has been set to 0.2. The computational complexity is of square, for calculating the cost matrix and calculating the distances for each pair of items. If the fuzzy Jaccard metric is used for the similarity check, then the computational complexity increases to mn^2, where m is the (largest) number of words contained in each item. Regarding to the Hungarian method, we utalize its optimized version, reducing its complexity from $O(n^4)$ to $O(n^3)$. Removing the items with a distance that is worse than the minimum confidence threshold is performed linearly. Thus, the overall computational complexity of the bipartite matching algorithm is $O(n^3)$ (Edmonds and Karp 1972).

4.3. *Prototypical implementation*

4.3.1. *User interface for natural language text to SPARQL queries*

A web interface can be used to access the attached data sources via natural language text (text-to-sparql). This client-side user interface is implemented using AngularJS (Google 2016) and is shown in Figure 4. As a result of the German data sets, the outputs of the user search ('Show all participations of organisation ORG1 in year 2016') are presented in the German language.The search tree within the ontology is presented directly underneath the automatically generated



```
SELECT DISTINCT ?s ?p ?o
WHERE {
  ?s ?p ?o.
  ?s a <http://fortiss.org/bm/ontology#Participation> .
  ?s <http://fortiss.org/bm/ontology#hasOrganization> ?org.
  ?s <http://fortiss.org/bm/ontology#hasBenchmark> ?bench.
  ?org <http://fortiss.org/bm/ontology#hasName> ?name.
  ?bench <http://fortiss.org/bm/ontology#hasYear> ?year.
  FILTER
   ( (str(?name) = "Org1") &&
   (str(?year) = "2015") )
 }
```

Show all participations of organisation Org1 in the 2015

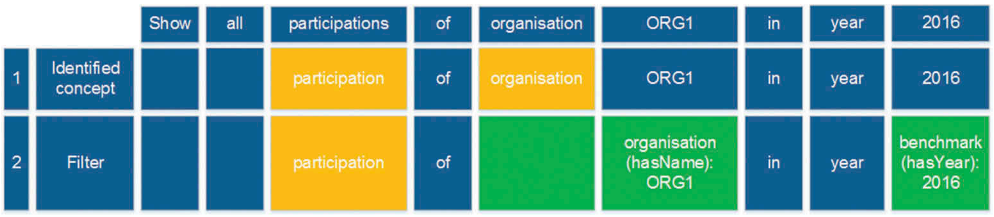
🟦 All Concepts
🟨 Identified Concepts
🟥 Searched Concepts
▬ Search Path

Partcipation of ORG1 in 2015

type	Participation
Benchmark	VOICE 2015 : 2015
Organisation	Org1
Werte	[1418]: 9999 Anzahl Server (Server dediziert, klein): 4 Patchzyklen (Server dediziert, klein): 57 Patchzyklen (Server dediziert, mittel): 56 Patchzyklen (Server dediziert, groß): 4 RZ-Umlage (Server dediziert, klein): 45

SHOW MORE 142 ELEMENTS

Figure 4. Client-side user interface for ontology-based data access.



		Show	all	participations	of	organisation	ORG1	in	year	2016
1	Identified concept			participation	of	organisation	ORG1	in	year	2016
2	Filter			participation	of		organisation (hasName): ORG1	in	year	benchmark (hasYear): 2016

Figure 5. Stepwise identification and assignment of identified tokens.

SPARQL query. Blue nodes represent the corresponding concepts in the ontology that the user was searching for data sets.

In this previous example, the search string 'Show all participations of organisation ORG1 in year 2016' is parsed and processed by the NLP module. In the first step, concepts that the user searched for are identified by comparing all words within the search string with the *label* description of all concepts. Note that all already specified concepts of the system are already lemmatized within a *CachingDictionary* as lemmatization of all concepts for every single user search would be very time consuming.

As shown in Figure 5, the Levinshtein distance of each lemmatized word within the search string and the implemented concepts is calculated. In the next step, these distances are evaluated against the operations needed to transform the lemmatized word into a concept. Only if this is possible by less than three NLP operations is the entered word identified as a concept. In Figure 5 all identified concepts are highlighted using yellow background color. Analogous to the concept identification, the remaining words are analyzed to identify literals that are specified within the ontology. Consequently, the identified literals are transformed into filter parameters such as *subject, predicate, and object*. The *subject* specifies the concept for which the filter is set, the *predicate* specifies the *rdfs:type*, and the *object* is set by the literal itself. The following example shows the filter results for the identified literal 'ORG1'. In the last step, all identified literals are marked as 'processed' (indicated by the green background color in figure 4.3.1).

```
Filter
    Type URI:          ,,http://fortiss.org/bm/ontology#Organization''
    Predicate URI:     ,,http://fortiss.org/bm/ontology#hasName''
    Value:             ,,ORG1 GmbH''
Filter
    Type URI:          ,,http://fortiss.org/bm/ontology#Organization''
    Predicate URI:     ,,http://fortiss.org/bm/ontology#hasName''
    Value:             ,,ORG1''
```

