Ontology Metadata for Ontology Reuse

Elena Simperl*

Karlsruhe Institute of Technology,
Institute AIFB,
Englerstr. 11, Building 11.40,
76128 Karlsruhe, Germany
E-mail: elena.simperl@kit.edu
*Corresponding author

Cristina Sarasua

Vicomtech-IK4 - Visual Interaction Communication Technologies,
Paseo Mikeletegi 57, Parque Tecnológico Miramon,
20009 San Sebastián, Spain
E-mail: csarasua@vicomtech.org

Rachanee Ungrangsi

Shinawatra University,
99 Moo 10 Bangtoey Samkok Pathumthani,
12160, Thailand
E-mail: rachanee@siu.ac.th

Tobias Bürger

Capgemini,
Carl-Wery-Str. 42,
81739 Munich, Germany
E-mail: tobias@tobiasbuerger.com

Abstract: Most ontologies are poorly documented, thus not being optimally accessible to potential users and ontology reuse services. A first step towards ontology documentation is the development of an explicit schema for the systematic and comprehensive description of ontologies. To provide an informed background for such a schema, we performed a survey of the most recent ontology engineering technology, and the types of information for describing ontologies. We introduce an ontology metadata schema, which as part of the OMV standard, was evaluated through professional reviews. A second step is the provision of automatic techniques to acquire such documentation. For this purpose we have developed the OMEGA (Ontology MEtadata GenerAtion) algorithm, which automatically generates metadata about arbitrary ontologies on the Web and is available as Web application and REST Web service. It was evaluated with very promising results, providing thus a core building block for the realization of fully-fledged ontology location services.

Keywords: ontology, metadata, reuse, schema, havervesting, extraction, generation, documentation, repositories, search engine

Reference to this paper should be made as follows: Elena Simperl, Cristina Sarasua, Rachanee Ungrangsi. and Tobias Bürger. (2011) ‘Ontology Metadata for Ontology Reuse’, *Int. J. of Metadata, Semantics and Ontologies*, Vol. 1, Nos. 1/1, pp.11–111.

1 Introduction

As semantic technologies mature, an increasing number of private and public sector organizations are developing ontologies to capture their knowledge about a domain of interest in a machine-understandable way. These ontologies are expected to act as communication interfaces between humans and machines. They are commonly agreed, shared conceptualizations through which humans express the intended meaning of the types of things they talk about in a specific domain; at the same time they form the basis for the resolution of interoperability issues between IT systems. Consequently, it is understandable that the question of how to ensure the reusability of ontologies beyond the context in which they have been initially developed has often been put in relation to the impact semantic technologies might potentially have in various business sectors. This question can be addressed from multiple viewpoints, ranging from knowledge representation languages over ontology reuse technology to community and organizational processes (Simperl 2009, Simperl & Bürger 2010, Pinto & Martins 2000).

While finding a solution to the problematic trade-off between application setting-motivated usability and global reusability is often considered an art more than a science, the latter can be significantly improved by resorting to several design principles, which have proved their relevance across several fields in computer science confronted with similar challenges: modularity and decomposition along abstraction and functionality levels, standards compliance, careful documentation of the development process, as well as up-to-date, meaningfully organized artifact repositories. Recent research achievements in the Semantic Web area offer a feasible basis for many of these principles to be put into practice. The usage of knowledge representation standards in conjunction with the ubiquitous, URI-based access of Semantic Web resources are a fair starting point to facilitate the cross-application usage of ontologies (Bojars et al. 2008, Gracia et al. 2010, Heath & Bizer 2011). Semantic Web search engines and ontology repositories play an equally important role in this context, as they provide potential ontology users with the query, search and browse support they need to identify ontologies potentially relevant to their own needs (Fikes & Farquhar 1999, Ding & Fensel 2001, Hartmann et al. 2009, Baclawski & Schneider 2009, Noy et al. 2008). The work presented in this paper provides the theoretical and technical foundations to allow ontology engineering environments, including the two types of systems just mentioned, to solve one of their currently most stringent limitation: the lack of useful metadata describing ontologies; this metadata is an important ingredient for the implementation of useful retrieval and navigation functionality.

Most ontologies, independently of their provenance and location, are poorly documented, while additional descriptive information is spread across the Web in various forms, thus not being optimally accessible to potential users and ontology reuse services. Available services in this area offer a basic set of search, browse and navigation features to the ontological resources administrated. However, their functionality is negatively influenced by the absence of reuse-relevant information about ontologies in a machine-readable, structured form. A first step towards the alleviation of this situation is the development of an explicit schema for the systematic and comprehensive description of ontologies so as to enable more effective access, management and usage of them across the Web. To provide an informed background for such a schema, we survey the most recent ontology engineering technology, including ontology repositories, Semantic Web search engines, and ontology documentation tools, and the types of information they provide to describe ontologies. We then introduce an ontology metadata schema, which is not only aligned with the related formats implicitly or explicitly used by state-of-the-art technology and tools, but also combines and extends the best elements of these formats in order to maximize the expressivity and usefulness of the unifying result. As part of the OMV (Ontology Metadata Vocabulary) standard, this metadata schema was evaluated through professional reviews, and is supported by ontology location services such as Oyster[1] and Watson.[2] A second step towards more informatively documented ontologies is the development and usage of automatic techniques to acquire such documentation in order to scale to the number of ontologies constantly being built in various sectors. For this purpose we have developed the OMEGA (Ontology MEtadata GenerAtion) algorithm, which is available as Web application and REST Web service. The algorithm follows a heuristic approach to automatically generate a wide range of metadata information about arbitrary ontologies available on the Web or indexed by Semantic Web search engines. It was evaluated in terms of coverage, precision, recall, and overall user-perceived quality in several studies with very promising results. The evaluation revealed that automatic ontology metadata generation is feasible to a much larger extent than it is done in state-of-the-art Semantic Web search engines if the wealth of knowledge resources freely available on the Web is taken into account in the process, making OMEGA a core building block for the realization of fully-fledged ontology location services.

In a nutshell, the contribution of this work is twofold. First, we provide a comprehensive overview of the types of information used in state-of-the-art ontology engineering environments to describe ontologies, identifying potential limitations at the level of ontology metadata in current ontology repositories, Semantic Web search engines, and ontology documentation tools. Second, we investigate how ontology metadata could be generated algorithmically in order to prevent the metadata acquisition bottleneck with which ontology location tools are typically confronted.

The remainder of this article is organized as follows: First we review current state-of-the-art ontology engineering technology with respect to ontology metadata they support and the means to acquire it (Sections 2 to 4). In particular, we introduce OMV (Ontology Metadata Vocabulary), which is used by our approach for automatic metadata acquisition. In Sections 5 and 6 we present the OMEGA algorithm and its implementation. Its evaluation and main findings are the presented in Section 7. We conclude with a short discussion of the limitations of the current approach, and sketch the planned future work in Section 8.

2 Metadata schemas used to document ontologies

As a first step of the ontology reuse process the ontology engineering team typically resorts to conventional or Semantic Web-oriented search engines, and browses repositories of ontological resources in order to build a list of potential reuse candidates. The participants specify a minimal set of desired features in terms of machine-understandable queries, and review the results delivered by ontology location services. Important for the execution of this step are core ontology features such as the domain of the ontology, its availability and licensing conditions, its development status and the release date (Paslaru-Bontas et al. 2005, Simperl 2009).

In a second step the reuse candidates are subject to an in-depth evaluation. A detailed documentation of the ontologies and the associated engineering process provides real added value for the effective operation of this primarily human-driven, knowledge-intensive endeavor. Various aspects of an ontology can be taken into account, also depending on the evaluation methodology that is applied. Obviously relevant is the content of the ontology, in terms of the domain modeled, the key concepts, and the number of ontological primitives of a specific type (Gómez-Pérez 2001). Equally important is the language in which the ontology is formalized and its semantics (Paslaru-Bontas et al. 2005, Simperl & Mochol 2007, Lozano-Tello & Gómez-Pérez 2004). Last, but not least it is recommended to consider the application scenario and the original purpose of the ontology, as using an ontology in a different context typically requires some degree of customization.

The selected ontologies are then adjusted to the current application scenario. This includes aspects as diverse as segmentation, translation to different knowledge representation languages, alignment, merging, or integration. As suggested for instance in (Mochol 2009), this last step could capitalize on ontology metadata to make an informed decision about the ontology alignment tools that are likely to deliver optimal results. Graph-oriented metrics such as the total size of the ontology, the number of classes, properties, axioms and instances are correlated with the performance and scalability of the alignment task. Further on, some tools are explicitly targeted at a specific set of ontological primitives, at a particular input format, or at natural language.

To summarize, the quality of the documentation about an ontology influences its reusability in several ways. On the one hand, it supports ontology engineers in their attempt to understand what a particular ontology is about, and whether the knowledge it captures is relevant wrt. their requirements. On the other hand, the availability of structured documentation forms the basis for the implementation of rich ontology location tools, including ontology search engines and ontology repositories. In the following sections we will survey various metadata schemas, vocabularies and ontologies, as well as the degree of documentation offered by different ontology environements and tools.

2.1 General-purpose metadata schemas

Metadata is often defined as data about data. A metadata record generally consists of a set of attributes or elements describing a resource. Metadata can be created manually or automatically. It can be integrated into the actual resource or stored separately in a metadata database. Several metadata schemas and vocabularies, specifying the set of elements allowed, as well as their value ranges and intended meaning, have been proposed. A good overview of metadata standards for describing electronic content is provided in (National Information Standards Organization 2004). From the impressive number of existing metadata formats we mention some of the most important standards capturing information about textual documents, including those ones relevant for the Semantic Web.

The Dublin Core **DC** metadata standard is a simple, yet effective element set which can be used for a wide range of networked resources.[3] It differentiates among two levels of detail: simple (with fifteen elements) and qualified, including an additional element, as well as a group of element refinements (or qualifiers) that adjust the semantics of the elements for resource discovery purposes. The metadata that can be declared with DC refers mainly to the creation and publication of a resource. Thus, the elements included in the DC Metadata Element Set provide a mechanism to indicate the entity or entities that created, contributed to or published the resource, the date and the location of the resource, some features for describing its content (e.g., title, subject, description, format, language, type and related resources), and the way in which the resource can be further used (e.g., identifier, rights and coverage). Table 1 shows the set of fifteen elements of DC. These elements can be applied using different formats, due to the different serializations of the standard that have been implemented, which include XML, RDFS and OWL. When the RDFS and OWL versions are employed, the resources can be described by means of RDF descriptions. In such cases, the DC elements are exploited as `rdf:Properties` and `owl:DatatypeProperties` respectively, whilst the values of these properties can be strings.

