A Workbench for Ontology Management and Exploitation

Luis Pedro Estrela Costa

Abstract. Ontologies encode data about any domain of interest in terms of its metadata (classes, relationships between classes and class attributes) and the actual domain "facts" (objects of the domain, as instances of classes). This data can be extracted using query languages and reasoning engines, and consequently, interesting domain information reports can be delivered to users interested in ontologies. The tools available to explore data from ontologies, such as Protégé, work on the assumption that the user has a significant level of knowledge about ontology structures in order to understand the information being presented. This imposes a technical barrier in the access to ontology information for users who have no knowledge of such tools but understand the domains being represented by ontologies. In this work, I propose a simple to use web application that facilitates the access to ontology information for this kind of users, allowing them to create and configure reports to suit their needs in order to explore data in any ontology, without the assumption that they have any knowledge on its primitives and the required tools.

Keywords: Ontology, Risk Management, Reasoning, OWL, Enterprise Architecture, Report

1. Introduction

Ontologies are useful to represent, organize and share information about any domain of interest. The combination of ontology information in different domains can be mashed together to create massive knowledge bases, and this information can then be processed by software, such as reasoning engines, providing us with additional information about domains that humans were unaware of. The tools available to us today to exploit the information in ontologies, such as Protégé, are powerful to do this exploitation, however, they assume that the user has a deep understanding of the structural elements of ontologies, in order to understand the information being presented. This creates a technical barrier to the access of this information for the type of users who have a clear interest in ontologies and the domain they are describing, but lack the know how to use these tools. In an attempt to solve this problem, in this paper, I present a workbench that makes the access to this information easier for these users, allowing them to easily create and configure reports to extract ontology information. By doing this, I also present some of the lessons learned in working with this technology and discuss the current software support to exploit ontology information.
2. Proposed Solution

In order to achieve the objectives outlined, a workbench that allows the creation and management of any type of ontology, as well as the ability to create, configure and execute reports on such ontologies was developed. The workbench uses the SPARQL-DL API\(^1\) to create and execute queries on ontologies, and the HermiT reasoner\(^2\) to perform inference tasks. The workbench should provide a set of services through Application Programming Interface (API) endpoints that should be used by a web application which should be easy to use and understand by users.

Figure 1 presents a blackbox view of the developed solution, with its main use cases.

By developing this workbench, one should expect to gain more insights and understandings over the current software support for querying and inference mechanisms on ontologies, as well as the current advantages and shortcomings of using such technology.

Additionally, for demonstration and validation purposes, the workbench was applied in two specific domains: first, it provides a special report for ontologies specifically created for the risk management domain, which is called the risk assessment matrix report. This is described in section 5.2. Secondly, it provides another special report which is called archimate risk impact analysis for ontologies generated from domain models created with the Archimate domain modeling language\(^3\). This is described in section 5.3.

3. Semantic Technology

Today, the web consists of millions and millions of documents. Browsers render these documents by parsing the HyperText Markup Language (HTML) present in web pages. A web consisting solely of HTML documents represents a static model which is human-friendly but not machine-processable. The Semantic Web\(^1\) movement offers a new approach, where web pages are not just connected by hyperlinks, but by semantic connections. A web page in the semantic web represents domain knowledge that can be shared across applications and organizations.

An ontology, defined by Borst as a “formal specification of a shared conceptualization”,\(^2\), is the core technology that allows the representation of this domain knowledge. Ontologies can be created with a myriad of languages, each of which offering different levels of reasoning support, decidability, and expressibility. The core languages are the Resource Description Framework (RDF)\(^4\), the Resource Description Framework Schema (RDFS)\(^5\) and the Ontology Web

\(^1\) http://www.derivo.de/en/resources/sparql-dl-api/
\(^2\) http://hermit-reasoner.com/
\(^3\) http://www.opengroup.org/subjectareas/enterprise/archimate
\(^4\) https://www.w3.org/RDF/
\(^5\) https://www.w3.org/TR/rdf-schema/
Language (OWL). These languages build on top of each other. OWL adds additional functionality on top RDFS, and RDFS adds additional functionality on top of RDF. OWL is the standard to create ontologies, because it is the most complete ontology language, providing more primitives and expressibility power.