4.3.2. *Data source configuration and mapping recommender*

The configuration of the mapping between an ontology and corresponding data sources is supported by an administrator user interface (see Figure 6). For each data source this configuration needs to be performed before the mapping of concepts to generators can be conducted. For consistency and data loss prevention reasons, all changes of the mapping between data sources and the ontology are stored temporarily and need to be confirmed separately after the configuration procedure. The mapping is performed stepwise, following the workflow shown in Figure 7.

(A) An external data source needs to be selected first. In this step, all already configured data connections are available for selection

(B) Based on the selected data source, different editing options for the mapping are available, depending on the different work-flow states.

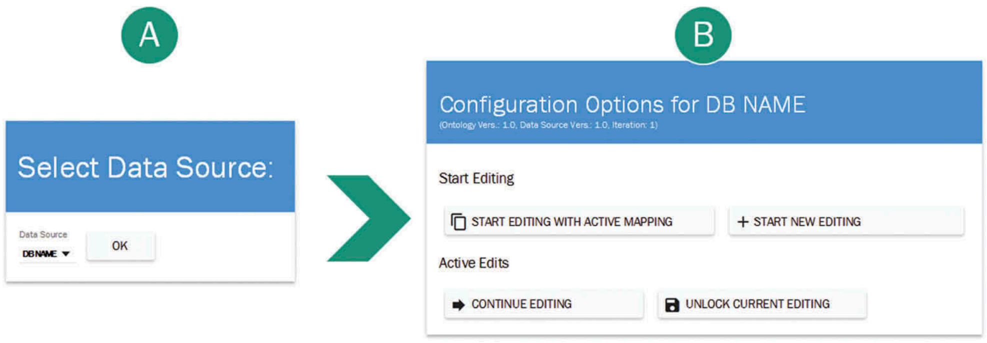

A
B
Select Data Source:
Data Source
DBNAME
OK
Configuration Options for DB NAME
(Ontology Vers.: 1.0, Data Source Vers.: 1.0, Iteration: 1)
Start Editing
START EDITING WITH ACTIVE MAPPING
+ START NEW EDITING
Active Edits
CONTINUE EDITING
UNLOCK CURRENT EDITING

Figure 6. Admin: mapping configuration.

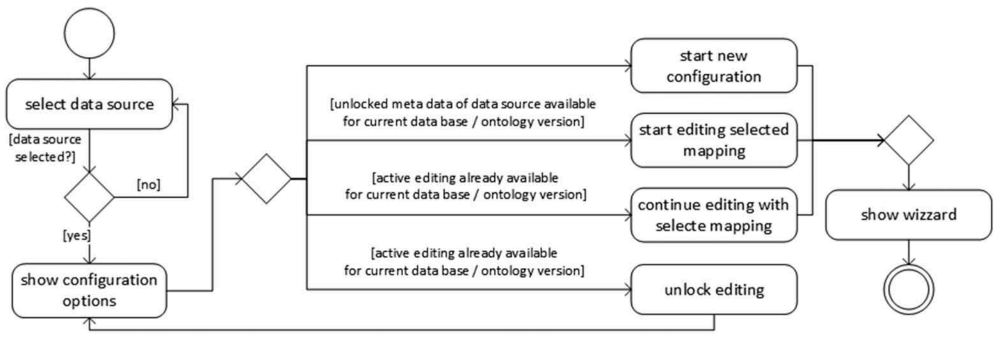

select data source
[data source selected?]
[no]
[yes]
show configuration options
[unlocked meta data of data source available for current data base / ontology version]
[active editing already available for current data base / ontology version]
[active editing already available for current data base / ontology version]
start new configuration
start editing selected mapping
continue editing with selecte mapping
unlock editing
show wizzard

