Project No. 249024

NETMAR

Open service network for marine environmental data

Instrument: *Please tick*	CA	✓ **STREP**	IP	NOE

ICT - Information and Communication Technologies Theme

D3.2 Review of available ontology tooling

Reference: D3_2_review_ontology_tooling_r1

Due date of deliverable: M0 + 3
Actual submission date: 19 May 2010

Start date of project: 1 February 2010 Duration: 3 years

British Oceanographic Data Centre (BODC), National Environment Research Council, United Kingdom

Revision 1

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

European Commission
Information Society and Media

NETMAR

Open service network for marine environmental data
Project Reference: 249024
Contract Type: Collaborative Project
Start/End Date: 01/03/2010 - 31/01/2013
Duration: 36 months

Coordinator: Prof. Stein Sandven

Nansen Environmental and Remote Sensing Center
Thormøhlensgate 47, Bergen, Norway
Tel.: +47-55205800
Fax. +47 55 20 58 01
E-mail: stein.sandven@nersc.no

Acknowledgements

The work described in this report has been partially funded by the European Commission under the Seventh Framework Programme, Theme ICT 2009.6.4 ICT for environmental services and climate change adaptation.

Consortium

The NETMAR Consortium is comprised of:

- Nansen Environmental and Remote Sensing Center (NERSC), Norway (coordinator).
  Project Coordinator: Prof. Stein Sandven (stein.sandven@nersc.no)
  Deputy Coordinator: Dr. Torill Hamre (torill.hamre@nersc.no)
  Quality Control Manager. Mr. Lasse H. Pettersson (lasse.pettersson@nersc.no)
- British Oceanographic Data Centre (BODC), National Environment Research Council, United Kingdom
  Contact: Dr. Roy Lowry (rkl@bodc.ac.uk)
- Centre de documentation de recherche et d'expérimentations sur les pollutions accidentelles des eaux (Cedre), France.
  Contact: Mr. François Parthiot (Francois.Parthiot@cedre.fr)
- Coastal and Marine Resources Centre (CMRC), University College Cork, National University of Ireland, Cork, Ireland.
  Contact: Mr. Declan Dunne (d.dunne@ucc.ie)
- Plymouth Marine Laboratory (PML), United Kingdom.
  Contact: Mr. Steve Groom (sbg@pml.ac.uk)
- Institut français de recherche pour l'exploitation de la mer (Ifremer), France
  Contact: Mr. Mickael Treguer (mickael.treguer@ifremer.fr)
- Norwegian Meteorological Institute (METNO), Norway.
  Contact: Mr. Jan Ivar Pladsen (janip@met.no)

Author(s)

- Dr Adam Leadbetter, BODC (alead@bodc.ac.uk)
- Mr Oliver Clements, BODC (daol@bodc.ac.uk)

Document approval

- Document status: Release 1
- WP leader approval: 2010-05-19
- Quality Manager approval: 2010-05-19
- Coordinator approval: 2010-05-19

Revision History

Issue	Date	Change records	Author(s)
Draft1	2010-04-08	First draft of the report.	Adam Leadbetter Oliver Clements
Draft2	2010-04-22	Revised draft incorporating comments from Roy Lowry and initial comments from NERSC.	Adam Leadbetter Oliver Clements
Draft3	2010-04-23	Revised executive summary	Roy Lowry
Draft4	2010-05-12	Comments from project partners and Advisory Board review incorporated	Adam Leadbetter Oliver Clements
Draft 5	2010-05-17	Final comments from Roy Lowry addressed	Adam Leadbetter Oliver Clements
1	2010-05-19	Final release.	Roy Lowry Torill Hamre

Executive Summary

The Open Service Network for Marine Environmental Data (NETMAR) project aims to develop a pilot European Marine Information System (EUMIS) for searching, downloading and integrating satellite, in situ and model data from ocean and coastal areas. EUMIS will use a semantic framework coupled with ontologies for identifying and accessing distributed data, such as near-real time, forecast and historical data. This report is a review of available tools for the creation, maintenance, serving, querying and browsing of semantic web ontologies and tools for bridging ontologies and human languages.

A range of computer languages exist in which to represent a published ontology. These fall into two categories: those based in the Resource Description Framework (RDF) and those that are not.

Recommendation 1: Due to the broad compatibility between its members, its recommendation by the World Wide Web Consortium (W3C) and its large software base we recommend that the NETMAR project uses the RDF family of languages to represent its ontologies.

A series of query languages have been developed to produce subsets of and interrogate ontologies in the ontology languages efficiently. Some query languages depend on the ontology language used; others depend on the server used to publish the ontology. Benchmarking of some query languages has been undertaken to assess their relative performance.

Recommendation 2: The SPARQL Protocol and RDF Query Language (SPARQL) is the recommended query language for general purpose usage due to the ubiquitous nature of its support and its high level of extensibility. If the Mulgara server is chosen for ontology publication, then the interactive Tucana Query Language (iTQL) is equally as good for retrieval but with the additional advantage of its ability to update concept databases as standard.

Ontology servers provide a storage mechanism and retrieval methods or services for ontologies. These may be a layer over a relational database or a bespoke method such as using a text indexer to store references to RDF triples.

Recommendation 3: The Mulgara server has the easiest mechanisms for entering data and then querying it, with a simple HyperText Transfer Protocol (HTTP) interface to a SPARQL and iTQL endpoint. However it does not use a relational database to store the data, which may cause scalability issues. If the scalability of a relational database is a requirement then Sesame is a better option, while if a cluster-based server is available 4store is recommended.

Where ontology servers provide one necessary piece of tooling, ontology frameworks provide the complete system. That is, they provide a storage mechanism, an editing mechanism and a querying mechanism. Utilising a complete system reduces the risk of interoperability issues.

Recommendation 4: Jena is a powerful complete ontology framework that is considered the best. It provides two mechanisms for storing data, one as a wrapper to a relational database the other using a bespoke system similar to the Mulgara server. The querying mechanism is an extension of SPARQL called ARQ which provides access to Extensible Stylesheet Language Transformations (XSLT) functions that make complex queries possible. Jena also provides classes for creating and editing ontologies programmatically.

An array of applications for editing ontologies has been developed. Many of these are standalone applications, written in the Java language to allow them to run on a variety of platforms. These Java applications tend to be large and particularly memory intensive when used to edit large vocabularies (20,000+ terms), without even considering the mappings required to build an ontology. Other software solutions exist for editing ontologies, such as web browser plugins and web applications built around SQL.

Recommendation 5: If a specific ontology editor is required by small to medium size ontologies in NETMAR then Semantic Turkey is recommended. It has good levels of scalability, it simply drops into the Mozilla Firefox web browser and it is freeware. However, it is anticipated that NETMAR will require large ontologies, which will require an editing tool working on the underlying database (Structured Query Language (SQL) based database tools or bespoke database editors such as the Natural Environment Research Council (NERC) Vocabulary Editor) or an ontology query language with update functionality (iTQL is the only one identified) .

Concept mapping tools are used by businesses in mind mapping exercises as well as by semantic web developers. Consequently, high quality free tools exits to produce visual representations of concepts and their interrelationships. Ontologies may be bridged using concept mapping techniques, with standalone software and web services available for this purpose.

Recommendation 6: The concept mapping tool recommended for NETMAR is the CMAPTools Ontology Editor due to its ability to export visual concept maps as Web Ontology Language (OWL) documents. The Marine Metadata Interoperability (MMI) project's Vocabulary Integration Environment (VINE) may be utilised to build bridges between ontologies stored in the MMI Ontology Registry and Repository and other semantic resources.

The development of tooling for the following needs has also been reviewed: (1) Ontology browsers, (2) Conversion of text to RDF, (3) Tool chaining of ontologies, and (4) Multilingual support. Software for these activities is very limited, and is unlikely to fully meet the requirements of the NETMAR project.

Recommendation 7: Existing software for ontology browsing, converting text to RDF and tool chaining for ontology development is unlikely to meet the requirements of NETMAR. Consequently, tooling will either have to be developed within the project (if feasible within resources constraints) or the level of semantic functionality matched to what is available. In particular, in order to allow cross human language domain concept bridging, multilingual ontologies will be required, which is still very much in the research rather than the operational domain. It is recommended that active research is monitored to identify any appropriate additional tools that become available during the course of the project.

Recommendation 8: In conclusion, it is recommended that an RDF based ontology be developed for use by the NETMAR project. This ontology should be queried by SPARQL and served using the Mulgara server or the Jena framework, unless a cluster server is available, in which case 4store is recommended. Editor tools such as Semantic Turkey or a query language web interface should be used and to bridge or extend existing ontologies the CMAPTools Ontology Editor and MMI VINE software should be used. Tooling will either have to be developed within the project for other semantic requirements (if feasible within resources constraints) or the level of semantic functionality matched to those tools which are available.

Contents

1 Introduction

1.1 Background

The Open Service Network for Marine Environmental Data (NETMAR) project (http://www.netmar-project.eu/) aims to develop a pilot European Marine Information System (EUMIS) for searching, downloading and integrating satellite, in situ and model data from ocean and coastal areas. It will be a user-configurable system offering flexible service discovery, access and chaining facilities using Open Geospatial Consortium (OGC), Open-source Project for a Network Data Access Protocol (OPeNDAP) and World Wide Web Consortium (W3C) standards. It will use a semantic framework coupled with ontologies for identifying and accessing distributed data, such as near-real time, forecast and historical data. EUMIS will also enable further processing of such data to generate composite products and statistics suitable for decision-making in diverse marine application domains. Figure 1-1 illustrates how observations, derived parameters and predictions are retrieved from a distributed service network through standard protocols, and delivered through the EUMIS portal using ontologies and semantic frameworks to select suitable products and where new products can be generated dynamically using chained processing services.

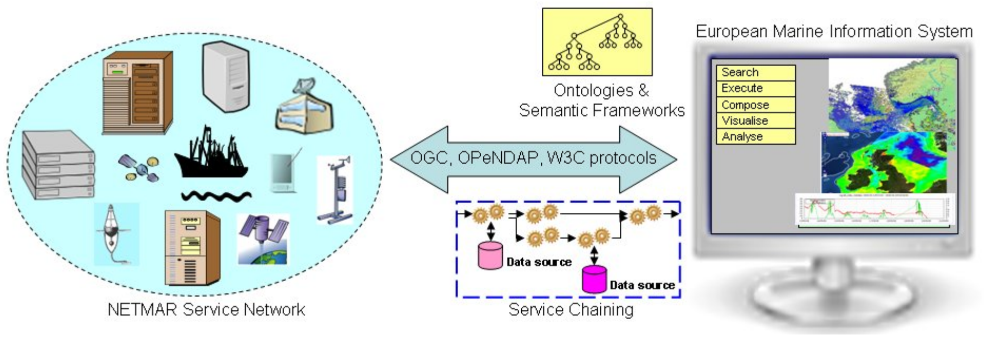


Figure 1-1 The NETMAR system concept.