Ontologies alone would not be useful without technology to extract information from them. Ontology query languages, such as SPARQL and SPARQL-DL, allows an SQL-like querying interface that extracts data from ontologies. Another approach to extract ontology information is to use ontology reasoners, such as HermiT and Pellet, to obtain additional data not made explicit by the ontology designer, thus uncovering hidden, but implicit, relationships in the ontology.

4. Risk Management & Enterprise Architecture

Every project within an organization is subjected to risks. Risks affect the path that organizations undergo in order to achieve their objectives. A risk is

---

6 https://www.w3.org/TR/owl2-overview/
7 https://www.w3.org/TR/sparql11-overview/
9 http://www.hermit-reasoner.com/
10 https://www.w3.org/2001/sw/wiki/Pellet
the combination of the likelihood of an event materializing and the impact of that event on the organization, which is a consequence[4].

This risk information is typically structured in risk registers. Since risk registers usually follow the same structure across domains, the metadata of these risk registers can be more formally structured in a domain model, producing a Risk Domain Model. Consequentially, these risk domain models can be represented with ontologies: the metadata of the risk registers is created with ontology classes and relationships between them, and the actual data is represented with ontology individuals.

The workbench developed in this work accepts this kind of ontologies and produces risk assessment matrix reports, which are the typical reports in this domain. This is described in section 5.2.

Enterprise Architecture11 (EA) is defined as the discipline of “identifying and analyzing the execution of change toward desired business vision and outcomes” [5]. In EA, it is typical to use domain modeling languages to represent domain information. Archimate is one such language. Online tools are useful to translate archimate domain models into OWL ontologies. The workbench accepts such ontologies, and produces a specific type of report, as is discussed in section 5.3.

5. Workbench design

The workbench developed consists of two core components: a backend application, written in Java, and a frontend application, written in Javascript. Figure 2 depicts the overall communication between components.

The backend application provides four core services that are used by the frontend application to store and manage ontologies, as well as to create and execute reports on them:

- **Repository Manager**: This service is responsible for handling requests to upload and delete ontologies from a repository. This repository is hosted in a Secure File Transfer Protocol (SFTP) Server. This service is called every time the user navigates to the home page, in order to retrieve all the available ontologies and their associated data.

- **Extractor Manager**: This service has the responsibility of extracting all the considered relevant ontology data for the user. The information that was considered relevant were the namespace, classes, individuals and properties declared and imported by the ontology, as well as the namespaces of imported ontologies. Additionally, the reports available (created by the user) for execution are also retrieved.

- **Reports Manager**: This class is responsible for the creation, management, and execution of reports on ontologies. When executing a report, the associated SPARQL-DL query of that report is executed and then the results from this query are filtered by report restrictions that the user can configure. Report restrictions are only applied on data properties (an example is presented next)

11 http://www.gartner.com/it-glossary/enterprise-architecture-ea/
Archimate Manager: This class is responsible for dealing with requests for the archimate risk impact analysis reports (described in section 5.3) on ontologies that were automatically generated from Archimate domain models, using online converters\(^{12}\).

5.1 Example reports on real world ontology

The user has the ability to execute a predefined report on all ontologies: **Listing report**, which presents a list of all individuals of the ontology, as well as values for data properties of each individual.

Consider the Ontology for Plant Protection (PPOntology)\(^ {13}\). Figure 3 presents an excerpt of the result of the execution of a listing report that extracts the individuals of the class **Disorder** as well as the data property values for individuals of this class.