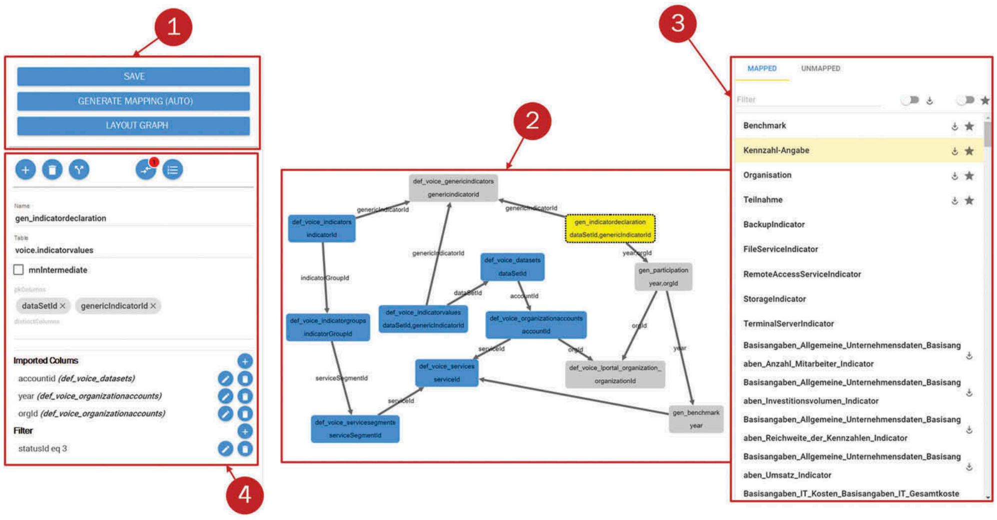
Figure 7. Workflow of the administration wizard.

- A new mapping can be started by ‘Start Editing’, or an active mapping can by modified by ‘Start Editing With Active Mapping’. In both of these cases, the active mappings between ontology concepts and generators are overwritten by a new configuration.
- If not already finished and stored, an existing mapping configuration can be edited and locked or unlocked to prevent data loss.

Once the configuration of the mapping is finished, the user is forwarded to the actual mapping web interface (see Figure 8. This interface can basically be divided into four sections.

- The first section (1) contains all of the actions that are available, such as saving the manually generated mappings; replotting the graph, which is shown in (2); and starting the semi-automatic mapping recommender (see Section 3.2).
- The second section (2) shows the graph and all connections of the generators for the previously selected data source.
- The third section (3) shows all concepts within the ontology that can be mapped to generators.
- The fourth section (4) provides the details for a selected entity (concept, connection or generator) and configuration options to implement the mapping.

The mapping of a selected entity can be displayed and configured using the linkage button (highlighted by a red ‘one’ in Figure 8. The number represents how many mappings already exist for this selected entity. If a generator and one or more concepts are selected in combination, the number indicates all mappings that exist for the selected pairings. Because various possibilities


1
2
3
4
SAVE
GENERATE MAPPING (AUTO)
LAYOUT GRAPH
MAPPED
UNMAPPED
Filter
def_voice_genericindicators genericindicatorid
def_voice_indicators indicatorId
gen_indicatordeclaration dataSetId,genericIndicatorId
def_voice_datasets dataSetId
gen_participation year,orgId
def_voice_indicatorgroups indicatorGroupId
def_voice_indicatorvalues dataSetId,genericIndicatorId
def_voice_organizationaccounts accountId
def_voice_services serviceId
def_voice_lportal_organization_ organizationId
def_voice_servicesegments serviceSegmentId
gen_benchmark year
genericIndicatorId
indicatorGroupId
dataSetId
accountId
year,orgId
orgId
year
serviceId
serviceSegmentId

Name
gen_indicatordeclaration
Table
voice.indicatorvalues
☐ mnIntermediate
pkColumns
dataSetId × genericIndicatorId ×
distinctColumns

Imported Colums
accountid *(def_voice_datasets)*
year *(def_voice_organizationaccounts)*
orgId *(def_voice_organizationaccounts)*
Filter
statusId eq 3

Benchmark
Kennzahl-Angabe
Organisation
Teilnahme
BackupIndicator
FileServiceIndicator
RemoteAccessServiceIndicator
StorageIndicator
TerminalServerIndicator
Basisangaben_Allgemeine_Unternehmensdaten_Basisangaben_Anzahl_Mitarbeiter_Indicator
Basisangaben_Allgemeine_Unternehmensdaten_Basisangaben_Investitionsvolumen_Indicator
Basisangaben_Allgemeine_Unternehmensdaten_Basisangaben_Reichweite_der_Kennzahlen_Indicator
Basisangaben_Allgemeine_Unternehmensdaten_Basisangaben_Umsatz_Indicator
Basisangaben_IT_Kosten_Basisangaben_IT_Gesamtkoste

Figure 8. Client-side administration wizard for the configuration of mappings.