1.2 Objective of this report

The objective of this report is to provide a description of available tools for ontology development and utilisation with particular reference to tools for bridging existing populated ontologies and for providing multilingual functionality.

1.3 Terminology

The term ‘ontology’ is used widely in this report. In this context, an ontology is the formal representation of a body of knowledge through the declaration of concepts from a given domain and defining the relationships between those concepts. It can be used both to describe and to infer knowledge about a given domain.

‘Concept mapping’ refers to the process of identifying the concepts which are to be represented in the ontology and the relationships between the concepts. This is often done in a graphical way, similar to mind mapping or spider diagrams. Figure 1-2 is taken from the Wikipedia entry for the term ‘concept map’ [Wi05].

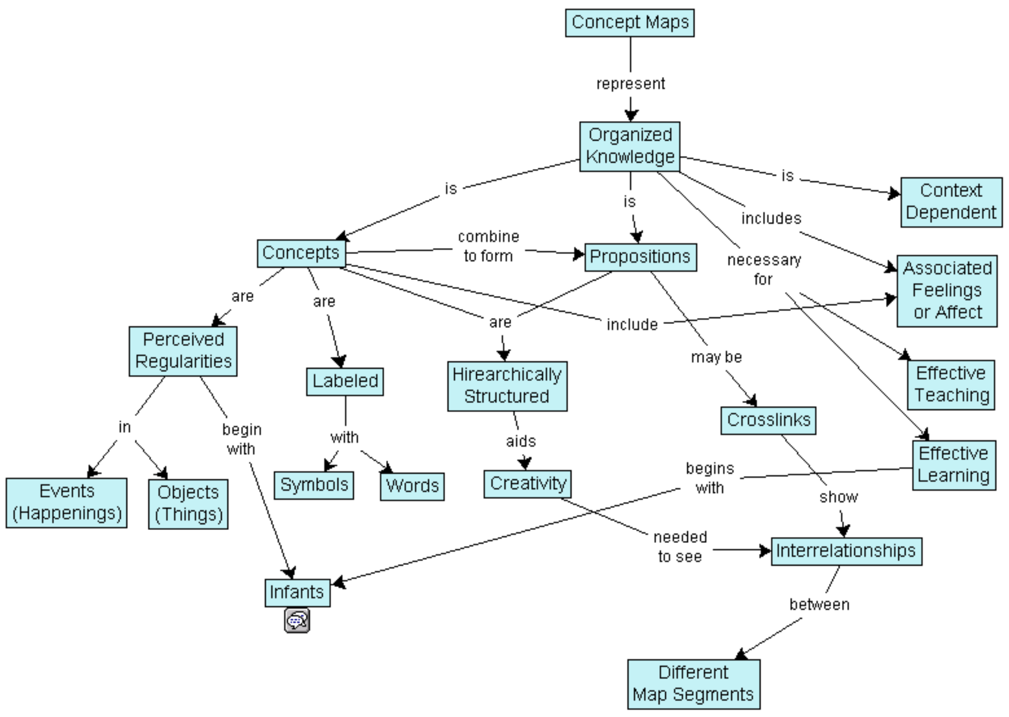


Figure 1-2 Illustration of the term 'concept map' (source: [Wi05]).

'Ontology bridging' or 'ontology extension' are the terms used to describe the mapping of concepts represented in one ontology to those concepts represented in a second ontology.

A 'software toolchain' is a set of computer programs used to create a product. They may, or may not, be used in a chain where the output from one program becomes the input to the next. 'Toolchaining of ontologies' is a phrase used in this report to describe the creation of software toolchains relevant to the development of semantic web ontologies.

1.4 Organisation of this report

This report is broken down into chapters reviewing the current state of development of a range of ontology tool types.

- Section 2 discusses the computer languages used to publish an ontology.
- Section 3 considers the languages which can be used to query a published ontology.
- Section 4 covers tools for the creation, viewing and editing of ontologies.
- Section 5 reviews software tools which can take plain text documents and turn them into ontologies.
- Section 6 covers tools for mapping the relationships between concepts in an ontology and between different ontologies.
- Section 7 presents the current state of chains of software tools to produce ontologies.
- Section 8 discusses software for serving ontologies across the world wide web represented in a standard ontology language.
- Section 9 considers the software frameworks and application programming interfaces for the creation and serving of ontologies.
- Section 10 discusses software which allows users to browse through ontologies served from the world wide web.
- Section 11 covers the tools and techniques available for the creation of ontologies which define their concepts in more than one human language.
- Section 12 presents the conclusions of this report.

2 Ontology Languages

2.1 Resource Description Framework (RDF)

RDF [He10] is a family of W3C specifications originally designed as a data model for metadata, which has since become a general method for modelling information which is then served as web resources [Wi10a]. An RDF document is built up of a number of statements which are made about the resources being described, in the form of subject-predicate-object expressions, known as RDF triples. The subject defines the resource; the predicate denotes properties of that resource and expresses the relationship between the object and the subject. For example, in the expression “the ocean is a form of water body”: the subject is “the ocean”; the predicate is “is a form of”; and the subject is “water body”.

It has been criticised for having an overly verbose form when expressed in Extensible Markup Language (XML) (although there are other ways of expressing RDF); the concept of the triple means that RDF can be both linguistically and computationally inefficient; and that RDF can be used to make ambiguous but possibly factually correct statements.

RDF is a recommendation of the W3C.

2.2 RDF Schema (RDFS)

RDFS extends RDF to provide the basic elements fro the description of ontologies which are represented in RDF. The main constructs of RDFS are the class (and subclass), the property and the utility property.

The RDFS Class declares a class for use by other resources. An example of this is the Friend Of A Friend (FOAF) person class, where a resource is defined as instance of the class using the rdf:type predicate, e.g. eg:Jeremy rdf:type foaf:Person. The rdfs:subClassOf syntax allows the creation of a hierarchy of classes within an RDFS ontology.

The RDFS Property construct is defined as the class which describes the properties of an RDF resource. Each member of the Property class is an RDF predicate, and there are three members: rdfs:domain, rdfs:range and rdfs:subPropertyOf. Domain declares the subject of the RDF triple to be of the class given by the object, while range gives declares the object to be of the class given by the subject. subPropertyOf is used to state that all resources related by one property are also related by another.

By making the assertion that ‘subject A’ ‘predicate rdfs:domain’ ‘object B’, then it must follow that in X A Y, X must be of type B. This is also true for rdfs:range.

There are two members of the RDFS Utility Property construct, rdfs:seeAlso and rdfs:isDefinedBy. Respectively they indicate a resource which may provide additional information about the current resource, and a resource which provides a definition of the current resource (this may be another RDF vocabulary).

Finally there are four more members of the RDFS specification which fall into a separate category. These are the: rdfs: label which renders the resource name in a human readable form; the rdfs:comment which gives a human readable description of the resource; the rdfs:Literal which allows storage of data values; and the rdfs:Datatype which is the class of datatypes and is a subclass of rdfs:Literal.

RDFS is a recommendation of the W3C.

2.3 Web Ontology Language (OWL)

The OWL [MvH04] family of ontology authoring languages is based on two semantic models which are largely compatible. There are three members of the OWL family: OWL Lite, OWL Description Logics (DL) and OWL Full.

OWL Lite and OWL DL are based on Description Logic, which brings with it a rich legacy of understanding and computational knowledge. OWL DL uses a model which preserves some compatibility with RDFS.

The following compatibility exists between the members of the OWL family:

- Every legal OWL Lite ontology is a legal OWL DL ontology
- Every legal OWL DL ontology is a legal OWL Full ontology
- Every valid OWL Lite conclusion is a valid OWL DL conclusion
- Every valid OWL DL conclusion is a valid OWL Full conclusion

OWL makes an open world assumption, meaning that if a statement cannot be proven to be true using the current knowledge base, we also cannot infer that it is false. This is in contrast to the closed world assumption of the Structured Query Language (SQL).

OWL is currently at version 2 [HKP09], which both extended and revised earlier versions of the language. OWL 2 adds new functionality with respect to OWL 1. Some of the new features are aimed at cleaning OWL’s syntax while others offer new expressivity, including:

- keys
- property chains
- richer datatypes
- data ranges
- qualified cardinality restrictions
- asymmetric, reflexive, and disjoint properties
- enhanced annotation capabilities

OWL 2 also defines three new profiles and a new, more-human readable syntax. In addition, some of the restrictions applicable to OWL DL have been relaxed; as a result, the set of RDF Graphs that can be handled by Description Logic reasoners is slightly larger in OWL 2.

An OWL 2 profile is a trimmed down version of OWL 2 that trades some expressive power for the efficiency of reasoning. The profiles are independent of each other. The choice of which profile to use in practice will depend on the structure of the ontologies and the reasoning tasks expected to be performed on those ontologies.

OWL 2 Expression Logic (EL) is particularly useful in applications employing ontologies that contain very large numbers of properties and/or classes. This profile captures the expressive power used by many such ontologies and is a subset of OWL 2 for which the basic reasoning problems can be performed in time that is polynomial with respect to the size of the ontology. Dedicated reasoning algorithms for this profile are available and have been demonstrated to be highly scalable.

OWL 2 Query Language profile (QL) is aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task. In OWL 2 QL, conjunctive query answering can be implemented using conventional relational database systems. Using a suitable reasoning technique, sound and complete conjunctive query answering can be performed in log space with respect to the size of the dataset.

OWL 2 Rule Language (RL) is aimed at applications that require scalable reasoning without sacrificing much expressive power. It is designed to accommodate OWL 2 applications that can trade the full expressivity of the language for efficiency, as well as RDF(S) applications that need some added expressivity.

Apart from the profiles specified here, many other possible profiles of OWL 2 exist — there are, for example, a whole family of profiles that extend OWL 2 QL. All OWL 1 Lite ontologies are OWL 2 ontologies, so OWL 1 Lite can be viewed as a profile of OWL 2. Similarly, OWL 1 DL can also be viewed as a profile of OWL 2.

2.4 Common Logic

Common Logic (CL) is an International Organization for Standardization (ISO) standard [Iso07] family of syntaxes for the representation of knowledge on the World Wide Web based on first order logic (FOL). The basic component of the CL standard is the sentence, which is the statement of an axiom about the resource. These sentences can be atomic (e.g. type_of(rain,precipitation)), Boolean (e.g. not has_part(Pacific_Ocean,Sargasso_Sea)), or quantified.

As of version 1.3, the Open Biological and Biomedical Ontologies (OBO) is rendered in Common Logic [Mu09], due to: the presence of multiple relations between subject and values in OBO; the difficulties of defining relations within OWL; and the body of literature which has grown up around FOL over the last century.