Additionally, the user can create and configure reports on his own. An example of the result of a report created and configured by the user is presented in Figure 4. This report is extracting individuals of the class **Microorganism** which are related with individuals of the class **Biotic_Disorder** (via the object property **factor-of**). Additionally, a data property restriction was created: from the set of the previous individuals, only list the microorganisms which include the

\(^{12}\) http://www.ifs.tuwien.ac.at/dp/process/projects/archi2OWL.html

\(^{13}\) https://sites.google.com/site/ppontology/
Fig. 3. List Report: Listing of individuals of the class Disorder, and the associated data property values name, zone and region.

string *gamininis* in its name (restriction on the data property *scientific_name*).

![List Report Table]

**Fig. 4.** Example of user-made report: Microorganisms which include the string *gamininis* in the name and that are cause of biotic disorders

5.2 Risk Management Use Case

This report creates the risk assessment matrix report, to depict risk information in terms of consequence impacts and event likelihoods. Typically, these values range between 1 (very low impact/likelihood) and 5 (very high impact/likelihood). Each risk is placed inside a 5x5 matrix cell. Each cell represents the risk level of that risk, i.e., the result of multiplying the risk impact and the likelihood of the event triggering that risk (from 1 to 25).

This report will be available if the ontology respects a specific structure. This structure contains the classes Event, Consequence, and Risk, along with the object properties Risk-hasEvent-Event, Risk-hasConsequence-Consequence and the data properties Risk-hasRiskLevel-Integer, Risk-hasOwner-String and Event-hasLikelihood-Integer, Consequence-hasImpact-Integer.

The execution of this report produces two results. First, a table containing all the risk information is presented, which includes the Risk IDs created auto-
matically by the application, useful to identify risks in the matrix. An excerpt of this table is presented in Figure 5.

![Risk Assessment Matrix Report: Full Risk Information](image)

Second, the actual matrix is presented to the user. Risks with low risk level are placed in the green area of the matrix. Risks with medium risk level are placed in the yellow area of the matrix, and risks with high risk level are placed in the red area of the matrix.

For the risk information in Figure 5, the workbench would produce the risk assessment matrix presented in Figure 6.

![Risk Assessment Matrix Report: Matrix](image)

### 5.3 Enterprise Architecture Use Case

If the user uploads ontologies generated from Archimate domain models, a special report will be available, the **archimate risk impact analysis report**, briefly described next.

The ontologies used in the risk management use case were related with these ontologies, with the objective of providing an answer to the question: "Assuming that there is some risk affecting some Asset in the organization, which organizational entity or entities of the enterprise architecture will also be affected?". The relationship between these two ontologies exists in the assumption that an individual of the class Asset of a risk domain ontology matches individuals of some class in the archimate ontology.

From this report results a set of individuals that the workbench, using the HermiT reasoner, inferred to be affected if there was a hypothetical risk in some
asset specified by the user, from the set of assets in the risk domain ontology. This asset is then matched automatically with one or more individuals of any class in the archimate ontology. The user also needs to select an archimate ontology class. The workbench will try to infer if any individual of that class will be affected if the hypothetical risk occurs in the matched individual of the archimate ontology.

The workbench will return the individuals of that class and the explanations that justify that entailment. These explanations are simply "paths" that the HermiT reasoner followed between individuals (starting from the matched individual in the archimate ontology) related via object properties. These explanations are automatically produced by the Hermit reasoner API.

Since this is a specific use case, the workbench is expecting archimate ontologies that import a specific upper ontology, the DIO ontology. The reasoner is configured to follow connections between individuals only through a specific set of object properties defined by the DIO ontology: dependsUp, dependsDown, assignment, and association. There are several object properties that extend each of these properties. The dependsDown relate individuals in a "downwards" manner in the archimate domain model (top layers to bottom layers) and the dependsUp relates the individuals in the inverse way. The object properties assignment, and association relate two individuals directly.