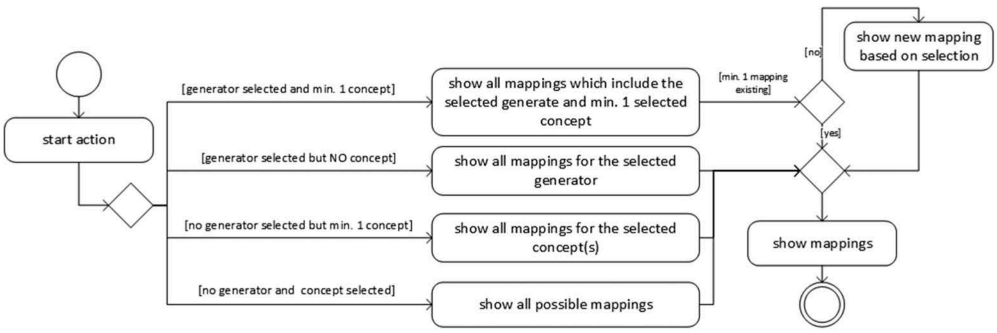

start action
[generator selected and min. 1 concept]
[generator selected but NO concept]
[no generator selected but min. 1 concept]
[no generator and concept selected]
show all mappings which include the selected generate and min. 1 selected concept
show all mappings for the selected generator
show all mappings for the selected concept(s)
show all possible mappings
[min. 1 mapping existing]
[no]
[yes]
show new mapping based on selection
show mappings

Figure 9. Mapping options based on different pairing possibilities.

exist for mapping configurations depending on the selected concepts or generators, Figure 9 shows the different mapping options based on different pairing possibilities.

After an entity is mapped manually or as a result from the mapping recommender, Figure 10 shows the user interface for a detailed overview on the mapping parameters. In this assignment interface for each mapping, the header (A) and the detailed mapping configuration (B) for this entity are shown. In this example, the header consists of the generator name and its mapped ontology concept. In the scope of this header interface, it is also possible to show/hide the details for the mapping; to copy the current mapping, which is use full if only 'Required Attributes' differ for a selected entity; and to mark this mapping for deletion. The deleting process is performed during the save operation of the entire mapping process. Within the detailed view, attributes are separated according to their allocation. On the left side, the generator is shown together with its 'Required Attributes'. On the right side of the detailed view, all mapped concepts are shown, together with their associated properties. The red overlay (1) indicates a previously performed deletion operation on this generator. Note that for all 'Required Attributes', only one value can be specified, whereas for the 'DataTypeProperties' (2), columns of the linked data sources can be specified (using,"#{..}" notation) as well as a free text. For 'ObjectProperties' (3), only the specification of corresponding generators is possible. Note that although it might be possible that a very

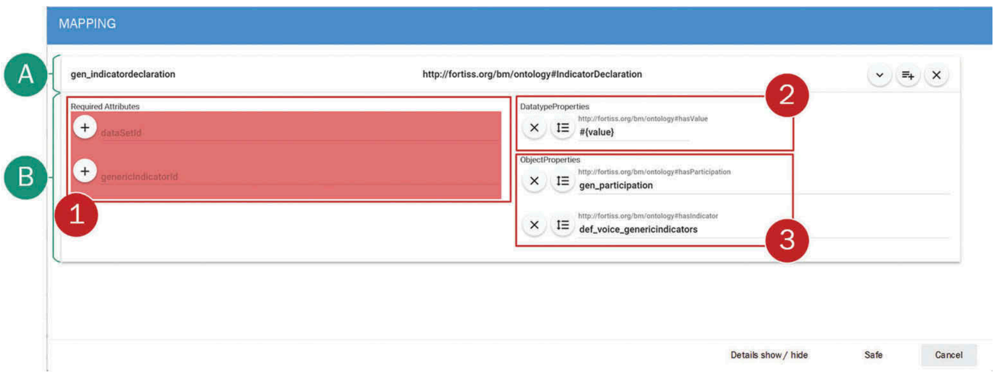

MAPPING
A
gen_indicatordeclaration
http://fortiss.org/bm/ontology#IndicatorDeclaration
B
Required Attributes
dataSetId
genericIndicatorId
1
DatatypeProperties
http://fortiss.org/bm/ontology#hasValue
#{value}
2
ObjectProperties
http://fortiss.org/bm/ontology#hasParticipation
gen_participation
http://fortiss.org/bm/ontology#hasIndicator
def_voice_genericindicators
3
Details show / hide
Safe
Cancel

Figure 10. Admin: interface for the mapping configuration of entities.

large number of nearly similar mappings need to be configured for a concept, it is possible to copy 'Data- and ObjectProperties' to reduce the configuration effort.

5. Conclusion and future work

Because there are numerous challenges related to data integration in the domain of ITBM and the related field of ITSM, this paper introduced an architecture for the (semi-)automatic and ontology-based integration of data from distributed data sources. To the best of our knowledge, the proposed system architecture and software prototype constitute the first approach to bridge the gap between a systematic characterization of IT services and their data-based valuation based on an ontology. Moreover, because the mapping of databases to ontology concepts is a very complex and time-consuming task, a semi-automatic mapping recommender was developed to support the user in this process. This recommender semi-automatically identifies similarities of possible mapping candidates and visualizes them in a graph to reduce the complexity of the mapping process for the system administrators. On the user side, the complexity for the use of such a system could also be reduced as it provides an easy way of to access data by using NLP techniques to translate natural language questions into SPARQL queries. This translation process is also implemented in a transparent manner by showing the generated SPARQL query and by visualizing the resulting search graph.