2.5 Simple Knowledge Organization System (SKOS)

SKOS is a family of languages designed for the easy representation of structured controlled vocabularies as web resources. The SKOS family is based upon RDF and RDFS, and is developed within the W3C framework.

SKOS is designed to be modular, and originally had three recognised components: SKOS Core, SKOS Mapping and SKOS Extensions. SKOS Core defines classes and properties in a manner which allows the representation of a controlled vocabulary. The building block within SKOS Core is the concept, each of which is defined as an RDF resource. Each resource has: one or more preferred label terms, in an equal number of natural languages; synonymous terms; definitions, including a specification of the natural language in which they are rendered. Using RDF predicates of ‘broader than’ or ‘narrower than’, hierarchies of concepts can be established within a SKOS thesaurus.

SKOS Mapping was intended to provide the possibility of mapping concepts from one scheme to another, while SKOS Extensions has been designed to increase the richness of the concept relationships beyond the simple ‘narrower-broader’ model. Both of these SKOS components were only maintained informally until SKOS became a W3C recommendation so have not been widely implemented in the past.

SKOS is now a recommendation of the W3C [MB09], and in this version there is no explicit namespace separation between the original SKOS Core and SKOS Mapping components. SKOS has been used to implement the General Multilingual Environmental Thesaurus (GEMET), developed for the European Environment Agency.

2.6 Recommendations

It is recommended that the RDF family be used for the ontologies developed by NETMAR. Beyond this, most ontology tools are happy with any flavour of the RDF family, so any of RDF, RDFS, OWL or SKOS could be used. As SKOS provides a legal OWL-Full ontology it would seem that this is a sensible choice of ontology language.

3 Ontology query languages

3.1 New Racer Query Language (nRQL)

nRQL is the query language for the Racer Description Logic reasoner, which is compatible with both OWL Lite and OWL DL ontologies [HMW04]. In nRQL, a query is built up of a series of atoms which define the concept to be retrieved from the knowledge base, the instances of those concepts which are to be considered by the query (variables); and any constraints placed on the query results. Where variables are included within the query, they are assumed to belong to the ontology being queried (the active domain assumption). nRQL also assumes that each variable name is unique across the entire domain (the unique name assumption). Union queries (familiar from SQL syntax) are possible to implement but can be very complicated in nRQL.

Benchmarking of Racer and nRQL has been undertaken on large datasets [We06]. If the query considers the complete knowledge base (including all relations) and this is over 10,000 individuals, the query becomes unfeasibly slow. If the completeness is sacrificed (i.e. the relations are ignored in the query) over 150,000 individuals can be loaded and queried in a reasonable time.

3.2 OWL Query Language (OWL-QL)

The OWL-QL specification is for a formal language and protocol which defines both a querying agent and an answering agent which enter into a query-answering dialogue for knowledge represented in OWL [FHH03]. It includes several assumptions about the nature of conducting queries on the semantic web.

The first assumption is that the query-answering service may access information in many formats. OWL-QL therefore allows the dialogues to have an answering agent which uses automated reasoning to derive answers to queries and which is capable of using knowledge bases spread across the semantic web even when those knowledge bases have not been specified in the query.
The second assumption is that, due to limitations in both knowledge and performance, some servers will only be able to give incomplete answers to queries. OWL-QL therefore allows the answering agent to deliver partial sets of answers to a query as they are generated.

OWL-QL anticipates that the server handling the query will be able to both select sources of knowledge bases which are reliable in order to answer those queries and to allow the client to request which sources are used to answer their query.

The specification for OWL-QL does not assume implementation of the language will necessarily be identical everywhere on the semantic web, rather it gives a description of the types of object that are passed in the query-answer dialogue, the required and optional components of each object type, and the server response to receiving a specific object type.

An OWL-QL query consists of an object which must contain a query pattern that specifies a collection of OWL sentences in which some URIs are considered to be variables. The object may optionally contain a list of variable that must be bound, a list of variables that may be bound, a list of variables that are not to be bound, query assumptions, and the knowledge base(s) to be queried. The answer to such a query may contain one or more bindings to URIs or literals which satisfy the query pattern sent to the server.

An example OWL-QL query to extract the preferred labels for concepts from the Natural Environment Research Council (NERC) Data Grid vocabulary P021 is:

```
<owl-ql:query
  xmlns:owl-ql="http://www.w3.org/2003/10/owl-ql-syntax#"
  xmlns:var="http://www.w3.org/2003/10/owl-ql-variables#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">

  <owl-ql:premise>
    <rdf:RDF>
      <skos:Concept rdf:about="#C">
      </skos:Concept>
    </rdf:RDF>
  </owl-ql:premise>

  <owl-ql:queryPattern>
    <rdf:RDF>
      <rdf:Description rdf:about="#C">
        <skos:prefLabel
          rdf:resource="http://www.w3.org/2003/10/owl-ql-variables#x"/>
      </rdf:Description>
    </rdf:RDF>
  </owl-ql:queryPattern>

  <owl-ql:mustBindVars>
    <var:x/>
  </owl-ql:mustBindVars>

  <owl-ql:answerKBPattern>
    <owl-ql:kbRef
      rdf:resource="http://vocab.ndg.nerc.ac.uk/list/P021/current/"/>
  </owl-ql:answerKBPattern>

</owl-ql:query>
```

3.3 RDF Data Query Language (RDQL)

RDQL has syntax which resembles SQL, with SELECT, FROM, WHERE, AND, USING clauses [OR04]. The SELECT clause specifies the variables from the knowledge base which are to be returned by the query. The RDQL SELECT clause also accepts SQL-like shortcuts, e.g. SELECT *. In the FROM clause the Universal Resource Identifier (URI) of the RDF knowledge base to be queried is specified. The WHERE clause indicates a list of RDF triple patterns which must be matched to provide a valid answer to the query, with the subject, predicate and object each being a URI or a variable. It is also possible for the RDF triple object to be a literal.

The AND clause of the query can be used to specify conditions to placed on the answer from the WHERE clause. These conditions may be Boolean, arithmetic, string equalities or regular expressions. AND clause conditions may be combined using logical operators and negation. In order to make a query more readable, the USING clause can be used to shorten the length of URIs in the FROM, WHERE and AND clauses by aliasing the URI to a short string.

An example RDQL query to extract the preferred labels for concepts from the NERC Data Grid vocabulary P021 is:

```
SELECT ?conceptName
FROM <http://vocab.ndg.nerc.ac.uk/list/P021/current>
WHERE (?conceptName, <http://www.w3.org/2004/02/skos/core#>,
  <http://www.w3.org/2004/02/skos/core#prefLabel>)
```

As in SQL, comments may be added to queries enabling them to be more easily human readable.

RDQL is a member submission with the W3C [Se04].

3.4 SPARQL Protocol and RDF Query Language (SPARQL)

SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns [Wi10b]. A SPARQL query may consist of PREFIX, SELECT, WHERE, CONSTRUCT, FILTER, BASE and OPTIONAL clauses. The PREFIX clause provides a short alias for the URI of ontology to be searched in order to make the SPARQL query syntax more compact. SELECT specifies which variables to return from the query and the WHERE clause specifies the RDF patterns which must be matched in order to satisfy the query. SPARQL's ability to be more than a simple query language is shown by the CONSTRUCT clause, which allows the query to return a triple or a set of triples, using the answer to the query to generate new RDF data. The FILTER syntax is used to add constraints to the answer to the query, e.g. age > 17 [Ga07]. The FILTER can use many operators, takes functions such as regular expressions, and can also be extended by SPARQL users. BASE defines the initial URI for the query and OPTIONAL makes the matching of part of a pattern non-mandatory.

SPARQL queries are globally unambiguous [Wi10b], which means that the query assumes that the ontologies used to describe the variables being returned eventually converge on the specification described in the PREFIX clause. SPARQL is highly extensible in nature, through the writing of SPARQL functions.

An example SPARQL query to extract the preferred labels for concepts from the NERC Data Grid vocabulary P021 is:

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?conceptName
FROM <http://vocab.ndg.nerc.ac.uk/list/P021>
WHERE {
  ?concept a skos:Concept.
  ?concept skos:prefLabel ?conceptName.
}
```

SPARQL is a recommendation of the W3C [PS08].

3.5 Interactive Tucana Query Language (iTQL)

The iTQL query language is associated with the Mulgara ontology server. iTQL is highly influenced by SQL, and is used to both query and update Mulgara databases. As such, its syntax includes both SELECT, and UPDATE / COMMIT / ROLLBACK keywords. In favour of iTQL is its clone-like resemblance of SQL meaning that anyone familiar with SQL syntax can immediately write queries in iTQL. The ability to update the database is also favourable as it allows for easy distributed management of the ontology.

An example SPARQL query to extract the preferred labels for concepts from the NERC Data Grid vocabularies is:

```
select $Subject
subquery
(
  select $Subject
  from <rmi://livlbl/server1#sampledata>
  where
```