For instance, consider the architectural domain model depicted in Figure 7. This domain model represents the business entities of an online bookstore. When this domain model is converted into an OWL ontology, the user can execute this report to answer the question “What individuals of the entity BusinessProcess would be affected if there is a risk in the entity EmailServer?”. The entity EmailServer would need to correspond to (or match with) an individual of the class Asset in some risk ontology uploaded to the repository. This report would return as result the individuals Contact_Submission, Provide_Answer, and Purchase_Book. The path that the reasoner takes to reach this conclusion is highlighted with red arrows in Figure 7. The object property usedBy is an inverse property of the object property uses, which is a subobject property of dependsDown and realizes is a subobject property of dependsUp.

6. Conclusions & Future Work

In this work, I presented a workbench that allows the management, creation, and execution of user-created reports on OWL ontologies. This workbench was created to provide an easier access to ontology information to users who lack the required technical knowledge to use the current software tools to exploit ontology data. The workbench makes no assumptions regarding the level of understanding that the user has on ontological structural elements.

The workbench was applied in two domains: In the risk management domain, to create risk assessment matrix reports and in enterprise architecture, where ontologies created from archimate domain models are used in conjunction with ontologies for the risk domain use case to provide a report that identifies what

---

14 http://timbus.teco.edu/ontologies/DIO.owl
are the impacts of having risks in one specific domain entity of the archimate model in the remaining entities of that domain.

By using software tools such as the SPARQL-DL API and the HermiT reasoner, I concluded that there is still a lack of support for SPARQL-DL querying on OWL DL ontologies. Even though there are several SPARQL query editors for ontologies created solely with RDF, such as the native SPARQL engine available in Protege, Apache Jena Fuseki SPARQL Server 15, Virtuoso SPARQL query editor 16, support for SPARQL querying on OWL ontologies is still lacking. Alternatives such as Protege plugins 17 or frameworks such as Snap-SPARQL [6] are interesting options, but they are lacking active development and lagging behind both on the OWLAPI 18 version and the JDK version.

There is still room for improvement in the SPARQL-DL specification because it does not yet support operators, provided by SPARQL, that would be useful to create the reports engineered in the workbench, such as OPTIONAL, FILTER or UNION. The FILTER 19 operator would be particularly useful in the designing of the workbench: because this operator is not provided natively by SPARQL-DL,

---

15 https://jena.apache.org/documentation/serving_data/
16 http://publications.europa.eu/webapi/rdf/sparql
17 http://protegewiki.stanford.edu/wiki/OWL2Query
18 http://owlapi.sourceforge.net/
19 https://jena.apache.org/tutorials/sparql_filters.html
the solution for providing report restrictions had to be implemented as a layer "on top" of the SPARQL-DL API.

Overall, the benefit of using semantic technology to create reports lies in using reasoning engines, which saves the programmer the cumbersome task of creating endless iterations and recursions in order to find implicit relationships between classes and individuals in the ontology. I also concluded, from the enterprise architecture use case, that software support for reasoning engines is still maturing, as is made clear by the fact the reasoner used in the workbench, HermiT, is not up to date to the latest version of OWLAPI, but its use and configuration is straightforward. Alternatives such as the Pellet reasoner\footnote{https://github.com/stardog-union/pellet} are appealing, because they support more recent versions of the OWLAPI (which means better reasoning support), but the installation, configuration, and usability of this reasoner is not as trivial as HermiT.

As future work, the workbench can be enhanced with other reasoners, and then investigate if these alternatives can be used to create different kinds of useful reports. The workbench can also be enhanced by providing reports that use all the operators provided by the SPARQL-DL API. Another useful feature to support the goal of easily provide ontology information for the type of users targeted by this workbench, the frontend application could provide assistance to the user while configuring the reports, explaining what each button and action is doing.

In the risk management use case, in order to eliminate the restriction of the specific structure expected from ontologies, a class names matching mechanism could be employed (the user would match classes expected by the system with the classes of the new ontology). Making the EA use case work with any ontology is not easy: either the system only accepts ontologies that import the DIO ontology (thus, always considering the same set of object properties for the reasoner path), or the set of object properties that the reasoner must consider is provided by the user apriori.

References