The proposed web-based system architecture for data integration allows numerous external data sources to be linked through the use of the domain ontology, which is a flexible way to link data sources without knowing the structures of already attached data sources. The separation of structural information provided by the ontology on the one hand and the data sources on the other hand addresses the need for flexibility in the case that the linkage must adapt to changes on both sides. In this way, already existing data sets from various data sources, such as MySQL databases, could be interlinked in terms of their semantic equivalence. At present, all non-administrator users are allowed to access all attached data sources. By using this client-/server-side implementation, based on web technologies, a more fine-grained access control could be implemented in the future. This would address possible security needs that could occur if the system is used beyond company boundaries. Moreover, it is conceivable that restrictions for the use of specific data sources and specific data points within a single data source could also be implemented to ensure that the attached data sources are only allowed to be used within a special context (benchmark) or by special users/organizations.

The ITBM ontology was developed on a large collection of ITBM documents and data set and covers various types of IT benchmarks and (IT) service descriptions from numerous organizations. Thus the developed ontology covers all aspects relevant for using it as universal link for the integration of different types of external benchmarking data. Because the quality of an ontology, in terms of its expressiveness and consistency, is highly dependent on domain knowledge, a broad range of different data are needed as a basis for the development process. Thus, the analysis of such an enormous amount of data, is generally extremely time consuming. This issue in the ontology construction process was already addressed by Pfaff and Krcmar (2015) using NLP techniques to populate the domain ontology and in this paper re-used to identify similar indicators in data sets across different IT benchmarks. In addition, the use of NLP also grounds the development process of an ontology and reduces the variations of an ontology that may occur if it is constructed manually by different domain experts. However, since an ontology is generally discontinuously changing over time, a periodic consistency check of the ontology and the linked data sources was also implemented. In the future, this already implemented consistency check could be developed further to automatically recognize changes upon their occurrence. Additionally, the mapping process for the ontology could also be extended to support and automatically resolve relations across different indicators that characterize the same concept. For now, the structural description of a benchmark within the ontology is limited to a hierarchical structure; this limitation could also be addressed in future research to enable the modeling of more complex coherence. Developing the capability of (semi-)automatic linkage with additional ontologies will be the next step in this research for the purpose of propagating a uniform description of domain knowledge in ITBM.

Note

1 https://github.com/Simmetrics/simmetrics.

Acknowledgments

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Institute of General Medical Sciences of the United States National Institutes of Health [GM10331601].

ORCID

Matthias Pfaff http://orcid.org/0000-0002-6231-9020
Helmut Krcmar http://orcid.org/0000-0002-2754-8493

References

Aldin, L., and S. Cesare. 2011. “A Literature Review on Business Process Modelling.” *Enterprise Information Systems* 5 (3): 359–383. doi:10.1080/17517575.2011.557443.

Asiaee, A. H., T. Minning, P. Doshi, and R. L. Tarleton. 2015. “A Framework for Ontologybased Question Answering with Application to Parasite Immunology.” *Journal of Biomedical Semantics* 6: 31. doi:10.1186/s13326-015-0029-x.

Cai, H., C. Xie, L. Jiang, L. Fang, and C. Huang. 2016. "An Ontology-Based Semantic Configuration Approach to Constructing Data as a Service for Enterprises." *Enterprise Information Systems* 10 (3): 325–348. doi:10.1080/17517575.2015.1070916.

Calvanese, D., B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, and Ó. Corcho. 2016. "Ontop: Answering SPARQL Queries over Relational Databases." *Semantic Web* 8 (3): 471–487. doi:10.3233/SW-160217.

Calvanese, D., G. De Giacomo, and M. Lenzerini. 2001. "Ontology of Integration and Integration of Ontologies." *Description Logics* 49: 10–19.

Camp, R. 1989. *Benchmarking: The Search for Industry Best Practices that Lead to Superior Performance*. Collection hommes et techniques. Milwaukee, Wis.: Quality Press.

Carpinetti, L., and T. Oiko. 2008. "Development and Application of a Benchmarking Information System in Clusters of Smes." *Benchmarking: An International Journal* 15 (3): 292–306. doi:10.1108/14635770810876601.