```
      $Subject <http://www.w3.org/2004/02/skos/core#externalID> 'P021'
    )
    from <rmi://livlbl/server1#sampledata>
    where
    $Subject <http://www.w3.org/2004/02/skos/core#prefLabel> $Object
    and $Subject $Predicate $Object;
```

Against iTQL is that it is coupled with the Mulgara server software, and therefore iTQL can only be used to query ontologies that are served from that engine. Also, it appears that if all concepts from a thesaurus of multiple controlled vocabularies are loaded into one table, the query language relies on the concepts containing a string value (not a URI) which can be used to specify the vocabulary they belong to.

3.6 Recommendations

Due to the ubiquitous nature of support for it and its high level of extensibility, SPARQL is the recommended query language is the recommended choice of query language from this review. However, if the Mulgara server is chosen, then iTQL remains an excellent option for ontology query language, especially with its ability to provide updates to concept databases as a standard feature of the language.

4 Ontology Editors

4.1 Introduction

In this section, and those that follow, the tools under discussion have been tested using vocabularies from the NERC Data Grid vocabulary server (http://vocab.ndg.nerc.ac.uk/), in particular the P0x vocabularies, paying close attention to the BODC P011 Parameter Usage Vocabulary (http://vocab.ndg.nerc.ac.uk/list/P011/current) and the SeaDataNet P021 Parameter Discovery Vocabulary (http://vocab.ndg.nerc.ac.uk/list/P021/current).

4.2 SWOOP

SWOOP is a free Java tool which is designed for browsing and editing ontologies in RDF and OWL. Under test, it loads small vocabularies and ontologies quickly, and loading a 20,000+ term vocabulary does not crash the software. However, the navigation of the concepts in the ontology is difficult as all the concepts are concatenated into one long list. This does improve dramatically once a concept has been located as it is possible to browse through the ontology in the main window.

The SWOOP editor can be downloaded from http://code.google.com/p/swoop/

4.3 Hozo

Hozo is a graphical ontology editor, again written in Java. As it does not support SKOS documents, it was only tested against the example OWL files with which it is shipped. Due to the very graphical nature of the editor, it seems to be quite slow and it is difficult to navigate through the concept maps which Hozo creates. In favour of Hozo, the developers provide an application programming interface (API) which allows third parties to develop applications over the top of the Hozo core.

The Hozo editor can be downloaded from http://www.hozo.jp/

4.4 CmapTools Ontology Editor

The CmapTools Ontology Editor (COE) is another Java application, which takes a very graphical approach to creating and editing ontologies. By mapping the ontology visually, the user is able to build up the ontology quickly. COE allows the addition of links between concepts within the ontology and the addition of properties to the concepts. Finally, when the ontology is ready for publication, it can be exported to an OWL file.

However, once the ontology is highly populated with many links between the concepts it represents, the maps become difficult to follow and the information too dense to be easily understood as a map.

The CmapTools Ontology Editor can be downloaded from http://coe.ihmc.us/groups/coe/

4.5 TopBraid Composer

The TopBraid Composer software is available in both commercial and free flavours. The free version was tested for this review.

In practice the software loads up reasonably quickly, but it does not display SKOS knowledge representations, which the vocabularies hosted by the British Oceanographic Data Centre (BODC) are currently rendered in. A nice touch within TopBraid is that it incorporates a SPARQL editor, so it is possible to easily design and test queries of RDF documents.

The TopBraid Composer can be downloaded from http://www.topquadrant.com/products/TB_Composer.html

4.6 Protégé

Protégé is free, developed in Java and designed to be an extensible platform for editing ontologies. In practice, Protégé is very slow when running on a standard desktop PC and using large RDF documents. Indeed, one of the standard vocabularies to be used in the NETMAR project simply refused to load into Protégé on a BODC desktop machine.

A SKOS editor plug in for Protégé also exists (SKOSEd, available for download from http://code.google.com/p/skoseditor/) but it appears to have been out of development since January 2009, and suffers the same problems as the Protégé base product when attempting to load large vocabularies.

The Protégé editor can be downloaded from http://protege.stanford.edu/

4.7 ThManager

ThManager is a Java utility for the viewing and editing of controlled vocabularies represented in SKOS. It provides an easy interface for navigating through a hierarchy of concepts. The editor is also easy to use, providing a good way of changing concept attributes and relations. However, this mode would be easier to use if the hierarchy was shown in the concept navigation window in editor mode.

Against ThManager is the fact that it appears not to have been updated since 2007.

The ThManager can be downloaded from http://thmanager.sourceforge.net/

4.8 SKOS Validation Service

Although not an editor, this is a useful tool for checking the validity of edits to SKOS documents. The experimental SKOS validation service can be accessed at http://www.w3.org/2004/02/skos/validation. Although not a fast service (a 2500 statement vocabulary took 40 seconds to check), it seems robust for what is advertised as being an experimental service. The validation service operates in both a basic integrity test case mode and a thesaurus compatibility test case mode. In the basic integrity test case the integrity of both the semantic relations and the labeling of the concepts are tested. The thesaurus compatibility test case adds testing of controlled vocabulary labeling and the structure of the SKOS scheme to these basic integrity tests.

4.9 SQL

For ontologies where the XML representation is generated dynamically at the time of request or on a regular basis, the conceptual data of the ontology may be stored in a relational database. If this is the case, then SQL can be used to manage the ontology. SQL is a good tool for this sort of work as it allows updates to be made quickly and easily and is designed to be comfortable with very large datasets. The downside is that, even with a modern tool such as SQLDeveloper, there is very little in the way of visual input to SQL.

4.10 NERC Vocabulary Editor

The NERC Vocabulary editor is currently only a test release available to BODC staff. It is a web based form which provides authorised users access to vocabulary lists which they then have the ability to edit. List editing functions comprise adding new terms, and modifying or deprecating existing terms. Updates are incorporated into the production version of the vocabulary list over night, and the new list version is published on the server. This system is an excellent way of maintaining vocabulary lists, but currently it is difficult to navigate through long lists as there is no filtering or sorting of lists available. A search from the browser's 'find' function is currently the best means of achieving intra-list navigation.

4.11 Semantic Turkey

Semantic Turkey is a platform for ontology development and semantic bookmarking developed at the University of Rome. Rather than being a standalone program, it is an extension for the Mozilla foundations Firefox web browser. It loads quickly, as it is simply a Firefox extension, and provides a good mechanism for browsing and editing ontologies. There is also a SPARQL editor and query engine which works well. However, as with the majority of the dedicated ontology tools, it seems to fall over when faced with a 20,000+ term vocabulary.

Semantic Turkey is free, and the code is open source. http://semanticturkey.uniroma2.it/

4.12 PoolParty

PoolParty is a web based ontology manager that acts as a central hub for the organization of knowledge concepts. It has facilities for the creation and editing of SKOS based vocabularies and provides an automatic SPARQL endpoint to the concepts held as well as HyperText Markup Language (HTML) form based editing. The interface has been designed to be accessible to users from a non computer science background.

The software provides access to a vocabulary through various means: alphabetically; visually; and through a search interface. This allows for crosswalk style searching when a user editing the vocabulary does not know the exact term they are looking for but does know the rough concept domain.

Thesauri may be imported into and exported from a PoolParty project from one of many formats, including RDF and comma-separated value files.

As this is a commercial product there are licensing considerations. Prices quoted in March 2010 are 1500 per month for use as a cloud computing service; 16,000 for an in-house server installation license; 4,800 per annum for maintenance of a server installation.

Overall this software provides a feature rich interface for the creation and editing of SKOS based vocabularies, but the licensing costs are prohibitively expensive.

Pool Party can be trialled at http://poolparty.punkt.at/

4.13 Recommendations

If a specific piece of software is required by NETMAR to act as an ontology editor, Semantic Turkey would be recommended due to its seemingly good levels of scalability, and the fact that it simply drops into Firefox, rather than being a large standalone application. It is also free, which is a positive compared with some of the other ontology editors available. However, none of the dedicated ontology editors are definitely scalable enough for the sheer volume of concepts and mappings that will be required by NETMAR. Therefore, an editing tool based on either SQL (either directly into the underlying database, or such as the NERC Data Grid Vocabulary Editor) or an ontology query language (iTQL is the only one which allows the update of ontologies as standard) is preferable.

5 Text to RDF Converters

5.1 *VOC2RDF*

MMI provides the VOC2RDF web application. It takes a vocabulary in an ASCII format and produces an RDF file of the results. This service also provides an ontology browser which is relatively easy to navigate. It seems, however, that the vocabulary then becomes published on the MMI site which may lead to multiple versions of the vocabulary existing on the web.

5.2 *Terminizer*