At a different end, the Creative Commons schema **CC** captures information about copyright licenses.[4] Defined in RDF, the CC schema establishes six different levels of protection depending on whether the author wants to avoid commercial use, the jurisdiction of the license and whether modifications are permitted or forbidden. There are six CC licenses that regulate the usage of resources. The *Attribution* license allows other people to copy, distribute, display and derive works from the original one, on condition that the author is recognized as she requested. The *Attribution-NoDerivs* license restricts the previous license stating that works can not be derived from the original one. The *Attribution-NonCommercialNoDerivs* license adds a new restriction stating that the resource can only be used for non-commercial purposes. The *Attribution-NonCommercial*

Table 1 Relevant metadata included in DC

Name	Description	Type
contributor	contributor of the document	string
coverage	spatial or temporal topic	string
creator	creator of the document	string
date	temporal information about the document	string
description	description of the document	string
format	physical information of the document	string
identifier	unambiguous reference	string
language	language used	string
publisher	publisher of the document	string
relation	related resource	string
rights	rights attached to the document	string
source	origin of the document	string
subject	topic of the document	string
title	name of the document	string
type	category of the document	string

license refers to the possibility of using the resource for non-commercial purposes only, without imposing any restrictions about derivative works. The *Attribution-Non Commercial-ShareAlike* license has the same meaning as the previous one, but includes a new restriction about the fact that the resource can be distributed only under the same license. Finally, the *Attribution-ShareAlike* license specifies that the work can be used if the author is correctly recognized and the distribution has to be performed under the same license.

Table 2 shows the properties defined in CC to waive these restrictions on online resources such as ontologies. The particular license attached to the resource can be identified using the `license` property, while its legal code can be pointed out by means of the `legalCode` property. Moreover, metadata about specific requirements, permissions and prohibitions related to the resource can also be indicated. The author and the real location of the resource is usually stated through the `attributionName` and `attributionURL` properties.

Table 2 Relevant metadata included in CC

Name	Description	Type
license	license attached to work	license
morePermissions	URL of other related permissions	string
attributionName	name of the creator	string
attributionURL	URL of the work	string
permits	permission over a license	permission
requires	requirement attached to license	requirement
prohibits	prohibition established by license	prohibition
jurisdiction	legal jurisdiction of license	jurisdiction
legalcode	URL of legal text	license
deprecatedOn	date of deprecation	license

2.2 *Ontology-specific metadata schemas*

Arguably, these standards, which have emerged outside of the ontology engineering community, are only to a certain extent suited to describe ontologies, as they do not take into account the very nature of ontologies as knowledge models (Hartmann et al. 2006, KnowledgeWeb European Project 2004). One of the first metadata schema developed with this motivation in mind was the Reference Ontology (Arpirea et al. 2000). It is a domain ontology that gathers, describes and has links to existing ontologies, using a common logical organization. A second one, which is based on the analysis in (KnowledgeWeb European Project 2004), is the Ontology Metadata Vocabulary (OMV), which is used as underlying metadata schema of the OMEGA algorithm.

The Ontology Metadata Vocabulary OMV (Hartmann et al. 2006, 2005) was developed and used over the last five years in a series of European research projects such as the EU IST thematic network of excellence Knowledge Web and the EU ICT integrated project NeOn.[5] It reflects the shared understanding and empirical findings of the ontology engineering community with respect to the metadata elements which should be essentially supported by ontology reuse technology.

Implemented as an OWL ontology, the latest version of the OMV vocabulary (2.4.1) contains a total of 16 classes, 62 properties and 50 individuals. Following a modular design, its developers distinguish between the OMV core ontology where common ontology metadata is included (OMV Core), and extensions accommodating specific aspects of the ontology engineering process or of a domain of interest (OMV Extensions) (Hartmann et al. 2006). OMV Core models metadata about the technological means used to develop the ontology, the context in which it was created, as well as its implementation code. For instance, OMV allows ontology engineers to provide information about the name of the ontology language, the acronym of the syntax (e.g., RDF/XML) and the ontology engineering methodology used (e.g., DILIGENT). Moreover, OMV can also show the name of the person or the organization that developed, supported or contributed to the development of the ontology. The license model attached to the ontology, as well as the official name, the locator and the creation and modification date can also be expressed using OMV. Table 3 and Figure 1 provide an overview of the most relevant ontology metadata included in OMV.

OMV Core is supported by several ontology reuse tools, such as Oyster, Cupboard, Watson and the Open Ontology Repository (OOR) Initiative,[6] whose main goal is to promote the global use and sharing of ontologies. Others, such as BioPortal, are considering to adopt the model as well. These examples show that OMV is considered as a de facto standard in the ontology engineering community; with more and more ontology repositories and search engines implementing OMV these tools can share and exchange ontology documentation, thus facilitating the reuse of ontologies independently of a particular system or platform. However, to capitalize on this advantageous state of the affairs, one still has to face the challenge of metadata acquisition.

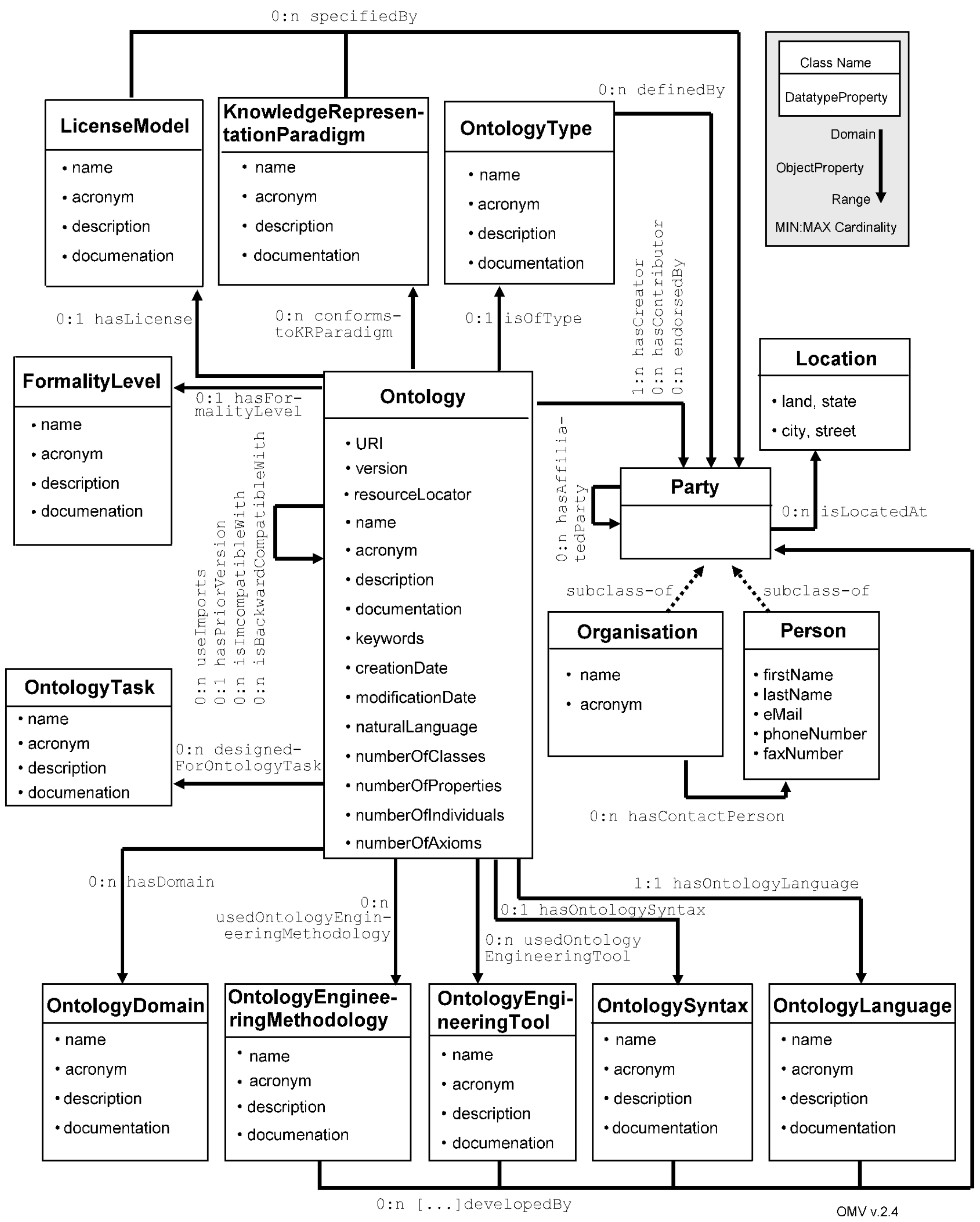


Figure 1 Excerpt of the OMV schema

Table 3 Relevant metadata included in OMV

Name	Description	Type
URI	URI of the ontology	xsd:string
version	version of the ontology	xsd:string
resourceLocator	URL where the ontology is available	xsd:string
name	name of the ontology	xsd:string
acronym	short name under which the ontology is known	xsd:string
description	a free text description about the ontology	xsd:string
creationDate	date when the ontology was created	xsd:date
modificationDate	date when the ontology was last modified	xsd:date
naturalLanguage	language used for labeling and commenting the ontology (e.g., English, German)	xsd:string
hasCreator	person or organization who created the ontology	Party
usedOntologyEngineeringTool	tool used for ontology development (e.g., NeOn toolkit, Protégé)	OntologyEngineeringTool
usedOntology EngineeringMethodology	methodology followed to develop the ontology (e.g., NeOn methodology, DILIGENT)	OntologyEngineeringMethodology
conformsToKnowledge RepresentationParadigm	KR paradigm used in the ontology (e.g., DL, F-Logic)	KnowledgeRepresentationParadigm
endorsedBy	organizations or individuals who support the ontology	Party
hasDomain	domain represented by the ontology (e.g., Tourism, E-Commerce)	OntologyDomain
isOfType	nature of the ontology according to established classifications (e.g., domain ontology, upper level ontology)	OntologyType
designedForOntologyTask	purpose for which the ontology was built (e.g., data annotation, data integration, semantic search)	OntologyTask
hasFormalityLevel	level of formality of the ontology	FormalityLevel
knownUsage	application where the ontology is being used	xsd:string
hasOntologyLanguage	ontology language used (e.g., RDFS, OWL-Lite, OWL-DL)	xsd:string
hasOntologySyntax	syntax used in the ontology (e.g., OWL-XML, RDF/XML)	xsd:string
hasLicense	license attached to the ontology (e.g., GPL, CCL)	LicenseModel
hasContributor	someone who contributed to the development of the ontology	Party
useImports	imported ontologies	Ontology
hasPriorVersion	other versions of the ontology	Ontology
isBackwardCompatibleWith	another ontology with which the ontology is backward compatible	Ontology
isIncompatibleWith	another ontology with which the ontology is not backward compatible	Ontology

2.3 Web 2.0 and Web of Data metadata schemas

The Simple Knowledge Organisation System **SKOS** proposes a vocabulary for describing knowledge organization systems, including ontologies but also taxonomies and thesauri, through RDF on the Semantic Web.[7] The basic element of the SKOS vocabulary, which has been developed in OWL Full, is the `skos:Concept`. Relations between concepts can be asserted to express broader, narrower or related links between them. Moreover, concepts can be labeled in three different ways (i.e.,, with the `prefLabel` property, the `altLabel` property or the `hiddenLabel` property). Concepts can be further described using the properties that the SKOS data model contains for notes (i.e.,, `note`, `scopeNote`, `historyNote`, `changeNote`, `editorialNote`), as well as with definitions and examples.

The Statistical Core Vocabulary **SCOVO** (Hausenblas et al. 2009) is a vocabulary to express statistical information on the Web of Data. It is based on SKOS and data is defined by means of datasets, dimensions and items. Given a data source about, let's say, book sales, SCOVO allows users to state things such as the number of books that have been sold in a particular country or in a given time period.

SCOVO, which has been developed in RDFS, is comprised of three classes and five properties. These elements enable developers to declare statistical datasets that describe available data sources. In SCOVO, the properties to be analyzed are declared as `svoco:Dimensions` and the dataset to be generated is instantiated as a `scovo: Dataset`. This `scovo:Dataset` is populated with `scovo:Item`'s, which come from the result of executing SPARQL queries against the existing data source that is being analyzed. SCOVO properties refer to the relation between a `scovo:Item` and a `scovo:Dataset` (`dataset` and `dataset Of`), and to the relation between an `scovo:Item` and a `scovo:Dimension` (`scovo:dimension`). Dimensions may have minimum and maximum values (`scovo:min` and `scovo:max`, respectively).

Complementarily, the Vocabulary of Interlinked Datasets **voiD** (Alexander et al. 2009), focuses on the description of linked datasets so as to provide relevant information for the discovery and usage of such datasets. Not only can voiD specify features about a particular dataset, but it can also relate the dataset to other linked datasets. Specified in RDFS, voiD establishes a set of three classes and thirteen properties. A data set can be documented by means of voiD if it is declared as a `void:Dataset`. Once instantiated as a dataset, its technical features can be pointed out, a subset of the dataset can be identified, and data about its sparql endpoint, the vocabulary that is being used within the dataset, example resources, a data dump, its URI look-up protocol, and a regular expression pattern of URIs. Moreover, the links between the dataset and other datasets can be defined by means of `void:Linkset`'s and the properties attached to them (`target`, `linkPredicate`, `subjectsTarget` and `objectsTarget`). Table 4 summarizes the metadata elements included in voiD.

2.4 Alignment of existing approaches

Even though the aforementioned schemas were designed for different purposes, they have some properties in common.

Table 4 Relevant metadata included in voiD

Name	Description	Type
statItem	item asserting statistical data	scovo:Item
feature	technical features supported	TechnicalFeature
subset	subset of given dataset	Subset
target	target dataset of link set	Dataset
sparqlEndpoint	SPARQL endpoint of dataset	rdfs:Resource
linkPredicate	RDF predicate of particular link set	rdfs:Property
exampleResource	representative resource	rdfs:Resource
vocabulary	RDFS/OWL ontology used	rdfs:Resource
subjectsTarget	source target of link set	Dataset
objectsTarget	sink target of link set	Dataset
dataDump	data dump	rdfs:Resource
uriLookupEndpoint	URI look-up protocol	rdfs:Resource
uriRegexPattern	regular expression pattern of URIs	XML Schema regular expression

Thus, several alignments could be defined between these models. It is worth emphasizing that the features shared by the vocabularies mainly refer to the process of creation and publication. More specifically, the information that these schemas have in common is related to the identification of the ontology, its version and location, the dates of creation and modification, the parties involved in its development and publication, and a brief description and the license attached to it. Table 5 illustrates the identified mappings between OMV and other schemas.

Table 5 Alignment between metadata standards

OMV element	Mapped elements
URI	dc:identifier
version	owl:versionInfo, schemaweb:namespace
resourceLocator	cc:attributionURL, schemaweb:location schemaweb:location
name	dc:title, rdfs:label, schemaweb:name schemaweb:name
description	dc:description, rdfs:comment, schemaweb:description, rdfs:comment, skos:note
creationDate	dc:date
modificationDate	dc:date
naturalLanguage	dc:language
hasContributor	dc:contributor
hasCreator	dc:creator, cc:attributionName
hasDomain	dc:subject, no:topic
isOfType	no:category
hasLicense	dc:rights, cc:license
hasPriorVersion	owl:priorVersion
isBackwardCompatibleWith	owl:backwardsCompatibleWith
isIncompatibleWith	owl:incompatibleWith

3 Ontology metadata in ontology development environments

Several ontology development environments have become available over the last decade, both public, such as Protégé, Apollo, WSMT, NeOn Toolkit, Swoop or IODT [8] and commercial, such as OntoStudio, TopBraid Composer or SemanticWorks.[9]

A survey we conducted revealed that almost all of them fail to provide rich ontology documentation functionality.[10] Some of them only permit free-text annotation of classes and slots. In the case of Apollo (v01a18), this annotation is declared as proprietary XML elements termed `documentation`. Similarly, SemanticWorks (2009) makes use of the RDFS constructs related to labels and comments to perform class, property and ontology annotation. SWOOP's documentation (v2.3-beta4) also includes information about the version of the ontology. Ontology development environments such as OntoStudio (v2.31) and NeOn Toolkit (v1.2.3) offer the possibility to define ontology annotations by means of the `owl:AnnotationProperty` construct. Through this option ontology developers can define arbitrary annotation properties, thus being able to freely choose the set of metadata that can be used to describe the ontology. Nevertheless, the usage of a particular OWL variant introduces some constraints. Thus, although OWL Full does not give any constraint on annotations, OWL DL establishes among other conditions, that annotations can be applied to classes, properties, individuals and the ontology element, as an instance of the `owl:AnnotationProperty`. Besides, both RDFS and OWL predefine several annotation properties that can be used by developers to describe data about the version of an ontology, its compatibility with other versions, general comments and related resources that contain additional information. Furthermore, the ontology metadata that can be declared in TopBraid Composer v3.1.0 is based on both the constructs that RDFS and OWL have for this purpose and the extension that can be accomplished with OWL annotation properties.

Finally, Protégé 4.0 allows the user to introduce ontology metadata using built-in annotations, Dublin Core based annotations and user defined annotations. Built-in annotations involve information related to OWL constructs, including compatibility, versions, deprecation, general comments and related resources. Whereas DC annotations include the Simple Dublin Core Metadata Element Set made up of fifteen elements. User defined annotations can be built using custom annotation URIs. Protégé was one of the first editors to consider ontology metadata, and although software has evolved in the past years, it is even now the most complete and easiest to use ontology editor, at least with regard to ontology metadata. What these tools have in common is that they support a semi-automatic approach to ontology metadata creation by presenting the user with a form that guides the manual input of information about the ontology currently being processed. All the information that is being captured through the forms is automatically being converted to ontological constructs, which are then stored in the original ontology as annotation properties, attached to the `owl:Ontology` construct. Nevertheless the number of supported metadata elements is still limited in all of the previously mentioned editors, whilst there is no guarantee that the user will provide the metadata information associated to the ontology she develops.

There is some data that both Protégé and NeOn Toolkit extract automatically from the ontology code, such as the number of classes, the number of properties, or the number of individuals. However, although this information is shown to the user during ontology development process (in the editor or by means of its OwlDoc plugin), it is not declared as ontology metadata. Table 6 provides an overview of the ontology metadata defined in the ontology tools mentioned above.

4 Ontology metadata in ontology repositories and Semantic Web search engines

Ontology repositories store and manage ontologies for future reuse. Whilst the number of ontologies hosted by such repositories has reached a critical mass in the last years – probably as a consequence of the increasing popularity of semantic technologies – the same positive trend can not be observed for ontology reuse, which is limited to a very restricted set of ontologies which are either straightforward to use (like FOAF) or very general in their content (like upper-level ontologies of the type DOLCE, SUMO and PROTON). Obviously, some might interpret this state of affairs as an indicator of the lack of high-quality ontologies in the space spanned by these two extreme examples. Nevertheless, without a useful documentation the existing ontological resources, already in their tens of thousands according to Watson, hardly stand the chance to be discovered and reliably assessed by potential users.

Descriptive information about ontologies can be entered by ontology developers manually at submission time in most of the repositories we surveyed. This information is, however, frequently missing. This in turn diminishes the quality of the retrieval algorithms and, in the longer run, on the reusability of the administrated ontologies. As a side note, a machine-understandable representation of the ontology metadata is often not available, which not only has consequences on the retrieval, but also on the ability to share this metadata across repositories and other ontology engineering tools.

In their first generation, ontology repositories or directories were plain Web sites managing a collection of links to (hundreds of) ontologies hosted on external servers. The DAML ontology library (no longer updated), Vocab.org (no longer updated), Protege OWL Library and SchemaWeb are some often cited examples of such repositories.[11] These systems offer very basic metadata, which is stored separately from the original ontologies; only SchemaWeb serializes ontology annotations in both human and machine-readable formats (based on an ontology for the latter). Metadata elements supported are the name of the ontology, its location, its namespace, a short textual description of its content, as well as details about submission and authorship. In addition, the DAML ontology library shows the number of classes, properties and instances, as well as their local names. Regarding submission, only SchemaWeb provides a means to upload ontologies and the associated metadata.

With semantic technologies gaining momentum, ontology repositories have significantly improved. Now at their second generation, they incorporate new features, and enhanced browsing and searching functionality. OntoSelect (Buitelaar et al. 2004) is an ontology library where users can search and browse ontologies submitted by authors. For each registered ontology the system provides further information about format, domain, language, number of classes, properties, labels, imported ontologies and location. Some ontology repositories are optimized to the characteristics of specific contexts; for instance, TONES is the ontology repository of the project with the same name, while BioPortal covers life sciences ontologies and ONKI Finnish ontologies.[12] TONES allows the user to filter the result set using automatically extracted metrics related to the syntax of the ontology, its classes, properties and individuals. BioPortal offers a collaborative, sophisticated environment for uploading, browsing and searching ontologies. It contains metadata – accessible through REST services – about format, name, location, version and contact data, together with links to related publications, projects and reviews. The technology underlying BioPortal has been developed as a domain-independent framework for ontology repositories that can be applied in arbitrary contexts, similar to Oyster (Palma et al. 2006), Cupboard (d'Aquin & Lewen 2009) and OLS2OWL (Garca-Castro et al. 2009). Oyster establishes a P2P framework where users can manage, search and share ontology metadata. The information that is made available through Oyster is compatible to OMV, and includes the name of the ontology, its location, type, language, status, syntax, license, authorship and related keywords. Cupboard, which builds upon Watson and Oyster, offers an online space where registered users can publish their own ontologies and related information. Conversely, ontology practitioners can search, evaluate and comment other ontologies. The ontology metadata provided by Cupboard includes a link, details about the development of the ontology, its syntax, type, number of classes, number of properties, number of individuals and number of axioms. The Protégé 4.0 OLS2OWL plug-in was conceived as a support tool for the knowledge acquisition phase of an ontology engineering process – it provides ontology developers access to online and local ontologies through a search interface, compares ontologies, and builds a customized ontology repository within the Protégé editor. Searches can be performed using concept labels as a starting point, whereas the results include available metadata and ontology statistics such as the number of classes, properties and instances.

Complementary to ontology repositories we can find several ontology search engines on the Web, among them Swoogle (Ding et al. 2004), Watson (d'Aquin et al. 2007), OntoSearch (Zhang et al. 2004), or Sindice (Tummarello et al. 2007). Ontology search engines are fundamental to ontology reuse, as they discover ontologies available on the Web (or within an intranet) and provide a controlled interface for potential users to access them. In this context, ontology metadata information is required for ontology users to be able to take an informed decision upon the appropriateness of an ontology delivered by the search engine. Acknowledging this, Semantic Web search engines such as Swoogle and Watson provide a small number of ontology metadata elements,

Table 6 Relevant ontology metadata in ontology development environments

Name	Type	Protégé	WSMT	NeOn	Swoop	IDODT	TopbraidComposer	SemanticWorks	OntoStudio
name or title	string	x	x	x	x	x	x	x	
identifier	rdf:Resource	x	x	x			x		x
namespaces	string	x		x			x	x	x
label	rdfs:Literal	x		x			x	x	
comment	rdfs:Literal	x		x	x		x		x
seeAlso	rdfs:Resource	x		x			x		x
isDefinedBy	rdfs:Resource	x		x			x		x
versionInfo	string	x		x	x		x		x
priorVersion	owl:Ontology	x		x			x		x
backwardsCompatibleWith	owl:Ontology	x		x			x		x
incompatible With	owl:Ontology	x		x			x		x
deprecated	xsd:boolean	x		x			x		x
imports	rdfs:Resource	x		x		x	x		x
contributor	string	x		x			x		x
coverage	string	x		x			x		x
creator	string	x		x			x		x
date	string	x		x			x		x
description	string	x		x			x		x
format	string	x		x			x		x
identifier	string	x		x			x		x
language	string	x		x	x		x		x
publisher	string	x		x			x		x
relation	string	x		x			x		x
rights	string	x		x			x		x
source	string	x		x			x		x
subject	string	x		x			x		x
type	string	x		x			x		x

which can be extracted automatically. OntoSearch provides even fewer details about the ontologies in the search results: the URI of the ontology, its format and the size of the file containing the ontology. Both Swoogle and Watson are able to display the URI of the ontology, its format, the size of the file, some comments or descriptions, number of classes, properties, and individuals. Apart from this, Swoogle can extract data about encoding, ontology ratio, number of statements, as well as some details related to the ontology discovery process. Watson on the contrary, is designed to show user reviews, locations and imports. Both manage their ontology annotations in HTML and RDF, according to a built-in schema and Dublin Core (for Swoogle), and OMV (for Watson). Sindice, is a powerful search engine for the Web of Data. In terms of metadata, it supports elements such as format, number of triples, size of the ontology file and release date.

To conclude 7 gives an overview of the ontology metadata defined in the ontology repositories and search engines mentioned above.

5 OMEGA: A Tool for the Automatic Metadata Acquisition

In order to overcome the ontology metadata acquisition bottleneck, we conceived the OMEGA algorithm.

The OMEGA approach consists of three steps: i) *Metadata Harvesting*, ii) *Metadata Extraction*, and iii) *Metadata Reuse*. In essence, the algorithm works as follows: First, an ontology is submitted to the algorithm as an input. The algorithm then harvests ontology metadata elements from the embedded meta tags within the ontology document itself and file headers by using parsers. In the second step, several OMV elements are derived with the help of special heuristics and publicly available reference knowledge sources: the Web searched using the Google engine through the Google APIs,[13] the Open Directory Project (aka the DMOZ directory),[14] or Wikipedia Categorical Index.[15] Extracting metadata automatically from the content of the ontology itself with the help of an inference engine such as Pellet[16] is also a part of this second step. The third step serves as the last resort for the remaining unknown OMV entries. Here we make use of existing metadata information provided by online ontology repositories such as the DAML Library and SchemaWeb and Semantic Web search engines such as Swoogle and Watson. As a result, the algorithm is not only able to provide nearly complete metadata entries for online ontologies, but also to generate a minimal set of meta-information for ontologies that are temporarily inaccessible or not machine-processable.

5.1 Step 1: Metadata Harvesting

In this step the algorithm parses ontology files to extract attributes or contents from specified tags, comments and file headers. In an ontology, users can record metadata as Dublin Core entries or as language-specific constructs. Examples in this category include: `dc:date`, `dc:description`, `daml:versionInfo`, `owl:imports`, and `rdfs:comments`. Some ontology editing tools by default produce metadata at the time an ontology file is created or updated without human intervention. In addition, meta-information embedded in the file header such as last modification date, character encoding and file size can be extracted automatically.

5.2 Step 2: Metadata Extraction

The metadata extraction step differentiates between **Ontology Mining** and **Ontology Classification**, as elaborated in the following.

5.2.1 Ontology Mining

At this stage the algorithm mines the content of the ontology to obtain metadata, particularly the OMV elements: `hasFormalityLevel`, `numberOfClasses`, `numberOfProperties`, `numberOfIndividuals`, `numberOfAxioms`, `containsTBox`, `containsRBox`, and `containsABox`.

The values for the element `hasFormalityLevel` are aligned with the ontological continuum proposed in (McGuinness 2002). This basically divides ontologies (or ontology-like structures) in *informal* and *formal* as follows:

- **Informal models** are ordered in ascending order of their formality degree as *catalogues*, *glossaries*, *thesauri* and *informal taxonomies*:
  - A *catalog* is an ontology that has only class definitions without labels or comments.
  - A *glossary* is an ontology that has class definitions with labels or comments.
  - A *thesaurus* is an ontology that has classes and class equivalent assertions.
  - An *informal taxonomy* is an ontology comprising classes and strict, yet informal hierarchical subclass relationships between classes.
- **Formal models** are ordered in the same manner: starting with *formal taxonomies*, which precisely define the meaning of the specialization/generalization relationship, more formal models are derived by incrementally adding *formal instances*, *properties/frames*, *value restrictions*, *general logical constraints*, *disjointness*, *formal meronimy* etc.

To determine the appropriate formality level the algorithm checks which types of constructs (classes, labels, comments, relationships of various kinds, value restrictions etc) are supported by the ontology and returns the most complex formality level satisfying the tests. In addition, OMEGA adds *Data entries* as an additional definition for ontology documents having no schema elements but containing solely instance data.

The remaining six metadata elements listed above can be computed by checking whether the corresponding type(s) of ontological constructs are available in the

Table 7 Relevant ontology metadata in ontology repositories and search engines

Name	**Type**	**Tools**
name or title	string	SchemaWeb, Vocab, Protégé, OntoSelect, TONES, BioPortal, Oyster, Cupboard, OLS2OWL
identifier	rdf:Resource	SchemaWeb, Watson, Vocab, Oyster, Cupboard, OLS2OWL
location	string	DAML, SchemaWeb, Swoogle, Watson, Vocab, Protégé, OntoSelect, TONES, BioPortal, ONKI, Oyster, Cupboard, OLS2OWL, OntoSearch, Sindice
description	string	DAML, SchemaWeb, Vocab, Protégé, BioPortal, ONKI, Oyster, Cupboard
language	string	OntoSelect, Oyster
ontology language	string	DAML, Swoogle, Watson, Vocab, Protégé, OntoSelect, TONES, BioPortal, Oyster, Cupboard
syntax	string	Swoogle, Vocab, OntoSelect, TONES, BioPortal, Cupboard
DL expressiveness	string	Watson, TONES, Cupboard
size	string	Swoogle, Watson, BioPortal
embedded ontologies	boolean	Swoogle, OntoSelect, Cupboard
included ontologies	rdf:Resource	Swoogle, Oyster, Cupboard
encoding	string	Swoogle
submission date	date	DAML, SchemaWeb
submission or discovery time	time	Swoogle
creation date	date	Vocab, BioPortal
last modification date	date	Swoogle, Oyster
version	string	Swoogle, Vocab, BioPortal,Oyster
keyword	string	DAML, Oyster
type	string	ONKI, Oyster
category	string	DAML, BioPortal
group	string	BioPortal
funder	string	DAML
owner or author	string	DAML, SchemaWeb, Vocab, BioPortal, ONKI, Oyster
contact email	string	DAML, SchemaWeb, BioPortal
submitting organization	string	DAML, Vocab
website	string	SchemaWeb, BioPortal
review	string	Cupboard
publications page	string	BioPortal
projects	string	BioPortal
license	string	Vocab, Oyster
number of triples	integer	Swoogle, TONES, Cupboard
number of classes	integer	Swoogle, OntoSelect, TONES, BioPortal, Oyster, Cupboard
number of properties	integer	Swoogle, OntoSelect, TONES, BioPortal, Oyster, Cupboard
number of instances	integer	Swoogle, TONES, BioPortal, Cupboard
number of axioms	integer	Swoogle, TONES, Oyster
number of subclasses	integer	TONES
number of equivalent classes	integer	TONES
number of disjoint classes	integer	TONES
number of data properties	integer	TONES
number of object properties	integer	TONES
number of transitive properties	integer	TONES
number of labels	integer	OntoSelect
ontology ratio	integer	Swoogle
notes	string	DAML, Swoogle, BioPortal

ontology and eventually computing the number of occurrences of each construct type. Pellet is employed as an external component to check the consistency of an ontology, which is specified in the OMV element (`isConsistentAccordingToReasoner`) and to derive language-related metadata information including the knowledge representation language, the level of expressivity and the knowledge representation paradigm.

5.2.2 Ontology Classification

In this step we extract the values of the following OMV elements: `isOfType`, `Keywords`, `KeyClasses`, `hasDomain`, and `naturalLanguage`. To do so we resort to heuristic techniques utilizing publicly available knowledge resources such as the DMOZ Directory Project, the Wikipedia Categorical Index, or the vast amounts of online knowledge accessible through a search engine interface, as explained in the remainder of this section.

With respect to the generality of the domain modeled by an ontology our algorithm differentiates between three ontology types (OMV element `isOfType`)

- *Upper-level ontologies:* describe general-purpose concepts and their properties. From a metadata generation perspective, an upper-level ontology is characterized by the usage of a notably abstract vocabulary. Examples of typical terms in upper-level ontologies typically include “object”, “unit”, “relation”, “abstract”, “quantity”, “process”, “thing” etc.
- *Core ontologies:* provide very basic concepts and properties of a knowledge domain. For example, the Semantic Network in UMLS contains general medical concepts such as disease, finding, syndrome, thus being a core medical ontology.[17]
- *Domain ontologies:* are used to model specific domains, and are more detailed than ontologies in the previous category.

To determine the type of an ontology automatically we use the hierarchical structure of categories provided by DMOZ as a reference.[18] At a basic level, the algorithm performs syntactic concept-matching between an ontology and the DMOZ directory. The matching can be either *exact* (the ontology class name is exactly the same as the DMOZ category name) or *fuzzy* (the ontology class name is a substring of the DMOZ category name). Then it determines the level of the directory in which the majority of concepts in the ontology can be feasibly matched. If all concepts are not found in the DMOZ directory, then that ontology is assumed to be an upper-level ontology. If most concepts are found the the root level or on the first level of the DMOZ directory (which contains categories such as *Science*, *Medicine* or *Computers*), then the ontology is assumed to be a core ontology. Otherwise, the algorithm considers the ontology processed to be by default a domain ontology.

Identifying the keywords of an ontology (corresponding to the OMV element `Keywords`) is a by-product of the ontology type classification process. OMEGA selects concepts that match DMOZ categories as keyword candidates. A keyword candidate is considered as a *meaningful* keyword if it is located at a high level in the corresponding sub-hierarchy in the DMOZ directory and if it is rarely used to describe other categories.

The weight of a keyword candidate k is calculated by

$$w(k) = \frac{1}{l(k)f(k)}$$

where $l(k)$ is the highest category level at which k appears and $f(k)$ is the frequency with which k appears as category name in the DMOZ directory. Currently, the system selects the ten highest-weighted keyword candidates as keywords.

Further on, in OMEGA, a class in an ontology can become a key class (OMV element `KeyClasses`) if its characteristics meet one of the following criteria:

- *By structure*: Taking into account the graph structure underlying each ontology this criterion applies the *degree centrality measure* (Wasserman & Faust 1994) to identify the importance of a class in the ontology based on the number of links connecting it with the rest of the ontology graph.
- *By readability*: Intuitively, when an ontology engineer builds an ontology, he will arguably ensure that important concepts are well described by defining several properties for both human (i.e., `rdfs:label` and `rdfs:comment`) and machine (i.e., `rdfs:property`, `owl(daml):ObjectProperty`, and `owl(daml):` `DatatypeProperty`) processing. As a result, a class that has a high number of properties is considered a key class.
- *By instance*: The way data is placed within an ontology can be another indicator for the importance of a class or concept. OMEGA adds classes with a large number of instances to the key class list.

The domain of interest of an ontology (OMV element `hasDomain`) is defined in terms of the top categories of the DMOZ directory. OMEGA makes use of the extracted keywords and key classes in the earlier step instead of using all classes to reduce computation complexity.[19] In general, an ontology keyword may appear in several categories in the DMOZ directory. For example, there are ten categories that contain the sub-category “animal”. To deal with this ambiguity issue OMEGA uses a maximum likelihood approach. For each keyword, the algorithm starts from computing full category paths of the sub-categories which match the respective class name. Then it counts the number of distinct top categories and computes the likelihood of each related domain. The top DMOZ categories that have the maximum likelihood based on all keywords previously computed are the possible domains of the ontology, and thus the values of the `hasDomain` metadata element.

The natural language of the human-readable content of an ontology (OMV `natural`

`Language`) is determined using a Google-supported heuristics. Given the keywords and key classes of the ontology OMEGA queries the Web using those terms and counts the Internet top-level domains. By exploiting the inherent language dependency of the top-level domains (such as those related to a particular country or to types of organizations), the algorithm maps each top-level domain to a natural language, thus determining the natural language primarily used in the analyzed ontology. National domains are mapped to the official language of the country, while general ones (.edu, .org, .com etc.) are ordered for historical reasons to English.

5.3 Step 3: Metadata Reuse

In this step OMEGA aims to determine the value of the remaining OMV elements by checking existing metadata information provided by ontology repositories such as the DAML Library and SchemaWeb, and Semantic Web search engines such as Swoogle and Watson. As a result, the algorithm is also able to provide some meta-information of ontologies that are temporarily inaccessible or not machine-parsable. For example, suppose that the ontology `http://www.kanzaki.com/ns/music` is temporarily unavailable, OMEGA is unable to generate the ontology metadata in Step 1 and Step 2. However, the metadata information of this ontology exists in Swoogle and the SchemaWeb Directory. Therefore, OMEGA can provide up to 13 out of 17 OMV metadata elements for the ontology `http://www.kanzaki.com/ns/music`.

OMEGA's metadata reuse algorithm details as follows: First, based on the pre-created mapping table between the OMV elements and elements in the metadata sources (i.e., ontology repositories and search engines as shown in Table 8), the algorithm looks for the entry that matches the missing OMV elements. If there are two or more matches, the algorithm will select the best answer according to the predefined ordered metadata sources. In the current implementation, the ordered list of metadata sources are Watson, Swoogle, DAML Library and SchemaWeb, respectively, which is sorted by the number of provided metadata elements and the system's last updated date. The algorithm then retrieves the metadata values by means of Web APIs or Web scraping techniques. Continued from the previous example, the value of `ResourceLocator` for Ontology `http://www.kanzaki.com/ns/music` can be founded in both Swoogle (see Figure 3) and SchemaWeb Directory (see Figure 2). The system will choose to retrieve the value from Swoogle as the answerhas been retrieved via Swoogle APIs. More metadata sources from Table 8 will be included into the OMEGA appliation as part of our future work.

To conclude this section, the OMV elements that are generated in each step of the algorithm are summarized in Table 9.

Table 9 OMV Elements Generated by OMEGA

OMV elements	Step 1	Step 2	Step3
	Harvest	Extract	Reuse
General Information			
Name	x	-	-
Description	x	-	x
Notes	x	-	x
Keywords	-	x	x
KeyClasses	-	x	-
CreationDate	x	-	-
ModificationDate	x	-	x
Provenance Information			
hasCreator	x	-	x
usedOntologyEngineeringTool	x	-	-
usedKnowledgeRepresentationParadigm	-	x	-
Applicability Information			
hasDomain	-	x	-
isOfType	-	x	-
naturalLanguage	-	x	-
hasFormalityLevel	-	x	-
Format Information			
hasOntologyLanguage	x	x	x
hasOntologySyntax	x	-	x
isConsistent	-	x	-
Expressiveness	-	x	-
Availability Information			
ResourceLocator	-	x	x
Version	x	-	-
hasLicense	x	-	-
Relationship Information			
useImports	x	-	-
hasPriorVersion	x	-	-
isBackwardCompatibleWith	x	-	-
isIncompatibleWith	x	-	-
Statistics Information			
containsTBox	-	x	-
containsRBox	-	x	-
containsABox	-	x	-
numberOfClasses	-	x	x
numberOfProperties	-	x	x
numberOfIndividuals	-	x	x
numberOfAxioms	-	x	x
Additional Information			
hasCharEncoding	x	-	x
fileSize	x	-	x

Table 8 OMV Elements Acquired from Repositories and Search Engines

OMV elements	DAML Library	SchemaWeb Directory	Swoogle	Watson	Vocab.org	Protégé OWL Library	OntoSelect	TONES	BioPortal	ONKI Finnish	Oyster	Cupboard	OLS2OWL	OntoSearch	Sindice
Description	x	x	-	x	x	x	-	-	x	x	x	x	-	-	-
Notes	x	-	-	-	-	-	-	-	x	-	-	-	-	-	-
Keywords	x	-	-	-	-	-	-	-	-	-	x	-	-	-	-
ModificationDate	-	-	x	-	-	-	-	-	-	-	x	-	-	-	-
hasCreator	x	x	-	-	x	-	-	-	x	x	x	-	-	-	-
hasOntologyLanguage	x	-	x	x	x	x	x	x	x	-	x	x	-	-	-
hasOntologySyntax	-	-	x	-	x	-	x	-	x	-	x	x	-	-	-
isConsistent	-	-	-	x	-	x	-	-	-	-	-	-	-	-	-
Expressiveness	-	-	-	x	-	-	-	x	x	-	-	x	-	-	-
ResourceLocator	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
useImports	-	-	x	x	-	-	-	-	-	-	x	-	-	-	-
numberOfClasses	-	-	x	x	-	-	x	x	x	-	x	x	-	-	-
numberOfProperties	-	-	x	x	-	-	x	x	x	-	x	x	-	-	-
numberOfIndividuals	-	-	x	x	-	-	-	x	x	-	-	x	-	-	-
numberOfAxioms	-	-	x	x	-	-	-	x	-	-	x	-	-	-	-
hasCharEncoding (additional)	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-
fileSize (additional)	-	-	x	x	-	-	-	-	-	-	-	-	-	-	-

6 Implementation

OMEGA is implemented in Java as Web application and REST Web service. The system is accessible at `http://research.siu.ac.th:8080/omega` which is hosted on an Apache Tomcat Web server. A MySQL database is used to persistently store and maintain ontology metadata entries automatically generated.[20] Ontologies available online, notably their URLs, are obtained from ontology repositories that are currently active. In the current release of OMEGA we use SchemaWeb, the DAML ontology library, Watson, and Swoogle, which result in more than 26,000 ontology metadata entries in the repository.

The OMEGA prototype provides the following functionality:

- It generates OMV-based ontology metadata information of arbitrary online ontologies in an automatic fashion.
- It allows browsing, searching and retrieving ontology metadata information by human users via a Web interface.
- It allows querying the ontology metadata information by software agents via a REST Web service.

Figure 4 shows the current interfaces of the OMEGA prototype. Figure 4(a) illustrates the result page when a user submits an ontology URL to generate an OMV entry or queries about the properties of a particular ontology in the OMEGA metadata repository. The result can also be saved as an OWL file by clicking on the provided link. Figure 4(b) shows the browsing page which enables users to explore the OMEGA repository by a number of ontology classifications. In addition, OMEGA enables user to locate ontologies that have particular properties via the ontology search page (see Figure 4(c)).

7 Evaluation

The automatic ontology metadata generation algorithm has been evaluated in terms of *coverage*, *precision*, *recall* and *overall quality*. This section explains the evaluation procedures and discusses the results.

7.1 Coverage

This criterion refers to the number of different metadata elements which can be automatically extracted by OMEGA and state-of-the-art ontology reuse technology. Table 10 gives a comparative overview of the number of ontologies and OMV elements that are covered by the current implementation of OMEGA, and other related tools. It clearly shows that OMEGA is able to provide much more ontology metadata information to users than the others.

7.2 Precision and recall

A user experiment has been conducted to evaluate the quality of the metadata extracted in Step 2 in OMEGA in terms of precision and recall. The test collection comprised 100 ontologies randomly selected from the OMEGA repository. The tests involved ten computer science students familiar with semantic technologies. Each of them was asked to tag ten of the given ontologies with OMV elements. We focused on those OMV elements which can not be computed directly from the content of an ontology:

Table 10 Coverage of metadata elements in OMEGA and related tools

		SchemaWeb Directory	DAML Library	Prot*égé*Library	OntoSelect	Watson	Swoogle	OMEGA	Vocab.org	TONES	BioPortal	ONKI Finnish	Oyster	Cupboard	OLS2OWL	OntoSearch	Sindice
No. of Ontologies		246	282	89	1652	25,000+	20,000+	26,000+	19	218	202	88	-	-	-	?	billions
OMV	General(11)	2	4	3	2	2	2	7	1		2	1	3	1			
	Provenance(6)	1	1	2	-	-	-	3									
	Applicability(6)	-	-	-	1	-	-	4		1	1			1			
	Format(4)	-	2	1	1	3	2	4	2	1	2		2	2			
	Availability(3)	1	-	-	-	1	2	3	1	1	1	1	1	1	1	1	1
	Relationship(4)	-	-	-	1	1	1	4					1				
	Statistic(7)	-	-	-	2	4	4	7		4	3		3	3			
	Additional(2)	-	-	-	-	1	2	2	1		1	1	1				

`keywords`, `keyClasses`, `isOfType`, `hasDomain`, and `naturalLanguage`. Contrarily, information such as the number of classes, the knowledge representation language or the level of expressivity can be obtained using core ontology technology (parsers, reasoners).

Metadata automatically generated by OMEGA was compared with the metadata produced by the test participants in terms of precision and recall. Precision measures how many of the automatic classifications are correct, whilst recall means how many of the manually assigned classifications were found in the results of the algorithm. In the experiment, the participants were asked to limit the number of the keywords and key classes to three, and this information was matched against the top three keywords and three key classes delivered by OMEGA.

Table 11 OMEGA precision and recall

OMV elements	**Precision**	**Recall**
naturalLanguage	1	0.961
isOfType	0.767	0.846
keyClasses	0.654	0.571
keywords	0.532	0.555
keywords (including synonyms)	0.627	0.662
hasDomain	0.413	0.714

As depicted by Table 11 OMEGA performs well when determining the natural language, the type and the key classes of an ontology. Interestingly, the algorithm provides high precision and low recall on extracting key classes. According to the experimental results, the algorithm performs better on ontologies having less than 100 classes. An explanation for this behavior could be that large-size ontologies cover several domains of interest, which in turn result in a larger number of eligible key classes. Around half of the manually generated ontology keywords coincide with class names in the ontology. In a second step we thus asked the participants to consider alternative synonym keywords, whilst extending the comparison so that it considers not only exact keyword matches between human and machine-generated metadata elements but also additional semantic relations as captured in WordNet. The precision and recall values increased by 10%. This confirms that the heuristic used to determine ontology keywords leads to *meaningful* results, at least as perceived by the test participants. The results also indicate that OMEGA obtains low precision and high recall value when classifying the domain of an ontology. This occurs because of three possible reasons. Firstly, most of the terms used as keywords and key classes appear multiple times in the DMOZ directory. Computing the maximum likelihood may alleviate this problem, but not to a satisfactory extent. On average, the algorithm returns two to three related domains and one of them is the correct answer. The second reason is the ontology size. In the dataset used for our experiments, there were both very small ontologies (containing less than five classes) and very large ontologies (containing hundreds of classes). When the ontology is too small, keywords and key classes do not provide a feasible basis for matching the ontology domain to one of the DMOZ top categories. At the other end of the spectrum, large ontologies are often multi-domain and thus difficult to classify, at least using the technique introduced in Section 5. Finally, due to the characteristics of the DMOZ directory, ontologies containing labels in languages other than English are always assigned to the *World* domain.

7.3 User-perceived quality

A second experiment was conducted to test the overall user-perceived quality of OMV entries produced by OMEGA. The experiment involved ten computer science students with basic knowledge in ontology engineering. They were provided with a set of ontologies and their associated OMEGA-generated metadata and were asked for each ontology to indicate how well each of the metadata elements in the corresponding record applied to the ontology. The scale used for the evaluation contained the following scores: *Very Poorly*, *Poorly*, *Well*, *Very Well*, or *Unsure*. In this second case the dataset contained 90 ontologies retrieved from Watson by issuing the query `University`. Each ontology/metadata record pair was reviewed by two to three participants leading to a total of 210 answers.

The average quality scores of each ontology are calculated and represented as a histogram in Figure 5. The figure shows that more than 75% of metadata automatically generated by

OMEGA were rated with Well (46%) or Very Well (31%). Most OMV entries that were rated Poor or lower contain limited metadata information, which was directly reused from existing metadata repositories or search engines

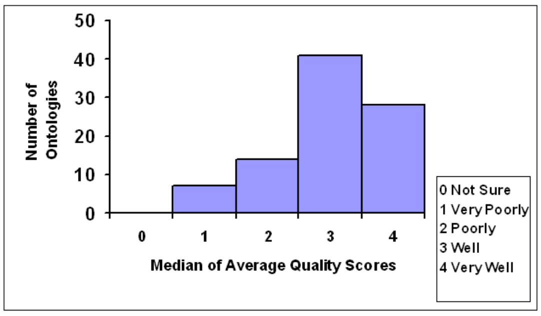


Figure 5 User-perceived quality

8 Conclusion

The availability of metadata information about ontologies available on the Web constitutes a fundamental step forward towards the realization of fully-fledged ontology repositories and search engines, and facilitates the reuse of ontological resources beyond the boundaries of their initial development scope. In this paper we surveyed some of the most important metadata schemas created in communities as diverse as digital libraries, ontology engineering, Web 2.0 and Web of Data to identify those aspects that are most relevant when it comes to supporting potential users in finding, understanding, assessing and eventually utilizing an ontology. Automatic methods to generate such metadata are the only feasible approach to scale at the number of ontologies constantly being developed in various sectors to date. For this purpose, we introduced the OMEGA (Ontology MEtadata GenerAtion) algorithm, which uses OMV (Ontology Metadata Vocabulary) as basic ontology metadata schema, and is available as Web application and REST Web service for further usage within semantic applications. The results of our evaluation experiments show that a heuristic approach to ontology metadata generation is feasible and that additional customizations to consider multi-linguality, ontologies of various sizes and an alternative category hierarchy are likely to enhance the results even further. Improvements could be obtained by exploiting the most recent features provided by the Watson tool, in particular its means to compute ontology similarities, which could be used as an additional criterion for testing the correctness of the OMEGA results, as well as the various statistics, which could extend the current coverage of the algorithm. Finally, the algorithm could take into account existing metadata from ontology repositories, or even taxonomy repositories such as the Taxonomy Warehouse,[21] to optimize its heuristics. We intend to develop our implementation in these directions to transform OMEGA into a core building block for ontology reuse technology.

Acknowledgments

Vicomtech-IK4 would like to thank the Basque Government for co-funding through the eTourgune project.

References

Alexander, K., Cyganiak, R., Hausenblas, M. & Zhao, J. (2009), Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets', *in* 'WWW 2009 Workshop: Linked Data on the Web (LDOW2009)', Madrid, Spain.

Arpirea, J., Gomez-Porez, A., Lozano-Tello, A. & Pinto, H. (2000), 'Reference Ontology and (ONTO)2 Agent: The Ontology Yellow Pages', *Knowledge and Information Systems* **2**, 387–412.

Baclawski, K. & Schneider, T. (2009), The open ontology repository initiative: Requirements and research challenges, *in* 'The Semantic Web - ISWC 2009 8th International Semantic Web Conference, ISWC 2009'.

Bojars, U., Breslin, J., Finn, A. & Decker, S. (2008), 'Using the Semantic Web for linking and reusing data across Web 2.0 communities', *Journal of Web Semantics* **6**(1), 21–28.

Buitelaar, P., Eigner, T. & Declerck, T. (2004), OntoSelect - A Dynamic Ontology Library with Support for Ontology Selection, *in* '3rd International Semantic Web Conference (ISWC 2004), Demo Session, Japan'.

d'Aquin, M. & Lewen, H. (2009), Cupboard - a place to expose your ontologies to applications and the community, *in* '6th European Semantic Web Conference ESWC', pp. 913–918.

d'Aquin, M., Sabou, M., Dzbor, M., Baldassarre, C., Gridinoc, L., Angeletou, S. & Motta, E. (2007), WATSON: A Gateway for the Semantic Web, *in* '4th European Semantic Web Conference (ESWC 2007), Poster Session, Austria'.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V. C. & Sachs, J. (2004), Swoogle: A search and metadata engine for the Semantic Web, *in* '13th ACM Conference on Information and Knowledge Management', pp. 58–61.

Ding, Y. & Fensel, D. (2001), Ontology library systems: The key to successful ontology reuse, *in* 'SWWS', pp. 93–112.

Fikes, R. & Farquhar, A. (1999), 'Distributed repositories of highly expressive reusable ontologies', *IEEE Intelligent Systems* **14**, 73–79.

Garca-Castro, A., Garca-Castro, L., Villaveces, J. M., Caldern, G. & Hepp, M. (2009), Ols2owl. a repository management facility, *in* '11th International Protege Conference'.

Gómez-Pérez, A. (2001), 'Evaluation of ontologies', *International Journal of Intelligent Systems* **16**(3), 391–409.

Gracia, J., Liem, J., Lozano, E., Corcho, O., Trna, M., Gómez-Pérez, A. & Bredeweg, B. (2010), Semantic techniques for enabling knowledge reuse in conceptual modelling, *in* 'International Semantic Web Conference ISWC2010', pp. 82–97.

Hartmann, J., Palma, R. & Gmez-Prez, A. (2009), Ontology repositories, *in* S. Staab & D. Rudi Studer, eds, 'Handbook on Ontologies', International Handbooks on Information Systems, Springer Berlin Heidelberg, pp. 551–571.

Hartmann, J., Paslaru-Bontas, E., R. Palma, R. & Gomez-Perez, A. (2006), Demo - a design environment for metadata ontologies, *in* '3rd European Semantic Web Conference, Montenegro'.

Hartmann, J., Sure, Y., Haase, P., Palma, R. & del Carmen Surez-Figueroa, M. (2005), Omv – ontology metadata vocabulary, *in* C. Welty, ed., '4th. International Semantic Web Conference ISWC 2005 - In Ontology Patterns for the Semantic Web'.

Hausenblas, M., Halb, W., Raimond, Y., Feigenbaum, L. & Ayers, D. (2009), Scovo: Using statistics on the web of data, *in* '6th. European Semantic Web Conference ESWC', pp. 708–722.

Heath, T. & Bizer, C. (2011), *Linked Data: Evolving the Web into a Global Data Space*, Morgan & Claypool.

KnowledgeWeb European Project (2004), 'Identification of standards on metadata for ontologies (Deliverable D1.3.2 KnoweldgeWeb FP6-507482)'.

Lozano-Tello, A. & Gómez-Pérez, A. (2004), 'ONTOMETRIC: A Method to Choose the Appropriate Ontology', *Journal of Database Management* **15**(2), 1–18.

McGuinness, D. L. (2002), Ontologies Come of Age, *in* 'Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential', MIT Press.

Mochol, M. (2009), The methodology for finding suitable ontology matching approaches, PhD thesis, Freie Universität Berlin.

National Information Standards Organization (2004), 'Understanding Metadata', NISO Press.

Noy, N. F., Griffith, N. & Musen, M. (2008), Collecting Community-Based Mappings in an Ontology Repository, *in* 'International Semantic Web Conference', pp. 371–386.

Palma, R., Haase, P. & Gómez-Pérez, A. (2006), Oyster: sharing and re-using ontologies in a peer-to-peer community, *in* '15th. International World Wide Web Conference WWW', pp. 1009–1010.

Paslaru-Bontas, E., Mochol, M. & Tolksdorf, R. (2005), Case Studies on Ontology Reuse, *in* 'Proceedings of the 5th International Conference on Knowledge Management IKNOW2005', pp. 345–353.

Pinto, H. & Martins, J. (2000), Reusing ontologies, *in* 'Proceedings of AAAI 2000 spring symposium series, workshop on bringing knowledge to business processes, SS-00-03'.

Simperl, E. (2009), 'Reusing ontologies on the semantic web: A feasibility study', *Data and Knowledge Engineering* **68**(10), 905–925.

Simperl, E. & Bürger, T. (2010), *Data Management on the Semantic Web*, Nova Science Publishers, chapter Ontology Reuse - Is it Feasible?

Simperl, E. & Mochol, M. (2007), *Semantic Web Methodologies for eBusiness Applications: Ontologies, Processes and Management Practices*, Information Science Reference, chapter A Case Study in Building Semantic eRecruitment Applications, pp. 83–104.

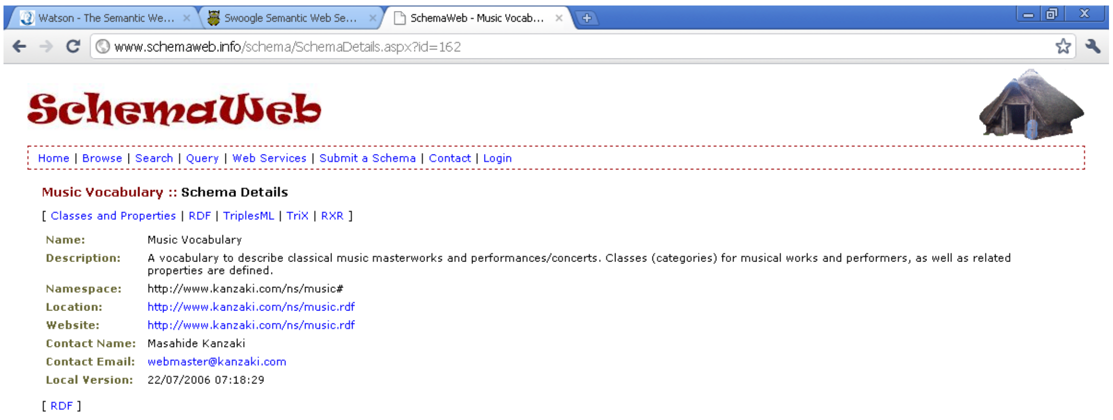
Tummarello, G., Delbru, R. & Oren, E. (2007), Sindice.com: Weaving the open linked data, *in* '6th. International Semantic Web Conference ISWC/ASWC', pp. 552–565.

Wasserman, S. & Faust, K. (1994), *Social Network Analysis*, Cambridge University Press, New York.

Zhang, Y., Vasconcelos, W. & Sleeman, D. (2004), Ontosearch: An ontology search engine, *in* '24th SGAI Int. Conf. Innovative Techniques and Applications of AI, UK'.

Note

[1]http://www.neon-toolkit.org/wiki/1.x/Oyster-menu
[2]http://watson.kmi.open.ac.uk/WatsonWUI/
[3]http://dublincore.org/
[4]http://creativecommons.org/ns
[5]KnowledgeWeb: http://knowledgeweb.semanticweb.org, NeOn: www.neon-project.org/
[6]OOR: http://ontolog.cim3.net/cgi-bin/wiki.pl?OpenOntologyRepository
[7]http://www.w3.org/2004/02/skos/
[8]Protégé: http://protege.stanford.edu, Apollo: http://apollo.open.ac.uk/index.html, WSMT: http://sourceforge.net/projects/wsmt, NeOnToolkit: http://neon-toolkit.org/wiki/Main_Page, Swoop: http://www.mindswap.org/2004/SWOOP/, IODT: http://www.alphaworks.ibm.com/tech/semanticstk
[9]LinkFactory: OntoStudio: http://www.ontoprise.de/en/home/products/ontostudio/, TopBraid Composer: http://www.topbraidcomposer.com, SemanticWorks: http://www.altova.com/products/semanticworks/semantic_web_rdf_owl_editor.html
[10]For the commercial tools, our analysis was in part based on evaluation versions, and publicly available documentation.
[11]DAML ontology library: http://www.daml.org/ontologies, Vocab.org: http://vocab.org/, Protégé OWL library: http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library, SchemaWeb directory: http://www.schemaweb.info
[12]TONES repository: http://owl.cs.manchester.ac.uk/repository/, BioPortal repository: http://bioportal.bioontology.org/, ONKI library service: http://www.yso.fi/
[13]http://code.google.com
[14]http://www.dmoz.org/
[15]http://en.wikipedia.org/wiki/Portal:Contents/Categorical_index
[16]http://www.mindswap.org/2003/pellet/
[17]www.nlm.nih.gov/research/umls/
[18]The Wikipedia Categorical Index can be used in a similar manner.
[19]Furthermore, the results from some preliminary tests showed that using all classes of the ontology in this step decreased classification accuracy.
[20]Of course other database management systems, particularly triplestores, can be used for this purpose as well.
[21]http://taxonomywarehouse.com/


Watson - The Semantic We...
Swoogle Semantic Web Se...
SchemaWeb - Music Vocab...
www.schemaweb.info/schema/SchemaDetails.aspx?id=162
SchemaWeb
Home | Browse | Search | Query | Web Services | Submit a Schema | Contact | Login

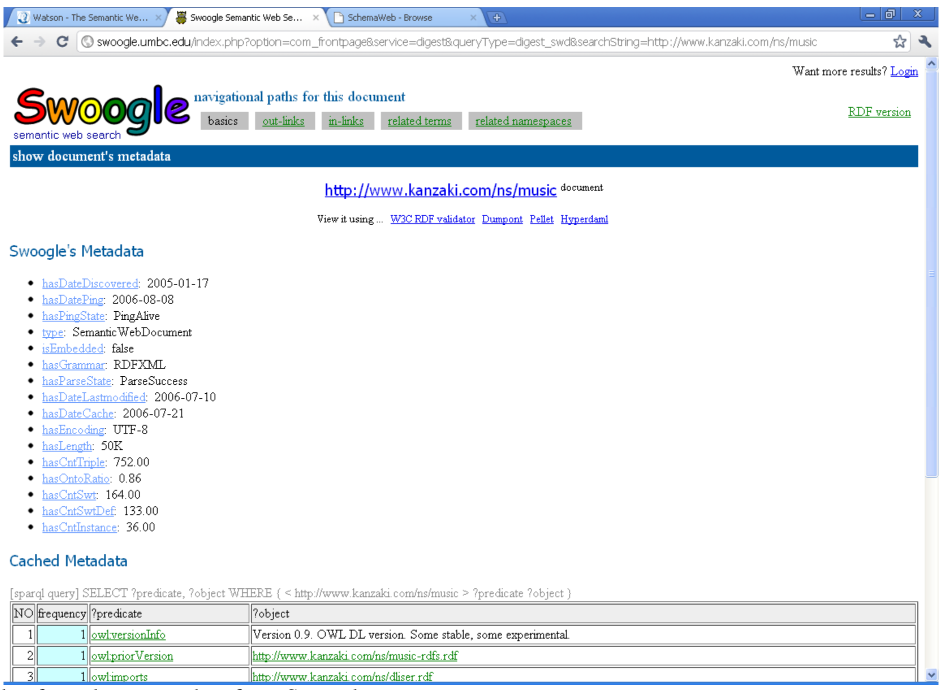
Music Vocabulary :: Schema Details

[Classes and Properties | RDF | TriplesML | TriX | RXR]

Name:	Music Vocabulary
Description:	A vocabulary to describe classical music masterworks and performances/concerts. Classes (categories) for musical works and performers, as well as related properties are defined.
Namespace:	http://www.kanzaki.com/ns/music#
Location:	http://www.kanzaki.com/ns/music.rdf
Website:	http://www.kanzaki.com/ns/music.rdf
Contact Name:	Masahide Kanzaki
Contact Email:	webmaster@kanzaki.com
Local Version:	22/07/2006 07:18:29

[RDF]

Figure 2 An example of ontology metadata from the SchemaWeb directory


Watson - The Semantic We...
Swoogle Semantic Web Se...
SchemaWeb - Browse
swoogle.umbc.edu/index.php?option=com_frontpage&service=digest&queryType=digest_swd&searchString=http://www.kanzaki.com/ns/music
Want more results? Login
Swoogle
semantic web search
navigational paths for this document
basics
out-links
in-links
related terms
related namespaces
RDF version

show document's metadata

http://www.kanzaki.com/ns/music document

View it using ... W3C RDF validator Dumpont Pellet Hyperdaml

Swoogle's Metadata

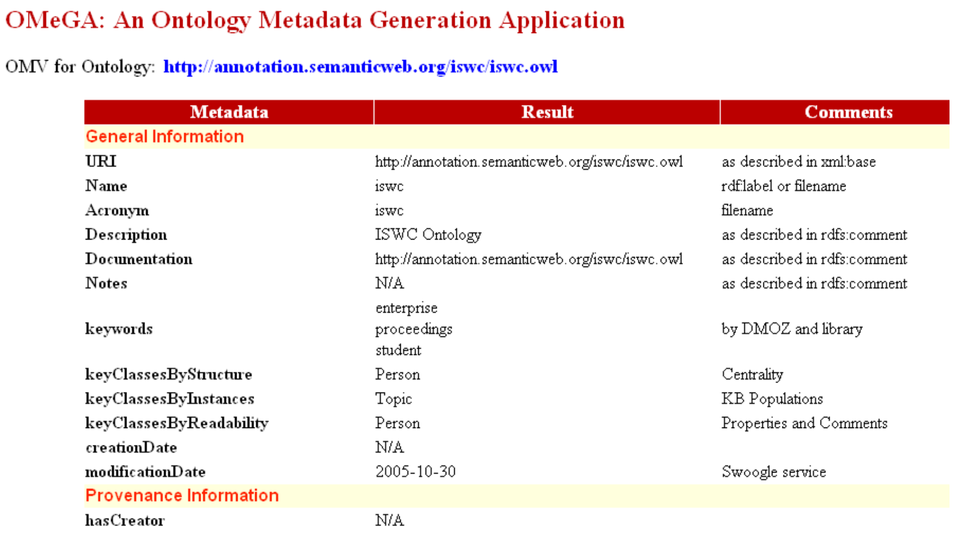
- hasDateDiscovered: 2005-01-17
- hasDatePing: 2006-08-08
- hasPingState: PingAlive
- type: SemanticWebDocument
- isEmbedded: false
- hasGrammar: RDFXML
- hasParseState: ParseSuccess
- hasDateLastmodified: 2006-07-10
- hasDateCache: 2006-07-21
- hasEncoding: UTF-8
- hasLength: 50K
- hasCntTriple: 752.00
- hasOntoRatio: 0.86
- hasCntSwt: 164.00
- hasCntSwtDef: 133.00
- hasCntInstance: 36.00

Cached Metadata

[sparql query] SELECT ?predicate, ?object WHERE { < http://www.kanzaki.com/ns/music > ?predicate ?object }

NO	frequency	?predicate	?object
1	1	owl:versionInfo	Version 0.9. OWL DL version. Some stable, some experimental.
2	1	owl:priorVersion	http://www.kanzaki.com/ns/music-rdfs.rdf
3	1	owl:imports	http://www.kanzaki.com/ns/dliser.rdf

Figure 3 An example of ontology metadata from Swoogle

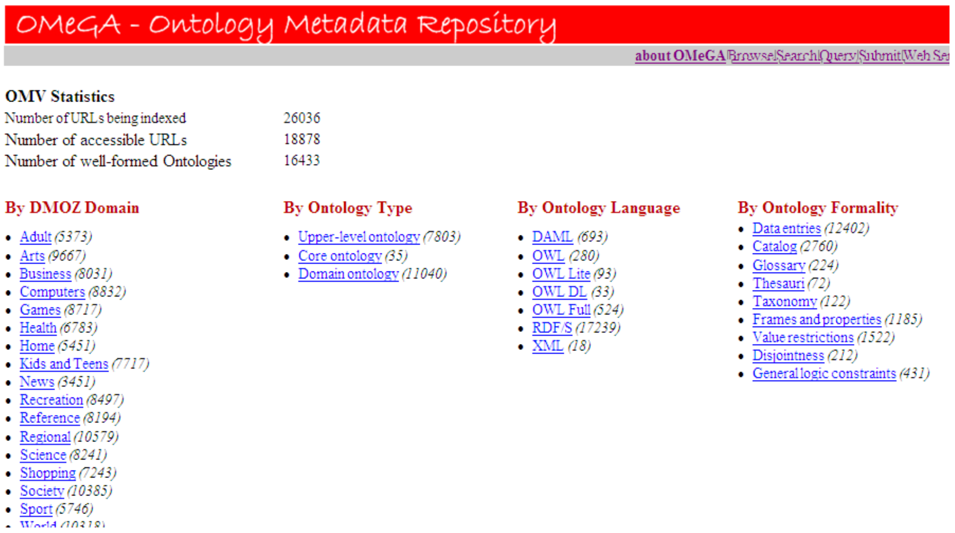


OMeGA: An Ontology Metadata Generation Application

OMV for Ontology: **http://annotation.semanticweb.org/iswc/iswc.owl**

Metadata	Result	Comments
General Information		
URI	http://annotation.semanticweb.org/iswc/iswc.owl	as described in xml:base
Name	iswc	rdf:label or filename
Acronym	iswc	filename
Description	ISWC Ontology	as described in rdfs:comment
Documentation	http://annotation.semanticweb.org/iswc/iswc.owl	as described in rdfs:comment
Notes	N/A	as described in rdfs:comment
keywords	enterprise proceedings student	by DMOZ and library
keyClassesByStructure	Person	Centrality
keyClassesByInstances	Topic	KB Populations
keyClassesByReadability	Person	Properties and Comments
creationDate	N/A	
modificationDate	2005-10-30	Swoogle service
Provenance Information		
hasCreator	N/A	

(a) Result page showing an OMV entry



OMeGA - Ontology Metadata Repository

about OMeGA|Browse|Search|Query|Submit|Web Se

OMV Statistics

Number of URLs being indexed	26036
Number of accessible URLs	18878
Number of well-formed Ontologies	16433

By DMOZ Domain

- Adult *(5373)*
- Arts *(9667)*
- Business *(8031)*
- Computers *(8832)*
- Games *(8717)*
- Health *(6783)*
- Home *(5451)*
- Kids and Teens *(7717)*
- News *(3451)*
- Recreation *(8497)*
- Reference *(8194)*
- Regional *(10579)*
- Science *(8241)*
- Shopping *(7243)*
- Society *(10385)*
- Sport *(5746)*
- World *(10318)*

By Ontology Type

- Upper-level ontology *(7803)*
- Core ontology *(35)*
- Domain ontology *(11040)*

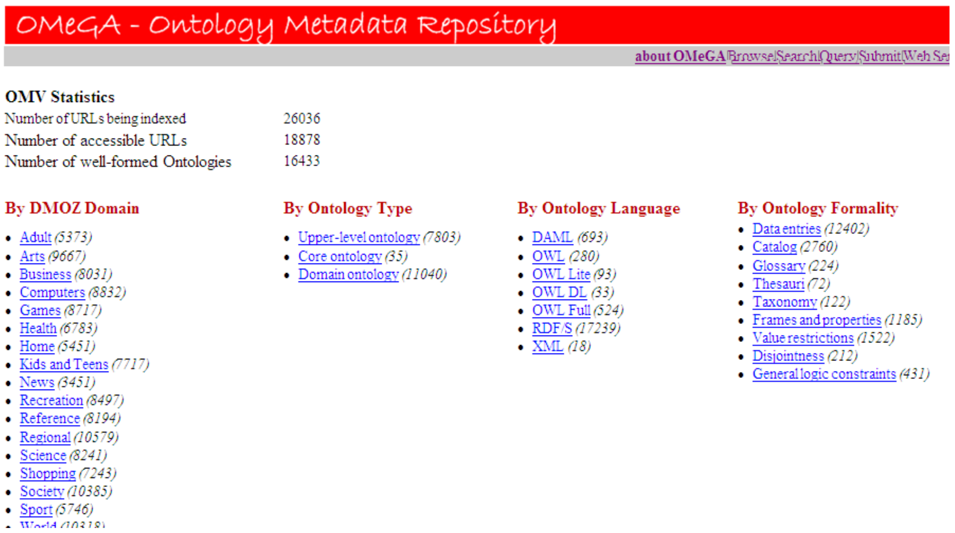
By Ontology Language

- DAML *(693)*
- OWL *(280)*
- OWL Lite *(93)*
- OWL DL *(33)*
- OWL Full *(524)*
- RDF/S *(17239)*
- XML *(18)*

By Ontology Formality

- Data entries *(12402)*
- Catalog *(2760)*
- Glossary *(224)*
- Thesauri *(72)*
- Taxonomy *(122)*
- Frames and properties *(1185)*
- Value restrictions *(1522)*
- Disjointness *(212)*
- General logic constraints *(431)*

(b) Browse page



OMeGA - Ontology Metadata Repository

about OMeGA|Browse|Search|Query|Submit|Web Se

OMV Statistics

Number of URLs being indexed	26036
Number of accessible URLs	18878
Number of well-formed Ontologies	16433

By DMOZ Domain

- Adult *(5373)*
- Arts *(9667)*
- Business *(8031)*
- Computers *(8832)*
- Games *(8717)*
- Health *(6783)*
- Home *(5451)*
- Kids and Teens *(7717)*
- News *(3451)*
- Recreation *(8497)*
- Reference *(8194)*
- Regional *(10579)*
- Science *(8241)*
- Shopping *(7243)*
- Society *(10385)*
- Sport *(5746)*
- World *(10318)*

By Ontology Type

- Upper-level ontology *(7803)*
- Core ontology *(35)*
- Domain ontology *(11040)*

By Ontology Language

- DAML *(693)*
- OWL *(280)*
- OWL Lite *(93)*
- OWL DL *(33)*
- OWL Full *(524)*
- RDF/S *(17239)*
- XML *(18)*

By Ontology Formality

- Data entries *(12402)*
- Catalog *(2760)*
- Glossary *(224)*
- Thesauri *(72)*
- Taxonomy *(122)*
- Frames and properties *(1185)*
- Value restrictions *(1522)*
- Disjointness *(212)*
- General logic constraints *(431)*

(c) Search page

Figure 4 Interfaces of the Web-based prototype system