Council, E. 2016. "FIBO. The Financial Industry Business Ontology." Accessed -February 2016. http://www.edmcouncil.org/financialbusiness/

Daraio, C., M. Lenzerini, C. Leporelli, P. Naggar, A. Bonaccorsi, and A. Bartolucci. 2016. "The Advantages of an Ontology-Based Data Management Approach: Openness, Interoperability and Data Quality - The Advantages of an Ontology-Based Data Management Approach." *Scientometrics* 108 (1): 441–455. doi:10.1007/s11192-016-1913-6.

Dattakumar, R., and R. Jagadeesh. 2003. "A Review of Literature on Benchmarking." *Benchmarking: An International Journal* 10 (3): 176–209. doi:10.1108/14635770310477744.

Edmonds, J., and R. M. Karp. 1972. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems." *Journal of the ACM* 19 (2, April): 248–264. doi:10.1145/321694.321699.

Erling, O., and I. Mikhailov. 2010. "Virtuoso: RDF Support in a Native RDBMS." In *Semantic Web Information Management*, edited by R. De Virgilio, F. Giunchiglia, and L. Tanca, 501–519. Berlin Heidelberg: Springer.

Fielding, R. T., and R. N. Taylor. 2000. "Principled Design of the Modern Web Architecture." In *Proceedings of the 22nd International Conference on Software Engineering, ICSE '00*, 407–416. New York, NY: ACM. doi:10.1145/337180.337228

Freitas, J. M., A. Correia, and F. B. E Abreu. 2008. "An Ontology for IT Services." In *Proceedings of the 13th Conference on Software Engineering and Databases (JISBD)*, edited by Ana Moreira, María José Suárez Cabal, Claudio de la Riva and Javier Tuya, 367–372.

Gacenga, F., A. Cater-Steel, W. Tan, and M. Toleman. 2011. "IT Service Management: Towards a Contingency Theory of Performance Measurement." In *International Conference on Information Systems*, edited by Dennis F. Galletta and Ting-Peng Liang, 1–18.

Gangemi, A. 2010. "The DOLCE+Dns Ultralite Ontology." Accessed -June 2016. http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite

Garcia-Crespo, A., B. Ruiz-Mezcua, J. Lopez-Cuadrado, and I. Gonzalez-Carrasco. 2011. "Semantic Model for Knowledge Representation in E-Business." *Knowledge-Based Systems* 24 (2): 282–296. doi:10.1016/j.knosys.2010.09.006.

Google. 2016. "Angularjs by Google, HTML Enhanced for Web Apps!" Accessed November 2016. https://angularjs.org

Gordijn, J., and H. Akkermans. 2001. "Designing and Evaluating E-Business Models." *IEEE Intelligent Systems* 16 (4, July): 11–17. doi:10.1109/5254.941353.

Guo, Q., and M. Zhang. 2009. "Question Answering Based on Pervasive Agent Ontology and Semantic Web." *Knowledge-Based Systems* 22 (6): 443–448. doi:10.1016/j.knosys.2009.06.003.

Hachicha, M., M. Fahad, N. Moalla, and Y. Ouzrout. 2016. "Performance Assessment Architecture for Collaborative Business Processes in BPM-SOA-Based Environment." *Data & Knowledge Engineering* 105 (C, September): 73–89. doi:10.1016/j.datak.2015.12.002.

Hepp, M. 2008. "Goodrelations: An Ontology for Describing Products and Services Offers on the Web." In *Knowledge Engineering: Practice and Patterns: 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, edited by Aldo Gangemi, and Jérôme Euzenat, 329–346. Berlin, Heidelberg: Springer.

Horkoff, J., A. Borgida, J. Mylopoulos, D. Barone, L. Jiang, E. Yu, and D. Amyot. 2012. "Making Data Meaningful: The Business Intelligence Model and Its Formal Semantics in Description Logics." In *On the Move to Meaningful Internet Systems: OTM 2012, Volume 7566 of Lecture Notes in Computer Science*, edited by R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, and I. Cruz, 700–717. Berlin Heidelberg: Springer.

Jaccard, P. 1901. "E´ Tude Comparative De La Distribution Florale Dans Une Portion Des Alpes Et Des Jura." *Bulletin Del La Soci´Et´E Vaudoise Des Sciences Naturelles* 37: 547–579.

Jung, J. J., C. Badica, N. T. Nguyen, and R. Slowinski. 2015. "Semantic Interoperability for Automated Enterprises." *Enterprise Information Systems* 9 (3): 300–302. doi:10.1080/17517575.2014.985614.

Khondoker, M. R., and P. Mueller. 2010. "Comparing Ontology Development Tools Based on an Online Survey." In *Proceedings of the World Congress on Engineering*. London, UK: World Congress on Engineering 2010 (WCE 2010).

Kuhn, H. W., and B. Yaw. 1955. "The Hungarian Method for the Assignment Problem." *Naval Research Logistics Quarterly* 2: 83–97. doi:10.1002/(ISSN)1931-9193.