The Terminizer software was developed from the OBO Foundry and takes a text or URI input and attempts to map terms from within the text to concepts within the ontologies within the software's knowledge base. While the term mapping is obviously highly skewed towards the biological and medical community, it does include several ontologies of interest to the environmental science community, and the mapping is particularly effective with these ontologies. Term mappings can be rejected or accepted through a web form interface, and the resulting mappings can be exported as example RDF (not standards compliant) or raw server response XML.

The Terminizer can be found at http://terminizer.org/

5.3 *Recommendations*

The existing tools for converting text to RDF are not strongly suited to the work of the NETMAR project, and therefore if this sort of tooling is deemed necessary it should be developed from within the project.

6 Concept Mapping / Ontology Bridging Tools

6.1 CmapTools Ontology Editor

COE, described in section 4.4, is an excellent tool for quickly mapping concepts in a graphical environment. The visual nature of this software is illustrated in Figure 6-1, which shows an ontology of geological terms in development. COE has the benefit of being able to export the concept maps directly into OWL.

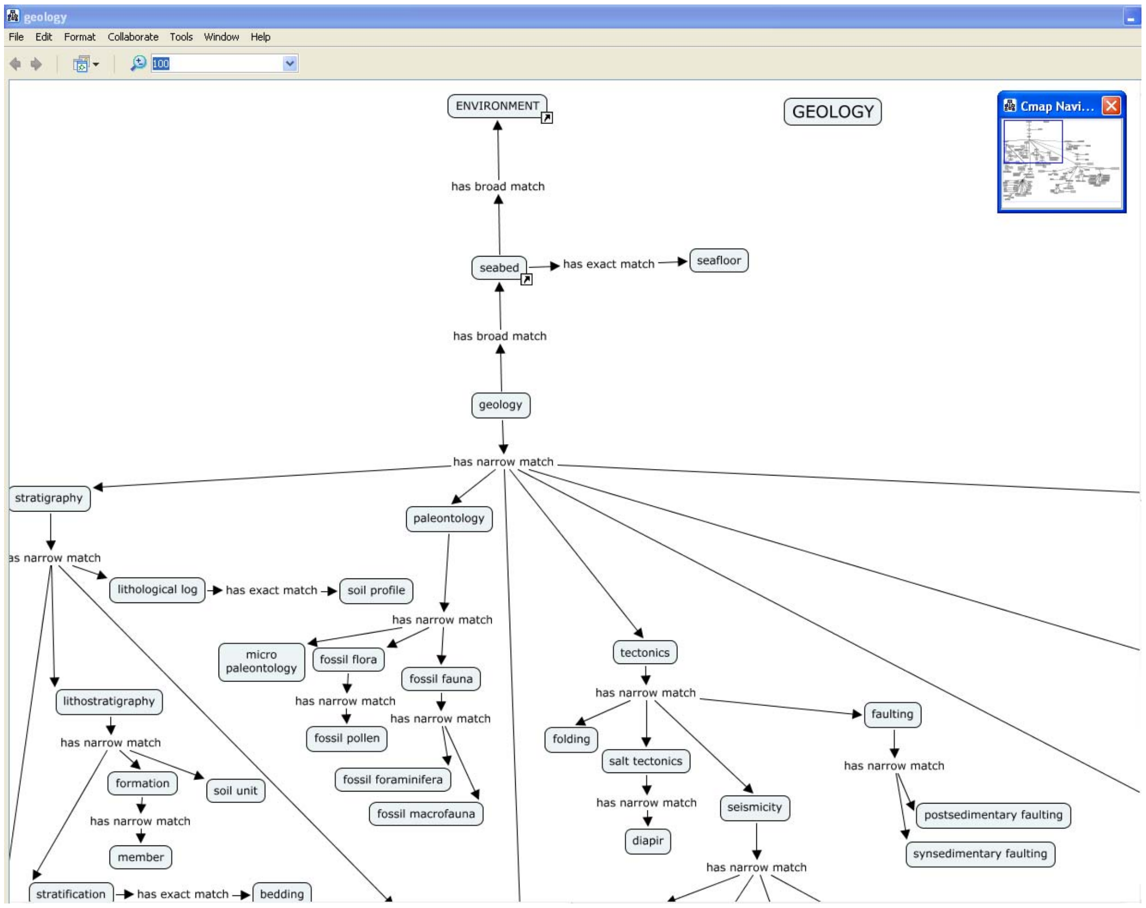


Figure 6-1 Screenshot from the CmapTools Ontology Editor.

6.2 Bubbl.us

Bubbl.us is a free web application for creating concept maps. It allows quick and easy mapping of concepts and relations between concepts, but it does lack the ability to export the concept maps as anything other than Joint Picture Experts Group (JPEG) or Portable Network Graphics (PNG) format images.

The Bubbl.us website is http://www.bubbl.us/ and https://bubbl.us/beta/

6.3 FreeMind

FreeMind is a free Java application which allows the creation of concept maps, including the hierarchy and the linking of concepts. FreeMind allows the export of concept maps via an Extensible Stylesheet Language Transformation (XSLT) which gives the possibility of creating a graphical concept map, and using it to produce an RDF document.

FreeMind can be downloaded from http://freemind.sourceforge.net/wiki/index.php/Main_Page

6.4 Terminizer

The Terminizer text to RDF tool described above searches a range of ontologies to produce its mappings of terms. This can therefore allow the exported RDF or XML from a Terminizer mapping session to build bridges between different ontologies.

6.5 Ontology MApping FRAmework Toolkit (MAFRA)

MAFRA is a standalone Java application which allows the creation of semantic relations between two (source and target) ontologies. In practice, MAFRA is not an intuitive tool to use. There is little in the way of help within the application itself, and the functions available within the menus and context menus are not self explanatory (Figure 6-2). The online documentation is also quite sparse.

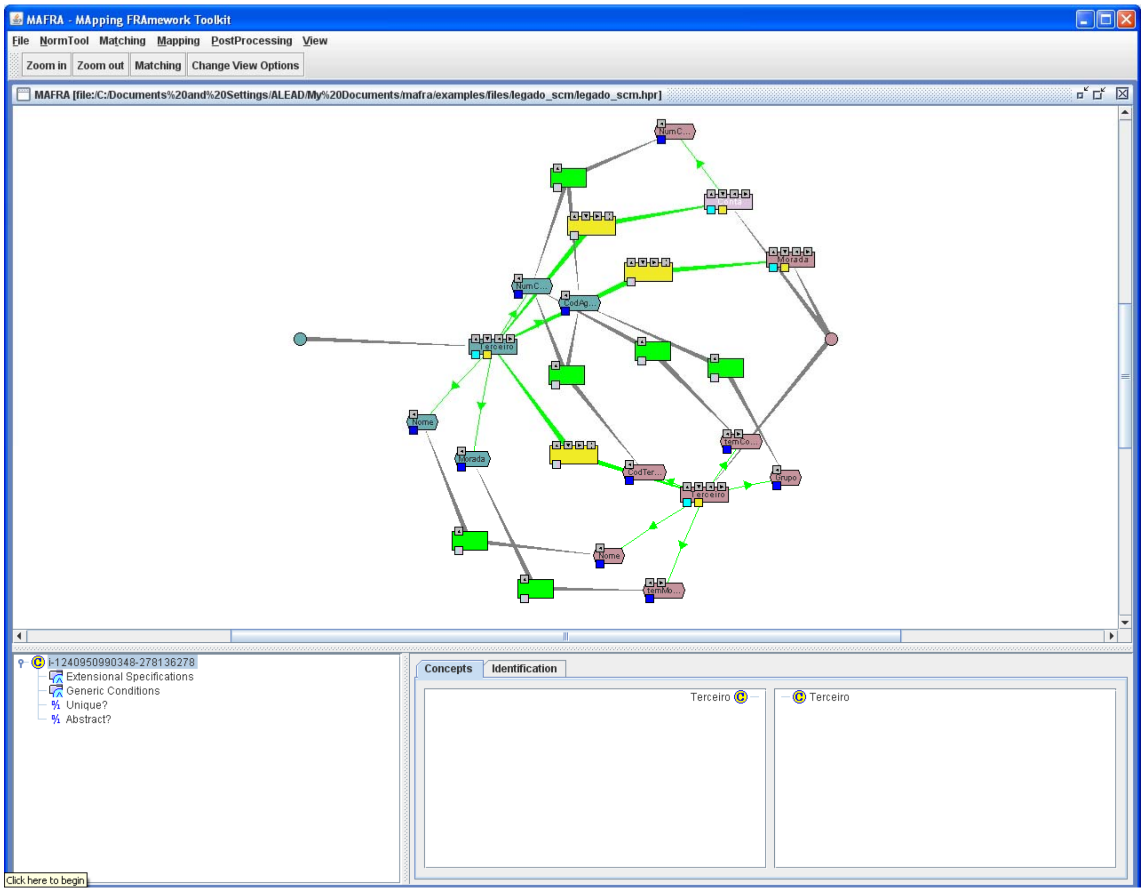


Figure 6-2 Screenshot from the Ontology Mapping FRAmework Toolkit.

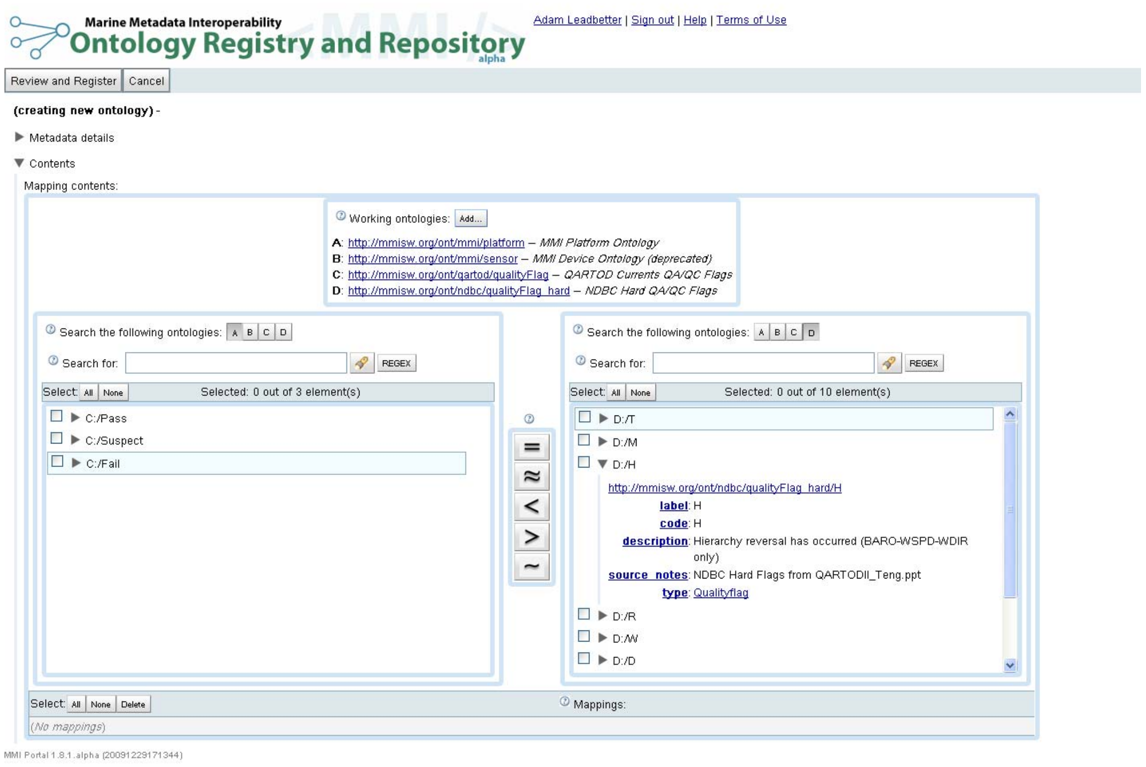
On the positive side, MAFRA allows the user to export the graphically created ontology bridge to RDF or RDFS, but as this is possible in other tools with a shallower learning curve this may not be a great benefit.

The MAFRA software can be found at http://mafra-toolkit.sourceforge.net/

6.6 Vocabulary Integration Environment (VINE)

VINE is a product of the MMI project. It began as a piece of standalone software, but has since been developed as a web application which supports other tools in the MMI semantic framework and integrates with MMI's ontology registry and repository (ORR). As such, VINE can be accessed at http://mmisw.org/orr. In order to use VINE, the user must create an MMI ORR account and log in.

In contrast to most other ontology bridging tools, VINE does not take a visual approach to displaying the ontologies being linked, instead listing the concepts from the ontologies in a hierarchical lists (Figure 6-3). While this is not as intuitive as having the concepts mapped in a visual manner, it does allow the simple searching of vocabularies and ontologies for desired concepts, including through the use of regular expressions which is particularly useful for large ontologies where graphical concept maps can become very difficult to navigate.


Marine Metadata Interoperability
Ontology Registry and Repository
alpha
Adam Leadbetter | Sign out | Help | Terms of Use
Review and Register
Cancel
Metadata details
Contents
Mapping contents:
Working ontologies:
Add...
Search the following ontologies: A B C D
Search for:
REGEX
Select: All None
Selected: 0 out of 3 element(s)
Selected: 0 out of 10 element(s)
=
≈
<
>
~
Select: All None Delete
Mappings:
MMI Portal 1.8.1.alpha (20091229171344)

(creating new ontology) -

A: http://mmisw.org/ont/mmi/platform – *MMI Platform Ontology*
B: http://mmisw.org/ont/mmi/sensor – *MMI Device Ontology (deprecated)*
C: http://mmisw.org/ont/qartod/qualityFlag – *QARTOD Currents QA/QC Flags*
D: http://mmisw.org/ont/ndbc/qualityFlag_hard – *NDBC Hard QA/QC Flags*

C:/Pass
C:/Suspect
C:/Fail

D:/T
D:/M
D:/H
http://mmisw.org/ont/ndbc/qualityFlag_hard/H
label: H
code: H
description: Hierarchy reversal has occurred (BARO-WSPD-WDIR only)
source_notes: NDBC Hard Flags from QARTODII_Teng.ppt
type: Qualityflag
D:/R
D:/W
D:/D

(No mappings)

Figure 6-3 Screenshot from the Vocabulary Integration Environment.

Mappings produced in VINE are stored in a new ontology, which can be served from the MMI ORR. This new ontology can contain concepts which are hosted both on the MMI ORR and on other servers, which is what is required to produce a truly distributed ontology. Equally, new mapping ontologies can be created in other tools, outside of VINE, and uploaded to be hosted and served by the MMI ORR.

6.7 Hypermedia Service

Hypermedia services or systems store and manage HTML hyperlinks separately from the documents in which they are referenced, meaning that they can be stored, transported, shared and searched separately from the document itself. The University of Southampton developed a Distributed Link Service (DLS) on this idea. This concept has been extended to cover ontological services, including mappings between concepts and synonyms, where the DLS uses ontology concepts and relationships to find terms from within a document [BSL05]. The reference implementation of the hypermedia service, Conceptual Open Hypermedia Services Environment (COHSE), is no longer being developed, but the philosophy employed lives on in VINE where links between ontologies are stored in a separate ontology.

6.8 PROMPT

PROMPT is a Protégé plug in for managing and mapping multiple vocabularies. When extending ontologies to include terms from another vocabulary or ontology, PROMPT will initially attempt to assist the user by generating a suggested list of mappings based on the class names of the two documents. New classes can then be created based on PROMPT's suggestions which merge the concepts from the two ontologies. However, as with the Protégé base product, there are scalability issues when using PROMPT with large vocabularies and to manage the complex mapping between them. PROMPT also appears to have been out of development since June 2005.

PROMPT is available for download from http://protege.stanford.edu/plugins/prompt/prompt.html.

6.9 Recommendations

Due to its ability to export visual concept maps to OWL documents, the CMAPTools Ontology Editor should be the software of choice for concept mapping exercises. Very little exists in terms of tooling for the bridging of ontologies, and as this is a key component of the NETMAR ontology concept, this is an area which must be developed from within the project, perhaps building on the technologies available in the CMAPTools Ontology Editor and VINE.

7 Toolchaining of ontologies

After researching this topic, it appears that no work has been done on creating toolchains for ontologies.

8 Ontology Servers

8.1 Kowari

Kowari is an Open Source, massively scalable, transaction-safe, purpose built database for the storage, retrieval and analysis of metadata. Kowari is written in Java and licensed under the Mozilla Public license. Kowari includes native support for RDF, the ability to have multiple databases per server and support for the interactive Tucana Query Language [Km04]. Currently there is no support for the W3C SPARQL standard but this is to be supported in future releases. A Jena interface was removed in the latest version and it is thought that once the SPARQL query support is added this would be the method for applications querying the RDF store.

There is support for the JRDF java library [Km04] which should provide all the functionality common with other frameworks (Jena, Sesame) such as querying and editing an RDF store. The biggest plus point for Kowari is the scalability of the storage. The backend storage is optimised for retrieval and includes multi-processor support, low memory requirements and can be tuned for either 64-bit or 32-bit architectures

Active development has stopped on Kowari due to copyright issues with the new owner Northrop Grumman. The project was forked into Mulgara and released under the Open Software License, this resolved the copyright issues and development on the Mulgara fork is ongoing [TP07].

8.2 Mulgara

Mulgara is a purely Java triple store database. It is open source, scalable and transaction safe. The storage works by creating Lucene indexes [Mulg] of the data which is an economical (both space and memory) way of storing and querying the data. The triples can be queried using a fully fledged SPARQL engine, which includes many additional functions over the standard. There is also the ability to query the store using iTQL, a bespoke query language originally designed for use on the commercial wing of this project that has a structure very similar to SQL.

During initial testing there were considerable delays when loading data in (~35mins for 200,000 triples) however once the system is stable querying the complete set with filters is extremely quick (almost instantaneous querying of the same triple graph).

8.3 Sesame RDF Framework

Sesame is an open source Java framework for the storage and querying of RDF data. The framework provides a fully extensible and configurable environment. It also offers a JDBC-like users API, streamlined system API and a Representational State Transfer (RESTful) HyperText Transfer Protocol (HTTP) interface supporting the SPARQL Protocol for querying RDF. The RESTful interface are only found on version 2.x which is *not compatible* with earlier releases of Sesame (i.e. 1.x).

Sesame has a flexible backend system. It can be deployed on top of a variety of storage mechanisms (relational database, in-memory, file systems, etc). Sesame makes a selection of query languages available to interrogate the RDF store, of those available (including SPARQL) the creators recommend using Sesame RDF Query Language (SeRQL) [Se09] as they claim it is the most powerful). All of this functionality (editing stored RDF and querying) is provided through Java libraries that are part of Sesame but can be utilised independently.