Kütz, M. 2006. *IT-Steuerung Mit Kennzahlensystemen*. Heidelberg: dpunkt.verlag.

Lasierra, N., F. Roldán, A. Alesanco, and J. García. 2014. "Towards Improving Usage and Management of Supplies in Healthcare." *Expert Systems with Applications* 41 (14): 6261–6273. doi:10.1016/j.eswa.2014.04.023.

Lehmann, J., R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, et al. 2015. "Dbpedia–A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia." *Semantic Web* 6 (2): 167–195.

Levenshtein, V. I. 1966. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals." *Soviet Physics Doklady* 10 (8): 707–710.

Mate, S., F. Kopcke, D. Toddenroth, M. Martin, H.-U. Prokosch, T. Burkle, and T. Ganslandt. 2015. "Ontology-Based Data Integration between Clinical and Research Systems." *Plos One* 10 (1): e0116656. doi:10.1371/journal.pone.0116656.

McGuinness, D. L., F. Van Harmelen. 2004. "OWL Web Ontology Language Overview." Accessed -July 2016. http://www.w3.org/TR/owl-features/

Michel, F., J. Montagnat, and C. Faron-Zucker. 2014. "A survey of RDB to RDF translation approaches and tools." Research Report, I3S.

Motik, B., P. F. Patel-Schneider, and B. Parsia. 2012. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation 11 December 2012*". Accessed July 2016. https://www.w3.org/TR/owl2-syntax/. .

Müller, O., T. Schmiedel, E. Gorbacheva, and J. Vom Brocke. 2016. "Towards a Typology of Business Process Management Professionals: Identifying Patterns of Competences through Latent Semantic Analysis." *Enterprise Information Systems* 10 (1, January): 50–80. doi:10.1080/17517575.2014.923514.

Nalepa, G. J. 2010, April. "Collective Knowledge Engineering with Semantic Wikis." *Journal of Universal Computer Science* 16 (7): 1006–1023.

Nissen, V., M. Petsch, D. Jung, and C.-P. Praeg. 2014. *Empfehlungen Fu¨R Eine Generelle IT Service-Katalog-Struktur*, Book section 8, 133–154. Fachmedien Wiesbaden: Springer.

Office, C. 2011. *Itil Service Operation: 2011*. Norwich: The Stationery Office.

Ong, E., Z. Xiang, B. Zhao, Y. Liu, Y. Lin, J. Zheng, C. Mungall, M. Courtot, A. Ruttenberg, and Y. He. 2017. "Ontobee: A Linked Ontology Data Server to Support Ontology Term Dereferencing, Linkage, Query and Integration." *Nucleic Acids Research* 45 (D1): D347–D352. doi:10.1093/nar/gkw918.

OpenLink. 2015. "Virtuoso Universal Server." Accessed July 2016. http://virtuoso.openlinksw.com

Osterwald, A., Y. Pigneur, and C. L. Tucci. 2005. "Clarifying Business Models: Origins, Present, and Future of the Concept." *Communications of the Association for Information Systems* 16: 1–25.

Paredes-Valverde, M. A., M. Rodríguez-García, A. Ruiz-Martínez, R. Valencia-García, and G. Alor-Hernández. 2015. "ONLI: An Ontology-Based System for Querying Dbpedia Using Natural Language Paradigm." *Expert Systems with Applications* 42 (12): 5163–5176. doi:10.1016/j.eswa.2015.02.034.

Pellet. 2015. "OWL 2 Reasoner for Java." Accessed August 2016. http://clarkparsia.com/pellet/

Peters, G. 1994. "Benchmarking Customer Service." In *Financial Times Management Series,*Financial Times/Pitman Publishing. London: McGraw-Hill.

Peterson, D., S. Gao, A. Malhotra, C. M. Sperberg-McQueen, and H. S. Thompson. 2012. "W3C XML Schema Definition Language XSD 1.1 Part 2: Datatypes." Accessed July 2016. https://www.w3.org/TR/xmlschema11-2/

Pfaff, M., and H. Krcmar. 2014. "Semantic Integration of Semi-Structured Distributed Data in the Domain of IT Benchmarking." In *16th International Conference on Enterprise Information Systems (ICEIS)*, 320–324. doi:10.5220/0004969303200324

Pfaff, M., and H. Krcmar. 2015. "Natural Language Processing Techniques for Document Classification in IT Benchmarking - Automated Identification of Domain Specific Terms." In *17th International Conference on Enterprise Information Systems (ICEIS)*, 360–366. doi:10.5220/0005462303600366

Pfaff, M., S. Neubig, and H. Krcmar. 2017. "Ontology for Semantic Data Integration in the Domain of IT Benchmarking." *Journal on Data Semantics*. under Review.

Play. 2016. "The High Velocity Web Framework for Java and Scala." Accessed November 2016. https://playframework.com/

Protégé. 2014. "Stanford University." Accessed June 2016. http://protege.stanford.edu/

Riempp, G., B. Müller, and F. Ahlemann. 2008. "Towards a Framework to Structure and Assess Strategic IT/IS Management." In *Proceedings of the 16th European Conference on Information Systems*, edited by Golden, W., Acton, T., Conboy, K., van der Heijden, H., and Tuunainen, V.K, 2484–2495.

Rodríguez-García, M. Á., R. Valencia-García, F. García-Sánchez, and J. J. Samper-Zapater. 2014. "Ontology-Based Annotation and Retrieval of Services in the Cloud." *Knowledge-Based Systems* 56: 15–25. doi:10.1016/j.knosys.2013.10.006.

Rudolph, S., and H. Krcmar. 2009. "Maturity Model for IT Service Catalogues an Approach to Assess the Quality of IT Service Documentation." In *Proceedings of the Americas Conference on Information Systems (AMCIS)*, edited by Robert C. Nickerson and Ramesh Sharda, 759–759. Association for Information Systems.

Sahoo, S. S., W. Halb, S. Hellmann, K. Idehen, T. Thibodeau Jr, S. Auer, J. Sequeda, and A. Ezzat. 2009. "A Survey of Current Approaches for Mapping of Relational Databases to RDF." *W3C RDB2RDF Incubator Group Report* 1: 113–130.

Smith, H. A., and J. D. McKeen. 1996. "Measuring IS: How Does Your Organization Rate?" *SIGMIS Database* 27 (1, February): 18–30. doi:10.1145/234611.

Souripriya, D., S. Seema, and R. Cyganiak. 2012. “R2RML: RDB to RDF Mapping Language.” W3C Recommendation, World Wide Web Consortium. http://www.w3.org/TR/r2rml/

Spendolini, M. J. 1992. *The Benchmarking Book*. New York, NY: Amacom.

Suárez-Figueroa, M. C., A. Gómez-Pérez, and M. Fernández-López. 2012. “The Neon Methodology for Ontology Engineering.” In *Ontology Engineering in a Networked World*, 9–34. Berlin Heidelberg: Springer.

Tatu, M., M. Balakrishna, S. Werner, T. Erekhinskaya, and D. Moldovan. 2016. “A Semantic Question Answering Framework for Large Data Sets.” *Open Journal of Semantic Web (OJSW)* 3 (1): 16–31.

Thomas, O., and M. Fellmann. 2009. “Semantic Process Modeling–Design and Implementation of an Ontology-Based Representation of Business Processes.” *Business & Information Systems Engineering* 1 (6): 438–451. doi:10.1007/s12599-009-0078-8.

Valiente, M.-C., E. Garcia-Barriocanal, and M.-A. Sicilia. 2012. “Applying an Ontology Approach to IT Service Management for Business-IT Integration.” *Knowledge-Based Systems* 28: 76–87.

Vom Brocke, J., A. M. Braccini, C. Sonnenberg, and P. Spagnoletti. 2014. “Living IT Infrastructures an Ontology-Based Approach to Aligning IT Infrastructure Capacity and Business Needs.” *International Journal of Accounting Information Systems* 15 (3): 246–274.

W3C. 2008. "SPARQL Query Language for RDF." W3C Recommendation, World Wide Web Consortium. Accessed July 2016. https://www.w3.org/TR/rdf-sparql-query/

Winkler, W. E. 1990. “String Comparator Metrics and Enhanced Decision Rules in the Fellegisunter Model of Record Linkage.” In *Proceedings of the Section on Survey Research Methods*, American Statistical Association, 354–359.

Wollersheim, J., M. Pfaff, and H. Krcmar. 2014. “Information Need in Cloud Service Procurement - an Exploratory Case Study.” In *E-Commerce and Web Technologies - 15th International Conference, EC-Web 2014, Munich, Germany, September 1-4, 2014. Proceedings*, edited by Martin Hepp, and Yigal Hoffner, 26–33.

Zhang, D., D. Hu, and Y. Xu. 2010. “A Framework for Ontology-Based Product Design Knowledge Management.” In *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference*, edited by Maozhen Li, Qilian Liang, Lipo Wang, and Yibin Song, 1751–1755. Vol. 4.

Ziaie, P., M. Ziller, J. Wollersheim, and J. Krcmar. 2012. “Mai. “Introducing a Generic Concept for an Online It-Benchmarking System”.” *International Journal of Computer Information Systems and Industrial Management Applications* 5: 137–150.