Currently the only relational databases supported are MySQL and PostgreSQL. Sesame supports two schemas, either a large 'monolithic' schema with all statements in a single table or a vertical schema that stores statements in a per-predicate table. In theory, there is no difference in performance between these schema (in an Oracle environment, at least) just differences in the data modelling approach.

8.4 4store

4store (http://4store.org/) is a custom RDF triple storage system. It utilises it own database back end and provides a SPARQL endpoint using the standard HTTP query interface. The software has been designed to run on a cluster system but will run equally well on a single machine if required. The design is such that it can be scaled hugely with some current implementations storing 15,000,000,000 triples.

The system allows for insert/update over HTTP from RDF files. This would allow multiple partners to load in RDF data from their local machines.

There are various client libraries in multiple languages (PHP, Ruby, Python, and Java) that allow access to most of the features. For instance the Java API allows querying of the store, creating a model within the store, adding data to a model/graph and deleting a model or graph.

From the 4store website:

"4store is written in American National Standards Institute (ANSI) C, and is designed to run on UNIX-like systems.

4store is optimised to run on shared–nothing clusters of up to 32 nodes, linked with gigabit Ethernet. However, it will also work on single machines, if your data requirements are not large."

Depending on the data requirements of NETMAR this may be a disadvantage unless access to such a cluster system is available. Overall the 4store system provides a stable, scalable and accessible triple store. Through the use of the various API's many client implementations could be built or a client could be built by using simple HTTP connection methods, which are available for all programming languages.

8.5 Talis Platform

Talis Platform (http://www.talis.com/platform/) is a hosted scalable data storage tool, designed as a Software as a Service architecture. As an RDF triplestore all of the standard features are included: HTTP methods for adding data to a store; updating data; and querying it using SPARQL. There are clear advantages of using a hosted service: in particular scalability and security issues are handled externally. The data a user stores on the Talis Platform can either be world-readable with edits by authorised users; or it can be entirely secured behind an access control system.

The service is primarily designed for public access data, the only rule being that data are licensed under either the Open Data Commons Public Dedication and License[1] or Creative Commons License[2]. There is a provision to have privately held data but at the time of writing no costing information was available.

[1] http://www.opendatacommons.org/licenses/pddl/1.0/

[2] http://creativecommons.org/choose/zero

8.6 OpenLink Virtuoso

OpenLink Virtuoso (http://virtuoso.openlinksw.com) provides a selection of data access, integration and management tools all packaged as one server, a so called Universal Server. This includes a “Hybrid Data Server for Relational, RDF-Graph, and Document (Full Text) data management” that would be of relevance to the NETMAR project. The RDF store capabilities include a full implementation of a SPARQL endpoint that can return results in a variety of formats, including JSON and XML.

However there are many other features that would not be used by the project, such as “Social Media enhanced Distributed Collaboration Services for effectively integrating Blogs, Wikis, Bookmarks, Feeds, Discussion Forums etc.”

8.7 Recommendations

Of the servers reviewed here the Mulgara server has the easiest mechanisms for entering data and then querying it, with a simple HTTP interface to a SPARQL & iTQL endpoint. However it does not use a relational database to store the data. If this is a requirement then Sesame is probably the best option, although if a cluster-based server was available then 4store would be recommended.

9 Ontology Frameworks and API's

9.1 Jena

Jena is a Java based framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS, OWL, and SPARQL it also includes a rule based inference engine. Jena is open source and is grown out of work with the HP Labs Semantic Web Programme [Jena].
Jena has two subsystems available for persisting RDF and OWL these are, the SQL database triple store (SDB) and the high performance, non-transactional RDF store (TDB).

- SDB acts as a wrapper around existing relational databases and provides either command line access or access through the Jena API's
- TDB is a storage and access solution in one. It supports the full suite of Jena API's and is the preferred storage method for use with the Jena framework. This is due to the scalability and the ease of setup compared to SDB

Jena offers a modified version of SPARQL called ARQ [Arq] this provides the full SPARQL specification for querying and adds onto it extra functionality through the use of XSLT functions. These functions are provided as standard, there is also the availability to create and use your own functions (for instance fn:substring('string', 3, 3) would return a 3 character substring of 'string' starting at character 3). ARQ also allows the use of advanced syntax such as LET (used to assign values to variables).

Jena is shipped with a complete Ontology API that provides methods/classes for creating, editing and querying ontologies.

9.2 Simple Ontology Framework API (SOFA)

SOFA is a simple but powerful ontology API that allows for inter-operation between several different ontology description formats [Sofa]. Additionally, SOFA is not tied down to a particular storage layer and can easily be integrated into any application that requires an ontology manager. Due to the structure of the API, virtually any Java object can be used to model ontology data type nodes, allowing the model to be as complex or simple as necessary. Features of the API include:

- Multiple inheritance, allowing the discovery of nodes beyond the first set of sub, or super-concepts
- Ontology inter-operation, so two ontologies in the same session can talk to each other and use the same resources
- Inferencing and reasoning about relationships
- Support for W3C OWL, Defense Advanced Research Projects Agency (DARPA) Agent Markup Language (DAML) and Ontology Interchange Language (OIL), and RDF and RDFS
- Ontology creation and querying

9.3 JRDF

JRDF is an attempt to create a standard set of APIs and base implementation to RDF using the latest version of the Java language [Jrdf]. JRDF provides the following features:

- "A Graph API (including graph comparison and graph set-based operations),
- Creating and manipulating Graph objects (Statements, Resources, Nodes, etc.),
- In memory and disk based graphs with a standard system level interface for storing triples,

- Inversion of Control support (currently using Spring 2),
- RDF Datatypes,
- Local (where nodes are tied to a graph/store) and global (where they are not) RDF statements
- Query Handling including SPARQL support (including results, transport, etc)”

The current state of the project is ‘in development’ and after some testing with the tools it is clear that they are not fully useable yet, for instance the SPARQL query engine does not take the OPTIONAL or LET keywords that other more advanced/stable query engines do. However the server technology seems quite good and provides an easy web-form mechanism, to create new ontologies and repositories that can then be queried extremely easily.

9.4 RDF API for PHP:HyperText Pre-processor (PHP) (RAP)

RAP is a Semantic Web toolkit for PHP developers. RAP started as an open source project at the Freie Universität Berlin in 2002 and has been extended with internal and external code contributions since then [Obw05]. Its latest release includes:

- A statement-centric API for manipulating RDF graphs as a set of statements;
- A resource-centric API for manipulating RDF graphs as a set of resources;
- Integrated RDF/XML, N3 and N-TRIPLE parsers;
- Integrated RDF/XML, N3 and N-TRIPLE serializers;
- In-memory or database model storage;
- Support for the RDQL query language;
- An inference engine supporting RDF-Schema reasoning and some OWL entailments;
- An RDF server providing similar functionality as the Joseki RDF server;
- A graphical user-interface for managing database-backed RDF models;
- Support for common vocabularies.

RAP offers two different programming interfaces for manipulating RDF graphs: The statement-centric Model API which allows you to manipulate an RDF graph as a set of statements; and the resource-centric ResModel API for manipulating an RDF graph as a set of resources.

The Model API supports adding, deleting, and replacing statements inside a model as well as adding entire models. Statement iterators allow sequential access to all statements within a model. However, RAP is reported to be no longer in active development.

9.5 OWL API

The University of Manchester OWL API is a reference implementation for creating, manipulating and serialising OWL ontologies written in Java.

The components listed on the website include:

- an API for OWL 2
- an efficient in-memory reference implementation RDF parser and writer
- OWL parser and writer
- OWL Functional Syntax parser and writer
- OBO Flat file format parser
- reasoner interfaces for several software products

The main use of the API would be in the production of XML representations of ontologies. It could be utilised by a tool for creating and editing OWL 2 based ontologies.

The OWL API is open source and is available under the Lesser General Public License, and is available from http://owlapi.sourceforge.net/.

9.6 Marine Metadata Interoperability (MMI) project semantic framework

MMI provide a ‘semantic framework’, part of which is the VINE ontology bridging tool discussed above. The MMI framework allows users to view and retrieve ontologies, make queries using the SPARQL ontology query language and to produce mappings between ontologies. It, however, does not allow users to programmatically interact with it in the sense of a true software framework or API.

9.7 Recommendations

Jena is a powerful complete ontology framework which provides two mechanisms for storing data, one as a wrapper to a relational database the other using a bespoke system similar to the Mulgara server. The querying mechanism is an extension of SPARQL called ARQ which provides access to XSLT functions that make complex queries possible. Jena also provides classes for creating and editing ontologies programmatically. As such, Jena is the recommended ontology framework / API for the NETMAR project.

10 Ontology web-browsers

10.1 jOWL

jOWL is a jQuery plug-in for navigating and visualising OWL-RDFS documents. The application loads an ontology from a document either stored locally or fetched from a URI. The ontology is then displayed in a tree structure with the user able to drill down through clicking of branches/elements of the ontology.

There are no editing features included in the application it is purely for display purposes. Displaying an ontology in a tree view makes it easy to navigate and follow related terms. However the system can be a bit slow and unresponsive when large lists (20,000+ terms) are loaded in.

jOWL can be downloaded from http://jowl.ontologyonline.org/

10.2 Flex ontology browser

This is currently only a proof of concept browser, testing the ability of Adobe Flash at visualising large ontologies. It is an extremely attractive offering. The ontology is rendered in a dynamic web display with the user able to drill down and see related terms by double clicking on a visible term. This type of display is in contrast to the simple tree structure used by the majority of ontology web browsers.

The test site currently does not allow you to load your own lists so the speed of use with a large list could not be tested.

The Flex ontology browser is available at http://labs.rgd.mcw.edu/?q=node/31

10.3 OwlSight

OwlSight is an OWL ontology browser that runs in any modern browser. It has been developed with Google Web Toolkit (GWT) and uses GWT extensions (GWT-Ext) as well as OWL-API (Java). Pellet is used as the OWL reasoner.

This is more of a textual browser instead of a visual browser like some of the browsers mentioned here. This provides a very clean frontend that is quick at rendering large lists. There are no editing features included in the application

OwlSight can be found at http://pellet.owldl.com/ontology-browser/

10.4 SWOOP

SWOOP is a Java web start based ontology browser and editor. It provides basic functionality such as viewing and editing existing ontologies and allows the creation of new ones. The interface is quite user friendly and the application managed fine when loading in multiple large lists. The options for editing are very comprehensive and the application takes care of creating the required changes to the ontology in any of the languages supported. This allows the developer to focus on the ontology and its members without worrying about using the correct syntax.

SWOOP can be downloaded from http://code.google.com/p/swoop/

10.5 Ontology-Browser (Manchester University Computer Science)

The Ontology-Browser allows the user to navigate around an ontology in an environment similar to OWLDoc [Od] pages, however they are generated dynamically on the fly. This creates a click through navigation mechanism that is both intuitive and user friendly. However the application can be slow when dealing with medium/large lists (1000+ terms). The browser only offers a textual/tabular mechanism for viewing the ontology.

The Ontology-Browser is available at http://code.google.com/p/ontology-browser/

10.6 Recommendations

The available tools in this area are numerous, however most are either still in development or are old proof of concept projects that have not been taken any further. jOWL provides a nice user interface and creates a tree like structure from an ontology which allows the user to navigate to more specific terms. However to provide the functionality needed for NETMAR a JavaScript based visual browser should be created.

11 Multilingual Ontology Mapping

"*The domain ontology can be extended to represent the concepts in multiple languages. The translation process has to be done manually, since current translation tools show rather inferior performance and are also quite unlikely to be applicable to specific domains"* [WH98]

11.1 Introduction

Currently there are two schools of thought for the creation of multilingual ontologies [FBO09]: the use of artificial intelligence techniques to automatically align two existing domain ontologies of different language or manually translating existing domain ontologies. The former option is still very much research based with no available implementations. For the purposes of this review we will focus on the available tools and frameworks for producing multilingual ontologies manually.

11.2 Google AJAX Language API

11.2.1 *Introduction*

With the asynchronous JavaScript and XML (AJAX) Language API, it is possible to detect and translate the language of blocks of text within a webpage using JavaScript [Go10]. The language API is designed to be simple and easy to use for the translation and detection of languages on the fly. The API provides methods to translate words entered through a web form interface. This could be utilised as a web based translation tool for creating a multilingual ontology. However it would be very labour intensive as each word would have to be translated individually. There may also be issues with domain specific terms that do not have simple translations or are simply not known by the Google translation engine.

11.2.2 *Using Google AJAX service as a HTTP service*

As the service is simply run as a set of Universal Resource Locators (URLs) secured by a Google API key it could be accessed using any programming language that can make HTTP calls. This makes the service more useful as a fully programmatic interface can be achieved. The calls can be either GET or POST requests, depending on the quantity of content you need translating (GET requests are limited to 2000 characters). Results are returned as JavaScript Object Notation (JSON) objects, for which there are numerous APIs available

11.2.3 *Supported languages:*

The version of the Google AJAX language API available on 21st April 2010 supports the following languages:

- Afrikaans
- Albanian
- Arabic
- Belarusian
- Bulgarian
- Chinese (Simplified and Traditional)
- Catalan
- Croatian
- Czech
- Danish
- Dutch
- English
- Estonian
- Filipino
- Japanese
- Korean
- Latvian
- Lithuanian
- Macedonian
- Malay
- Maltese
- Norwegian
- Persian
- Polish
- Portuguese
- Romanian
- Russian
- Spanish

- Finnish
- French
- Galician
- German
- Greek
- Hebrew
- Hindi
- Hungarian
- Icelandic
- Indonesian
- Irish
- Italian
- Serbian
- Slovak
- Slovenian
- Swahili
- Swedish
- Thai
- Turkish
- Ukrainian
- Vietnamese
- Welsh
- Yiddish

11.3 Microsoft Translator Services

11.3.1 *AJAX Interface*

This is practically the same as the Google AJAX API, allowing translation of text from web pages and web forms through a JavaScript AJAX call [Ms10]. This service suffers from the same limitations as the Google offering, namely the fact that terms would have to be entered through a web interface manually.

11.3.2 *Simple Object Access Protocol (SOAP) Interface*

The SOAP interface provides a strongly typed, web service standards based programming model. However it is focused around a client application written in .NET which is a restrictive platform for development as it is not platform independent.

11.3.3 *HTTP Interface*

The HTTP interface allows applications to integrate translation functionality by invoking HTTP GET & POST methods. The interface is technology agnostic; no particular operating system or programming language is needed to develop using it. The HTTP API uses the same protocols and verbs as the World Wide Web and is simpler than AJAX. This method provides the same level of interoperability as the Google AJAX service used over HTTP. The only distinguishing feature is the slightly smaller list of available languages.

11.3.4 *Supported languages*

The version of the Microsoft Translator Services available on 21st April 2010 supports the following languages:

- Arabic
- Bulgarian
- Chinese (Simplified and Traditional)
- Czech
- Danish
- Dutch
- English
- Finnish
- French
- German
- Greek
- Haitian Creole
- Hebrew
- Hungarian
- Japanese
- Korean
- Lithuanian
- Norwegian
- Polish
- Portuguese
- Romanian
- Russian
- Slovak
- Slovenian
- Spanish
- Swedish
- Thai
- Turkish

- Italian

11.4 Validating Translated Terms

Once a concept has been translated there is also a need to validate the translation, this is due to the fact that certain domain specific concepts may not easily translate using an automated service and as such would need validating by a domain specialist who is also a fluent speaker in the language the term was translated into. I believe using humans to translate all terms will be more labour intensive that simply using a human to validate a translation.

To facilitate this, a group of multilingual domain specialists will need to be assembled, most likely through an email list mechanism. Terms could be discussed once they have been translated and any erroneous translation can be corrected and this fed back into the translation service to improve its future output.

11.5 Recommendations

Previous attempts to build a multilingual ontology have been based around collaborative translation by domain experts who are also native speakers of a range of languages. However, this human translation is very time consuming and at least one of the potential base vocabularies for NETMAR has over 23,000 terms. It is therefore recommended that an approach is followed based on the software translation of terms, followed by native-speaking domain expert validation of a subset of the translated terms. In order to provide a level of confidence in the translation of the terms by software, a measure of the uncertainty in the translation should be provided along with the translated term, perhaps as a value from a binary skill score [SP08] and encoded in UncertML [UL10].

12 Conclusions

Due to the broad compatibility between its members; the fact that it is a recommendation of the World Wide Web Consortium; and the fact that there is a large software base aimed at it, we recommend that the NETMAR project uses the RDF family of languages to represent its ontologies.

Due to the ubiquitous nature of support for it and its high level of extensibility, SPARQL is the recommended query language from this review. However, if the Mulgara server is chosen for ontology publication, then iTQL remains an excellent option for ontology query language, especially with its ability to provide updates to concept databases.

The Mulgara server has the easiest mechanisms for entering data and then querying it, with a simple HTTP interface to a SPARQL and iTQL endpoint. However it does not use a relational database to store the data. If this is a requirement then Sesame is probably the best option.

Jena is a powerful complete ontology framework. It provides two mechanisms for storing data, one as a wrapper to a relational database the other using a bespoke system similar to the Mulgara server. The querying mechanism is an extension of SPARQL called ARQ which provides access to XSLT functions that make complex queries possible. Jena also provides classes for creating and editing ontologies programmatically. Therefore, Jena is the recommended ontology framework for NETMAR.

If a specific piece of software is required by NETMAR to act as an ontology editor, Semantic Turkey would be recommended due to its seemingly good levels of scalability, and the fact that it simply drops into Firefox, rather than being a large standalone application. It is also free, which is a positive compared with some of the other ontology editors available. However, none of the dedicated ontology editors are definitely scalable enough for the sheer volume of concepts and mappings that will be required by NETMAR. Therefore, an editing tool based on either SQL (either directly into the underlying database, or such as the NERC Data Grid Vocabulary Editor) or an ontology query language (iTQL is the only one which allows the update of ontologies as standard) is preferable.

Due to its ability to export visual concept maps to OWL documents, the CMAPTools Ontology Editor should be the software of choice for concept mapping exercises. The technologies used in the Marine Metadata Interoperability project’s Vocabulary Integration Environment (VINE) may be harnessed to build bridges between existing ontologies.

In order to allow cross human language domain concept bridging, multilingual ontologies will be required. This area is still very much research with two main areas of concentration. The first is to create a multilingual thesaurus type ontology but this raises many issues, the other is to use artificial intelligence to determine meaning for grammatical structure. This may not be very appropriate for single word terms. A mixture of these approaches would seem to be the best approach.

In conclusion, it is recommended that an RDF based ontology be developed for use by the NETMAR project. This ontology should be queried by SPARQL and served using the Mulgara server or the Jena framework, unless a cluster server is available, in which case 4store is recommended. Editor tools such as Semantic Turkey or a query language web interface should be used and to bridge or extend existing ontologies the CMAPTools Ontology Editor and MMI VINE software should be used. Tooling will either have to be developed within the project for other semantic requirements (if feasible within resources constraints) or the level of semantic functionality matched to those tools which are available.

13 References

[Arq] ARQ – Documentation and Resources. Retrieved April 6, 2010, from http://openjena.org/ARQ/documentation.html

[BG04] Brickley, Dan and Guha, R.V. RDF Vocabulary Description Language 1.0: RDF Schema. [Online] February 10, 2004. [Cited: May 12, 2010.] http://www.w3.org/TR/rdf-schema/

[BSL05] Bechhoffer, S.K., Stevens, R.D. and Lord, P.W., 2005. Ontology driven dynamic linking of biology resources. Pacific Symposium on Biocomputing 10:79-90.

[FBO09] Fu, Bo, Brennan, Rob and O'Sullivan, Declan, 2009. Multilingual Ontology Mapping: Challenges and a Proposed Framework. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour. Retrieved April 21, 2010 from http://hdl.handle.net/2262/30727.

[FHH03] Fikes, Richard, Hayes, Patrick and Horrocks, Ian, 2003. OWL-QL – A Language for Deductive Query Answering on the Semantic Web. Knowledge Systems Laboratory, Stanford University. Stanford, California. Technical Report. 03-14.

[Ga07] Gandon, Fabian. Sparql In A Nutshell. Slideshare. [Online] 2007. [Cited: March 17, 2010.] http://www.slideshare.net/fabien_gandon/sparql-in-a-nutshell.

[Go10] Google AJAX Language API (2010), Retrieved April 21, 2010, from http://code.google.com/apis/ajaxlanguage

[He10] Herman, Ivan. Resource Description Framework (RDF) – Semantic Web Standards. World Wide Web Consortium [Online] March 7, 2010. [Cited: May 12, 2010.] http://www.w3.org/RDF/

[HMW04] Haarslev, Volker, Moller, Ralf and Wessel, Michael, 2004. Querying the Semantic Web with Racer + nRQL. Ulm, Germany. Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04).

[HKP09] Hitzler, Pascal, Krotzsch, Markus, Parsia, Bijan, Patel-Schneider, Peter F. and Rudolph, Sebastien. World Wide Web Consortium. [Online] October 27, 2009. [Cited: May 12, 2010.] http://www.w3.org/TR/2009/REC-owl2-primer-20091027/

[Iso07] International Organization for Standardization. ISO/IEC 24707:2007 Information technology - Common Logic (CL): a framework for a family of logic-based languages. International Organization for Standardization. [Online] International Organization for Standardization, September 25, 2007. [Cited: March 11, 2010.] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39175

[Jena] Jena – A Semantic Web Framework for Java. Retrieved April 6, 2010 from http://jena.sourceforge.net/index/html

[Jrdf] JRDF – An RDF Library in Java. Retrieved April 6, 2010 from http://jrdf.sourceforge.net/index.html

[Km04] Tucana Technologies (2004) Documentation of system. Retrieved April 7, 2010, from http://kowari.sourceforge.net/oldsite

[MB09] Miles, Alister and Bechhofer, Sean. SKOS Simple Knowledge Organization System Reference. World Wide Web Consortium. [Online] August 18, 2009. [Cited: March 25, 2010.] http://www.w3.org/TR/2009/REC-skos-reference-20090818/.

[MvH04] McGuinness, Deborah L. and van Harmelen, Frank. OWL Web Ontology Language Overview. World Wide Web Consortium. [Online] February 10, 2004. [Cited: May 12, 2010.] http://www.w3.org/TR/owl-features/

[Ms10] Microsoft Translator Tools, Retrieved April 21, 2010 from http://www.microsofttranslator.com/tools

[Mu09] Mungall, C. OBO Format and Common Logic. Slideshare. [Online] July 2009. [Cited: March 15, 2010.] http://www.slideshare.net/cmungall/obo-and-common-logic.

[Mulg] Mulgara Overview (2007) Overview of Mulgara Semantic Store. Retrieved April 6, 2010, from http://docs.mulgara.org/overview/index.html

[Obw05] Bizer, Oldakowski and Westphal, 2005. RAP: RDF API for PHP. Retrieved April 6, 2010, from http://www.semanticscripting.org/SFSW2005/papers/Oldakowski-RAP.pdf

[Od] OWL-Doc. http://www.co-ode.org/downloads/owldoc/

[OR04] Oldakowski, Radoslaw. RDQL Tutorial. Software Environment for the Advancement of Scholarly Research (SEASR). [Online] October 2004. [Cited: March 11, 2010.] http://seasr.org/wp-content/plugins/meandre/rdfapi-php/doc/tutorial/rdql_tutorial.htm.

[PS08] Prud'hommeaux, Eric and Seaborne, Andy. SPARQL Query Language for RDF. World Wide Web Consortium. [Online] January 15, 2008. [Cited: March 10, 2010.] http://www.w3.org/TR/rdf-sparql-query/.

[Se04] Seaborne, Andy. RDQL - A Query Language for RDF. World Wide Web Consortium. [Online] January 9, 2004. [Cited: March 11, 2010.] http://www.w3.org/Submission/RDQL/.

[Se09] openRDF.org (2009) Overview of Sesame. Retrieved April 6, 2010, from http://www.openrdf.org/about.jsp

[So05] Alishevskikh and Subbaih, 2005. SOFA Design Whitepaper. Retrieved April 7, 2010 from http://sofa.projects.semwebcentral.org/doc/sofa.pdf

[SP08] Sohn, Keon Tae and Park, Sun Min (2008). Guidance on the choice of threshold for binary forecast modelling. Advances in Amospheric Sciences 25(1): 83-88.

[Tp07] TopazProject Ticket #26 Kowari Legal Status (2007). Retrieved April 7, 2010, from http://www.topazproject.org/trac/ticket/26

[UL10] UncertML (2010) UncertML: describing and exchanging uncertainty. [Online]. [Cited: May 12, 2010.] http://www.uncertml.org/

[We06] Wessel, Michael. nRQL Tutorial. Institute for Software Systems, Hamburg University of Technology. [Online] 2006. [Cited: March 10, 2010.] http://www.sts.tu-harburg.de/people/mi.wessel/papers/nRQL-tut3.pdf.

[WH98] Weigand, Hans and Hoppenbrouwers, Stijn (1998). Experiences with a Multilingual Ontology-based Lexicon for News Filtering, dexa, pp.160, 9th International Workshop on Database and Expert Systems Applications (DEXA'98.

[Wi05] Wikipedia. File:Concetmap.gif [Online] January 2005. [Cited: April 20, 2010.] http://en.wikipedia.org/wiki/File:Conceptmap.gif

[Wi10a] Wikipedia. Resource Description Framework. Wikipedia. [Online] February 2010. [Cited: March 9, 2010.] http://en.wikipedia.org/wiki/Resource_Description_Framework.

[Wi10b] Wikipedia. SPARQL. Wikipedia. [Online] February 2010. [Cited: March 10, 2010.] http://en.wikipedia.org/wiki/SPARQL.

Appendices

Appendix A. List of abbreviations

AJAX	Asynchronous JavaScript and XML
ANSI	American National Standards Institute
API	Application Programming Interface
ARQ	Jena specific SPARQL query language
BODC	British Oceanographic Data Centre
CL	Common Logic
COE	CMapTools ontology editor
COHSE	Conceptual Open Hypermedia Services Environment
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
EUMIS	European Marine Information System
FOAF	Friend of a Friend
FOL	First Order Logic
GEMET	General Multilingual Environmental Thesaurus
GWT	Google Web Toolkit
GWT-Ext	Google Web Toolkit Extension
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ISO	International Organization for Standardization
iTQL	Interactive Tucana Query Language
jOWL	JQuery plugin for navigating and visualising OWL-RDMS documents
JPEG	Joint Photographic Experts Group image format
jRDF	An RDF Library in Java
JSON	JavaScript Object Notation
MAFRA	Ontology Mapping Framework Toolkit

MMI	Marine Metadata Interoperability
NERC	Natural Environment Research Council
NETMAR	Open Service Network for Marine Environmental Data
nRQL	New Racer Query Language
OBO	Open Biomedical Ontologies
OGC	Open Geospatial Consortium
OIL	Ontology Inference Layer or Ontology Interchange Language
OpenDAP	Open-source Project for a Network Data Access Protocol
ORR	Ontology Registry and Repository
OWL	Web Ontology Language
OWL-QL	OWL Query Language
OWL DL	OWL Description Logics
OWL EL	OWL Expression Logic profile
OWL QL	OWL Query Language profile
OWL RL	OWL Rule Language profile
PHP	PHP:HyperText Pre-processor
PNG	Portable Network Graphics image format
RAP	RDF API for PHP
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	RDF Data Query Language
REST	Representational State Transfer
SeRQL	Sesame RDF Query Language
SDB	Jena SQL database triplestore
SKOS	*Simple Knowledge Organization System*
SOAP	*Simple Object Access Protocol*
SOFA	Simple Ontology Framework API
SPARQL	SPARQL Protocol and RDF Query Language

SQL	Structured Query Language
TDB	Jena high performance, non-transactional RDF store
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VINE	Vocabulary Integration Environment
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformation