Semantic Web Kick-Off in Finland

Vision, Technologies, Research, and Applications

Eero Hyvönen (editor)

May 19, 2002

HIIT Publications 2002–01

Semantic Web Kick-Off in Finland

Vision, Technologies, Research, and Applications

Eero Hyvönen (editor)

Helsinki Institute for Information Technology (HIIT)
Tammasaarenkatu 3, Helsinki
P.O. Box 9800
FIN-02015 HUT
FINLAND
`http://www.hiit.fi`

HIIT Publications 2002–01
ISBN 1458-951-22-6019-0
ISSN 1458-9451 (printed version)
ISSN 1458-946X (electronic verision)

Authors

Jyrki Haajanen
VTT Information Technology
Tekniikantie 4B
02150 Espoo
FINLAND
jyrki.haajanen@vtt.fi

Petteri Harjula
University of Helsinki
Department of Computer Science
P.O. Box 26
00014 UNIVERSITY OF HELSINKI
FINLAND
petteri.harjula@cs.helsinki.fi

Heikki Helin
Sonera Corporation
Corporate R&D
P.O.Box 970
00051 SONERA
FINLAND heikki.j.helin@sonera.com

Eero Hyvönen
University of Helsinki
Department of Computer Science
P.O. Box 26
00014 UNIVERSITY OF HELSINKI
FINLAND
eero.hyvonen@cs.helsinki.fi

Aki Kivelä
Grip Studios Interactive
Temppelikatu 6 B
00100 Helsinki

FINLAND
aki.kivela@gripstudios.com

Marja-Riitta Koivunen
W3C MIT/LCS
200 Tech. Square
Cambridge, MA 02139
USA
marja@w3.org

Ora Lassila
Nokia Research Center
5 Wayside Road
Burlington MA 01803
USA
ora.lassila@nokia.com

Heimo Laamanen
Sonera Corporation
Corporate R&D
P.O.Box 970
00051 SONERA
FINLAND
heimo.laamanen@sonera.com

Mikko Laukkanen
Sonera Corporation
Corporate R&D
P.O.Box 970
00051 SONERA
FINLAND
mikko.t.laukkanen@sonera.com

Sten Malmlund
University of Helsinki
Department of Computer Science
P.O. Box 26
00014 UNIVERSITY OF HELSINKI
FINLAND
sten.malmlund@helsinki.fi

Eric Miller
W3C MIT/LCS
200 Tech. Square
Cambridge, MA 02139

USA
emiller@w3.org

Vilho Raatikka
University of Helsinki
Department of Computer Science
P.O. Box 26
00014 UNIVERSITY OF HELSINKI
FINLAND
vilho.raatikka@cs.helsinki.fi

Janne Saarela
Profium Ltd
Lars Sonckin kaari 12
02600 Espoo
FINLAND
janne.saarela@profium.com

Karru Salminen
University of Helsinki
Department of Computer Science
P.O. Box 26
00014 UNIVERSITY OF HELSINKI
FINLAND
karru.salminen@cs.helsinki.fi

Mika Seppälä
University of Helsinki and
Florida State University Department of Mathematics
Yliopistonkatu 5
00014 UNIVERSITY OF HELSINKI
FINLAND
mika.seppala@helsinki.fi

Paula Silvonen
VTT Infomation Technology
Tekniikantie 4 B
02150 Espoo
FINLAND
paula.silvonen@vtt.fi

Petri Takala
University of Helsinki
Department of Computer Science
P.O. Box 26

00014 UNIVERSITY OF HELSINKI
FINLAND
petri.takala@cs.helsinki.fi

Santtu Toivonen
VTT Infomation Technology[1]
P. O. Box 1203
02044 VTT
FINLAND
santtu.toivonen@vtt.fi

Kim Viljanen
University of Helsinki
Department of Computer Science
P.O. Box 26
00014 UNIVERSITY OF HELSINKI
FINLAND
kim.viljanen@cs.helsinki.fi

Jouko Väänänen
University of Helsinki
Department of Mathematics
Yliopistonkatu 5
00014 UNIVERSITY OF HELSINKI
FINLAND
jouko.vaananen@helsinki.fi

[1]At the time of writing his article the author was working for Sonera Corporate Research.

Preface

The importance of the Internet is due to the services it provides to us. Services include messaging systems, such as IRC, ICQ, and other chatting systems, email and news groups, FTP and the World Wide Web for publishing and searching data, and more advanced interactive systems, such as electronic market places, public services, peer to peer (P2P) systems, etc. Since the mid 90's, the Internet and its services have been a driving force of the whole IT industry. Lots of resources have been spent, e.g., in trying create "dot.com" applications and in building up 3G networks upon which the future mobile services could based.

In order to serve us in an intelligent fashion, the machine should be able to understand the meaning of the data, messages, and processes it deals with. However, the web contents of today, such as HTML web pages, PDF-documents, images, music files, and software components are difficult to interpret algorithmically, and the protocols and messaging languages in use are primitive. The web is designed for presenting data in a nice form for the humans to use, not for the machines. From the technical viewpoint, this is a fundamental hinder for creating intelligent web services.

The Semantic Web is a vision for making the contents of the Web understandable to the machines. This would create a new basis for implementing intelligent web services in the future. The vision is actively pushed forward by the World Wide Web Consortium (W3C), among others. The idea is attributed to Tim Berners-Lee, the "father" of the WWW and the director of the W3C. In February 2001, W3C launched a special Semantic Web Activity in order to promote and coordinate the development of the next generation Web of machine processible semantics. Since then the idea has rapidly raised wide international interest. In addition to W3C, a major force in the USA for promoting the idea has been the DARPA Agent Markup Language project. In the summer 2001, the OntoWeb research network was established in EU and is coordinated by the Free University of Amsterdam. In October 31, 2001, the Semantic Web Kick-Off in Finland seminar was organized as a national follow-up of these international developments. An indication of the raising interest in the topic was that more than 200 researchers, developers, and students from universities, research centers, and companies participated in this event in Helsinki.

The goal of the kick-off was to initiate and boost Semantic Web research in Finland

- by presenting the Semantic Web vision and the W3C Semantic Web Activity program by distinguished international visitors,
- by providing a concise overview of the technologies underlying the Semantic Web endeavor, and
- by calling Finnish research groups together to present and discuss their work in the field.

The seminar program was organized in three consecutive sessions according to these goals. In the first part, the Semantic Web vision and Activity was presented with applications. After this followed an overview of Semantic Web technologies. Finally, Finnish research groups presented their work, applications, and future plans in the field. This book is based on the presentations — with a few additional complementary articles — and consists of the corresponding three parts.

Semantic Web Kick-Off in Finland was organized by Helsinki Institute for Information Technology (HIIT) and University of Helsinki, Department of Computer Science, together with the Finnish Artificial Intelligence Society, XML Finland, and Elisa Communications Ltd. The program committee was chaired by Eero Hyvönen (University of Helsinki and HIIT) and the members were (in alphabetical order) Heikki Hyötyniemi (Finnish AI Society and Helsinki University of Technology), Mika Klemettinen (XML Finland), Ora Lassila (Nokia), Kari Lehtinen (Elisa), and Martti Mäntylä (HIIT and Helsinki University of Technology). Marja-Riitta Koivunen from W3C presented the keynote lectures on W3C activities with contributions of Eric Miller, the director of the W3C Semantic Activity program.

On the behalf of the program committee I would like thank all authors of this publication and all presenters in the kick-off seminar. Thanks are also due to the audience that actively participated in the meeting until the late closing time. You all made made the event memorable.

Helsinki, May 19, 2002

Eero Hyvönen

Contents

Part I

The Semantic Web Vision

Chapter 1

The Semantic Web — The New Internet of Meanings

Eero Hyvönen

The WWW of today has been developed for the human reader. A machine cannot understand much of the contents of the web, but just offer them to people to interpret. The automatic interpretation of the contents is vital, however, for the development of intelligent Internet applications that are easy to use. The Semantic Web — the Internet of meanings — is a vision of the next generation web that may be used not just by humans, but also by machines. The vision has swiftly transformed into international research and standardization programmes and projects. With the help of the Semantic Web standards and tools we can represent meanings of web contents in a machine understandable way and implement the next generation of intelligent services and applications.

1.1 The Problem of Meaning

The data on the current WWW is mainly presented so that it is easy to convey and show to people. Web document languages such as HTML and PDF, are languages for the external layout of documents. For example, an HTML heading tag

```
<H2>The Problem of Meaning</H2>
```

says nothing about the heading contents without the interpretation of a human reader.

However, the Internet is used not only by people, but increasingly by search robots, agents of electronic commerce (shopbot), web crawlers, and other artifacts. It is hard to them to interpret or "understand" the unstructured information available through the Internet.

1.1.1 Understanding Web Content

The problem of understanding meanings has been approached in various research fields, such as information retrieval, information extraction, and data mining.

Information Retrieval

In *information retrieval* [24, 4], information that satisfies the user's need is searched for in a data repository, such as a database. Successful retrieval presupposes that meanings of the data can somehow be processed. On the Internet, information retrieval is based on two main methods:

- Information is found by following associative hyperlinks one by one, i.e., by "surfing".
- The pages can be searched for by keywords using a search engine, such as AltaVista[1], Lycos[2], or Google[3]. The engine will generate hit lists of web pages for closer human analysis.

The documents in hit lists are usually in abundance. In order to help the human reader, it is possible to order the hits according to their relevance by counting keyword frequencies. It is also possible to inspect the reference structure of the web and rank trusted addresses and frequently referenced pages higher. This approach is used e.g. in Google [8].

Information Extraction

In *information extraction* [31], the idea is to process the documents mechanically so that the structure of their meaning is found out. Natural language words can be recognized fairly easily. Syntactic phrase, sentence, and document structures can also be searched for, as well as semantic structures. Based on the extracted features, it is possible to identify or filter out the needed information from a bulk of data. For example, news items that fit the interest profile of a user can be extracted from news sources.

[1] http://www.altavista.com
[2] http://www.lycos.com
[3] http://www.google.com

Data and Web Mining

With the techniques of *data mining* [19], it is possible to find or learn new inner structures in databases and other repositories. For example, one can cluster existing data items into related groups and produce sets of rules for categorizing new data into the clusters. A related research area in connection with the WWW is *web mining* [9]. Here the target for machine learning or discovery is either to find out the structure are contents of the web (content mining), or the behavior of the users on the web (usage mining). This enables the development of, among other things, adaptive web pages that adapt to the user's routines.

In the research areas discussed above, the problem of meaning is approached by trying to understand the contents of the web as they are in the same way as we humans do. However, machine-based interpretation is often very difficult due to several reasons:

- The documents are typically written in natural languages that are rich in form and meanings. The text may also be incomplete, fuzzy, and contain errors. Recognizing the morphological and syntactic structures of the text alone is computationally difficult, not to mention semantic content.

- A large part of WWW contents is not textual in nature; images, sounds, music, videos and program components, for example. Understanding such non-textual forms, like an image presented as a bitmap, may even be more challenging than interpreting natural language texts.

- The contents of WWW pages cannot be interpreted on the basis of the information on the page alone. External human common sense knowledge and experience is usually needed, too. Teaching a computer this kind of context information and common knowledge in any general way has proved very difficult in the area of artificial intelligence research [34].

In practice, the automatic interpretation and data mining of documents is possible only for certain kinds of data and in narrow areas of application.

1.1.2 The Semantic Web Approach

Most data on the current WWW has been entered into the web in an informal form that is hard to interpret for the machine. We can develop ever more intelligent systems to understand such data, but on the other hand, why not meet the computer halfway? The underlying idea of Semantic Web technologies [13, 14, 2] is that data should be encoded in forms that make web contents (meaning, semantics) more understandable by algorithmic means.

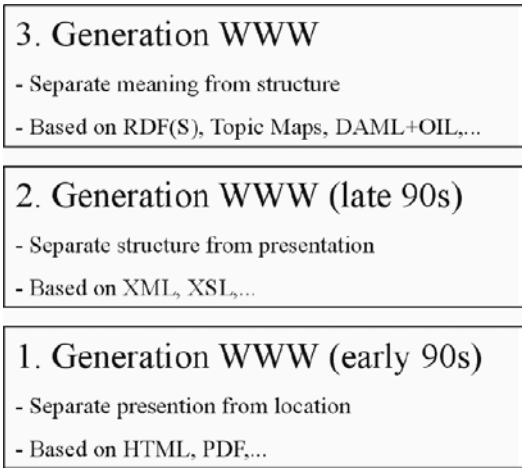


Figure 1.1: Generations of the WWW.

If semantic encoding is too much of a burden to the human author, the computer can help itself by helping the human in creating such descriptions. In this vision, explicit knowledge representation will form a basis for developing more intelligent web services and applications [26].

This development can be seen as a kind of third generation of the WWW (cf. figure 1.1). The main contribution of the original WWW was to make it possible to view documents across the Internet, i.e., the presentation was separated from the location of the document. Next, the XML revolution separated document structure from its presentation. In this view, a document is meant in the first place for storing data whose presentation can be different for different purposes. Finally, the idea of the Semantic Web is to separate meaning from structure, i.e., extend the idea of the document with machine-processible semantics.

Conceptual Layer Model

The Semantic Web is not just a vision of the future; we already have first concrete technologies with which the visions are being implemented. They can be organized on different levels as illustrated in figure 1.2. The WWW and its XML-based technologies for structured documents function as the technical basis. With the help of them, higher-level semantic meta data languages to describe web resources have been created. At the core of the Semantic Web concept are also the ontologies; shared specifications of terminology and conceptualizations needed in different areas of application. By using ontologies, the entities related to an application are represented. At the logical level, inferences based on these can be drawn and proofs for new inferred information be constructed. At an even higher conceptual level the

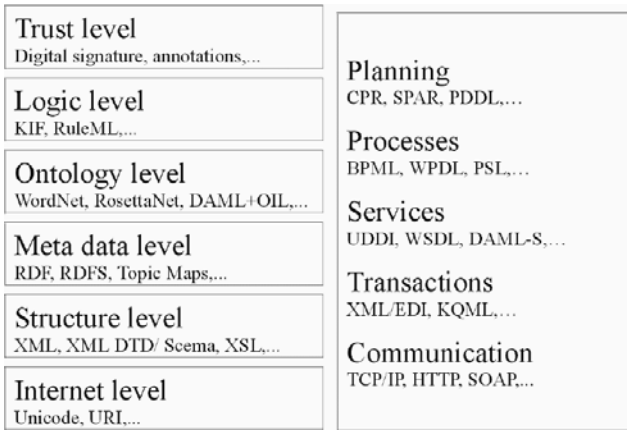


Figure 1.2: Conceptual levels and technologies of the Semantic Web.

questions of *trust* are in focus. The basic question here is: on what grounds can one rely on the data obtained from the web?

These levels form the W3C "layer model" of the Semantic Web. In addition, methods are needed for inter-system communications, transactions, description of services and processes, and planning actions. These technologies can be seen as a kind of functional infrastructure into which semantic representations can be embedded in order to create working applications.

The tools and standards for the Semantic Web are just the technological basis. They cannot solve the problem of presenting and utilizing meanings any more than a hammer can solve the problem of building a house. Lots of work will be needed in the future to actually materialize the promises of the Semantic Web vision.

Semantic Web and Artificial Intelligence

The ambitious practical goal of semantic web research is to create intelligent systems. Many intellectual roots and researchers of the Semantic Web area come from the field of artificial intelligence (AI) that shares the same general goal.

Since the 80's, extensive projects for building knowledge-based systems have been carried out to realize the Feigenbaumian dream of human-level intelligent systems. In retrospection, the general result of this endeavor has thus far not been a new generation over-intelligent machines that was initially expected by many. However, small intelligent systems and applications, such as fuzzy logic controllers in house hold appliances, have proliferated. From the research viewpoint, valuable insights into the difficulty of semantic data processing have been gained. The Semantic Web endeavor is not the old AI dream in a new disguise. The aim is not to realize intelligent systems that

should be as intelligent as man in order to be useful. The point of reference is not man but the current non-semantic web. In this framework, it is much easier to make progress by gradually adding semantic content descriptions on the Web. As a result, an ever more intelligence Internet emerges step by step.

1.2 Conceptual Levels

In the following, the main ideas of the Semantic Web are discussed according to the conceptual levels depicted in the left hand side of figure 1.2. The point of view is technological.

1.2.1 Structure Descriptions

Web of URIs

One of the central concepts of the WWW is the URI (Uniform Resource Identifier). A URI which unambiguously identifies any resource on the worldwide web, such as a web page, email address, or a picture. The WWW is the web of resources identified by URIs. The traditional URL (Uniform Resource Locator) identifiers (e.g., `http://www.cs.helsinki.fi`) used in WWW links are a central part of the URI system. URLs can be used to refer to web files by a given protocol, such as HTTP or FTP. According to Ted Nelson, the father of hypertext, the idea is that a resource may be used by linking when needed, which means that resources do not have to be copied or managed centrally. Full control of the resource stays with the original producer, which makes distributed maintenance and utilization of up-to-date information over the web possible.

XML Languages

XML (eXtensible Markup Language) [6, 11] is the web's new foundation. The Unicode character system used in it supports all the languages in the world, currently over 90,000 characters. XML is not a particular markup language, such as HTML, but a meta language for defining such languages for different purposes. For example, to present address data, an XML syntax in English could be specified, and the address could be written in the following way:

```
<ADDRESS>
   <NAME>John Smith</NAME>
   <PHONE> 123 456 </PHONE>
</ADDRESS>
```

With the help of XSL (XML Stylesheet Language), address data in this format could automatically be transformed into a normal HTML page that

νΠρρψΣΩΩξ
νφΠΦΣξ τω9φ ΩΥχ´9 νΗφΠΦΣξ
νχςΧφΣξ ΘθΙ ικκ νΗχςΧφΣξ
νΗΠρρψΣΩΩξ

Figure 1.3: John Smith's address tagging in Greek letters.

is readable with a browser, into a printed address book, or into other presentation formats. The idea is to separate the structure of the document from its visual manifestation.

The XML markup does not contain any formal semantics for the computer. It is rather the human user who gives meaning to such expressions as ADDRESS, PHONE etc. For the computer, the above address information is exactly as understandable as its version written in Greek letters in figure 1.3. The XML just describes the structure of the information, the syntax.

According to the Dictionary of Philosophy [3] "semantics" (*semainein* in Greek; to signify) is a discipline that studies how symbols refer to other objects. Here the meaning of symbols (intention) is typically defined in terms of the objects to which they refer (extension). Though XML is often said to be a language for presenting meanings, it is not semantic in the sense described above. The symbols of the XML language, such as ADDRESS above, do not refer to anything as such, but the reference is formed in the mind of the human reader. When we read the same address written in the Greek alphabet in figure 1.3, there is no point of reference and we cannot understand the meaning.

With the help of XML, shared vocabularies and syntax (*syntaxis* in Greek; order, structure) for presenting data can be developed in various fields of application. Hundreds of different international groups are currently specifying XML standards for their own application fields[4]. A common syntactic language is required for the systems to be able to co-operate and communicate. With the help of a shared language, the number of needed conversions between different languages decreases. If we want to specify the conversions from n different address formats, for example, to m other formats, one needs $n * m$ conversions. If, on the other hand, we use a common language, only $n + m$ conversions are needed: first n conversions into the intermediate language, and then m conversions from that to the target languages.

Swift increase of data in XML format on the WWW is to be expected in the near future. One practical problem will be the data management of large collections of documents in XML with the help of database techniques [18].

[4]See http://www.xml.org for examples.

1.2.2 Semantic Meta Descriptions

A language can be understood by a computer only if it has semantics. This means that the symbols and structures of the language must refer to an underlying model of some sort, implicit or explicit. The meaning exists only in relation to something. In logics, for example, meaning is based on the model theory, which consists of set of structures describing possible states of the world. The Semantic Web brings the idea of formal semantics into the world of the WWW — a major contribution from the logico-linguistic point of view.

RDF, Resource Description Framework

The best known semantic WWW language at the moment is RDF (Resource Description Framework) [25, 20] and its enhancement RDF Schema (RDFS) [7]. With RDF, meta data concerning web resources can be encoded.

At its core, RDF is a simple relational model. The data presented by it consists of object-attribute-value triples, whose each member may be either a literal symbol or a web resource (URI). Alternatively, from a linguisto-logical view point, the triples can be interpreted as subject-predicate-object tuples.

For example, the following two triples tell that "Jean Sibelius is the creator of Finlandia" and that "Finlandia is a piece of music".

```
SUBJECT                              PREDICATE OBJECT
<http://music.fi/pieces#Finlandia, Creator,  http://composer.org/Sibelius>
<http://music.fi/pieces#Finlandia, type,     music>
```

The specification of the RDF syntax [25] is based on XML. For example, the previous example is rewritten in XML below. Prefixes `rdf` and `dc` refer to XML name spaces.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="http://music.fi/pieces#Finlandia">
      <dc:Creator rdf:resource="http://composer.org/Sibelius"/>
      <rdf:type>music</rdf:type>
    </rdf:Description>
  </rdf:RDF>
```

However, it is essential to note that the data model of RDF is not the serialized syntactic tree of XML, where the branches must be presented in the order specified by the XML DTD or schema. The data model is a set of triples. This model is independent from the serialized XML presentation, and can graphically be imagined to form a directed web. The use of W3C syntax recommendations alone enables the writing of one RDF model in several different formats. Furthermore, also other syntactic specifications than the serialized XML recommendation by W3C can be developed and used. One

possibility is the Notation 3 (N3)[5]. It is based directly on the triples and is easy to read for the human, but cannot be parsed with the usual XML and RDF parsers.

In RDF, the resources to be referred to can also be other RDF expressions. This means that the RDF network can be hierarchical. For instance, personal beliefs of trust in some data can be expressed without arguing that the data holds. Asserting such higher order meta data concerning lower level statements is called "reification".

RDF Schema

RDF does not offer tools for specifying the vocabulary used in RDF descriptions, such as "music" in the example above. RDF Schema (RDFS) [7] is an extension to RDF with which it is possible to:

- Describe the concepts used in the RDF application.
- Set type constraints for the objects and values in the triples.

RDFS can be seen as a kind of introduction of the object oriented modeling paradigm into WWW. The predefined attributes `rdfs:Class` and `rdfs:subClass` can be used for defining class hierarchies and their instances can be pointed out with the `rdf:type` attribute of RDF.

In a similar way, attributes `rdfs:domain` and `rdfs:range` constrain the types of resources that can be used in the object and value positions of the RDF triples. The idea is related to the notion of data integrity in database systems. This feature makes it possible to validate RDF(S) statements, not only XML syntax.

In RDFS descriptions, normal RDF language is used. The primitives `Class`, `subclass`, `type` etc. give the RDF expressions an intended semantic interpretation common in object oriented modeling. They tell us how these concepts stand hierarchically in relation to each other, and what restrictions there are on presenting the attributes. RDFS is application domain independent in the same way as classes and objects in programming.

RDF(S) offers a foundation on which application designers can build their applications. Its data model is more general than that of XML and hence more versatile. The actual intelligence is created on the application level, where RDF(S) descriptions are fed into the main memory with the help of a parser, and where they are given an operational interpretation by the application. With the help of the XML name space mechanism, the descriptions can make use of vocabularies compiled by different standardizing groups. The Dublin Core[6], for example, is a standard that originated in the field of

[5]http://www.w3.org/2000/swap/Primer.html

[6]http://www.dublincore.org

library science. In Dublin Core, the general meta data of a document — such as `Creator`, `Subject`, etc — may be described with the help of fifteen attributes.

The XML syntax of RDF(S) is clumsy for presenting information from the viewpoint of a human user. The Semantic Web vision cannot be realized if the makers of WWW pages were supposed to encode meanings with it. However, meta descriptions can also be produced through programs. For example, with the help of the RDFPic Java applet developed by W3C, meta data in the form of RDF text can be embedded in the files of JPEG images. Adobe has started to use RDF as the meta data format in its eXtensible Metadata Platform (XMP)[7] that will be used eventually in all Adobe application. In the time of writing this, general search engines (Google, Altavista, Lycos etc.) are not using RDF meta descriptions. First RDF plug-ins for browsers are available, e.g., for Mozilla[8].

An important aspect of the RDF data model is that it enables combining the meta descriptions in different parts of the web into a single graph. Separate descriptions for one resource (URI), such as the data for a piece of music, can be gathered into one common RDF graph. With XML trees this kind of merges would be much more difficult to make. A different DTD would probably be needed for the merged data, for example.

RDF data model also offers an abstract level with which it seems to be easier to transform XML documents from one format into another [30]. Writing the transformations directly at XML level using XSL can be quite laborious even when considering simple formats.

The amount of RDF(S) documents on the WWW will increase in future. To manage information in the RDF format, database solutions (such as Redland[9]) are being developed similarly to databases and query languages (such as XQuery) for XML languages. Sesame[10] is probably the first database solution supporting RDF Schemas. It uses RQL query language specially designed for RDFS data. RQL can be seen as a kind of mixture of SQL and logic programming languages, such as Prolog. In addition to the object oriented paradigm, RDFS also introduces ideas of logic programming into the world of WWW through query mechanisms.

Topic Maps

The Topic Maps[11] (TM) [33] is a meta data representation scheme for describing web resources in the same spirit as RDF. Both approaches have the

[7] http://www.adobe.com/products/xmp/main.html
[8] http://www.mozilla.org
[9] http://www.Redland.opensource.ac.uk/
[10] http://sesame.aidministrator.nl/
[11] http://www.topicmaps.net

same goal: to provide a tool for dealing with the information overload of web contents.

The idea of TM was born in the early 90's before the WWW came into general use. The original goal was to develop standards for merging book indexes. These historical roots are still visible in the TM scheme, which essentially enriches the idea of the book index into a kind of electronic semantic associative index. Topic maps were standardized under the name "Topic Navigation Maps" by ISO (International Organization for Standardization) in the year 2000 (ISO/IEC 13250). This standard was originally based on an SGML Data Type Definition (DTD). In order to create an XML-based serialization language, a new organization called TopicMaps.org was immediately established after this. The result of its work, XML Topic Maps (XTM) standard[12], was published in February 2001.

The core of the TM consists of *topics*, *associations*, and *occurrences*:

Topic A topic may be any subject, concept, person, thing, occurrence etc. to which information can be attached. Topics correspond to the index words used in traditional book indices.

Association Associations connect related topics with each other. The association "Creator", for example, can connect Finlandia to Sibelius, the composer, or King Arthur lore to Lord Tennyson, the author. Associations may also be topics on their own.

Occurrence The occurrences are the topic's different incarnations. The topic of King Arthur, for example, can manifest itself in a painting, a poem or an article on the subject.

The underlying data model of a topic map is a directed graph structure in the same way as in RDF. In this view, topic maps can be seen as a kind of semantic nets that are used for knowledge representation [36] in artificial intelligence (AI).[13]

A topic is characterized by its name[14], by its associations with other topics, and by its occurrences. A sample application could be the "Knights of the Round Table" that would gather the characters and themes of the King Arthur lore, as well as works of art inspired by it and factual research work on it. The information search on the topic could be made by following associations from one topic to another. There are also query language proposals for topic maps, such as Topic Maps Query Language TMQL[15] and tolog [16].

[12] http://www.topicmaps.net/xtm/1.0/

[13] The concept "semantic net" is widely used in AI and should not be confused with "semantic web".

[14] A topic may also be nameless, i.e., have only identity.

[15] http://www.y12.doe.gov/sgml/sc34/document/0227.htm

While TMQL is based on SQL, tolog has its roots in logic programming and resembles and in this sense RQL, a query language for RDF repositories [22].

A topic can belong to one or several topic types. Excalibur, for example, can refer to King Arthur's sword, or a company that produces leather products, or a hotel in Las Vegas. The interpretation depends on the *scope* used in interpreting the topic map.

Topic maps are meta descriptions that are physically separate from the documents they describe. A map can therefore be developed without moderating the target documents. The underlying data model is essentially a graph, like with RDF. As a result, it is easy to merge maps together.

The relation between the W3C RDF and the ISO Topic Maps standard has given cause to discussions [32], even heated ones. TM is more application oriented due to its historial roots in book indices. It is a formalism for describing semantic content structures, which is illustrated in the use of the concepts "topic", "association", and "occurrence". RDF, as can be seen by its name, is a generic lower level "framework" and a web data model on which one can create application specific languages and applications. RDF(S) and Topic Maps are based on the graph data model and have XML syntax specification, but it is unclear, how to map the languages from one to another. For example, the notion of TM "scope" has no direct counterpart in RDF(S). On the other hand, TM lacks the predefined specification mechanisms for class hierarchies of RDFS. The discussion on the pros, cons, and possible combination of the two approaches will continue.

1.2.3 Ontologies

The concept of ontology [36, 14] is one of the most central ones in the Semantic Web vision. Ontology can be defined as follows [17]:

> An ontology is a formal, explicit specification of a shared conceptualization.

Attributes "formal" and "explicit" enable the automatic machine-based interpretation of the conceptualization, "shared" enables the sharing, combination, and integrated use of ontological information. These features are quite vital in the future WWW used by machines.

In practice, ontologies are thesauri [15] and more advanced terminological concept hierarchies. They determine the terms, concepts, and the relations between them in different fields of application. Ontologies vary a great deal depending on the domain, purpose of use, and knowledge representation mechanisms used [28, 14]. Examples of ontology types are listed below:

- Ontologies for sciences, such as biology, electronics, etc.

- Business ontologies for representing products, business models etc.
- Cultural ontologies for representing art, artifacts etc.
- Meta data ontologies, e.g., for representing the publishing data of documents.
- General, horizontal common sense ontologies.
- Meta concept collections, such as the meta concepts used in describing ontologies themselves.
- Ontologies for dynamic phenomena, such as tasks, processes, and services.

Some well-known large ontologies are the WordNet[16], containing over 100,000 concepts in English, and the RosettaNet[17] of the IT and electronics industry, the CYC[18], and the Standard Upper Ontology (SUO) of IEEE, a standardization project in progress for conceptually high level ontologies[19].

Several kind of tools are needed for creating and maintaining ontologies [12, 23]:

- Editors with which the ontologic descriptions are created manually or semi-manually.
- Annotation tools to link information sources with meta data.
- Reasoning tools for ontology validation and management.
- Ontology libraries and environments for creating new ontologies from older ones and for managing their evolution.

As for ontology languages, much attention has lately been paid to the European OIL[20] (Ontology Inference Layer), the American DAML[21] (DARPA Agent Markup Language), and their combination DAML+OIL. The aim is to finally reach a common standardization recommendation for the W3C.

DAML and OIL are logical specification languages for concept hierarchies based on *description logics*[22]. In OIL, the problem of concept subsumption, central to description logics, is decidable, and an efficient inference engine called FaCT (Fast Classification of Terminologies) [21] has been implemented

[16] http://www.cogsci.Princeton.edu/-wn/
[17] http://www.rosettanet.org
[18] http://www.cyc.com/
[19] http://suo.ieee.org/
[20] http://www.ontoknowledge.org/oil/
[21] http://www.daml.org
[22] http://dl.kr.org

for the language. DAML+OIL combine the ideas of object oriented modeling and logic programming with WWW languages, especially RDF(S) and XML. They offer a basis for representing and sharing ontologies on the web. With the help of a high-level logical ontology language like DAML+OIL, meta descriptions of WWW resources can be created more easly. The transformation into lower-level RDF(S) and XML expressions can be left to the machine.

To create ontologies, a great number of editors have been developed, such as Protégé[23] and OilEd[24]. In addition to a graphic user interface that is easy to use, editors also offer, among other features, tools for checking the internal consistency of the ontology and for combining different ontologies. Back-end generators can be added to the Protegé editor, to transform the general ontology description into different languages, such as RDFS.

Several practical difficulties arise with the development of ontologies. The standardization of terminologies is difficult in general due to the different needs and preferences of different parties and interest groups. Ontologies also tend to become large. In addition, they change in time, and the management of them may prove difficult. Czechoslovakia, for example, was until recently part of the ontology of nations, and the term has been used to index a great deal of information. However, Czechoslovakia no longer exists. The development techniques for ontologies [29], the shared use of ontologies, and their combination, as well as the management of ontologies, present a great challenge to the implementation of the Semantic Web vision [37].

1.2.4 Logic, Proof, and Trust

An ontology defines the concepts and objects of interest in a field of application, but does not tell us how they can be utilized. An electronic buying agent, for example, should be able to deduce that, if the purpose is to order ten PCs, and it has ordered four machines from company A and six from company B, then it can let the customer know that the task has been completed. This sort of knowledge is not ontological in essence, but related to actions, processes, and logical deductions concerning the entities of the ontology. Such reasoning takes place at the logic level of the Semantic Web (cf. figure 1.2).

Work on standards for the logic level of the Semantic Web has been initiated but is still immature. Among others, the markup language RuleML (Rule Markup Language) for deduction rules is under development at W3C.

With logic it is possible to deduce new implicit information not explicitly coded. As a side effect of the logical *proof*, the user can be convinced that the conclusions acquired are correct given the premise data on the web are correct. However, an evermore difficult problem with the WWW is: what

[23] http://protege.semanticweb.org/

[24] http://img.cs.man.ac.uk/oil/

public information there is right and can actually be trusted? One would rather trust the meta descriptions, annotations, of a well-known person or enterprise than the information on a random page written by someone we do not know. The problem of trust is in the very hearth of the WWW, whose strength lies in the fact that anyone can publish information and opinions on the web, as well as have access to the information provided by others.

Trust can be enhanced by using meta data of web resources. The basic mechanisms for annotating WWW resources are being developed, e.g., in the Annotea project[25] at W3C . The idea is to offer WWW users the tools to evaluate and comment web pages. Based on such evaluations and annotations, it is easier for other users to evaluate the reliability of those web resources. With the help of *digital signatures* the identity of the person who made the annotation can be proved. In this way, the reliability of the annotation and the related resource can be evaluated.

1.3 Fields of Application

Semantic web techniques, such as meta data descriptions and ontologies, can be applied in many fields of application including [14]:

- Information search and retrieval.
- Knowledge management.
- Web commerce in the B2C sector.
- Electronic business in the B2B sector.

In the following, these application fields are briefly discussed.

One of the main problems of the WWW is finding the information or service the user needs from the huge, unstructured information mass of the Internet. Associative hypertext links [1] and meta descriptions of contents give us new possibilities for data search based on contents, and thus enhance the traditional methods based on keyword search [24]. The new approach makes it possible to construct, for example, semantic portals , such as the intranet of the Karlsruhe University AIFB Institute [27].

The current interest in knowledge management [35] applications comes from the increasing need of companies and organizations to purchase, maintain, find, and utilize their own knowledge. The goal is to achieve competitive advantage and efficiency. Some of the challenges here are the informality of documents in the organization's data warehouse, and their distribution along with globalization.

Advantages of Semantic Web techniques include:

[25] http://www.w3.org/2001/Annotea/

- Information can be searched on the basis of contents, not just with keywords.

- Answers to complicated information needs can be constructed in stead of just generating hit lists.

- Documents can be exchanged between systems using different standards (e.g. with the help of XSLT transformations).

- Different viewpoints can be created for documents.

By creating shared ontologies, process descriptions, and communication languages interoperability of systems in different fields of application can be pursued. For example, patient data in different hospitals can be combined, company and product catalogues can be merged, museum collection databases be made compatible, etc.

Some of the innovations of web commerce [38] are on-line markets, buying agents and auctions. The Internet provides an alternative distribution and marketing channel, which enables new types of business models to be used. The Semantic Web techniques will have an increasingly important role in the description of products, services, and processes in B2C web commerce.

In electronic business [38, 10], the focus of development is on the management of B2B business transactions. Another important field of development is product and service descriptions and catalogues along with their catalog services. With the technologies and ontologies based on XML, content standards can be created for various business fields. These standards can easily be integrated into the normal document management and Internet communications of the enterprise. Through shared terminology and communication languages it will be possible to create common portals for different areas of businesses. For example, Interiore.net project[26] is an attempt to create a common product catalog model for the furniture industry in Finland. Its vision is to provide customers with a single access point for finding product information of different vendors.

Lots of international standardization is going on in the field on electronic business. For example:

- The XML version of EDI (Electronic Data Interchange), XML-EDI[27]. EDIFACT has not met the original expectations, but has rather proved to be a clumsy and isolated solution. Help is sought in the world of XML.

[26] http://www.interiore.net

[27] http://www.oasis-open.org/cover/xml.html#xml-edi

- Framework standards of electronic commerce.[28] The ebXML (Electronic Business XML)[29] is a large modular project for standardization, with the ambitious goal of enabling the creation of global electronic market places for enterprises of all sizes and types. The project is headed by OASIS and UN/CEFACT. UDDI (Universal Description, Discovery and Integration) is a related standardization project headed by enterprises like Ariba, IBM, and Microsoft. With its specifications, companies may publish their services in a special registration centre, as well as find and use the services they need. WSDL (Web Services Description Language) is a language for describing the services. For example, a service from a stock exchange service provider may produce the latest market report on a given company.

- Structured and knowledge-based communication. SOAP[30] (Simple Object Access Protocol) is a standard based on the HTTP protocol, with which systems can exchange messages in XML format. KQML[31] (Knowledge Query and Manipulation Language) is a framework for the communication between intelligent agents. It is based on similar pragmatic structures as human communication (speech acts). FIPA[32] (Foundation for Intelligent Physical Agents) and OMG[33] (Object Management Group) each on their part develop languages for agent communication.

- Product ontologies. With the help of the standards listed above, the general frameworks and operational mechanisms for electronic commerce and communications can be described. However, the standards are application independent, horizontal in nature and do not take into account of the nature of the products or services involved, or what the message contents are. With the help of business specific, vertical ontologies, the systems can be given an understanding of the subject of business and communication. The most well-known vertical standardization effort is probably the RosettaNet[34] that has been developed in the IT and electronics industry.

[28] http://www.xml.org/xml/resources_cover.shtml
[29] www.ebxml.org
[30] http://www.xml.org/xml/resources_focus_soap.shtml
[31] http://www.cs.umbc.edu/kqml/
[32] http://www.fipa.org
[33] http://www.omg.org
[34] http://www.rosettanet.org

1.4 Conclusions

The ultimate goal of the Internet is to offer services. Recent fall of "dot.com" companies and problems in developing commercially viable business service models in the area of mobile computing (e.g., the WAP standard) show that this not as easy as expected.

The Semantic Web gives a new starting point for developing intelligent web services. The area has lately been the object of intense research as well as practical standardization efforts. Consensus on standards is essential for the global use of technologies in the WWW environment.

In February 2001, W3C started the specialized Semantic Web Activity program[35] to promote and coordinate the development in the field. Along with this opening, a cover story appeared in the journal Scientific American written by among others Tim Berners-Lee, known as the father of the WWW [5]. Creating networks for co-operation right from the start is vital in an endeavor like this.

In Europe, in the summer of 2001, the specialized OntoWeb research network[36] was established to enforce the co-operation between research facilities, companies, and scientists. In the USA, DARPA is funding the large DAML program[37] (DARPA Agent Markup Language). In a short period of time, a great number of coalitions have been formed to create the languages, technologies and standards for the Semantic Web concept[38]. There is even a risk that excessive standardization turns on itself; the existence of two overlapping standards already means that there is no real standard.

In semantic web research, several interests, possibilities and challenges are fruitfully combined with each other:

Business interests The world of business has a great need to produce appealing services and businesses in future Internet and mobile networks.

Technological possibilities The WWW and Internet platforms, as well as the standards, techniques and tools based on XML, enable the global implementation of the Semantic Web vision.

Scientific challenges This area offers the possibility to utilize the results of the methods of both intelligent systems and knowledge technology in a new context, the Internet, that is central to the field of computer science.

[35] http://www.w3c.org/2001/sw
[36] http://wwwontoweb.org
[37] http://www.daml.org
[38] http://www.semanticweb.org

National interests A prerequisite for an intelligent WWW is, among others, the development and implementation of technologies for different national languages.

The Semantic Web is at the core of the new WWW research when looking at the web from the point of view of applications and services. The envisioned revolution in technology offers a new qualitative opportunity to develop intelligent and easy to use systems based on knowledge contents on the web.

It remains to be seen how soon the vision of the Semantic Web, the Internet of meanings, will be implemented as applications in practice. The attitude of the developers in the field is humble in comparison with some earlier "next generation" visions related to artificial intelligence. The idea is simply to gradually improve of the present situation by the addition of semantic descriptions to the web. This starting point is promising in spite of the extent of the visions and challenges.

Acknowledgements

Marina Kurten has helped in translating an earlier Finnish version of this paper into English.

Bibliography

[1] M. Agosti and A. Smeaton, editors. *Information retrieval and hypertext.* Kluwer, New York, 1996.

[2] Kal Ahmed, Danny Ayers, Mark Birbeck, Jay Cousins, David Dodds, Joshua Lubell, Miloslav Nic Daniel Rivers-Moore, Andrew Watt, Robert Worden, and Ann Wrightson. *Professional XML Meta Data.* Wrox Press Inc., 2001.

[3] P. Angeles. *Dictionary of Philosophy.* Harper Reference, New York, 1981.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, New York, 1999.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[6] N. Bradley. *The XML Companion.* Addison-Wesley, 2000.

[7] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[8] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[9] R. Cooley, M. Bamshad, and S. Jaideep. Web mining: Information and pattern discovery on the world wide web. http://www-users.cs.umn.edu/ mobasher/webminer/survey/survey.html, 1997.

[10] H. Deitel, P. Deitel, and H. Nieto. *e-Business & e-Commerce. How to program.* Prentice Hall, New Jersey, 2001.

[11] H. Deitel, P. Deitel, and H. Nieto. *XML. How to program.* Prentice Hall, New Jersey, 2001.

[12] Y. Ding, D. Fensel, M. Klein, and B. Omelayenko. The semantic web: Yet another hip? *Data and Knowledge Engineering*, 2002 (forthcoming).

[13] D. Fensel (ed.). The semantic web and its languages. *IEEE Intelligence Systems*, Nov/Dec 2000.

[14] D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce*. Springer-Verlag, 2001.

[15] D. J. Foskett. Thesaurus. In *Encyclopaedia of Library and Information Science, Volume 30*, pages 416–462. Marcel Dekker, New York, 1980.

[16] Lars Garshol. *tolog — A topic map query language*. XML Europe 2001, 2001. http://www.ontopia.net/topicmaps/materials/tolog.html.

[17] T. R. Gruber. A translation approach to portable ontology spesifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[18] A. Halevy. IEEE Data Engineering, special issue on XML data management, June 2001.

[19] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2000.

[20] J. Hjelm. *Creating the Semantic Web with RDF*. John Wiley & Sons, New York, 2001.

[21] I. Horrocks and P. Patel-Schneider. Optimizing description logic subsumbtion. *Journal of Logic and Computation*, 9(3):267–293, 1999.

[22] G. Karvounarakis, V. Christopides, D. Plexousakis, and S. Alexaki. Querying community web portals, 2000. http://139.91.183.30:9090/RDF/publications/sigmod2000.html.

[23] M. Klein. Combining and relating ontologies. In Stuckenschmidt and Uschold [37].

[24] R. Korfhage. *Information storage and retrieval*. John Wiley & Sons, New York, 1997.

[25] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[26] A. Levy and D. Weld. Intelligent Internet systems. Artificial Intelligence, special issue, Vol. 118, 2000.

[27] A. Maedche, S. Staab, N. Stojanovic, R. Struder, and Y. Sure. Semantic portal — the SEAL approach. Technical report, Institute AIFB, University of Karlsruhe, Germany, 2001.

[28] N. Noy and C. Hafner. The state of the art in ontology design. *AI Magazine*, (3), 1997.

[29] N. Noy and D. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford University, Medical Informatics, 2000.

[30] B. Omelayenko and D. Fensel. A two-layered integration approach for product information in B2B e-commerce. In G. Pernul K. Bauknecht, S.-K. Madria, editor, *Electronic Commerce and Web Technologies, Proceedings of the Second International Conference on Electronic Commerce and Web Technologies (EC WEB-2001)*, LNCS 2115, pages 226–239. Springer-Verlag, 2001.

[31] T. Pazienza, editor. *Information extraction: a multidisciplinary approach to an emerging information technology*. Springer-Verlag, Berlin, 1997.

[32] S. Pepper. Topic Maps and RDF: A first cut, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.

[33] Steve Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe 2000, Paris, France*, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.

[34] S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, New Jersey, 1995.

[35] R. Smith and A. Farquhar. The road ahead for knowledge management. an AI perspective. *AI Magazine*, (Winter):17–40, 2000.

[36] J. Sowa. *Knowledge representation. Logical, philosophical, and computational foundations*. Brooks/Cole, 2000.

[37] H. Stuckenschmidt and M. Uschold, editors. *Ontologies and information scharing*, Working notes, IJCAI-2001 workshops. AAAI, 2001.

[38] E. Turban, J. Lee, D. King, and H. M. Chung. *Electronic commerce. A managerial perspective*. Prentice-Hall, New Jersey, 2000.

Chapter 2

W3C Semantic Web Activity

Marja-Riitta Koivunen and Eric Miller

2.1 Introduction

The World Wide Web contains huge amounts of information created by many different organizations, communities and individuals for many different reasons. Users of the Web can easily access this information by specifying URI addresses, searching, and following links to find other related resources. The simplicity of usage is a key aspect that made Web so popular; so popular in fact, it is hard to imagine life without it anymore.

This simplicity of the current web has a price. It is very easy to get lost, or discover irrelevant or unrelated information with all that is available. For instance, if we search for something as simple as research papers written by a person named "Eric Miller" we will find all kinds of other information starting from Web diaries or phonebooks that mention "Eric" and/or "Miller" somewhere. Similar problems arise if you search for resources about "Marja" as "Marja" could equally well refer to a first name of a person, or to a berry in Finnish.

The goal of the Semantic Web is to develop enabling standards and technologies designed to help machines understand more information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks. With Semantic Web we not only receive more exact results when searching for information, but also know when we can integrate information from different sources, know what information to compare, and can provide all kinds of automated services in different domains from future home and digital libraries to electronic business and health services [7].

With the Semantic Web we can associate semantically rich, descriptive information with any resource. For instance, by adding metadata about doc-

ument creation, we can search for documents that have metadata specifying Eric Miller as a "writer". With a bit more metadata we can also search only documents under the category of "research papers". In Semantic Web we not only provide URIs for documents as we have done in the past, but to people, concepts and relationships. In the above case for example, by giving unique identifiers to the person, the role "writer" and the concept of "research paper" we make very clear who the person is, and the corresponding relation between this person and a particular document. Furthermore, by making clear which person we are talking about we can differentiate the plethora of "Eric Miller's". We can also combine descriptive information from different sites and learn more about this person in differing contexts; in his roles as an author, as a manager, as a developer, etc.

The Semantic Web provides the means to add specific information to the Web to aid in the automation of services, in the above case to discover and correlate, and the means of declaring the kind of information one might trust. An objective of the W3C Semantic Web activity is to standardize the key technologies that enable the non-centralized development while making sure that all the pieces fit together. In the following, we will first explain the main Semantic Web principles, then the key technological layers, and finally talk about the activity itself. In the end we will present a couple of sample Semantic Web applications.

2.2 Semantic Web Main Principles

Principle 1: Everything can be identified by URI's

People, places, and things in the physical world can be referred to in the Semantic Web by using a variety of identifiers. Anyone who has control over a part of Web namespace can create a URI and say that it identifies something in the physical world. Some people have insisted that only a small part of Web URI namespace is permissible for this purpose; e.g. those URIs starting with 'urn:'. The Semantic Web does not require, nor does it enforce, such restrictions. We can refer to physical entities also indirectly. For instance, we can refer to the person whose common name is "Marja-Riitta Koivunen" by using the URI to her e-mail inbox in a sentence such as: "The person whose email address is mailto:marja@w3.org and whose name is Marja-Riitta Koivunen.". We can then proceed to specify a great many more things about this person without ever having to assign another identifier to her. Similarly it is possible to identify a place, such as the city Helsinki, by referring to the URI of a page containing information about Helsinki maintained by the city offices. This is a pragmatic approach, with more metadata it is possible to describe the relation of the real city and the URI in more details.

Figure 2.1: Resources from the physical world and their useful identifications in the Semantic Web.

The vocabularies we choose to use for describing resources are also defined by URIs. We may choose to Dublin Core's [1] unique identifer for 'author/creator' to express the relationship between the authors and this paper. The resources and their identifications presented in Figure 2.1 with some vocabulary for an event and a presentation could be used to explain, for instance, this presentation of the W3C Semantic Web activity during the Semantic Web kick-off event in Finland.

Principle 2: Resources and links can have types

The current Web consists of resources and links (see Figure 2.2). The resources are Web documents targeted for human consumption and do not commonly contain metadata explaining what they are used for and what are their relationships to other Web documents. While a knowledgeable human may realize easily that one resource is conceptually an invoice and another one is a novel or a research paper this information is often unavailable for a machine. Similarly a user can guess what kinds of relationships the resource has by reading the text around the link, but it is hard for the machine to make these same guesses. More informative relationships would be, for instance, "depends on", "is version of", "has subject", "authors".

As can be seen in Figure 2.2, also the Semantic Web consists of resources and links. However, now the resources and links can have types which define concepts that tell a bit more to the machines. For instance, some links may tell that a resource is a version of another resource or written by a resource that describes a person or that a resource contains software that depends on some other software. Here we have written the type inside the node to emphasize the similarities. The types are usually defined by a type link to a node with a URI address as illustrated in the detailed model in Figure 2.2.

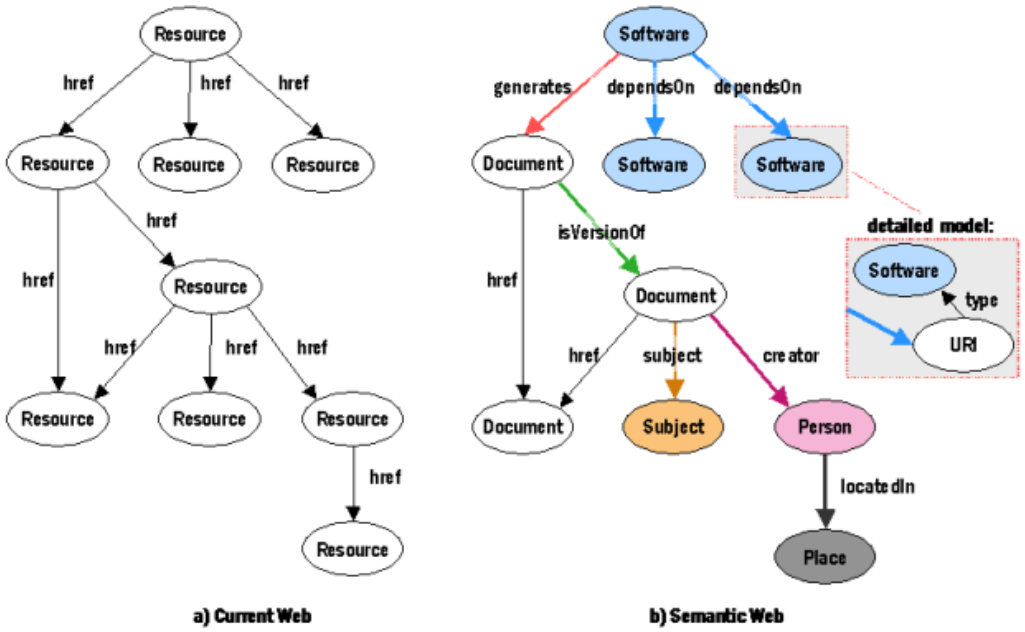


Figure 2.2: Resources and links can have types in the Semantic Web.

Interestingly the different types for resources and links were already present in the original proposal for Web by Tim Berners-Lee as shown in Figure 2.3. He presented in this proposal that the Web of relationships amongst named objects can greatly unify and enhance the information management tasks [6].

Principle 3: Partial information is tolerated

The current Web is unbounded: it sacrificed link integrity for scalability. Authors can easily link to other's resources as they don't have to worry about the links back to their resource. With no way to inform the linkers when the resources are moved we accept that we may get the 404 links presented in Figure 2.4 informing us that the link no longer leads to some Web resource.

Similarly the Semantic Web is unbounded: anyone can say anything about anything and create different types of links between resources. There will always be more to discover. Some of the linked resources may cease to exist or the addresses may be reused. The Semantic Web tools need to tolerate this data decay and be able to function in spite of that. For instance, we should be able to learn about Eric Miller's role in W3C Semantic Web Activity and use that information when making conclusions even if some other information linking to his other achievements is missing. Often the Semantic Web tools can work on selected information islands.

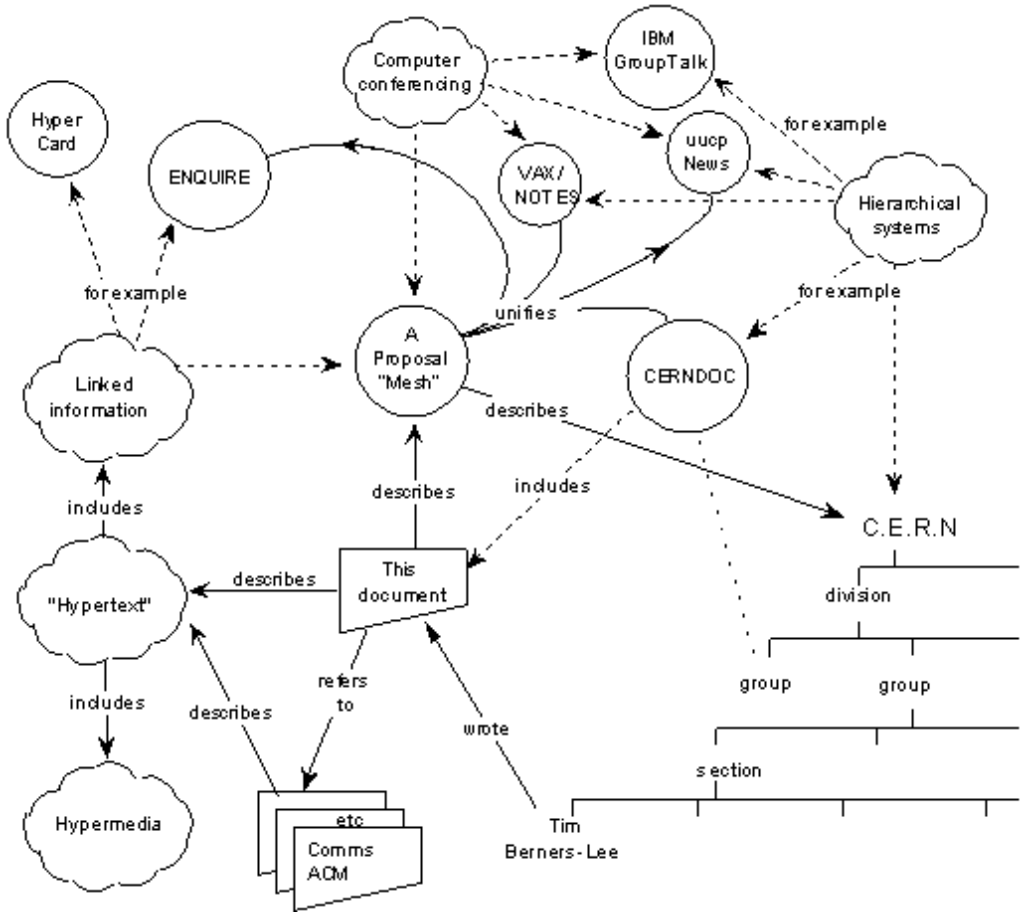


Figure 2.3: The Web as envisioned in 1989 by Tim Berners-Lee.

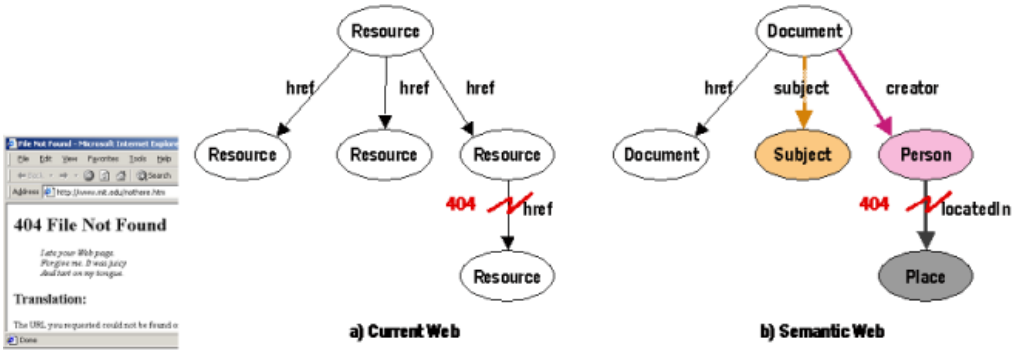


Figure 2.4: Both the current Web and the Semantic Web handle partial information.

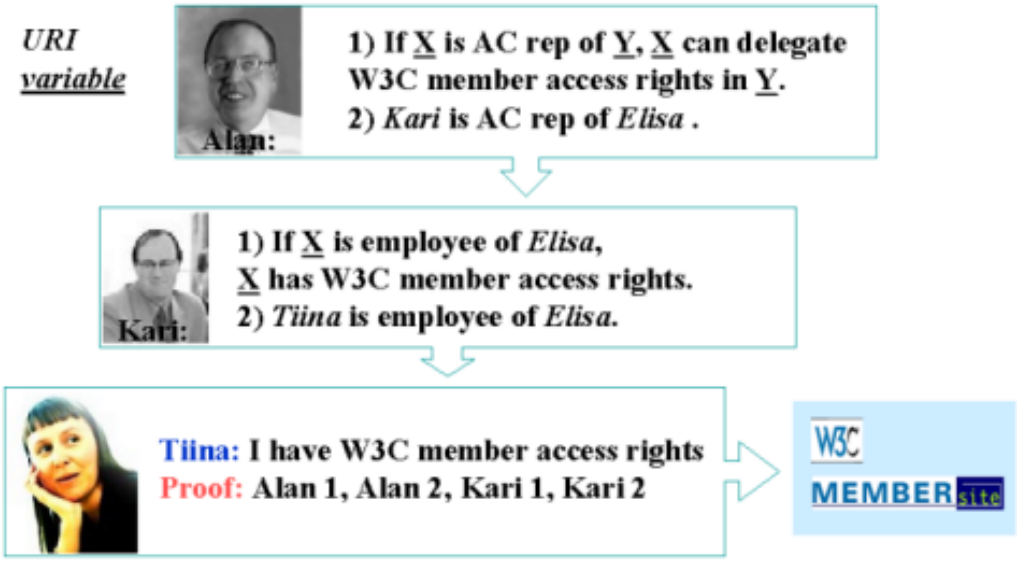


Figure 2.5: The chain of trust can define access rights.

Principle 4: There is no need for absolute truth

Not everything found from the Web is true and the Semantic Web does not change that in any way. Truth - or more pragmatically, trustworthiness - is evaluated by each application that processes the information on the Web. The applications decide what they trust by using the context of the statements; e.g. who said what and when and what credentials they had to say it.

In Figure 2.5 Tiina is an employee of Elisa who wants to visit W3C's member page. To do that she needs to give a proof that she has the rights to access the page. She does that by referring to the four statements made by Kari and Alan, who again are defined to have rights to give such statements, because of their roles as Elisa's Advisory Committee representative and W3C Associate Chairman respectively. The W3C application accepting the proof knows that it can trust Alan's statements and therefore also Kari's statements as Alan has delegated the responsibility for defining Elisa's list of employees with member access rights to Kari.

Principle 5: Evolution is supported

It is common that similar concepts are often defined by different groups of people in different places or even by the same group at different times. It would often be beneficial to combine the data available on the Web that uses these concepts. The Semantic Web uses descriptive conventions that can expand as human understanding expands. In addition, the conventions allow effective combination of the independent work of diverse communities even

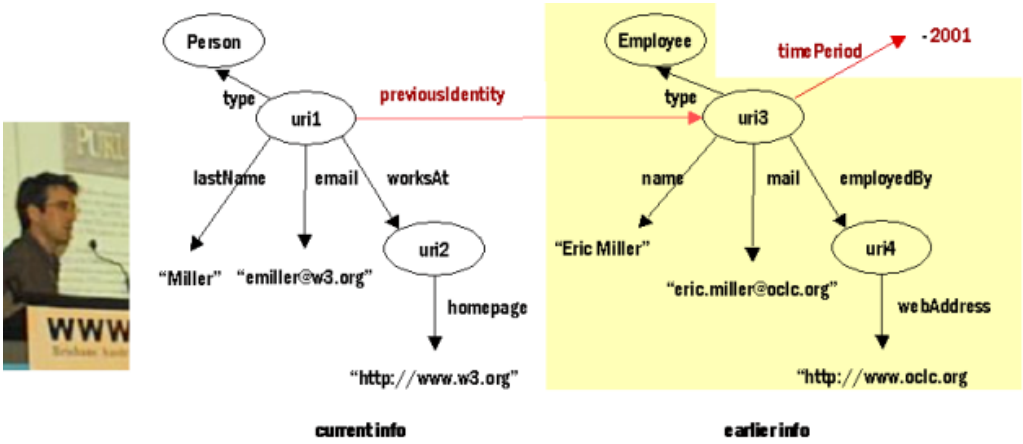


Figure 2.6: Combining new information with old when the old information cannot be changed.

when they use different vocabularies. The Semantic Web provides communities tools that can be used to resolve ambiguities and clarify inconsistencies. Also new information can be added without insisting that the old has to be modified.

In Figure 2.6 we present the current information of a person with last name Miller by using one vocabulary. At the same time some earlier information about the same person can be found in the Web. This information can be combined with the current information in many ways. We can add a new property "previousIdentity" that links the two persons together. We can also define a transformation from our current vocabulary to the vocabulary used by the earlier information e.g. to say that "worksAt" and "employedBy" define the same relation. It is also possible to add a new property defining a time period when the information was valid if the information does not already contain that. This additional information can be stored anywhere, where the relevant applications can find it, as it just refers to the URI of the earlier info. If the understanding of a property evolves over time, that too can be recorded by the Semantic Web facilities as properties can be first class objects with URIs.

Principle 6: Minimalist design

The Semantic Web makes the simple things simple, and the complex things possible. The aim of the W3C activity is to standardize no more than is necessary. This approach enables the implementation of simple applications now that are based on already standardized technologies (e.g. Dublin Core [1], RSS [3], MusicBrainz [5]). At the same time there is research that plans for future complexity. When we use the Semantic Web technologies the result

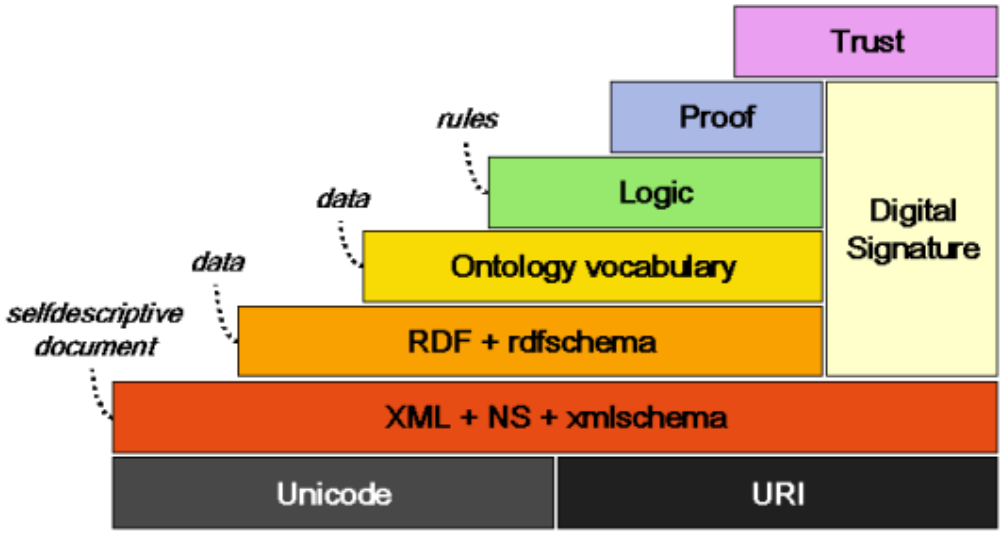


Figure 2.7: The Semantic Web layers.

should offer much more possibilities than the sum of the parts.

2.3 Semantic Web Layers

The Semantic Web principles are implemented in the layers of Web technologies and standards. The layers are presented in Figure 2.7. The Unicode and URI layers make sure that we use international characters sets and provide means for identifying the objects in Semantic Web. The XML layer with namespace and schema definitions make sure we can integrate the Semantic Web definitions with the other XML based standards. With RDF [10] and RDFSchema [8] it is possible to make statements about objects with URIs and define vocabularies that can be referred to by URIs. This is the layer where we can give types to resources and links. The Ontology layer supports the evolution of vocabularies as it can define relations between the different concepts. With the Digital Signature layer for detecting alterations to documents, these are the layers that are currently being standardized in W3C working groups.

The top layers: Logic, Proof and Trust, are currently being researched and simple application demonstrations are being constructed. The Logic layer enables the writing of rules while the Proof layer executes the rules and evaluates together with the Trust layer mechanism for applications whether to trust the given proof or not.

2.4 W3C Semantic Web Activity

The Semantic Web Activity is part of the W3C Technology and Society Domain. The aim of this activity is to design technologies that support machine facilitated global knowledge exchange. The main focus is information that can be consumed and understood by machines and means that make it cost-effective for people to record their knowledge. The focus of this work is on short term deployment while keeping an eye toward longer term research issues. With Semantic Web we can hopefully change the following situation familiar to many users.

> *"The bane of my existence is doing things that I know the computer could do for me."*
> – Dan Connolly, The XML Revolution

The Semantic Web Activity offers an environment for cooperation and collaboration. Currently it has two Working Groups that define enabling standards and technologies: The RDF Core Working Group [2] and The Web Ontology Working Group [4]. The Semantic Web Advanced Development projects explore prototype development in pre-competitive phase. These will be discussed more later. In addition, there are also public interest groups and other education and outreach activities that explain and clarify designs and goals, create implementation guidelines and try to understand policy implications.

RDF Core Working Group (RDFCore)

The current Resource Description Framework (RDF) standard [10] provides a common framework for representing metadata across many applications. The aim of RDFCore is to clarify and improve RDF's abstract model [9] and XML syntax according to feedback from implementors. This group is chartered to complete the RDF vocabulary description in the RDF Schema Candidate Recommendation [8]. The Working Group also explains the relationships between the basic components of RDF (Model, Syntax, Schema) and the larger XML family of recommendations.

Web Ontology Working Group (WebOnt)

The WebOnt working group standardizes means that can be used to define Web ontologies that describe structures of concepts. The DAML+OIL work supported by DARPA's DAML Initiative is given as input to this group. WebOnt builds on RDF Schema (classes and subclasses, properties and subproperties) while extending these constructs to allow a more complex relationships between entities. For instance, it can limit the properties of classes

with respect to number and type. It also offers means to infer that items with various properties are members of a particular class and offers a well-defined model for property inheritance.

Semantic Web Advanced Development (SWAD)

The SWAD projects explore prototype ideas in pre-competitive phase. It is a venue for liaison with research community and a testbed for early implementations of Working Drafts. It also offers a collaborative development environment for the Team and Members to explore ideas together. It stimulates the development of more Semantic Web infrastructure components. Some work has been done or is still done at least in the following areas:

- Developer's Tools
- Resource Description
- Annotation, Collaboration, and Web of Trust
- Access Control Rules
- Logic and Proof Calendaring and Scheduling
- Work-flow and Dependency Tracking
- Transformation and Extraction Utilities
- Integration with XML infrastructure

2.5 Sample Applications

Formalized expression of simple vocabularies

> *"These ambiguities, redundancies, and deficiencies recall those attributed by Dr. Franz Kuhn to a certain Chinese encyclopedia entitled Celestial Emporium of Benevolent Knowledge. On those remote pages it is written that animals are divided into (a) those that belong to the Emperor, (b) embalmed ones, (c) those that are trained, (d) suckling pigs, (e) mermaids, (f) fabulous ones, (g) stray dogs, (h) those that are included in this classification, (i) those that tremble as if they were mad, (j) innumerable ones, (k) those drawn with a very fine camel's hair brush, (l) others, (m) those that have just broken a flower vase, (n) those that resemble flies from a distance."*
> – Essay: "The Analytical Language of John Wilkins", in La Nación, 8 February 1942

The story above defines a simple vocabulary classifying animals to classes in two hierarchical levels. Figure 2.8 presents this vocabulary as an RDF model. In the Semantic Web it is easy to define vocabularies in a formal way. In this example the "subClassOf" property is used for defining the class hierarchy.

Formalized support for ontology merging

In the Semantic Web it is easy to integrate information from several sources. The following example illustrates how easy it is to merge RDF based information.

DMOZ is an open directory project maintained by a large community of volunteers on the Web. Figure 2.9 presents the user interface of a DMOZ page that defines the category "Resource Description Framework - RDF". It will be referred simply as RDF category in the followingdiscussion. The first section of the page defines the category hierarchy starting from the "Top" parent class and ending with the RDF category. The subclasses of the RDF category itself, such as "Applications", "Standards Documents", and "Documentation", are presented in the next section of the page. The page also has a list of closely related categories under the "See also:" section.

The RDF model of some of the categories in Figure 2.9 is presented in Figure 2.10. The "rdfs: subClassOf" relation defines the category hierarchy and the "rdfs:seeAlso" relation cites the related categories that are not directly part of this hierarchy.

The DMOZ page also lists all the Web pages whose category is RDF (see Figure 2.11). The category for these pages is defined by using the "dc:subject" relation from the Dublin Core vocabulary [1]. It is easy to integrate this additional information into the RDF model presented earlier in Figure 2.10. Figure 2.12 presents the new RDF model with three sample pages belonging to RDF category.

All the information presented so far is from the same source, the DMOZ directory. However, it is as easy to add information from other sources to the RDF data. For instance, we could use RDF to model the favorites hierarchy in the Web browser as illustrated in Figure 2.13. The model here uses a vocabulary defining a "bm:includes" relation for both subclasses and categories.

As the RDF category in both favorites and DMOZ RDF models has the same URI and therefore is the same concept, it becomes possible to merge the categories from the favorites directories into the information in the DMOZ directories as we have done in Figure 2.14. With ontologies it will be possible also to define the correspondence between the vocabularies in both models. If our client browser is capable of searching this information either from available Web servers and providing it in some form to the user,

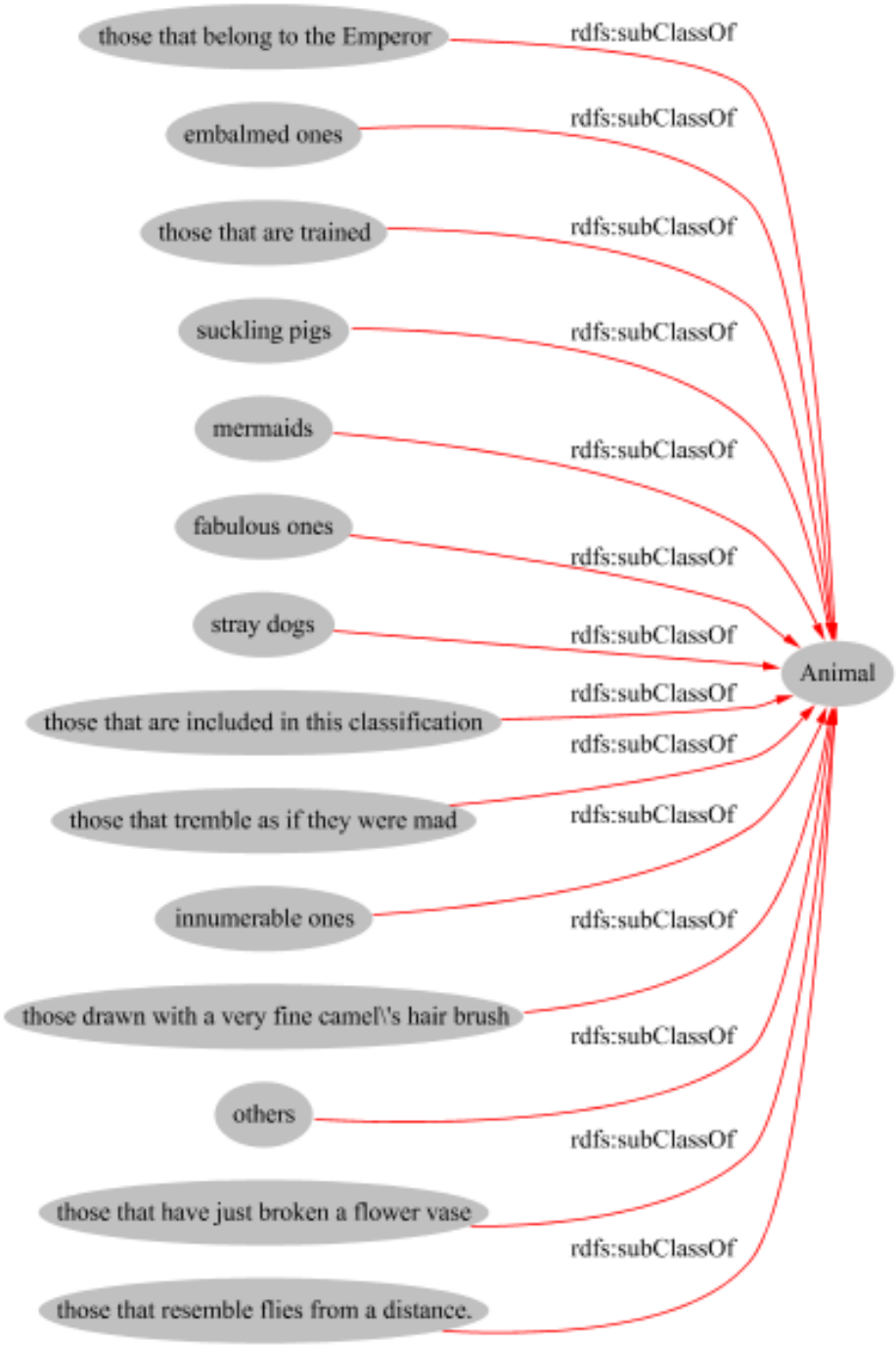


Figure 2.8: RDF model of a simple vocabulary.

Figure 2.9: The DMOZ categories related to RDF category.

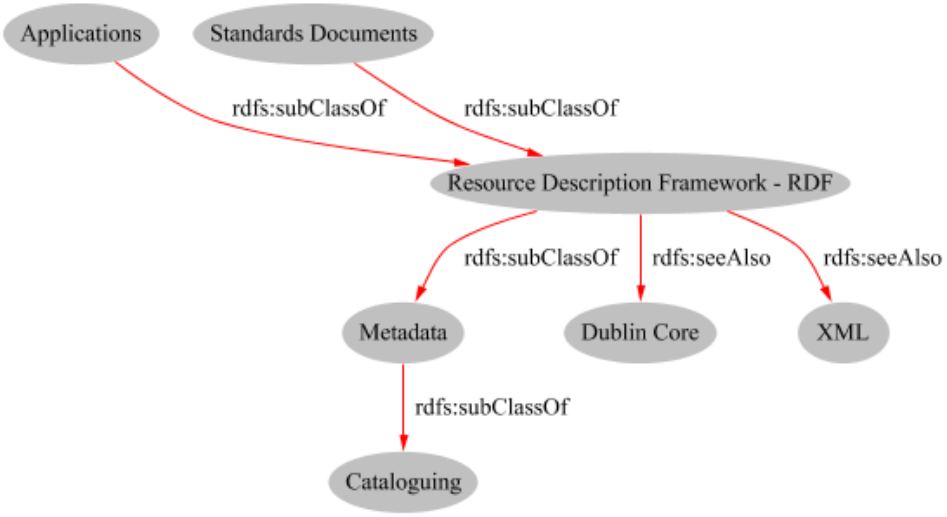


Figure 2.10: RDF model of the DMOZ categories in Figure 2.9.

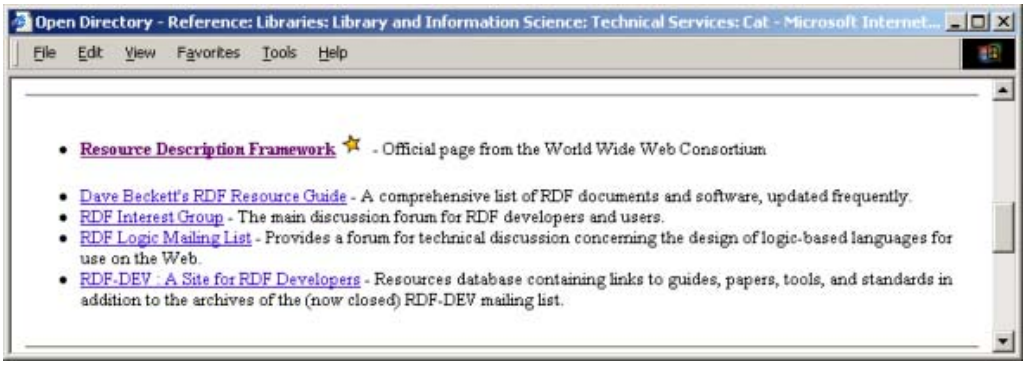


Figure 2.11: DMOZ pages with the RDF category.

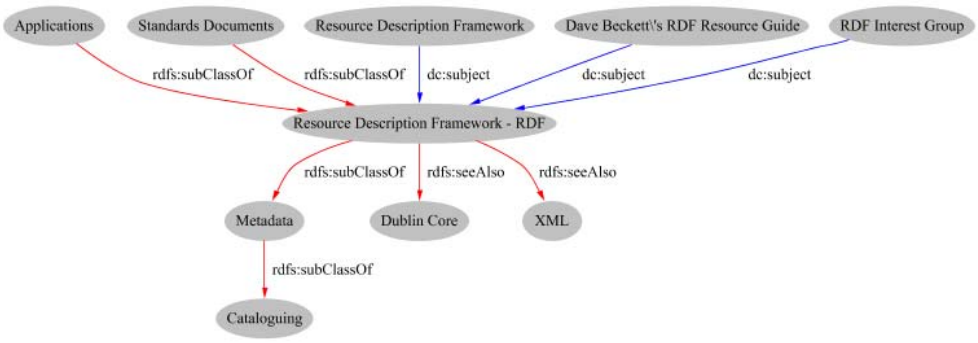


Figure 2.12: Web pages with RDF category are integrated to the RDF model in Figure 2.10.

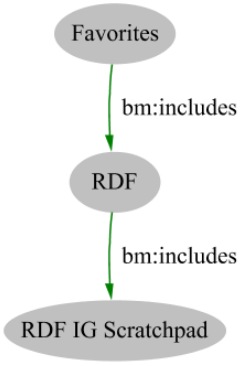


Figure 2.13: An RDF model for browser 'favorites'.

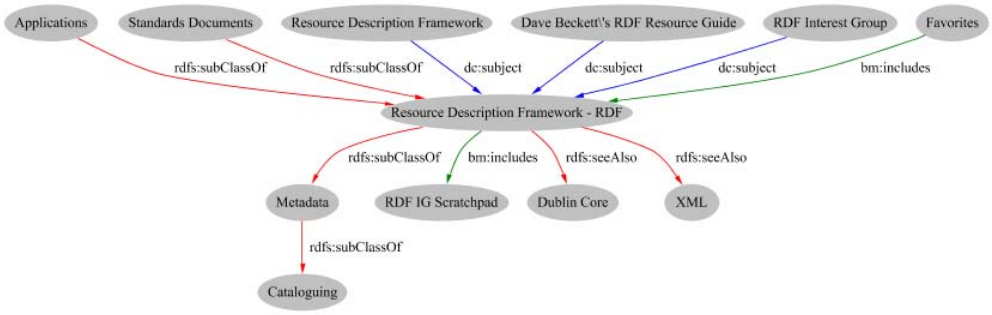


Figure 2.14: Integrating the RDF model for favorites to the DMOZ model.

the user will be able to get not only the information he or she stored in the RDF category but also the information stored by the whole DMOZ open directory community. Naturally it is also possible to filter the information, for instance, only use the information created by some trusted contributors from the community. The RDF datastores can usually handle specialized queries; the clients just need to offer flexible interfaces for users to take full advantage of the possibilities.

2.6 Conclusions

A lot of exiting things are happening right now in the W3C Semantic Web Activity. Working groups are established and technologies are standardized on the lower technological layers that are well understood. It is already possible to implement concrete applications based on this work. At the higher technological layers more research and consensus building is needed and information is gathered from experimental demonstrations.

In addition to the technologies, work is needed to develop tools and easy user interfaces that support users in understanding the metadata and adding the metadata into the Web. This support and automation will be critical in the deployment of the Semantic Web. When more and richer metadata appears there will be huge amounts of opportunities for various applications.

2.7 Acknowledgements

We would like to thank Jose Kahan, Ralph Swick, and Dan Connolly for their comments and help. Also special thanks to Elisa Communications for supporting Marja-Riitta Koivunen's work at MIT.

Bibliography

[1] Dublin core metadata initiative, Dublin Core metadata element set, version 1.1. http://purl.org/dc/documents/rec-dces-19990702.

[2] RDFCore working group. http://www.w3.org/2001/sw/RDFCore/.

[3] RSS 1.0. http://groups.yahoo.com/group/rss-dev/files/specification.html.

[4] Web-ontology working group. http://www.w3.org/2001/sw/WebOnt/.

[5] MusicBrainz 2.0, 2001. http://www.musicbrainz.org/MM/.

[6] T. Berners-Lee. Information management: A proposal. CERN, March 1989, May 1990. http://www.w3.org/History/1989/proposal.html.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001. http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html.

[8] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[9] P. Hayes (ed.). RDF model theory, September 2001. W3C Working Draft, http://www.w3.org/TR/rdf-mt-20010925/.

[10] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

Part II

The Semantic Web Technologies

Chapter 3

Representing Metadata about Web Resources

Eero Hyvönen, Petteri Harjula, and Kim Viljanen

This paper discusses techniques for representing metadata on the World Wide Web (WWW). Explicit metadescriptions concerning the web contents are necessary for making the resources algorithmically understandable. We consider the notions of semantics and metalanguages and present general features of metadata architectures used in the WWW context. Major approaches to representing metadata are presented, compared with each other, and application areas of metadata descriptions are outlined.

3.1 Introduction

3.1.1 Semantics, Metalanguages, and Metadata

The idea of the Semantic Web [5] is to provide the WWW with machine-processible semantics. This means that we need "semantic" formal languages for expressing meanings. Formal languages are defined at two major levels. First, a syntax specification is needed to define the structure of the expressions used in the language. A natural choice for syntactic specifications on the WWW is to use XML Data Type Definitions (DTD) or XML Schemas [7]. Second, a semantic specification or interpretation is needed. It tells what the syntactic expressions mean in terms of some underlying *model*. This model represents in one way another the application domain being modeled. By using the mapping between syntax and the semantic model, language expressions, such as fact assertions, rules, queries, etc. can be interpreted by computers.

For example, the semantics of propositional logic are defined by the truth tables of the connectives in the propositional model theory. The truth value of an arbitrary well-formed propositional expression, i.e., its meaning, can easily be determined based on this simple model. In relational databases [27] the semantics for SQL (Structured Query Language) queries are defined in terms of the underlying relational model. Notice that XML is *not* a semantic language from the machine processing view point. XML-languages do not automatically have an underlying semantic model, only syntax. The meaning of XML elements and attributes is embedded in the procedures processing the XML parse trees. The tags alone have a meaning only in the mind of the human reader.

A language is called a *metalanguage* if it is used for expressing facts concerning some other underlying language. The term is used in two related connotations that should not be confused:

- Definitional connotation. Here "meta" refers to the definitional aspects of the language. For example, according to Deitel et al. [12, p. 540] "XML is a metalanguage — a language for creating other languages".
- Descriptional connotation. Term "meta" can also refer to the fact that the metalanguage is used for expressing metadata concerning expressions (extension) of the underlying language. For example, RDF [24, 19] is a framework for expressing metadata of web resources such as WWW pages, their internal parts, images, on-line database records, email addresses, etc.

In this paper the latter connotation is in focus. We consider languages for expressing metadata about WWW resources. In this view, the web essentially consists of literals, represented in Unicode, and Uniform Resource Identifiers[1] (URI) pointing to the resources.

3.1.2 Metadata Architectures

The idea of adding semantics on the WWW is not new. Any software system using the web must have some sort of metaknowledge of its domain. For example, in search engines metaknowledge is encoded in the inverse indices that map keywords to web pages represented by their URL[2] addresses.

Attaching metadata to web resources can be accomplished by using different architectural approaches. They can be considered from various viewpoints.

Firstly, there is the distinction between implicit and explicit metadata:

[1] Also the phrase "Universal Resource Identifier" is sometimes used.

[2] Uniform Resource Locator, a special form of the URI used, e.g., to pointing to file resources when using the Hypertext Transfer Protocol (HTTP).

- *Implicit* metadescriptions embed the meaning of language structures in the algorithms. For example, a Java program can access and provide meaning to XML-documents through XML Domain Object Model (DOM) Application Program Interface (API) or SAX (Simple API for XML). There is no distinct representation that tells what the document elements mean. Such implicit procedural semantics cannot be reasoned about by machines and are difficult to share and modify by software agents.

- *Explicit* metadescriptions can be accessed and processed by different applications. An explicit metadescription is algorithmically interpretable and modifiable and can be shared by different applications.

In the Semantic Web, explicit metadata is needed in order to make meanings understandable to machines. The idea is to separate content semantics from structure and processing algorithms by using explicit semantic metadata about web resources. This is a natural next follow-up step after the XML-revolution, where the idea was to separate structure from presentation.

Metadata architectures [3] can be viewed from

- the *resource* or

- *client* view points.

From the resource perspective, metadata can be either *external* or *embedded*:

- External metadata is represented in separation from the resources it describes. For example, a separate yellow pages catalog can be created external to the actual services it lists.

- Embedded metadata is written within the document it describes. For example, in HTML documents the `<META>`-tag can be used for describing the content.

From the client perspective, metadescriptions can be either *centralized* or *distributed*:

- Centralized data is retrieved from a single source. For example, search engine indices are centralized metadata. An UDDI registry[3] stores company service profiles in a centralized manner.

[3] http://www.uddi.org/

- Distributed descriptions, such as web page annotations[4], are from the clients logical viewpoint distributed even if they were stored physically in the same server. From the resource perspective, annotations are external.

In the following, we first review approaches to representing metadata of web resources along these distinctions. After this the two major metadata language standardization efforts going on are in focus:

1. W3C develops Resource Description Framework (RDF) and RDF Schema (RDFS).
2. ISO develops the Topic Maps standard.

In conclusion, major application areas of metadata languages for web resources are discussed.

3.2 Classical Approaches to Web Semantics

The WWW has its roots in Ted Nelson's idea of hypertext [2]. It is a very "semantic" idea. Hyperlinks are semantic associations by which related contents of documents can be linked with each other in order to facilitate content-based information retrieval by "surfing". A problem soon found in the idea, and especially in its incarnation as the WWW, is that when information becomes abundant, one easily gets "lost in the hyperspace". Searching information by following one link at a time just is not efficient enough.

The standard solutions to the search problem on the web are *portals* and *search engines*.

3.2.1 Portals

Portals create semantic structure on the WWW by organizing its contents according to some specific view of interest, such as shopping, a hobby, a research area, etc. A prominent multi-portal is Yahoo![5] and dmoz[6]. Their ambitious goal is to give a classification structure to basically any topic on the web. The classification is created and maintained by an army of human administrators that organize web topics and pages into huge semantic hierarchies based on their content. The meaning to portal categorizations is given in the mind of human surfers and are not intended for machines to use.

A problem with Yahoo!-like portals is that the world cannot be represented within a single intuitive hierarchy. A hierarchy typically provides

[4]http://www.w3.org/2001/Annotea/
[5]http://www.yahoo.com
[6]http://dmoz.org

only one view of the data, and people tend to disagree on what categories should be used. In practice, the taxonomies grow huge and need lots of human work to create and to maintain.

3.2.2 Search Engines

Search engines are among the first web systems that make use of machine-processable semantics. The idea is to use a web crawler that visits systematically web pages, inspects their contents, and creates inverse indices mapping content keywords or other expressions to web pages. This gives explicit meaning to content expressions in terms of web addresses (URL). As a consequence, pages related to a query can be found quickly from billions of potential pages.

Keyword-based search methods are fast and often useful, but have several shortcomings [14]:

1. A keyword in a document does not necessary mean that the document is relevant. For an extreme example, a page may contain the phrase "This page is not about *politics*", which means that a search using the keyword *politics* will include the page in the hit list.

2. The engines cannot differentiate between *synonyms*. For example, pages discussing "Morning star" or "Evening star" are not found using the keyword "Venus". This lowers the recall[7] rate of information retrieval.

3. The engines do not understand *homonyms*[8]. For example, the keyword "Nokia" would find not only pages related to the Finnish telecom company, but also pages related to a city in Finland. This leads to low precision[9] in information retrieval.

4. The engines do not understand general terms or phrases. For example, if your are interested in finding out pages discussing "AI programming techniques" you probably have to be able to enumerate the techniques by name *beforehand*.

5. Relevance. It is difficult to evaluate the relevance of a document with respect to a query. A list of 10,000 hits is not very useful unless they can be ordered according to their relevance to the user.

[7]Recall r is defined as the ratio $r = f/a$, where f is the number of retrieved relevant documents and a the number of all relevant documents. [4]

[8]A homonym is a word with several meanings.

[9]Precision p is defined as the ratio $p = f/n$, where f is the number of retrieved relevant documents and n the number of all retrieved documents. [4]

6. Implicit information. A page is found only if it contains the explicit keyword. For example, one may be interested in Albert Einstein's work. A page describing the relativity theory is not found unless it happens to mention Albert Einstein.

7. Distributed information. Queries that refer to the contents on several pages are difficult to manage. For example, one may want to find all companies X that have business relations with another company Y. This information may be distributed in several ways on the pages of the companies involved.

Web documents are usually written in HTML or its later XML-based version XHTML. The content of such pages can be described by using a special tag `<META>`. Using this tag, search engines can be provided with explicit indexing keywords and other metadata that is more precise than what can be extracted from the document text.

The `<META>`-tag has attributes `NAME` and `CONTENT`. `NAME` attribute tells what kind of metadescription is given. `CONTENT` gives the actual contents of the description. For example, tagging

```
<META NAME="keywords" CONTENT="university, Helsinki"\>
<META NAME="content" CONTENT="Home page of the University of Helsinki"\>
```

enumerates content keywords for search engines to index the page, and a short natural language metadescription about the page.

Practice has shown that this nice facility is very often misused by content providers on the Internet. For example, a company may tag its product pages with keywords related to its competitors' products. Pages are often tagged with lots of irrelevant and wrong keywords in order to cheat web crawlers and end-users. As a result, `<META>`-tags are often simply ignored as untrustworthy[10] by search engines, which corrupts the whole idea. `<META>`-tags are of course more useful in intranets where content providers can be trusted.

In XHTML also links can be assigned with metadescriptions by using the Resource Directory Description Language RDDL[11]. Here the other end of a link can be described by attributes that tell its nature `xlink:role` (e.g., an XML DTD) and purpose `xlink:arcrole` (e.g., a home page).

Embedded distributed metadescriptions, such as `<META>`-tags, cannot be searched efficiently without aggregating them into a centralized repository. This can be, for example, a relational database, an XML or RDF(S) repository or a Topic Map (these metadescription languages will be discussed in

[10]This is an example of the general problem of "trust" on the web, which is one of the high level concerns of the Semantic Web Activity at W3C.

[11]http://www.rddl.org

more detail later). For this purpose, one needs a web crawler that systematically scans pages and extracts their metadata. Furthermore, contents from different sources must be transformed into a compatible vocabulary, formatted and be compiled together into data structures supporting fast information retrieval.

Search engines suffer from the problems of low precision and recall. However, even if these problems could be solved, the problem of long hit lists and need for manual list processing afterwards remains. It is difficult to determine the relative *relevance* of the hits.

Most search engines sort hit lists according to a relevance measure based on the counts of keywords found in the documents. However, the web linking structure implicitly encodes metadata concerning its content, which can be used as a source of additional information. In Google[12], for example, the relevance and trustworthiness of a document is evaluated based on the links pointing to it from other documents [9].

3.2.3 Logical Metadescriptions

`<META>`-tags in HTML can be used for expressing content keywords. The idea can be extended beyond keywords to more elaborate logical content descriptions. In the following we briefly describe one such system, OntoBroker.

Ontobroker[13] [15, 11] provides a metalanguage called $HTML^A$ for embedding annotations in HTML documents. Annotations are written using a new `onto`-attribute in the anchor element `<A>`.

The annotation language used in the tag is a frame-based logic language. When creating an application, one first defines the concepts related to its contents by an ontology. A web page can then be annotated according to this ontology by creating its class instances and objects, and by assigning values to their attributes. URLs are used to identify objects.

For example, the following tagging tells that John Smith, identified by his home page, is an instance of class Employee and has the given email address given as an object property:

```
<a onto='"http://www.helsinki.fi/john.smith/":Employee'></a>
<a onto='"http://www.helsinki.fi/john.smith/"
   [email:?mailto:john.smith@helsinki.fi]'></a>
```

Given a set of web pages that have been annotated using the ontology, a web crawler is used to extract the embedded metadescriptions, to parse them, and to add them into a global centralized repository. The user can then make queries to the repository using a logical *query languge*, which is a subset of the language used for defining the underlying ontology. The query responses

[12] http://www.google.com

[13] http://www.aifb.uni-karlsruhe.de/www-broker

are generated by a logical *inference engine*; they are value substitutions of the variables in the query, like in logic programming languages, such as Prolog.

OntoBroker makes use of a semantically rich logic language for annotating web pages. Best known current systems using the same idea include OIL[14] (Ontology Inference Layer) and DAML[15] (DARPA Agent Markup Language) and their combination DAML+OIL being standardized at W3C. These languages are based on a lower level Semantic Web data model, Resource Description Framework (RDF) and RDF Schema (RDFS). In the following, RDF and RDFS are briefly discussed.

3.3 Resource Description Framework

3.3.1 RDF Model and Syntax

Data Model

Resource Description Framework (RDF) [24, 19, 3] is a generic scheme for representing metadata about web *resources*. A resource is identified by its URI (with an optional anchor identifier). A resource can be anything accessible on the web, such as a HTML page, its parts, a picture, email address etc. For example, "http://www.helsinki.fi" is the URI identifying the web site of the University of Helsinki. A resource may also be an object not accessible through the web, such as a work of art.

An RDF description essentially consists of ¡resource, property, value¿ triples. Each triple assigns a named *property* (attribute) with a *value* to a *resource*. For example, the fact that the University of Helsinki is located in Finland can be expressed by the triple below, where "Location" is the property with the value "http://www.finland.fi", the URI identifying the country.

```
<http://www.helsinki.fi, Location, http://www.finland.fi>
```

A triple can intuitively be interpreted as a directed arc that points from a resource to another resource or label (value) with a labeled arc, the property. In this view, an RDF description — a set of triples — constitutes a *directed graph*, whose nodes and arcs are labeled with URIs or literal symbols.

In addition, there is still a kind of linguistic interpretation of the idea of the RDF triple as a *statement*, where the resource to be described is called the *subject*, the property is the *predicate*, and the value is the *object*. In the statement "The University of Helsinki is located in Finland", "University of Helsinki" is the subject, "is located in" is the predicate, and "Finland" is the object.

[14] http://www.ontoknowledge.org/oil/
[15] http://www.daml.org

Syntax

The RDF data model can be represented in many syntactic ways. The original RDF specification [24] already defines two alternative XML-based syntaxes: *serialization syntax* based on nested element structures and *appreviated syntax* in which a subset of the data model can be represented in a more compact form by using XML attributes.

In addition to these two syntaxes, triple-based notations[16] reflecting more directly the underlying data model have been developed.

The following is an example showing an RDF description of a fictional resource located at `http://www.helsinki.fi/john.smith/paper.html`.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:DC = "http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about = "http://www.helsinki.fi/john.smith/paper.html">
        <DC:Title>Using RDF to describe web resources</DC:Title>
        <DC:Creator>John Smith</DC:Creator>
        <DC:Date>2001-01-01</DC:Date>
        <DC:Subject>RDF, Metadata, Semantic Web</DC:Subject>
    </rdf:Description>
</rdf:RDF>
```

Two namespaces are used in the example: the RDF namespace (rdf) and the Dublin Core namespace (DC) that defines a set of basic properties of documents. Since a property name in different application areas may have different meanings, the property must be identified uniquely with exactly one XML namespace. The whole RDF description is given as a set of `rdf:Description`-elements within the `<rdf:RDF>`-tag. The `about` attribute identifies the resource that the `rdf:Description`-element describes. The subelements `DC:Title`, `DC:Creator`, etc. are the properties of the resource and their data are the corresponding values. Here the `rdf:Description`-element boils down to four statements (triples). This example is illustrated as a graph in figure 3.1.

RDF can be embedded in HTML by using the abbreviated syntax. This format ensures that a browser which does not support RDF metadata will not print the metadata as a part of the text of the page. In the abbreviated format, all metadata items are presented as attributes inside the `Description` element, as in the following example.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:DC = "http://purl.org/dc/elements/1.1/">

    <rdf:Description
        rdf:about = "http://www.helsinki.fi/john.smith/paper.html"
        DC:Title = "Using RDF to describe web resources"
```

[16]For example, the Notation 3 (N3), `http://www.w3.org/2000/swap/Primer.html`, and N-Triples, http://www.w3.org/2001/sw/RDFCore/ntriples/.

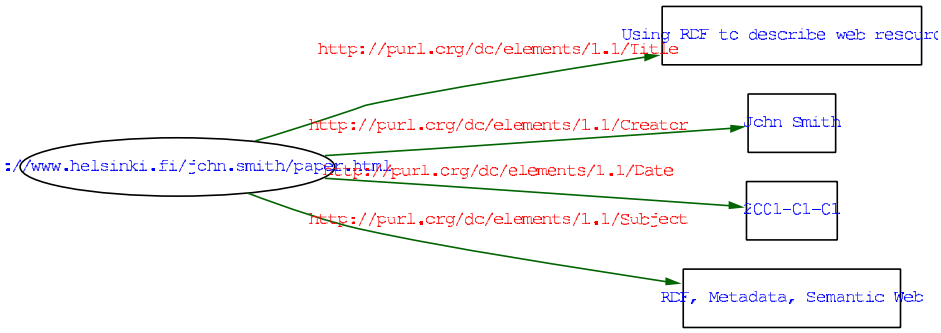


Figure 3.1: RDF metadata concerning John Smith's paper. The graph is produced by the W3C RDF Validator: `http://www.w3.org/RDF/Validator`.

```
        DC:Creator = "John Smith"
        DC:Date = "2001-10-02"
        DC:Subject = "RDF, Metadata, Semantic web"
    />
</rdf:RDF>
```

There are initiatives for implementing RDF support in browsers, for example, in Mozilla. In addition, some experimental web search services claim to support RDF metadata to some extent. They will be facing the problem encountered earlier with the `<META>`-tagged information on HTML pages: webmasters are tempted to include cheating keywords to their web site's meta information in order to make their URL appear more often in search results.

Additional Elements of the RDF Model

RDF has some still some additional constructs that increase the power of the basic model:

Containers RDF offers the possibility of using collections of resources by using *bags* (unordered list), *sequences* (ordered list), and *alternatives* (choice list). A sequence, for example, gives the possibility to define several authors for a document if their order of appearance is relevant. An alternative can define, e.g., a set independent sites for downloading a document.

Reification It is possible to make statements about statements, i.e., express higher-order metadata about metadata. Such a statement is called *reified*. For example, it is possible make the reified statement "John Smith believes: 'Helsinki is located in Sweden' " without asserting that "Helsinki is in Sweden", which would not be true.

3.3.2 RDF Schemas

The RDF model makes it possible to assign named property values to resources. There are, however, no mechanisms for defining such properties, classes of resources to which the properties apply, or constraints on using the properties with them. These extensions to the RDF model are defined in the RDF Schema (RDFS) specification of W3C [8]. This candidate recommendation does not specify domain specific classes or properties, such as "Date" or "Creator", but generic mechanisms for defining such vocabularies. RDFS itself is a specification written in RDF.

Core Classes and Properties

In RDF, resource instances belong to one or more classes. The instance-class relationship is expressed by using the property `rdf:type`.

RDF Schema introduces a type system of hierarchical classes into RDF in the same spirit as classes are used in object-oriented programming languages, such as C++ or Java. The recommendation defines a small set of generic *core classes* and *core properties* by which the user can define her/his own vocabularies for different applications.

The core classes are:

- `rdfs:Class`, the general notion of the class.
- `rdfs:Resource`, the class of resources being described.
- `rdf:Property`, the class of RDF properties.

The set of core properties includes, e.g., the following:

- `rdf:type`, for indicating the class of a resource instance
- `rdfs:subClassOf`, for indicating the superclass of a class

In the following example, an application RDF Schema for defining the class `Mammal` with subclasses `Bear` and `Wolf` is presented. These classes are resources identified by the ID attribute values in the `rdf:Description` elements. The definitions are based on the core classes `Class` and `Property`.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:ID="Mammal">
  <rdf:type rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf
      rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>
```

```
<rdf:Description rdf:ID="Bear">
  <rdf:type rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Mammal" />
</rdf:Description>

<rdf:Description rdf:ID="Wolf">
  <rdf:type rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Mammal" />
</rdf:Description>

</rdf:RDF>
```

In RDFS, also properties form a class hierarchy. Property hierarchy is expressed by the core property `subPropertyOf`. In the following example, property `Mother` is a subproperty of the the more general `Parent` property.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:ID="Parent">
  <rdf:type rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Property" />
</rdf:Description>

<rdf:Description rdf:ID="Mother">
  <rdf:type rdf:resource="http://www.w3c.org/2000/01/rdf-schema#Property" />
  <rdfs:subPropertyOf rdf:resource="#Parent"/>
</rdf:Description>

</rdf:RDF>
```

Constraints

RDFS also makes it possible to express constraints on using properties with classes. A small set of *core constraints* are defined for the purpose:

- `rdfs:ConstraintProperty`,the general class of properties used to specify constraints.
- `rdfs:range`,indicates the value range of a property as a class.
- `rdf:domain`,the class on whose members a property can be used.

In the following example, the value of the `Author` property of an instance of the class `Book` is restricted to the instances of the class `Person`.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:ID="Author">
  <rdf:type
     rdf:resource="http://www.w3c.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#Book"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Description>

</rdf:RDF>ssss
```

The idea of using constraints is to define properties in terms of classes on which they apply. This property-based approach makes it easy to create new properties for describing existing resources. One thing that is missing in the recommendation is the definition of atomic data types, but these will probably be provided by the W3C effort concerning XML data typing.

An RDF schema defines a vocabulary within an application area based on classes, properties, and constraints on their usage. The definition is based on the core classes, properties, and constraints of the candidate recommendation. Vocabularies defined and maintained by different organizations or persons share the common definition mechanism. It is therefore possible to merge different vocabularies and metadata descriptions in order to describe data in a semantically richer setting.

RDFS extends the RDF model with an object-oriented approach to resource description. By building complex class hierarchies the developer is able to express application knowledge in more detail. This makes it possible for the machine to process the metadata in a more efficient and meaningful way.

RDF Model Theory

Recently, a specification of model-theoretic semantics for RDF(S) (excluding reification and containers) has been presented [13]. It's goal is to provide a precise semantic theory for RDF(S). With such a rigid basis, it is possible to define and analyze the semantic properties of RDF(S), such as the logical notions of consequence and inference. An RDF graph can be interpreted in terms of logic. If an arc labeled with p connects a node to another, then it maps directly into the atomic assertion that relation p holds true between the expressions corresponding to the nodes.

3.3.3 Examples of RDF in Practice

The real value of RDF(S) will be evaluated in real-world applications that ease our lives in one or another way. In the following, some examples of RDF(S) applications are given.

Dublin Core

One widely used schema is the Dublin Core metadata standard [31]. It defines a basic set of elements commonly needed in a wide variety of application domains, especially in describing document-like resources. Dublin Core was not originally an RDF schema but a collection of metadata elements used in library science. The original Dublin Core Element Set is described in [22].

Dublin Core is being developed by the Dublin Core Metadata Initiative (DCMI). The name is due to a workshop held in Dublin, Ohio, in 1995.

DCMI is "an organization dedicated to fostering the widespread adoption of interoperable metadata standards and promoting the development of specialized metadata vocabularies for describing resources to enable more intelligent resource discovery systems" [33].

The Dublin Core vocabulary consists of two types of terms, *elements* and *qualifiers*. The element set contains elements such as Title, Subject, Creator, Publisher, Date, Language, etc. There are altogether fifteen core elements developed for a wide variety of application domains and disciplines. The elements are described in [31] using ISO/IEC 11179 -defined attributes. For example, the core element Title is defined as follows:

```
Element: Title

Name: Title
Identifier: Title
Definition: A name given to the resource.
Comment: Typically, a Title will be a name by which the
         resource is formally known.
Version: 1.1
Registration Authority: Dublin Core Metadata Initiative
Language: en
Obligation: Optional
Datatype: Character String
Maximum Occurrence: Unlimited
```

Qualifiers are terms that either refine basic elements or identify encoding schemes, such as date formats. If a system does not recognize a qualifier, it can treat the data ignoring the qualifier and assume that it might still be useful for a human reader. This "dumb-down" principle is one cornerstone of the Dublin Core model. Additional or more accurate metadata can be expressed with the aid of qualifiers, but the interoperability should never be endangered. Any additional qualifiers may be ignored and the metadata is still useful and understandable. The qualifiers are described in [32].

DCMI has been very careful with their model to retain it as useful as possible in different areas of industry and science.

vCard Schema

One vocabulary in use is vCard [10, 20] which basically defines the elements that are used in common business cards. This basic set of information can be used in applications requiring descriptions about persons. The vCard-specified elements can easily be translated into an RDF schema. The RDF model of vCard represents all vCard elements as RDF properties. These include FN (Full Name), NICKNAME, TITLE, and EMAIL, to name a few. Elements may have container values, such as multiple e-mail addresses. The vCard schema also gives the possibility to use images as passport photos in the cards.

RDFPic

The RDFPic software[17] is used to attach metadata to digital photos. RDFPic follows the W3C note for describing and retrieving photos using RDF and HTTP [23], and is in fact an example implementation of that note. The idea behind RDFPic is that attached metadata would automatically provide a non-visual, textual representation of an image for search purposes. Traditional search techniques do not work with binary image data. A goal of this example application is to explain and demonstrate the power of metadata on the web.

JPEG standard allows for an unlimited amount of 64-kilobyte comment blocks to be attached to image files. RDFPic exploits this possibility and embeds the RDF metadata directly into the JPEG file. RDFPic uses a combination of it's own elements and elements from the Dublin Core schema to describe the photos. Dublin Core elements describe the photo as a document. The additional elements describe technical photographic details and content classification, such as the lens used, development method, and whether the picture is a portrait or a landscape shot.

The following is an example of the metadata produced by the RDFPic software.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:s0="http://purl.oclc.org/dc/documents/rec-dces-199809.htm#"
   xmlns:s1="http://sophia.inria.fr/~enerbonn/rdfpiclang#">
   <rdf:Description rdf:about="http://www.pictures.com/pic1.jpg">
     <s1:xmllang>en</s1:xmllang>
   </rdf:Description>
   <rdf:Description rdf:about="#image">
     <s0:_Subject>Group-Portrait</s0:_Subject>
     <s0:Title>The team</s0:Title>
     <s0:Description>Group shot of the whole team.</s0:Description>
     <s0:Type>image</s0:Type>
     <s0:Format>image/jpeg</s0:Format>
     <s0:Creator>John Smith</s0:Creator>
   </rdf:Description>
 </rdf:RDF>
```

In addition, the Jigsaw server is employed in order to serve metadata and photos, depending on the queries that the web clients make.

3.4 Topic Maps

Today we are facing the problem of the huge information masses with rich semantic content such as the World Wide Web. Without efficient ways to

[17]The software is freely downloadable at `http://jigsaw.w3.org/rdfpic/` with source code and executables.

organize the information, we eventually end up with inability to find information at all. One tool to organize information masses for meaning-based information retrieval is the Topic Map model [26, 6, 21].

A topic map[18] is a network of topics with connections between topics themselves and between topics and information resources. The goal is to create a structure for the information resources and make it easier to navigate between the resources.

This navigational network constitutes an external layer over the information resources and does not affect the information resources. This separation between topic maps and the resources gives the possibility to create multiple topic maps for the same information resources and also to use topic maps as *portable semantic networks*, that can be overlaid on multiple information pools or even used without the information resources.

The key elements of topic maps are topics, associations and occurrences.

Topics

The most central concept of Topic Maps is the *topic*. A topic is an internal representation for an subject, where the subject can be anything (e.g. physical things like the tower of Eiffel or immaterial things like the European Union). For example, if the subject is the country "Finland", then the topic "Finland" is an internal marker for this external (physical) subject.

The optimal situation would be, that every subject is represented by one topic, and vice verse.

Topics can be thought of as hubs to which everything about the subject of the topic is connected. For example, all publications about Finland can be thought of as *occurrences* of the topic "Finland". Other topics, such as the topics for the countries "Sweden" and "Norway" have occurrences of their own.

A topic is defined by its *characterizations* such as the name of the topic, associations with other topics, and connections to the occurrences.

A topic can have any number of *names* in any languages and contexts connected to it. These names indicate different views of the same topic. It is also possible to have a topic without a name, which is useful if there are things, topics, or occurrences that are related but there is no meaningful explicit name for the collection.

Each topic has an attribute called *type*. It describes what kind of topic is in question. For example, if "Finland" is the subject of a topic in a topic map, then "country" could be its type. The types are also defined as topics.

[18] We use capital letters (Topic Map) when we refer to the Topic Map model and lower case (topic map) when we refer to an instance of the model.

Associations

A connection between two topics is called an *association*. It indicates that the topics somehow relate to each other. Associations can have a *type* just like the topics have types.

For example, we can say that Helsinki "is the capital of" Finland, which is useful information as such. This association can be contained in a topic map relating topics Helsinki and Finland and resources discussing these topics, but without affecting their content. Some common types of relations between topics x and y are that topic x "is a part of" topic y and that topic x " is a subclass of" topic y.

The notion of association is central in topic maps. It is also the reason why topic maps can be thought a a kind of portable semantic networks [30] that are useful even without the information resources beneath the topic map. By following associations from one topic to an other the user can navigate through the information, like on the WWW. The difference between Topic Maps and the WWW is that the relations between topics of a topic map are defined whereas the links between Internet resources are not. Therefore the navigation of a topic map is more predictable.

Occurrences

Information resources that are external to the topic map, but contain relevant information to the subject of some topic can be linked from the topic. Such information resources are called *occurrences*. Occurrences can be, for example, web pages and text documents, photographs, software, topic maps etc. The only thing that these resources must have is a URI address identifying it, so that linking them is possible.

Occurrences can also have types, that are called *occurrence role type*. The types can be, for example, "article" or "illustration". They are defined as topics just like the types of topics and associations.

Scope

Topic characteristics, i.e., names, occurrences, and associations, can have different meanings in different contexts. For example, the name "Java" might mean either the island or the programming language. In topic maps the *scope* can be used to establish contexts for distinguishing the island from the programming language topic. The scope of the island could be "geography" and the scope of the programming language could be "computer science".

A scope can be specified either explicitly as a set of topics, or implicitly, in which case it has unconstrained scope [18] containing all the topics in a topic map.

The figure 3.2 illustrates the use of topics, names of the topics, associations between topics, and occurrences of the topics.

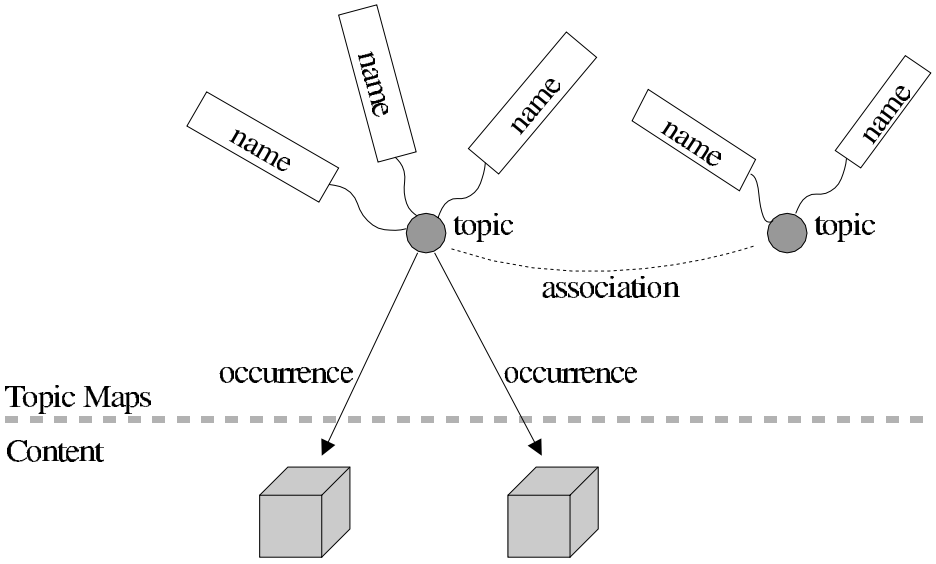


Figure 3.2: The Topic Map model

3.4.1 Topic Maps and XML

The Topic Map model described above is formally described in the ISO 13250 [21] standard, that was published in 2000. The standard is based on SGML, which is a more complex and less known markup language than XML. Therefore the working on an XML-based specification of the topic map model began immediately after the publishing of the ISO specification. It is called XML Topic Maps (XTM) [18].

The goal of the XTM specification[19] is to enable using topic maps on the Internet and to make it easier for the users to use topic maps. The XTM specification is compatible with XML, XLink, and the ISO 13250 specifications.

The following example shows how topics, associations, and occurrences are described in XTM. (Basic knowledge of XML [7] is needed to understand the example.)

An XTM Example

The map represents some key information about Finland:

[19]Available at: http://www.topicmaps.org/xtm/1.0/

- Finland is the international name of the country, while the name in Finnish is "Suomi" or more officially "Suomen tasavalta".
- The country code is "fin" and the "home page" of Finland is the Virtual Finland web page, located at http://www.finland.fi.
- The capital of Finland is Helsinki.

This information can be described with XTM in the following way.

```
<!-- A new topic with the id ''fin'' -->

<topic id=''fin''>

<!-- The type of the topic is ''country''. -->

  <instanceOf>
    <topicRef xlink:href=''#country''/>
  </instanceOf>

<!-- The name of the topic is in Finnish ''Suomen Tasavalta''
     and ''Suomi''.?-->

  <baseName>
    <scope>
      <topicRef xlink:href=''fi''>
    </scope>
    <baseNameString>Suomen tasavalta</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <topicRef xlink:href=''fi''>
    </scope>
    <baseNameString>Suomi</baseNameString>
  </baseName>

<!-- The name of the topic is in English ''Finland''. -->

  <baseName>
    <scope>
      <topicRef xlink:href=''en''>
    </scope>
    <baseNameString>Finland</baseNameString>
  </baseName>

<!-- An occurrence of the topic: the web page called Virtual Finland. -->

  <occurrence>
    <instanceOf>
      <topicRef xlink:href=''#html-version''/>
    </instanceOf>
    <resourceRef xlink:href=''http://www.finland.fi/''/>
  </occurrence>
</topic>
```

Now we make an association between Finland and the capital of Finland, Helsinki. The topics "Helsinki" and "capital-of" would be declared in the

same way as the topic "Finland" earlier.

```
<!-- The id of the association is ''helsinki-finland''. -->

<association id=''helsinki-finland''>

<!-- The type of the association is ''capital-of''.-->

  <instanceOf>
    <topicRef xlink:href=''#capital-of''/>
  </instanceOf>

<!-- The first of the two topics associated here is a
     ''city'' with the id ''helsinki''. -->

  <member>
    <roleSpec>
      <topicRef xlink:href=''#city''/>
    </roleSpec>
    <topicRef xlink:href=''helsinki''/>
  </member>

<!-- The second part of the association is a ''country''
     with the id ''fin''. -->

  <member>
    <roleSpec>
      <topicRef xlink:href=''#country''/>
    </roleSpec>
    <topicRef xlink:href=''fin''/>
  </member>
</association>
```

3.4.2 Creating and Using Topic Maps

Creating Topic Maps

Creating and managing topic maps can be either done manually, partially automated or automatically.

The simple manual way for creating topic maps is to use a text editor, such as Emacs or Notepad. Partially automated tools don't exist yet, but they could be "Topic Map aware" in a similar way to HTML editors that know the syntax of HTML and can assist the user in creating web pages [3].

A partially automated approach to creating topic maps is to use existing data repositories and to write (manual part) a program that translates (automatic part) the information into a topic map. Such program can be an effective way to creating large maps, but such a program can probably be used in just one application without rewriting it.

A solution to create automatically topic maps is to use RDF as the source and translate the RDF information into a topic map. This should be possible, because both RDF and topic maps are created for describing relationships

between entities with identity. Some solutions using this idea are presented in [25, 3].

Merging Topic Maps

One underlying idea of the topic map model is that there is nothing wrong with different world views. For example, a geographical topic map probably differs from a computer science related topic map even if there are some common names of the topics, such as "Java". It is more practical to create several topic maps each with one defined goal than a single, huge topic map that would contain the relations that exist in the world around us.

If the information needs of the user are the same as the scope of the topic map, then a single topic map may contain enough information. It is, however, possible to merge topic maps so that two (or more) topic maps can be used together. For example, if a geographical topic map and a computer science related topic map are merged, it would be possible to tell that "Java" is a computer language and that it is also an island. If a coffee related topic map would be also merged with the previously mentioned topic maps, then it would be possible also to tell, that coffee from "Java" is tasty.

Viewing topic maps

After a topic map has been created, a rendering software is needed to visualize it for the user and give the user the possibility to navigate around the map.

One way to implement a navigator application is to create it as a web service, where every topic is presented as a web page with links to associated topics and resources of the selected topic. By clicking on the links the user can navigate through the map and find out the desired information.

The topic maps can also be visualized as graphs, that can be navigated more visually (with the mouse) than the web-based solution above. For example, it can be possible to zoom and rotate the network, expand interesting parts of the networks, and hide less interesting parts [17].

We tested the Ontopia Topic Map Navigator[20] with our previous XTM example. With the Ontopia Navigator it is possible to browse XTM files with a normal web browser. By using a predefined rendering scheme, the HTML-pages of figures 3.3 and 3.4 could be created. A separate web page for each different topic can be created. On topic pages, links to related occurrence resourced as well as association links to related topic pages can be found. Information retrieval can be accomplished by surfing the associations. The product is implemented as a Java Servlet using Java Server Pages -technology (JSP) [16]. With a tagging language [29], one can create customized rendering schemes for different applications. The tool contains

[20]Available from: http://www.ontopia.net/

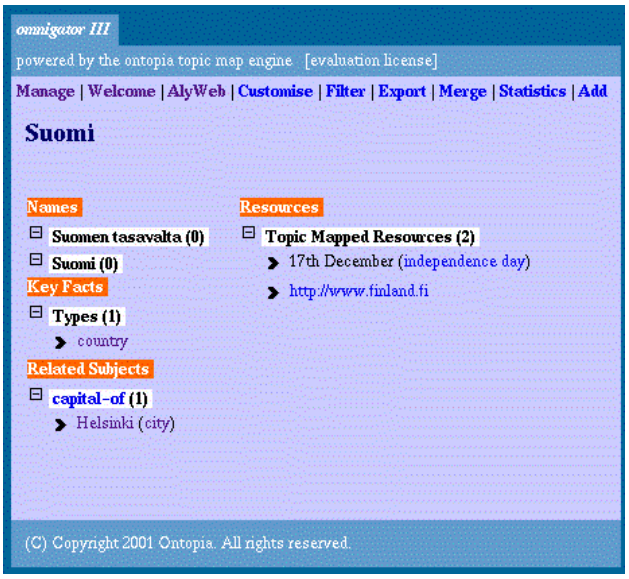


Figure 3.3: Ontopia Navigator showing the topic "Suomi" (Finland).

lots of additional features, such as filtering mechanisms, topic map merging, and a query language.

3.5 Application Areas

In this paper metalanguages for web resource description have been discussed. In conclusion, some of the most important application areas of the technologies are listed.

Enhancing Search Capability

Semantic metadata attached to resources would enhance recall and precision rates of search engines and web crawlers. For example, consider the problem of finding documents written *by* John Cook. The keywords `John Cook` can be used in the search but will generate lots of extra documents *about* John Cook (not to mention pages related?tURo cooking). If the documents were tagged with the Dublin Core[21] `Creator` property, the semantic distinction *by* vs. *about* can be made.

Semantic Navigation

Knowing the meaning of documents makes it possible to enhance navigational capabilities of computer systems. Semantic metadata can be used for creating

[21] http://www.dublincore.org

Figure 3.4: Ontopia Navigator showing the metaindex of the underlying XTM topic map.

semantic portals, intelligent indexes, directories and catalogs in which the needed resources can be found by semantic descriptions natural to the human user. In a semantic catalog, related categories can be linked with each other based on their content.

Personalization

Metadata on user interest is required for high quality personalized services. For example, Platform for Privacy Preferences (P3P)[22] is an RDF-based standard by which a browser (e.g., Internet Explorer 6.0) can compare the privacy policy of a web site with the preferences of the user. The user may require, for example, that the service may not use email addresses for direct marketing. First major company sites supporting P3P include, e.g., Microsoft[23], ATT [24], and HP[25].

[22] http://www.w3c.org/P3P/
[23] http://www.microsoft.com
[24] http://www.att.com
[25] http://www.hp.com

Device Profiling

Communication between the various kind of agents connected to the Internet is constrained with the hardware, software, and data connections available. For example, it does not make sense to download a video stream into a cellular phone if the video cannot be displayed in it. A way is needed for describing the capabilities of devices, their software configurations and communicational capabilities. Composite Capabilities/Preference Profile Specification (CC/PP)[26] is an RDF-based standardization effort for this purpose.

Processes and Services

Process descriptions, e.g., a description of a web service, are an important type of metadata. Exchanging metainformation on processes is often needed when integrating applications with each other. For example, a shopbot application requesting a service may need to know when and how the service will be provided. A process description tells (1) a sequence of actions to be performed for a given goals and (2) defines how such actions are accomplished.

Standardization on process descriptions is difficult. However, standards are desperately needed, e.g., in eCommerce applications, and several international projects are going on:

- Project Specification Language (PSL)[27] is an ISO standard proposal (ISO 18629) of a collaborative project led by the National Institute for Standards and Technology (NIST).
- Unified Modeling Language[28] (UML) developed by Object Management Group[29] contains description of process semantics and the standard XML metadata Interchange (XMI) for representing processes in XML.
- W3 Consortium is working on a standard DAML-S[30] for describing web services. XML-languages are a natural choice for representing processes and services.
- Workflow Management Coalition[31] (WfMC) has specified XML Process Definition Language (XPDL) for the exchange of workflow process specifications.

[26] http://www.w3c.org/mobile/CCPP/
[27] http:www.nist.gov/psl
[28] http://www.omg.org/technology/uml/
[29] http://www.omg.org
[30] http://www.daml.org/services/daml-s
[31] http://www.wfmc.org

Annotations

By annotations one can add external side notes to web documents for others to view and use without modifying the annotated data. For example, a group of developers could jointly add comments to a set of web pages under development. Or the trustworthiness of web services could be evaluated by an authority for others to use. W3 Consortium has a special Annotea project[32] for developing annotation mechanisms and annotations are a focus of the Text Encoding Initiative[33] (TEI).

3.6 Conclusions

The RDF model, extended with RDF Schema is a powerful, general way of expressing metadata about web resources. RDF can be expressed in XML syntax, which makes it easy to use in the cross-platform environments of the web. It is possible to merge multiple RDF graphs into one and use vocabularies defined in several different schemas.

RDF Schema enriches the RDF model with an object-oriented class system and a constraint definition mechanism for properties; RDFS provides a means of defining domain specific vocabularies.

The topic map model has its roots in library systems and originates from the early 90's. It can be seen as a kind of semantic generalization of book indices. The idea is to provide a tool by which contents of semantically rich and abundant materials can be organized in a meaningful way for search purposes. This background and original motivation is different from RDF(S) that has its roots in computer science, artificial intelligence, and web technologies. RDF(S) is more technical in nature and backed with more academic research. Both systems are not bound to any particular application domain and their serialization is (usually) based on XML.

RDF(S) and Topic Maps are explicit representations of metadata. From the resource perspective, RDF is used both externally and embedded within the documents they describe while Topic Maps focus on external usage. From the client perspective, RDF and Topic Maps may be distributed and be merged into a centralized repository.

There are many ways of using metadescriptions. A simple way is to embed RDF into the resources themselves, such as HTML or PDF documents or JPG images. Another way is to provide the descriptions as independent documents in an external repository, from which they can be fetched by the applications for different usages.

There are lots of important application areas and possibilities for metadata on the web, such as semantic navigation, personalization, device profil-

[32] http://www.w3.org/2001/Annotea/

[33] http://www.tei-c.org

ing, annotations, and product, service, and process descriptions for business and commerce. Search engines would become more accurate if there were metadata widely used on the web and collected by web crawlers. However, a way of evaluating the trustworthiness for distributed metadescriptions is needed.

From the practical application viewpoint, an important step towards global usage of metadata on the web is Adobe's decision (2001) to support RDF metadata within it's Extensible Metadata Platform (XMP) technology [1]. It is already supported in many Adobe's products, including Acrobat 5.0, InDesign 2.0, and Illustrator 10 [28]. First start-up companies and commercial applications of both RDF[34] and Topic Maps[35] are already there on the market, and lots of standardization work is going on within various industries.

[34]E.g., http://www.profium.com
[35]E.g., http://www.ontopia.net/

Bibliography

[1] Adobe. *Extensible Metadata Platform (XMP). http://www.adobe.com/products/xmp/main.html*, 2001.

[2] M. Agosti and A. Smeaton, editors. *Information retrieval and hypertext.* Kluwer, New York, 1996.

[3] Kal Ahmed, Danny Ayers, Mark Birbeck, Jay Cousins, David Dodds, Joshua Lubell, Miloslav Nic Daniel Rivers-Moore, Andrew Watt, Robert Worden, and Ann Wrightson. *Professional XML Meta Data.* Wrox Press Inc., 2001.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, New York, 1999.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[6] Michel Biezunski and Steven R. Newcomb. XML Topic Maps: Finding aids for the web. *IEEE*, 8(2):104–108, April-June 2001.

[7] N. Bradley. *The XML Companion.* Addison-Wesley, 2000.

[8] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[10] F. Dawson and T. Howes. IETF RFC 2426 – vCard MIME directory profile, 1998. http://www.imc.org/rfc2426.

[11] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology based access to distributed and semi-structured unformation. In *DS-8*, pages 351–369, 1999. citeseer.nj.nec.com/article/decker98ontobroker.html.

[12] H. Deitel, P. Deitel, and H. Nieto. *XML. How to program.* Prentice Hall, New Jersey, 2001.

[13] P. Hayes (ed.). RDF model theory, September 2001. W3C Working Draft, http://www.w3.org/TR/rdf-mt-20010925/.

[14] D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce.* Springer-Verlag, 2001.

[15] Dieter Fensel, Stefan Decker, Michael Erdmann, and Rudi Studer. Ontobroker: The very high idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida*, 1998.

[16] D. K. Fields, M. A. Kolb, and S. Bayern. *Java Server Pages.* Manning Publications Co., 2002.

[17] Bénédicte Desclefs-Le Grand and Michel Soto. Visualizing topic maps. In *XML Scandinavia 2000*, 2000.

[18] TopicMaps.Org Authoring Group. XML Topic Maps (XTM) 1.0, TopicMaps.Org Specification, 2001. http://www.topicmaps.org/xtm/1.0/.

[19] J. Hjelm. *Creating the Semantic Web with RDF.* John Wiley & Sons, New York, 2001.

[20] R. Iannella. Representing vCard objects in RDF, 2001. http://www.w3.org/TR/vcard-rdf.

[21] ISO/IEC. ISO/IEC 13250, Topic Maps, 1999. http://www.y12.doe.gov/sgml/sc34/document/0129.pdf.

[22] J. Kunze. IETF RFC 2413 – Dublin Core Metadata Initiative, 2001. http://www.ietf.org/rfc/rcf2413.txt.

[23] Y. Lafon and B. Bos. Describing and retrieving photos using RDF and HTTP, 2000. http://www.w3.org/TR/photo-rdf.

[24] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[25] Graham D. Moore. RDF and Topic Maps — an exercise in convergence. XML Europe 2001, May 2001.

[26] S. Pepper. Topic Maps and RDF: A first cut, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.

[27] G. Riccardi. *Principles of database systems with Internet and Java applications*. Addison-Wesley, 2001.

[28] Janne Saarela. Semantic Information Router (SIR). In *Semantic Web Kick-Off in Finland*, 2002.

[29] G. Shachor, A. Chase, and Magnus Rydin. *JSP Tag Libraries*. Manning Publications Co., 2001.

[30] J. Sowa. *Knowledge representation. Logical, philosophical, and computational foundations*. Brooks/Cole, 2000.

[31] Dublin Core Workgroup. Dublin core metadata element set 1.1, reference description, 1999. http://dublincore.org/documents/1999/07/02/dces.

[32] Dublin Core Workgroup. Dublin Core qualifiers, 2000. http://dublincore.org/documents/2000/07/11/dcmes-qualifiers/.

[33] Dublin Core Workgroup. Dublin Core Metadata Initiative frequently asked questions, 2001. http://dublincore.org/resources/faq/.

Chapter 4

XML, RDF(S) and Topic Map Databases

Vilho Raatikka, Karru Salminen, and Eero Hyvönen

The web is huge repository of documents whose form is optimized for presenting them to humans with browsers. In order to facilitate better data management and more accurate information retrieval, web languages for representing content instead of layout are needed. In this paper, we will discuss such languages from the database perspective, especially XML, RDF(S), and Topic Maps. The basic questions addressed are how to store documents written in these languages, and how to retrieve information from such repositories.

4.1 The Web as a Data Repository

The HTML Revolution

One of the key ingredients of the WWW is the Hypertext Markup Language (HTML). Its first draft was presented in 1993 by Tim Berners-Lee and Daniel Connolly. They proposed the language for publishing written documents, news, email, and hypermedia locally on the web after which the contents could be viewed by a HTML browser globally through the Internet.

The popularity of HTML increased rapidly and led to the situation where millions of individual persons, companies, and other organizations are using the Internet for distributing HTML- and other documents. By the end of the year 2001, there were already some two billion static web pages on the web, and the "hidden web" of dynamically generated pages is much larger. The web is a huge distributed repository of unstructured data whose structure and contents are rapidly evolving. Retrieving needed information from there has become an evermore serious problem.

From the information retrieval viewpoint, a major problem of the web is that HTML does not describe the meaning, semantics, of web pages, but only its proposed layout. The lack of semantics means that the query mechanisms for the web cannot make use of an underlying data model in the same way as, for example, SQL is based on the relational data model [29]. As a result, information retrieval on the web has to be based on simple free text indexing techniques [24, 4] that map keywords to web pages in which they occur. The query forms for search robots are typically boolean expressions of keywords. Such expressions cannot necessarily identify contents accurately, which leads to the well-known problems of low precision and recall rates in web queries [16]. Both data and the user get "lost in the hyperspace". It seems that database techniques for the WWW would be of use, and lots of research is going on towards this direction [18].

The Need for Database Techniques

In order to facilitate better data management and more accurate information retrieval, web languages for representing content instead of layout are needed. In this paper, we will discuss such languages from the database perspective, especially XML [8], RDF(S) [21], and Topic Maps [28]. The basic questions addressed are how to store documents written in these languages, and how to retrieve information from such repositories.

Structured content data coded in these languages enables, among other things, the creation of more powerful search capabilities, intelligent software agents, and better content management. To acquire these benefits, efficient access to data must be provided. Efficiency means the possibility to search and update the necessary data without interference of other possible users. One user should not affect the system so that the data is not accessible for other users at the operating time. The data must remain consistent even if multiple updaters are operating the data at the same time. The data repository must also be able to handle large amounts of data. One useful feature would be interoperability which would enable data integration of separate repositories without changing the data management system.

Web document repositories are often large and continuously evolving. For example, an XML repository can describe the behavior and contain the documentation of a large system, such as a paper machine, or a product catalog of a company. Updating individual documents or parts of them becomes very hard if it must be done manually to files. The file which consists of the wanted document must be locked from other users even if only one element were being updated. The database management system avoids these pitfalls and offers usually other usable features, such as application programming interfaces (API) for several languages, support for network protocols, and graphical user interfaces for management purposes. General

database properties, such as Atomicy, Consistency, Isolation, and Durability[1] are also reached with the use of database techniques.

In the following, storage management possibilities for XML documents are first discussed. We look at the properties of relational databases, LDAP, and native XML databases and then overview most prominent XML query languages. After this, the idea of the Semantic Web is shortly introduced as a follow up for the XML revolution. Techniques for storing RDF(S) data in databases and the corresponding query languages are in focus. Also developments in Topic Map metadata storage and query languages are discussed.

4.2 XML Databases

4.2.1 Separating Content from Layout

XML (eXtensible Markup Language) [8] approaches the problem of representing web semantics by separating document content from layout. The content is written using a syntax defined by a Data Type Definition (DTD) or an XML Schema. The same content can be rendered in different ways by using XML Stylesheet Language (XSL). In this way, XML provides a solution, for example, for multi-channel publishing where the same data has to be rendered in many presentation formats.

XML is likely to be the universal format for structured documents and data on the future Web. The standard is based on a tree-like data model. There is the root element having zero or more child elements. Each element has a name and a value. The value can be a literal value or it consists hierarchically of child elements. An element can also have attributes, each of which consists of a name and its value. The XML example below illustrates the idea. Here the DTD file `dtdfilename.dtd` tells that in this particular XML language, the root element `RootElement` has the child element `Element` with an attribute `attribute`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE RootElement  SYSTEM "dtdfilename.dtd">
<RootElement>
  <Element attribute="value">
    <childElement>childElementValue</childElement>
  </Element>
</RootElement>
```

4.2.2 XML Storage Systems

The majority of web pages published by companies are dynamically generated from databases. A usual approach is to define layout templates in HTML with data placeholders, and fill them with suitable data fetched from

[1]For further information about these "ACID" properties see [31].

a database. The data and the layout are separated both logically and physically. The data is stored into — and fetched from — the database, and the layout files are stored in the file system.

XML provides the web with a data model: XML element node tree. This tree model can be used as the basis for query languages and raises hope for more accurate information retrieval results than are possible with HTML. As XML takes its place in web publishing, things get a little bit more complicated from the data storage view point. Storing XML data documents simply as files in the file system — like static HTML web pages — does not match the needs of efficient and reliable data management. On the other hand, since XML data includes a flexible structure, an element tree whose depth and number of branches may vary, it is more difficult to split the documents into pieces of "raw" data to be stored in a database. Storing XML into safe and highly usable databases has become a challenging research topic [20].

The requirements of efficient and reliable data management and retrieval lead towards the idea of using database management systems (DBMS) of some sort for storing and retrieving web documents. There are many possible database architectures to choose from. The relational system is the strongest candidate for storing structured documents due to the dominant role of relational DBMSs in the market. Besides, there are numerous alternative choices, such as object, main memory and native XML databases[2], directory products, such as Light weigth Directory Access Protocol (LDAP) [22] and JNDI-trees[3]. Furthermore, there are many different strategies for storing structured documents within each particular architecture.

XML Document Types

In [7] the difference between data- and document-centric XML is emphasized. A document is *data-centric* if XML is used as the data transporting format. A data-centric document is created for the use of computers.

The structure of a data-centric document is usually quite simple. The document can be, for example, a product catalogue, a time table, or a list of sales orders. Below is an example of a data-centric XML document [7]:

```
<SalesOrder SONumber="12345">
  <Customer CustNumber="543">
    <CustName>ABC Industries</CustName>
    <Street>123 Main St.</Street>
    <City>Chicago</City>
    <State>IL</State>
    <PostCode>60609</PostCode>
```

[2]http://www.internetworld.com/magazine.php?inc=071501/07.15.01technology3.html, February 2002

[3]http://java.sun.com/products/jndi/overview.html, February 2002

```
  </Customer>
  ...
</SalesOrder>
```

It is not very important for data-centric data that it is presented particularly in XML format. It is also usual, that the order of sibling elements is not very important.

A *document-centric* document is usually intended for human use. It may be a book, a report, or an email. Document-centric documents usually have a rich content and a less regular or an irregular structure. The next example depicts a document-centric XML document [7]:

```
<Product>
  <Name>Turkey Wrench</Name>
  <Developer>Full Fabrication Labs, Inc.</Developer>
  <Summary>Like a monkey wrench, but not as big.</Summary>
  <Description>
    <Para>The turkey wrench, which comes in <i>both right- and
    left-handed versions (skyhook optional)</i>, is made of the <b>finest
    stainless steel</b>. The Readi-grip rubberized handle quickly adapts
    to your hands, even in the greasiest situations. Adjustment is
    possible through a variety of custom dials.</Para>
    <Para>You can:</Para>
    <List>
      <Item><Link URL="Order.html">Order your own turkey wrench</Link></Item>
      <Item><Link URL="Wrenches.htm">Read more about wrenches</Link></Item>
      <Item><Link URL="Catalog.zip">Download the catalog</Link></Item>
    </List>
    <Para>The turkey wrench costs <b>just USD19.99</b> and, if you
    order now, comes with a <b>hand-crafted shrimp hammer</b> as a
    bonus gift.</Para>
  </Description>
</Product>
```

It is not always easy to resolve whether a document is data- or document-centric. Data-centric documents may include parts, which are document-centric, and vice versa. For example, an invoice may consist of regularly structured elements, such as the price, the name and the serial number element. Furthermore, it may include an irregularly structured element, e.g., the description about the product. The basic rule is that if the document is of the data-centric type, it can be stored into a traditional — relational or object-oriented — database. Otherwise a native XML database, a content management system, i.e., an application designed to manage documents and built on top of a native XML database, or some other repository solution, such as a directory service, is preferred.

If the data exists in a database and one wants to express it in XML form, it must be translated into the preferred form. In this case, the traditional database is probably used. If the data resides outside the database and the goal is to store it from the document to the database, then it may be desirable to select some native XML database and store the whole document into it.

XML and Relational Databases

When XML data is retrieved from or stored into a relational database, it must be mapped from one data model to another at the runtime. This translation is done with a piece of software which may be included into the database, or it can be a database-independent transferring software. In both cases it is usual that some information is lost in the transformation. There are two major mapping schemas [7]:

- In *template-driven* mapping data is transformed into XML when retrieved from the database.
- In *model-driven mapping* XML data is transformed when stored into the database.

Transforming Database Data into XML

Database data can be transformed into an XML document by using a template. The template is an XML document with embedded SQL statements. These statements are executed and the results are inserted in place of the query in the document. Below is an example of a simple template:

```
<?xml version="1.0"?>
<persons>
  <SQLQuery>SELECT name, address from person</SQLQuery>
</persons>
```

If the data fetched from the database consists of the two rows

```
["John Smith", "Elm street 23"]
["Alice Smith"]
```

then the following document is obtained by placing the data into the placeholder of the template:

```
<?xml version="1.0"?>
<persons>
  <person>
    <name>John Smith</name>
    <address>Elm street 23</address>
  </person>
  <person>
    <name>Alice Smith</name>
  </person>
</persons>
```

In template-driven mapping the result set can be placed anywhere in the resulting document, loops and conditional statements are allowed, parametrizing is possible, and variables can be used.

Storing XML into a Relational Database

It is possible to store XML into a relational database in a general way without using the meta data of the document DTD. One such model-driven solution is the table-based mapping[4]. It is a simple but limited way to split XML data into fields, rows, and tables of a database. In this scheme, a document of the form

```
<dbname>
  <tablename>
    <field1>value</field1>
    .
    .
    .
  </tablename>
</dbname>
```

is stored into the table `tablename` of the database `dbname`.

Table-based mapping suits well for data which is to be transferred from one database to another database, but cannot be used for documents with irregular or deep hierarchical structure.

Another simple and general solution, called "the edge approach" [13], is based on the idea that each document includes it's own table. The rows of the table include at least the following fields:

- parent elements identifier (for the root element it is "1")
- the sequential number of the element
- the identifier of the element
- the value of the element.

This mapping doesn't support concurrent access to same document because of the numerous nested operations of the table. Still another mapping solution, called "the binary approach" eases the strain by dividing the data into different tables. The distribution is done so that each element in the source document is stored into its own table, i.e., every value of an element "A" is stored into the table named "A".

There are numerous additional ways to do mapping from XML data into a relational database. Many of them are more sophisticated than the solutions presented above. It is also usual to analyze the possibly available document schema and base the mapping on it. Anyway, the mappings presented above are efficient enough for many purposes, especially when simple document "field-filling" takes place, i.e., when an XML template exists with placeholders for values which are to be fetched from a database. The database schema must also be quite simple in order to make the general-purpose mappings useful.

[4]http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html, February 2002.

XML and LDAP

Relational databases are not a straightforward solution for storing XML data due to the differences between the relational and the tree data models. The mapping between the models is complicated or thriftless especially with documents whose structure is irregular or which are strongly document-centric. The Lightweight Directory Access Protocol (LDAP) [22] data model lies close to Document Object Model (DOM) of XML, which makes the mapping simple. As a result, LDAP is claimed to be a more natural and efficient choice for storing and processing queries concerning XML data.

The XML data presentation and query model in LDAP is described in [27]. XML data can be stored efficiently in LDAP without having to change the schema of the LDAP system. The data can be queried with XPath[5] statements. XPath is a language designed for addressing parts of XML documents.

The LDAP data model corresponding to XML consists of the three classes below:

```
XMLNode OBJECT-CLASS ::= {
  SUBCLASS OF {top}
  MUST CONTAIN {oc, oid, name}
  TYPE oc OBJECT-CLASS
  TYPE oid DN
  TYPE name STRING}
XMLElement OBJECT-CLASS ::= {
  SUBCLASS OF {XMLNode}
  MUST CONTAIN {order}
  MAY CONTAIN {value}
  TYPE order INTEGER
  TYPE value STRING}
XMLAttribute OBJECT-CLASS ::= {
  SUBCLASS OF {XMLNode}
  MUST CONTAIN {value}
  TYPE value, DN, STRING}
```

The `XMLNode` is a the most upper class and both `XMLElement` and `XMLAttribute` are subclasses of it. Each `XMLNode` must have three attributes which express the type of particular node (`oc`), an unique identifier (`oid`) and a name (`name`). Additionally an `XMLElement` must have one attribute, order, which tells the exact location of an object among all children of it's parent object. It also may have a value attribute. The `XMLAttribute` has a mandatory value attribute; there is no such thing as an attribute with the empty value.

The generality of the model is based on two things. Firstly, to the use of attributes to store information about XML nodes and secondly, to the hierarchical nature (at the instance level) of the LDAP model, where each node tells its exact location in the hierarchy.

[5] http://www.w3.org/TR/1999/REC-xpath-19991116, February 2002.

Native XML Databases

Term "Native XML database" refers originally to a trade mark used in a campaign for AG Software's Tamino XML database[6]. Since then, the meaning of term has become broader and nowadays the common understanding is that a native XML database is one with no other data model than the structure of the document it is stored into. It should include at least elements, attributes, XML data types, such as PCDATA, and the document order.

There are many reasons to use native XML databases:

- The documents may be strongly document-centric and difficult to transform into relational form.
- Faster retrieval of the documents is needed than is possible with a simple file system.
- The need for receiving an XML document as the result of a query.
- The need for supporting an XML query language.

Satisfactory retrieval speed is not always possible. Especially when elements are fetched in a different order than they are presented in the target document, the performance of a native XML database may collapse.

Native XML databases fall into two categories [7].

- In a *text-based database* the document is stored as text. An example of the text-based XML database is presented in [14].
- In a *model-based database* the internal data model is built from the document, and the model is then stored into the database.

According to this definition, the LDAP would fall into the model-based native XML database category. A solution for the model-based (main memory) XML database is presented in [6]. For more information about the topic see [7].

4.2.3 XML Query Languages

There are already several generations of XML query languages coming from both the database and document communities. In the following, an overview of two of them, XQL and XQuery is given.

[6]http://www.softwareag.com/tamino/, February 2002.

XQL

XQL [30] is the XML query language having the largest number of different implementions at the moment. It was first introduced in 1998 and took some ideas from earlier languages XML-QL [15] and Lorel [1]. The formulation of XQL queries is based on the tree structure of XML-documents, i.e., hierarchy, sequence, and position of document parts. Its addressing scheme is closely related to XPath[7]. New structural forms can be produced in queries by joining subtrees.

XQL makes queries over one or more XML documents (the *context*) and produces a list of XML document nodes as the result. XQL syntax mimics the URI directory navigation syntax and is similar to that of XML-QL. In the next examples the results of queries are produced from the following XML-document:

```
<?xml version="1.0"?>
<paintings>
   <painting style="cubism">
      <year> 1909 </year>
      <name> Seated Nude </name>
      <painter>
         <name>
            <first> Pablo </first>
            <last> Picasso </last>
         </name>
         <birth>
            <year> 1881 </year>
            <month> 10 </month>
            <day> 25 </day>
         </birth>
         <death>
            <year> 1973 </year>
            <month> 4 </month>
            <day> 8 </day>
         </death>
         <spouse>
            <name>
               <first> Olga </first>
               <last> Koklova </last>
            </name>
            <divorce> 1937 </divorce>
         </spouse>
         <spouse>
            <name>
               <first> Jacqueline </first>
               <last> Roque </last>
            </name>
         </spouse>
      </painter>
   </painting>
   <painting>
      <name> The Three Dancers </name>
      <year> 1925 </year>
      <painter>
```

[7] See http://www.w3.org/TR/1999/REC-xpath-19991116 for details.

```
      <name>
         <last> Picasso </last>
      </name>
      <birth>
         <year> 1881 </year>
         <month> 10 </month>
         <day> 25 </day>
      </birth>
   </painter>
</painting>
<painting>
   <name> Weeping Woman </name>
   <year> 1937 </year>
   <painter>
      <name>
         <first> Pablo </first>
         <last> Picasso </last>
      </name>
      <birth>
         <year> 1881 </year>
         <month> 10 </month>
         <day> 25 </day>
      </birth>
   </painter>
</painting>
<painting>
   <year> 1941 </year>
   <name> Me and My Parrots </name>
   <painter>
      <name>
         <first> Frida </first>
         <last> Kahlo </last>
      </name>
      <birth>
         <year> 1907 </year>
         <year> 1910 </year>
         <month> 7 </month>
         <day> 6 </day>
      </birth>
      <death>
         <year> 1954 </year>
         <month> 7 </month>
         <day> 13 </day>
      </death>
      <spouse>
         <name>
            <first> Diego </first>
            <last> Rivera </last>
         </name>
         <divorce> 1942 </divorce>
      </spouse>
   </painter>
</painting>
<painting>
   <name> House over the Bridge </name>
   <year> 1909 </year>
   <painter>
      <name>
         <first> Diego </first>
         <last> Rivera </last>
      </name>
      <birth>
```

```
            <month> 8 </month>
            <day> 12 </day>
          </birth>
        </painter>
    </painting>
</paintings>
```

Hierarchy

The query is performed within a context, and document substructures in the tree hierarchy are pointed to using XPath, where "/" indicates hierarchy level. For example, expression

```
painter/birth/year
```

would find all the year-elements of all the birth-elements of all the painter-elements in the context. If the hierarchy structure is not known in detail, operator "//" can be used to signify any number of intervening levels:

```
painter//year
```

This would find not only all painter/birth/year-elements, but also, for example, all the painter/death/year-elements. If there were elements of type painter/spouse/divorced/year, they would be returned, too.

The root element of a document is signified by "/" at the beginning of the path expression:

```
/paintings
```

This would find the root element paintings from the document.

"//" used in the beginning of a path signifies search starting from the root and searching all levels. If it is used for the current context, it must be preceded by the current context operator ".".

Sequence

In XQL, "before", "after", and "list concatenation" operators can be used for convenient handling of sequences. They have two parameters each and are used in form (X before Y), (X after Y), and (X,Y), respectively. Two first ones are used to specify the query, and the third one is used to specify the result of a query. For example,

```
painter/birth/(year, month, day)
```

would find the birthdays of all painters, returning first the year, then month, then day. If there are multiple nodes of some item, they all are listed before the next item. For example, Frida Kahlo has two birth years[8], and

[8]Frida Kahlo was born in the year 1907, but she claimed to have been born 1910 because she wanted her life to begin the same year as the new Mexico.

the result for her would be: year, year, month, day.

Position

Brackets are used to indicate the position of a node in a query. For example, the expression

```
/painting/painting[2 to 4, 6, -1]
```

would find the second, third, fourth, sixth, and last painting in the paintings root element.

Names

Namespace prefixes can be declared using a variable declaration in the following way:

```
a := http://www.imaginaryaddress.net;
//a:paintings
```

Namespaces are preserved in the result of a query. To change the namespaces of nodes in the result, they can be renamed. The wildcard character "*" can be used in place of a namespace or a node name. Expression

```
*:*
```

would find all elements for which a namespace has been declared.

Filtering

Nodes to be searched for can be of various types: element, attribute, text, processing information, etc. Elements are the basic nodes without a specialized type. The search tree can be pruned by filtering the branches, which can be done using brackets after the element name. The pruning condition is placed inside the brackets. In the condition, one or more descendant nodes are evaluated, and only nodes which satisfy the condition are searched further and may contribute to the result. For example:

```
painting[@style='cubism']/
      painter[name contains "picasso"]/spouse[divorce]/name
```

This would find the names of (divorced) spouses of Picasso-named painters who have painted at least one cubistic painting. The "@"-operator signifies the attribute of an element, which can also be treated as a child node (without forgetting the "@").[9]

[9] See http://metalab.unc.edu/xql/xql-proposal.xml for what filtering operators are supported.

Functions and methods

In XQL two kinds of functional "commands" have been proposed: functions and methods. Functions evaluate the subtree of the context, whereas methods evaluate to a property of the reference node in the search context. There are some basic functions included in XQL, such as "attribute()", "text()" and "id()". For implementing XQL-based systems, function "function()" is provided for defining new functions.

Variables and renaming

Variables can be used for filtering elements. Binding is done in brackets after the element in which the binding is done. Variables have prefix "$". For example:

```
painting/painter[$s := spouse/name] { name and //painter[name=$s]/name }
```

This would find the name of a painter and the painter's spouse, if the spouse is also a painter, i.e.:

```
<xql: result>
   <name>
      <first> Frida </first>
      <last> Kahlo </last>
   </name>
   <name>
      <first> Diego </first>
      <last> Rivera </last>
   </name>
</xql: result>
```

Renaming is done using the operator "->", with the queried context on the left side and the new name on the right side:

```
painting/painter[$s := spouse/name]
      -> PainterCouple { name and//painter[name=$s]/name }
```

This would find exactly the same thing as the previous query, but the result nodes would have different names.

Results

The result of an XQL query is a list of nodes preserved in the original (context) order, hierarchy, and identity, to the extent that these are defined. XQL query results can be serialized into well-formed XML documents. The results of a serialized query are wrapped in an <xql:result> element.

Basicly, queries return the nodes, which satisfy the query conditions, in same form as in the original document. For example,

```
painting/painter/name/
```

would return

```
<xql: result>
   <name>
      <first> Pablo </first>
      <last> Picasso </last>
   </name>
   <name>
      <last> Picasso </last>
   </name>
   <name>
      <first> Pablo </first>
      <last> Picasso </last>
   </name>
   <name>
      <first> Frida </first>
      <last> Kahlo </last>
   </name>
   <name>
      <first> Diego </first>
      <last> Rivera </last>
   </name>
</xql: result>
```

Handling query result can be done using the following operators: "|" (union); "intersect" (intersection); "∼" (both); "or" (or); "and" (and); "before" (before); "after" (after); "," (list concatenation). These operators are placed in parentheses, with one query on each side of it.

```
/painting/painter/name(first ~ last)
```

This would return union of first and last name for each painter who has made a painting, but only if the painter has both first and last name. If a painter has only last name, result for that painter is empty. "or" and "and" operators yield boolean values. Grouping of results is done by using curly braces around the query to be joined in the result.

```
//painter {spouse/name/last}
```

This would find last names of spouses of every painter.

```
<xql: result>
   <painter>
      <last> Koklova </last>
      <last> Roque </last>
   </painter>
   <painter>
      <last> Rivera </last>
   </painter>
</xql: result>
```

If queried without join

```
//painter/spouse/name/last
```

the result would be

```
<xql: result>
   <last> Koklova </last>
   <last> Roque </last>
   <last> Rivera </last>
</xql: result>
```

Joins over multiple sources are done simply by grouping the results of queries over different sources.

Since the results of an XQL query can be serialized into XML documents, they can be used as a context for further XQL queries. This means that queries can be embedded.

XQuery

XQuery[10] is still under standardization at W3C, but there are already a few experimental implementations. The following overview of XQuery is based on the W3C working draft [11]. The examples and information given should be considered as general ideas and not necessarily as final and complete statements.

XQuery came from the document community, that often has more interest in reformatting the queried data than the data community. XQL has the same background community, but still the main difference of expressiveness between XQL and XQuery is that XQuery allows for more flexible result formatting.

Path expression

In XQuery, hierarchy navigation, sequencing, and positioning are done in the same fashion as in XQL; both languages base their path expressions on XPath.

There is, however, an extra operator introduced in XQuery: the dereference operator "=>". It allows references in attributes of elements to be followed. Attributes to be followed must be of type IDREF or IDREFS, and the values to be matched in referenced elements are attributes of type ID. For example,

```
//painter/@spouse=>painter/death/year
```

[10] http://www.w3.org/XML/Query

wouldn't find anything in the original document. If in the original document `painter` elements would have `spouse` attribute of type IDREF referencing to another `painter` element, then it would find the death year of every painter's every spouse. Actually, the expression would return several nodes for some spouses' death years because of the structure of the original document.

Namespaces

Namespaces can be declared in XQuery in the following fashion:

```
NAMESPACE xsd = "http://www.imaginaryaddress.net"
```

Filtering

XQuery queries use the FLWR-syntax, i.e., a query consists of clauses FOR, LET, WHERE, and RETURN. The FOR-clause consists of variables and expressions. Each variable is bound to an expression for further handling. LET-clause also binds variables to expressions, but it does not iterate. Filtering is done in the WHERE-clause using a variety of operators[11]. The results are defined in the RETURN-clause. In the WHERE-clause filtering can be done

Functions

XQuery provides some useful built-in functions, such as avg(), sum(), count(), max(), and min()[12]. The user can also define her/his own functions. This is done using the following syntax:

```
DEFINE FUNCTION functionname(datatype $variable) RETURNS
namespace:datatype
{
   # Function definition
}
```

Functions can be recursive, or even mutually recursive (several separate functions can call each other).

Variables and renaming

Variable bindings are done in the FOR- and LET-clauses in the following manner:

[11] See http://www.w3.org/TR/xquery or a detailed list of these operators.

[12] For a detailed list of these, see http://www.w3.org/TR/xquery-operators

```
FOR $p IN document("www.imaginaryaddress.net")//painting)
LET $y := $p/year
WHERE $y < 1940
RETURN
      <renamed_year>
           {$y}
      </renamed_year>
```

Variables have prefix "$". In the FOR-clause they are followed by keyword IN, after which follows an expression. In the LET-clause there is operator ":=" between a variable and a value expression. The query would find for all the paintings which were painted before 1940 from an XML-document in address `www.imaginaryaddress.net` where painting-nodes have a year-node to represent the painting year:

```
<renamed_year>
   <year> 1909 </year>
   <year> 1925 </year>
   <year> 1937 </year>
   <year> 1909 </year>
</renamed_year>
```

Renaming is done in the LET-clause. The user can define a name for the returned elements which contain queried data. In the previous example, the query returned <renamed_year> -nodes with values derived from //painting/year -nodes.

Results

Results are formatted in the RETURN-clause. The RETURN-clause is executed once for each tuple of bindings that is generated by the FOR and LET-clauses, and satisfies the condition in the WHERE-clause, preserving the order of these tuples. Order or values of results can be reformed by operators and functions. For example, the query

```
FOR $name IN //painting/painter/name
LET $first := $name/first, $last := $name/last
WHERE not(empty($first)) AND not(empty($last))
RETURN $first, $last
```

would return the union of the first and last name for each painter who has made a painting, but only if the painter has both first and last name.

In conclusion, a few examples follow to illustrate how different operators and expressions can be used.

```
<productive_painters>
   {
   NAMESPACE paint = "http.imaginaryaddress.net"
   FOR $painter IN distinct(document(paint)//painter)
   LET $painting := document(paint)//painting[painter = $painter]
   WHERE count($painting) > 2
```

```
   RETURN
      <productive_painter>
         {$painter/name/last}
         {$painter/name/first}
         {$painter/birth/year}
      </productive_painter>
   SORTBY (year ASCENDING)
   }
</productive_painters>
```

This query would return an element <productive_painters>, which would include all the painters, which have painted over 2 paintings, with elements for their first and last names and birth years. The list is finally sorted in the result element according to the birth years:

```
<productive_painters>
   <productive_painter>
      <last> Picasso </last>
      <first> Pablo </first>
      <year> 1881 </year>
   </ productive_painter>
</ productive_painters>
```

Note the inclusion of the query as a part of an element constructor to wrap the result in the element.

The following query would return all the paintings with their name, except cubistic paintings with their style.

```
FOR $p IN //painting
RETURN
   <painting>
    {IF ($p/@style = "cubism")
     THEN $p/@style
     ELSE $p/name
    }
   </painting>
```

Finally consider the query:

```
FOR $p IN //painting
WHERE SOME $pntr IN $p/painter
      SATISFIES (contains($pntr/name/last, "picasso")
RETURN $pntr/name
```

This would return from the example document all paintings which Picasso has made. If the document contains paintings which have several painters, and Picasso is one of them, then the query would return them, too.

Summary

At the moment, XQuery is probably the most prominent XML query language. Figure 4.1 depicts its development. XQuery is derived from Quilt [12], which in turn was influenced by several other query languages: path

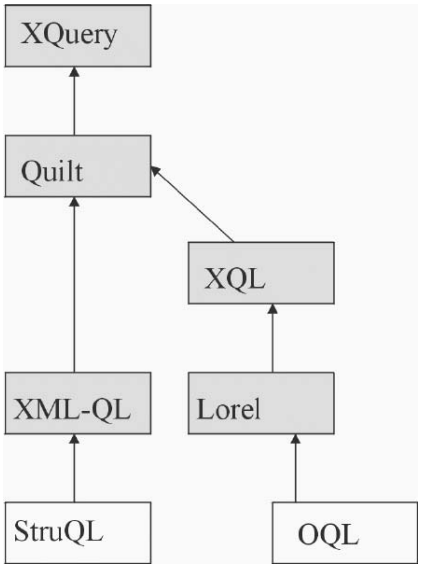


Figure 4.1: The pedigree of some XML and RDF query languages.

expression syntax from XQL; from XML-QL the idea of binding variables and then using them to create new structures; The pattern for restructuring the data from SQL. XML-QL and Lorel were evolved from query languages for other types of data. StruQL [17] and OQL [3] are earlier query languages.

4.3 RDF(S) Databases

4.3.1 Separating Semantics from Syntax

The XML model is based on syntax, not on semantics. The schema designer can only name the tags by which the element to be characterized is surrounded. This syntactic basis of XML means that also queries to XML repositories must be formulated in terms of the structure of the documents. However, from the user's viewpoint, semantic characterizations of information needs would be more natural. For example, the same semantic address information about company employees can represented in many syntactic ways. In the DTD, the telephone number may be represented as a child element or as an attribute, and different naming conventions may be used. However, such syntactic choices are not of interest to the telephone catalog user and should not affect her/his queries.

Common dictionaries give the word "semantics" the meaning "the study of meanings"[13]. According to [5] the *"Semantic Web is an extension of*

[13]Cf. e.g. Merrian-Webster's Collegiate Dictionary.

the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation". To enhance human-machine cooperation, the data on the web must be made from being just machine-readable (as it is now) to be machine-understandable. In this perspective, "semantic" means "machine-understandable".

In order to give web information a well-defined, machine-processable meaning, languages for representing semantics are of central importance. Representing and annotating web documents with these languages would gradually change the current web of HTML pages into a better-structured repository of data or document database.

The Resource Description Framework (Schema), RDF(S)

The emerging Resource Description Framework (RDF) [26] and RDF Schema (RDFS) [9, 21, 2] hold the promise to become the next breakthrough technology of the Semantic Web. They provide a way to build a whole class/object schema upon the actual resource data.

RDF(S) makes it possible to tell what things really mean and, as a consequence, can support more expressive query languages based on meanings. From the syntactic perspective, RDF(S) descriptions are serialized in XML and also need a place where they can be stored and from where they are available for use.

The main idea and objective of RDF is to define a mechanism for describing web resources. RDF makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain [26]. RDF is a generic framework, which is in priciple applicable to any application domain.

RDF is suitable for enhancing the precision of search engines and for cataloging, classifying and rating the content available in web sites or digital libraries. It can be used by intelligent software agents, e.g., to facilitate knowledge sharing and exchange.

An RDF-statement can be thought as a triplet which consists of a *predicate*, a *subject*, and an *object* just like a proposition in a natural language. Alternatively, a triple can be seen as an attribute and its value attached to an object. For example, in the proposition "A Philisave is a shaver" the subject is "a Philisave", "is" is the predicate, and "a shaver" is the object. The same statement looks like this when written in formal RDF:

```
<rdf:Description rdf:about="http://www.shavingproducts.com/Philisave">
  <rdf:type rdf:resource="http://www.bathroomschema.org/schema#Shaver"/>
</rdf:Description>
```

RDF defines a data model for describing relationships between resources in terms of named properties and values, but does not provide a mechanism for defining the meaning of these constructs. The RDF Schema (RDFS)

extends RDF by offering tools for describing RDF vocabularies and relationships. For example, using the pre-defined RDFS terms *Class* and *subClassOf* we can say that all shaving tools are electrical equipments, too. In RDF Schema, *an agreement is made on the semantics of certain terms and thus on the interpretation of certain statements* [10].

4.3.2 RDF Storage Systems

RDF-statements can be and usually are written in XML [26]. So why do we need a separate database for RDF(S) documents? There are a few reasonable reasons.

An XML database is not the best choice for storing RDF(S) documents due differences in the underlying data models and the special semantics of RDF(S). The strategies for mapping XML data into relational form are based the tree-like structure of the document. In XML such strategies make sense, since the structure of the document is similar to the undelying XML data model. However, in RDF the data model is a set of triples, and structure based mapping of RDF statements would not necessarily map the triples in a meaningful way into the relational tables.

In principle, "normal" RDF triplets form a data-centric document that can be stored easily into a traditional relational database. For example, for each predicate used in the triples, a table of subject-object-pairs could be created. However, problems arise when RDF Schema is used. RDFS looks like RDF, but the underlying data model is an object oriented one with the class hierarchy, instances and inheritance mechanisms, which is more difficult to transform into the relational database model.

Even if the RDF(S) database management would be done with an XML database, the need for distinct query languages remains. XML query languages are not designed for querying the relationships of the class instances, neither is traversing of the object hierarchy an inherent feature of XML query languages. The object oriented vocabulary of RDF(S) is meant to be as generic as possible, which is not usually the case with XML languages focusing on particular application domains. Generic semantics could be supported in the query language.

Sesame, an RDF(S) Database

The Sesame architecture [10] is capable of storing and processing RDF(S) data. The structure of Sesame consists of three layers. The most upper layer contains HTTP and SOAP protocol handlers. The middle layer consists of administration, query, and export modules. The lowest layer closest to the database is database specific and separares database details from the rest of the system. It is called the Repository Abstraction Layer (RAL).

The implementation of the Sesame database[14] lays on the PostgreSQL object-relational database. However, changing the repository abstraction layer RAL makes it possible to change the current repository to any kind of DBMS, existing RDF stores, RDF files, or RDF network services.

RDF and RDF Schema are conceptually divided when data is stored into the database. Schema classes (*RDFS-class* in future) and relations, i.e., *Class*, *subClassOf*, and *Property*, have classes of their own in the database (*db-class* in future). Objects in db-classes are the actual RDFS-classes and -relations used in a documents. For example, the db-class *Class* consists of data about the RDFS-classes described in a particular RDF Schema. For every new RDFS-class description, a new db-class is created on the RDF side of the database. If the RDFS-class is a subclass of another RDFS-class, the new db-class is defined to be a subclass of particular parent db-class.

The figure 4.2 depicts an example database schema from [10].

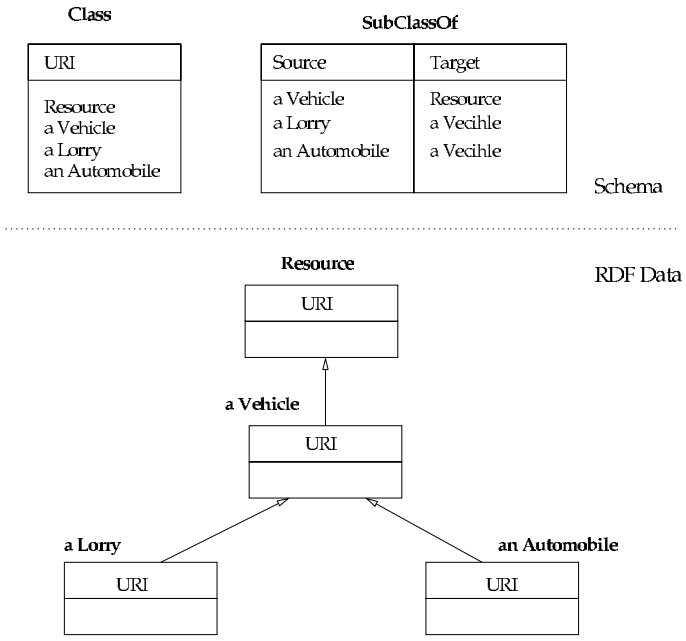


Figure 4.2: RDF(S) database schema in PostgreSQL.

RSSDB, RDF, and an Ontology Database

Sesame does not support DAML/OIL-statements written in RDF(S) or any other ontology language. In contrast, RSSDB[15] does. Unfortunately it lacks in support of RDFS features. For example, the core property *rdfs:subClassOf* of RDF Schema is missing. In RSSDB the RDF Schema is used for creating a

[14]See http://sesame.aidministrator.nl/ (February 2002) for details.

[15]See http://139.91.183.30:9090/RDF/RSSDB/ (February 2002) for details.

database schema into an object oriented database (PostgreSQL is used in the implementation). The size of the database grows linearly and it is told to be relatively efficient when compared to other similar implementations. RSSDB is available for download at `http://139.91.183.30:9090/RDF/RSSDB/`

4.3.3 RDF Query Languages

The idea of RDF query languages is rather new and has not (yet) had as much attention as XML query languages. Only few RDF query languages have been defined and implemented. A prominent proposal at the moment is RQL [23], the first proposal for a query language for both RDF and RDF Schema. RQL is used, for example, in Sesame[16].

In the following, the syntax of RQL presented based on the paper by Karvounarakis et al. [23].

RQL

RQL uses the SELECT-FROM-WHERE-query structure of SQL. In the SELECT-clause the variables are introduced. Mappings to these variables will be returned as the result of the query. In the FROM-clause, a path expression is given and variables are bound to a path. In the WHERE-clause filtering is done.

Basicly, an RDF(S) repository consists of *classes*, their *instances* and *properties*. Classes can be thought of as any sort of concepts, and properties as their (binary) relations, which leads to the {source}property{target} triple model of RDF. Properties are directed, which means that their source and target are distinquished. For example, if {source}does{target}, it doesn't follow that {target}does{source}.

For example, the query

```
SELECT X, Y
FROM {X}paints{Y}
```

would find all the painter-painting pairs that are connected by the property `paints` in the repository. Variable X on the left is the source and Y is the target.

Variables can be defined not only for resources but for properties and classes, too. Prefix "@" is attached to a property variable, and "$" to a class variable. The query

```
SELECT X, Y, @P, $Z
FROM {X}@P{Y:$Z}
WHERE $Z <= Avant-garde AND @P = paints
```

[16]The Sesame database implementation can be tested at the address `http://sesame.aidministrator.nl/`

would find all the painter-painting pairs in which the painting belongs to a subclass of "Avant-garde". The possible class restriction given by a class variable is given after the colon ":".

If it not desired that class hierarchies are expanded, i.e., subclasses are taken into account, one could use functions `domain()` and `range()`. For example,

```
SELECT domain(@P), @P, range(@P)
FROM Property{@P}
WHERE domain(@P) <= artist
```

would return triples such as `artist, creates, artifact; painter, paints, painting` whereas without domain and range functions also subclasses would be expanded, creating triples such as `artist, creates, painting` (painting being subclass of artifact). Query "Property" returns all the found properties.

Finally, properties can be chained with the operator "." to produce more versatile queries. The following query would find the names and the periods to which paintings date to:

```
SELECT X, Y
FROM {X}paints.dates_to{Y}
```

4.4 Topic Map Databases

4.4.1 Topic Maps

RDF(S) is the meta data representation standard of W3C. In addition, ISO (International Organization for Standardization) has the Topic Maps[17] (TM) [28] meta data representation scheme for describing web resources. The original TM standard in the year 2000 (ISO/IEC 13250) was originally based on an SGML Data Type Definition (DTD) but was soon converted to XML Topic Maps (XTM) standard[18] in February 2001.

The idea of TM originated from the need to develop standards for merging book indexes. The TM scheme essentially enriches the idea of the book index into a kind of electronic semantic associative index to be used in the WWW context.

Database techniques are needed also for storing large topic maps. The underlying data model of a topic map is a directed graph structure in the same way as in RDF. The arc triples of the graph could be stored easily in a relation database as in pure RDF. There is no inherent notion of classes and

[17] http://www.topicmaps.net

[18] http://www.topicmaps.net/xtm/1.0/

inheritance in topic maps like in RDF Schema, which makes the mapping between the TM data model and the relational datamodel simpler. However, there are other semantic constructs in TM that have to be modeled.

The problem of managing topic map databases has not been discussed much in the literature, but there are commercial implementations on the market. For example, the Ontopia Topic Map Engine[19] has a (separate) RDBMS back-end add-on for persistent and scalable storage of large topic maps.

4.4.2 Topic Map Query Languages

TMQL

TMQL (Topic Maps Query Language) is under standardization by ISO; at the moment it's requirements[20] are being worked on. The idea is to create a language based on SQL, which makes querying, viewing, creating, and updating topic maps possible. In below, the idea of quering topic maps is explained by using illustrative examples. It is assumed that the reader is familiar with the basic notions of topic maps [28].

The query language syntax to be used follows the proposal of Rafal Ksiezyk [25]. The queries will be of the following type:

```
SELECT topic[x]
FROM topic[x]
WHERE x="puccini"
```

This would return all Puccini topics.

In topic maps, query targets are typically associations between topics. For example, to query "what has Puccini composed" the association "composed by" from the topic "Puccini" to some other topic is in focus. In this association "Puccini" and the composition play different roles:

```
SELECT topic[x]
FROM
 topic["puccini"].assoc-role[ar1].assoc[a].assoc-role[ar2].topic[x],
 Assoc[a].assoc-type["composed by"],
 Assoc-role[ar1].assoc-role-type["who"],
 Assoc-role[ar2].assoc-role-type["what"]
```

TMQL queries return topic maps. The SELECT clause may have a more complex structure, for example:

[19] http://www.ontopia.net

[20] http://www.y12.doe.gov/sgml/sc34/document/0227.htm

```
SELECT topic-type["artist"].topic[x]
FROM topic-type["artist"].topic[y], topic-type[y].topic[x]
```

This would return topics, whose type is "artist".

A useful way to filter the data would be to store queries in profiles. These profiles can be used later in WHERE-clauses.

tolog

tolog is a topic map query language proposed in 2001 by Lars Marius Garshol [19]. The language is closely based on first-order predicate logic programming language Prolog. This leads to a quite different approach form TMQL that was based on SQL.

Queries in tolog are given in the form *clause(value, value)*. Here *value* can be a variable or a topic. The result is given as a list of matches. For example, the query "What has Puccini composed?" is given below:

```
composed-by($A, puccini).
```

The result could be:

```
[{'A': 'gianni-schicchi'},
 {'A': 'madame-butterfly'},
 {'A': 'tosca'},
 {'A': 'edgar'},
 {'A': 'la-boheme'},
 {'A': 'il-trittico'},
 {'A': 'le-villi'},
 {'A': 'la-rondine'},
 {'A': 'suor-angelica'},
 {'A': 'la-fanciulla-del-west'},
 {'A': 'il-tabarro'},
 {'A': 'manon-lescaut'},
 {'A': 'turandot'}]
```

In a query, commas can be used as 'and' operators. For example:

```
composed-by(\$A, puccini), written-by(\$A, \$B).
```

Writing two separate clauses in one query can be used as an "or" operator. The results are presented as lists of matches in which every list element contains mapped values for the variables in the query.

A query without variables produces either a single empty match, if the associations can be matched in the database, or an empty list of matches, which means that associations are not found. These values correspond to the logical values "true" and "false", correspondingly. For example, the query "Has Puccini composed Tosca?"

```
composed-by(tosca, puccini).
```

produces the result "true":

```
[{}]
```

If Puccini hadn't composed Tosca (or if the association were not present in the underlying topic map), the result would be:

```
[ ]
```

There are two predefined rules in tolog. "Has-type" verifies whether a topic is an instance of a class, and "/=" means "is not". New rules can be implemented like in Prolog. For example:

```
used-work-by($A, $B):- composed-by($OPERA, $A),
                       based-on($OPERA, $WORK),
                       written-by($WORK, $B).
```

Here a virtual association `used-work-by` is created. It means that A has used-work-by B, if A has written an opera based on a work of B. Also recursive rules can be implemented, for example:

```
descendant-of($A, $B) :- child-of($A, $B).
descendant-of($A, $B) :- child-of($A, $C), descendant-of($C, $B).
```

In the rules many variables can be used, but only the variables of the queried clause are used in the result. For example, the query "Whose work has Puccini used?"

```
used-work-by(puccini, $A).
```

evaluates the result:

```
[{'A': 'dante'},
 {'A': 'belasco'},
 {'A': 'sardou'},
 {'A': 'musset'},
 {'A': 'murger'},
 {'A': 'gold'},
 {'A': 'prevost'},
 {'A': 'gozzi'}]
```

In the queries, association role types can be given for better filtering. For example, the previous rule can be rewritten as:

```
used-work-by($A : composer, $B : writer) :-
              composed-by($OPERA : opera, $A : composer),
              based-on($OPERA : result, $WORK : source),
              written-by($WORK : work, $B : writer).
```

Here variable A matches only topics of type `composer`, B only topics of type `writer`, and so on. Without these contraints, the query could actually produce results such as {'A' : 'metallica', 'B' : 'trumbo'}, because Metallica's song "One" is based on Dalton Trumbo's book "Johnnie Got His Gun", which has nothing to do with opera. This of course depends on the underlying topic map.

4.5 Conclusions

The Semantic Web is an effort to make data on the web more useful for machines and, as a consequence, for people using the machines. Data formats such as XML, RDF(S), Topic Maps, and DAML+OIL are key factors when building multi-layered data models including the data, the meta data, and the ontologies. The need to store and retrieve data on the web efficiently and in a disciplined manner becomes more and more important, when the amount of the data to be managed increases and is used by more and more end-users.

For XML data there are two main solutions from which to select. For data-centric documents, a traditional (i.e., relational) database is likely to be the right choice. If the document is document-centric and highly irregularly structured, a native XML database would propably be the best option. From the various XML query languages developed, XQuery is becoming more and more widely accepted.

There is only one system that supports satisfyingly both RDF and RDF Schema: the Sesame database. It is available as a web service via HTTP and SOAP interfaces. An alternative for storing and managing RDF is the RSSDB database, which does include slight support for RDF Schema, but do support the OIL-ontology language. As for RDF query languages, RQL seems to be most developed for the time being.

Database solutions and query languages for topic maps are also emerging, including the SQL based TMQL and logic programming based tolog.

RDF(S)- and topic map -based databases and query languages are not as widely developed and used as XML-based ones. However, their usability can potentially be greater because of the greater the expressive power of the underlying data models.

Both large relational database manufacturers and small, specialized database companies fight for a storage management markets for structured documents. Relational databases have a long history; there are already three

generations of professional relational database designers and programmers. It is possible that the relational empire just swallows structured documents to it's already wide application area. Small and thus effective XML/RDF(S) databases are new to the market but there are already dozens of new alternative products available.

Bibliography

[1] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[2] Kal Ahmed, Danny Ayers, Mark Birbeck, Jay Cousins, David Dodds, Joshua Lubell, Miloslav Nic Daniel Rivers-Moore, Andrew Watt, Robert Worden, and Ann Wrightson. *Professional XML Meta Data*. Wrox Press Inc., 2001.

[3] A. M. Alashqur, S. Y. W. Su, and H. Lam. OQL: A query language for manipulating object-oriented database. In *Proceedings of the 15th Conference on Very Large Databases*. Morgan Kaufmann, San Francisco, 1989.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, New York, 1999.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[6] P. Bohannon, H. Korth, and P. Narayan. The table and the tree: Online access to relational data through virtual XML documents. In *Proceedings of the WebDB 2001 Workshop on Databases and the Web*, May 2001. http://www.bell-labs.com/user/bohannon/Papers/Rolex.ps.

[7] Ronald Bourret. XML and databases. 2001. http://www.rpbourret.com/xml/XMLAndDatabases.htm.

[8] N. Bradley. *The XML Companion*. Addison-Wesley, 2000.

[9] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[10] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An architecture for storing and querying RDF data and scheme information. Technical report, Free University of Ansterdam, 2001.

[11] D. Chamberlin, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu (Eds). Xquery 1.0: An XML query language. W3C Working Draft, 2001. http://www.w3.org/TR/2001/WD-xquery-20010607/.

[12] Donald D. Chamberlin, Jonathan Robie, and Daniela Florescu. Quilt: An XML query language for heterogeneous data sources. In *WebDB (Informal Proceedings)*, pages 53–62, 2000.

[13] Surajit Chaudhuri and Kyuseok Shim. Storage and retrieval of XML data using relational databases. In *Tutorials of the 27th International Conference On Very Large Data Bases, Roma, Italy, September 11-14, 2001*, pages 1–70. September 2001.

[14] Brian F. Cooper, Neal Sample, Michael J. Franklin, Gísli R. Hjaltason, and Moshe Shadmon. A fast index for semistructured data. In *Proceedings of the 27th VLDB Conference, Roma, Italy 2001*, pages 341–350, September 2001. http://www.vldb.org/conf/2001/P601.pdf.

[15] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. XML-QL: A query language for XML. W3C note., 1998. http://www.w3.org/TR/NOTE-xml-ql.

[16] D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce*. Springer-Verlag, 2001.

[17] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language and processor for a web-site management system. In *Proc. Workshop on Management of Semistructured Data*, Tucson, 1997. citeseer.nj.nec.com/fernandez97query.html.

[18] D. Florescu, A. Levy, and A. Medelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 3(27):59–74, 1998.

[19] Lars Garshol. *tolog — A topic map query language*. XML Europe 2001, 2001. http://www.ontopia.net/topicmaps/materials/tolog.html.

[20] A. Halevy. IEEE Data Engineering, special issue on XML data management, June 2001.

[21] J. Hjelm. *Creating the Semantic Web with RDF*. John Wiley & Sons, New York, 2001.

[22] T. A. Howes, M. C. Smith, and G. S. Good. *Understanding and Deploying LDAP Directory Services*. Macmillan Technical Publishing, USA, 1999.

[23] G. Karvounarakis, V. Christopides, D. Plexousakis, and S. Alexaki. Querying community web portals, 2000. http://139.91.183.30:9090/RDF/publications/sigmod2000.html.

[24] R. Korfhage. *Information storage and retrieval*. John Wiley & Sons, New York, 1997.

[25] R. Ksiezyk. *Answer is just a question [of matching Topic Maps]*. XML Europe 2000, 2000. http://www.gca.org/papers/xmleurope2000/pdf/s22-03.pdf.

[26] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[27] Pedro José Marrón and Georg Lausen. On processing XML in LDAP. In *Proceedings of the 27th VLDB Conference, Roma, Italy 2001*, pages 601–610, September 2001. http://www.vldb.org/conf/2001/P601.pdf.

[28] Steve Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe 2000, Paris, France*, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.

[29] G. Riccardi. *Principles of database systems with Internet and Java applications*. Addison-Wesley, 2001.

[30] J. Robie, J. Lapp, and D. Schach. *XML Query Language (XQL)*. WWW The Query Language Workshop (QL), Cambridge, MA, 1998.

[31] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. Database system concepts. pages 440–442. McGraw-Hill International Editions, 1997.

Chapter 5

Ontological Theories for the Semantic Web

Aki Kivelä and Eero Hyvönen

This paper overviews ontology and ontological theories from different perspectives. The origins of ontology as a philosophical discipline are first discussed. Ontology has more recently also been studied in other sciences, such as linguistics and computer science. Our focus is on ontology in computer science and, especially, in its application to creating the Semantic Web. Two prominent languages for creating ontological theories in this field are briefly discussed: RDF Scheme and Ontology Inference Layer (OIL) on top of it. Classification schemes for ontological theories are also presented in order to give a broader perspective to the semantic contents of ontologies.

5.1 Perspectives to Ontology

"Ontology" [9] is a slippery term referring to various different topics depending on the context. First, the term refers to an established field of science studying the definitive classifications of entities that exist. Secondly, the term is also used to refer to specific theories, i.e., actual entity classifications that the field-of-science-ontology has produced. Contemporary adaptation of ontology in the computer science and especially in artificial intelligence [33] has made the terminological tangle even more confusing. Often an ontology application is also called as "ontology" [19].

In the following, the traditional perspective of ontology as a philosophical discipline is first discussed.

Category	*Example*
Substance	A cat
Quality	The cat is black
Quantity	The cat is two-feet high
Relation	The cat is a half of the size of the dog
Where	The cat is in the house
When	The cat came out yesterday
Position	The cat sat
Having	The cat has a rat
Action	The cat is running
Passion	The cat desires fresh fish enthusiastically

Table 5.1: The ten categories of Aristotle's ontological theory [5].

5.1.1 Philosophical Perspective

Philosophical ontology is a branch of metaphysics that specifies the most fundamental categories of existence [20]. Philosophical ontology classifies concepts and examines distinctions that underlie every phenomenon in the world [22].

The philosophical perspective to ontology dates back to Aristotle's (384-322 B.C.) works on metaphysics discussing the nature of being, i.e., discovering the ultimate essence and the reason for being behind the nature (physis) as we perceive it. Aristotle's ontological theory of the world consisted of the ten different categories of the table 5.1.

Although the first ideas of ontology were outlined by Aristotle, the term "ontology" was first introduced 1613 independently by Rudolf Göckel, in *Lexicon Philosophicum* and by Jacob Lorhard, in *Theatrum Philosophicum* [34]. The term originates from Greek and means the "General Doctrine of Being", also translated as the "Science of Being". Due to the fundamental nature of ontology, it is also called as the "First Philosophy" and the "First Science"[1] [7]. On some occasions ontology is also treated as a synonym for the metaphysics [22, 34]. Since Göckel and Lorhard both defined the ontology very shortly in their dictionaries, a number of historians credit Johannes Clauberg as the inventor of the ontological discipline. Clauberg's merit was that he differentiated ontology from theology, and gave ontology the justification for being a research area of it's own. In the prologue of his *Elementa Philosophiae sive Ontosophiae*, published in 1647, Clauberg writes[2]:

> "Since the science which is about God calls itself Theosophy or Theology, it would seem fitting to call Ontosophy or Ontology

[1] Center for Commercial Ontology, http://www.acsu.buffalo.edu/~koepsell/center.htm, November, 2001

[2] http://www.formalontology.it/

> that science which does not deal with this and that being, as distinct from the others owing to its special name or properties, but with being in general."

German philosopher Christian Wolff (1679-1750) popularized ontology in his *Philosophia Prima sive Ontologia*, published in 1730. The work attempted to describe the methodology of ontology and made a distinction between ontology and other branches of metaphysics. As a mathematician and a logician, Wolff stated that ontology should use a rational and a deductive method while investigating the nature of entities [7].

The major classical categories of ontology developed before the 20th century are materialism, idealism, and realism [29].

- *Materialism* is a category of ontological theories asserting that the universe consists of only physical objects interacting with each other. Perhaps the most well-known example of materialism is the philosophy of Thomas Hobbes (1588-1679). In his view, the universe is corporeal, i.e., that all that is real is material, and what is not material is not real [22].

- *Idealism* contradicts materialism and claims that everything is fundamentally spiritual. Objective idealism states that the spirituality is independent of our conscious. According to subjective idealism — also called phenomenology — the spiritual and ideal world is dependent on the sensing subject. George Berkeley (1685-1753) manifested under phenomenology that "Being is to be seen", Esse est percipi.

- *Realism* is a category of ontological theories describing that reality exists independently of human consciousness. Realism is not a distinct theory from idealism and materialism but certain parts of idealism and materialism can be seen also in context with the realism. For example, subjective idealism is one such part.

Formal ontology [16] is perhaps the most active contemporary school of philosophical ontology. Formal ontology is defined as

> "a systematic, formal, axiomatic development of the logic of all forms and modes of being."

Formal ontology deals with categories that are merely a form of being. This is different from material ontology dealing with material realizations of the categories. Material ontology is often seen as a counterpart of formal ontology. Formal ontology relies heavily on the concepts of *mereology* and *topology*. Mereology is a theory of part-whole relations while topology studies

the connection relation. Formal ontology can be seen as a study of object distinctions.

Contemporary formal ontology originates from the works of two different schools of philosophy. The school of analytic philosophy includes numerous philosophers agreeing upon the idea of descriptive metaphysics proposed by Strawson. Descriptive metaphysics tries to explain the actual structure of our thought about the world. Analytic approach employs formal methods, such as formal logic. Another school that has strongly influenced formal ontology is the school of Manchester relating closely to phenomenology in the tradition of Bretano and Husserl, where the research focuses on fundamental categories, such as object, state of affairs, part and whole, relations between parts and the whole, and their laws of dependence[3] [16].

5.1.2 Linguistic Perspective

Contemporary research on natural language processing (NPL) [25] is also addressing increasing interest in ontology. The motivation behind the interest grows from major difficulties encountered in robust speech recognition, fluent machine translation, and other branches of machine aided natural language processing. The principal difficulty in processing language lies in dealing with meaning. Although the contemporary applications handle language morphologically and even syntactically well, they are unable to sufficiently "understand" the meaning, semantics, underlying the expressions and language pragmatics. Especially, applications lack common sense, i.e., adequate semantic knowledge of real world domains required in trivial and everyday needs [25].

In the linguistic perspective, ontology is seen as a method to improve NPL applications' semantic understanding of natural languages. Ontology and — more importantly — ontological theories attempt to represent human knowledge in a structured way. It is assumed that the ontological theories bring in semantic knowledge of language entities and improve the overall quality of natural language processing. This very practical and problem-oriented usage of ontological theories is different from the philosophical view of ontology aiming at universally valid generic knowledge.

The study of meaning is generally called *semantic theory*. Semantic theory provides two different methods to describe meaning explicitly. Semantic discourse captures the meaning of natural language expressions in a way similar to encyclopedias and dictionaries, for example. The fact that phrasal explanation techniques are rarely formal enough to be computer-traceable has given a rise to a new method called semantic representation. The representational method describes the meaning of linguistic expressions and their

[3] http://www.formalontology.it/

Number	*Class*
1.	Abstract relations
2.	Space
3.	Matter
4.	Intellect
5.	Volition
6.	Affections

Table 5.2: The six top-level categories of Roget's Thesaurus [32].

relations using formal notation systems [40]. An ontological theory is a special kind of categorial representation system written in a knowledge representation language [35]. Categorial representation systems are a subclass of representational methods.

The linguistic approach to ontology derives from Peter Mark Roget's work *Thesaurus of English Words and Phrases Classified and Arranged so as to Facilitate the Expression of Ideas and Assist in Literary Composition*, published in 1852. Roget's motivation was to develop a tool that helps the analysis and classification of ideas and communication between people [4]. Both intended purposes are objectives of contemporary ontological theories, too.

The top-level of Roget's thesaurus[4] [32] contains the six broad classes of table 5.2.

At the bottom level, the thesaurus contains concept categories each of which is described by using similar words and example phrases [4]. Words in each category entry are separated into nouns, verbs, adjectives, and other parts of speech. Neighboring categories are semantically related. For example, category 266 is "Journey" and category 267 is "Navigation", both under the more general supercategory of "Motion", a subcategory of the 2. top-level class "Space". The edition [32] contains over 100,000 words indexed in this fashion into 1000 categories. As a consequence of the classification technique, Roget's thesaurus can be seen as a network of related words according to a hierarchical concept ontology. For example, the class Space has the following structure:

```
CLASS 2. Space
  I Space in general
    Abstract space
      180 Indefinite space
      181 Definite region
      182 Limited space
    Relative space
      183 Situation
      ...
```

[4]Online version of Roget's thesaurus is available at http://www.thesaurus.com/. Lexico LLC, Thesaurus.com, 2002.

Symbol	*Relationship*
SN	Scope note
USE	Equivalent to "see" reference
UF	Use for, reciprocal of USE
BT	Broader term, in a hierarchical array
NT	Narrower term, in a hierarchical array; the reciprocal of BT
RT	Related term, expressing any useful relation other than BT/NT

Table 5.3: More or less standard symbols and relationships used in thesauri [10].

```
   Existence in space
     ...
II Dimensions
  General
    192 Size
    ...
  ...
II Form
  ...
IV Motion
  ...
```

Thesauri organize words. This is contrast with conceptual ontologies that organize concepts underlying the words. Such an ontology is language independent in nature. For example, a single conceptual ontology can be manifest itself as a set of similar thesauri in different languages.

Linguistic terminological thesauri, such Roget's thesaurus, are rarely formally defined. Such a thesauri typically employ only a small number of relationships to organize the terms, such as those listed in table 5.3. Also references to synonymity, antonymity, and homonymity may be explicitly presented.

The vocabulary of a word-oriented ontology, thesaurus, contains lexicalized concepts shared by a linguistic community. Such a vocabulary excludes, for instance, ad hoc concepts, such as "flying giraffe", not found in dictionaries. Terminological ontologies are also called as lexical ontologies. [25]

Terminology itself is a relative new research field having it's roots in linguistic and cognitive science. Terminology is defined as [11]

> "a theory concerned with those aspects of the nature and the functions of language which permit the efficient representation and transmission of items of knowledge in all their complexity of concepts and conceptual relationships."

The objective of terminology is to enhance human communication. A practical application of terminology is language translation. Descriptive terminology tries to collect and describe terms. Terminology science still uses

informal description methods to capture the meaning of terms. As a result, human users are needed for term interpretation and computational support for terminology maintenance and navigation is difficult. Traditional terminologies are criticized of their lack of expressive mechanisms to represent, maintain, and reason about complex knowledge in an explicit form. Contemporary ontology could promote the use of computerized and more powerful knowledge representation systems to explicitly formalize conceptual knowledge in terminologies. This could also ease the maintenance of text corpuses[5] because, for example, formal definitions enable automated consistency checking. Also artificial agents could utilize formally defined corpus knowledge, which would, for example, improve the quality of NLP applications [11].

5.1.3 Information System Perspective

Ontology and ontological theories have also found a promising application area in developing information systems. The term "information system" refers to a wide range of resources within organizations related to the collection, management, use, and dissemination of information [27, 8]. Information systems is a multi-disciplinary research field combining primarily computer and management sciences.

Ontology as a science was first discovered by the computer science community in the late sixties. Probably the first appearance of ontology occurred in 1967 in G. H. Mealy's paper *Another look at data* where he brings in the question of what is data and how it is related to the world. Mealy recognizes the question as an ontological issue. Ontology has an important role in clarifying the concepts of information and information systems. [34]

Also ontological theories, i.e. ontologies have important role in information systems. Ontologies can be transformed into reusable information system components and be used as a basis of several information systems. Although component-based re-usage reduces the costs creating information systems and improves their quality finding a suitable ontological component may be difficult because information systems are often domain specific.

Ontological theories are also used as tools for improving the development process of an information system. Ontologies provide information system designer a CASE-like tool for conceptual analysis. Ontologies help not only information system engineering but also re-engineering by better maintainability [18]. Ontological theories may also be used within information systems in a similar way as schemas used in database systems. For example, they enable vocabulary detaching in the user interface, and assist communication between software agents [18].

The rapid expansion of the World Wide Web (WWW) in the 90's has created several application areas for ontologies. Most information on the WWW

[5] A large collection of language material in machine-readable form is called a corpus.

is in the form of unstructured text. Natural language processing techniques must be used to extract the meaning from the textual data. Unfortunately, these techniques are still immature when dealing with semantics. As a result, data extraction and mining easily produce false interpretations and is inefficient. Tagging web content with explicit semantics, i.e., metadata assists semantic interpretation significantly. Motivated by this observation, the Semantic Web activity of the World Wide Web Consortium (W3C)[6] is developing content annotation technologies, such as the Resource Description Framework (RDF) [26] and RDF Schema [2], and ontology technologies and languages, such as DAML+OIL[7].

5.1.4 Knowledge Engineering Perspective

Knowledge Engineering (KE) is an application-oriented branch of Artificial Intelligence [33] concentrating on the development of Knowledge-Based Systems (KBS). A goal of KE is to increase the quality of these engineering artifacts. The initial paradigm of KE focused on the problem of transferring human knowledge into an implemented knowledge base. The transfer approach assumed that all the knowledge already existed and that the engineer's task was to collect the knowledge, formalize it, and store it into a computer. The assumption was wrong partly due to the substantial role of the common sense and tacit knowledge the experts tend to use when solving problems. Contemporary knowledge engineering has taken a step forward and sees the engineering task as a modeling activity, where an adequate model of the expert's problem solving methods is constructed instead of a simulation or the complete model of the expert's cognitive processes. [36]

In the beginning of the 90's, knowledge engineering community was urgently looking for new techniques. At that time most knowledge-based systems were constructed from the scratch; there was an undeniable need to develop the engineering process required to realize bigger and more reliable systems cheaply from re-usable components [36, 12]. Ontology had already been noticed in linguistics and it was only a natural step to adopt ontology in knowledge engineering. Knowledge engineering obtained ontology's concept of ontological theory and a toolkit of ontological methods.

Broadly speaking an ontological theory — an ontology — is a representation of a given domain and captures certain knowledge of that domain. Ontologies are used in knowledge engineering as special kinds of knowledge bases that offer a reusable and shareable framework for knowledge-based systems. A single ontological theory may be a fundamental component of several different knowledge bases. Ontological engineering is a branch of knowledge

[6] http://www.w3.org/2001/sw/
[7] http://www.daml.org

engineering using the principles of formal ontology to build ontological theories. [19]

In it's strongest form, an ontological theory tries to capture universally valid knowledge. This form of requirement originates from the philosophical traditions of ontology. Knowledge engineering community soon realized that the philosophical demand was too strict in order to utilize ontology in practical knowledge-based systems. A weaker requirement avoiding the pitfalls of universal knowledge states that ontological theory should capture human common sense knowledge. The weakest but most practical view claims that ontological theories should capture domain specific knowledge. These three levels are also discovered during the discussion of ontological theory classifications later in this paper. Although ontological theories are often used to model the domain knowledge, ontological theories may model problem-solving knowledge, too. Ontological theories focusing to problem solving methods are called *method ontologies* [36].

In the late 90's, the knowledge management community found ontological theories suitable for organizational memories. An organizational memory is a comprehensive computer system that captures the accumulated know-how and other forms of knowledge assets of an organization and makes them available to enhance the efficiency and effectiveness of knowledge-intensive work processes [39]. The most important knowledge assets of many organizations are related to human resources. Typically such knowledge is only partially explicit, there is no single interface to access the knowledge, and knowledge assets are distributed all over the company. As a consequence, the knowledge is often inefficiently used and sensitive to organizational changes, such as down-sizing. This has given the motivation to adopt new strategies to maintain valuable knowledge assets of an organization. Ontological theories provide a framework for organizational memories [39].

It is useful to investigate different levels of knowledge representation in order to understand the difference between general and ontological knowledge bases. Nicola Guarino has proposed five different levels of knowledge representation based on Ron Brachman's classification [16, 15]. Guarino's levels of knowledge representation are logical, epistemological, ontological, conceptual, and linguistic (cf. the figure 5.1).

The logical layer of knowledge representation expresses the knowledge using logical primitives such as propositions, predicates, and logical operations allowing their formal interpretation. Epistemological layer uses structuring relations to collect formally defined primitives of the logical level to conceptual units that must be taken as a whole. Both logical and epistemological levels have no strict interpretation with respect to the application domain in question. The vast number of possible interpretations of the logical and epistemological knowledge representation is the motivation for the ontological level. Ontological level specifies explicitly all ontological commitments

Level	Primitives	Interpretation	Main feature
Logical	Predicates, functions	Arbitrary	Formalization
Epistemological	Structuring relations	Arbitrary	Structure
Ontological	Ontological relations	Constrained	Meaning
Conceptual	Conceptual relations	Subjective	Conceptualization
Linguistic	Linguistic terms	Subjective	Language dependency

Figure 5.1: Ontological levels of knowledge according to N. Guarino [16].

of representation primitives and restricts the number of possible interpretations of them. Ontological level knowledge is inspected against the ontological categories used to describe the domain giving a real life meaning for the knowledge primitives. Primitives at the conceptual level have a cognitive but language-independent interpretation. Linguistic level primitives are language dependent words and sentences. [16, 15]

5.1.5 Pragmatic Perspective

Previous subsections have discussed the ideas of ontology and ontological theories and their usage in philosophy, linguistics, information systems, and knowledge representation. This chapter summarizes the arguments for promoting ontology and ontological theories and investigates problems faced.

Ontology is an old discipline, dating back to Aristotle, and has been revised many times along the history by philosophers, such as Christian Wolff, Edmund Husserl, and Franz Bretano. More recently, other research fields, such as linguistics and computer science, have become increasingly interested in ontology and especially in applications of ontology. Ontology provides methods to model the real world. As a consequence, ontology has become a multi-disciplinary research field.

The question of the real world's relation to it's model is very difficult and often neglected in ontological theories. Traditionally, computer science and especially artificial intelligence views the world distinct of the model and takes it granted that the model is able to address real world entities with symbolic and computational expressions.

Bateman discusses this issue [1] and is concerned with the metaphysical and semiotic problems of ontology. Metaphysical problems arise from ontology's objective to model reality independent of our knowledge about it. The problem is that ontology — at least the traditional philosophical ontology — tries to build up a world classification that should be independent of our vision of the world. Bateman manifests that in order to account ontology to be a concern of knowledge engineering, we should first be able to specify the relation between things-themselves and our knowledge of them. Unfortu-

nately, such a relation has not been found. Instead, many philosophers have argued that the world of human being is essentially committed and intersubjective. Our world is already organized ontologically in intersubjective terms of human interest [1].

Bateman's semiotic problem addresses the incomplete and misleading interpretation of the concepts of the world and natural language. This hinders the construction of ontological theories. Traditionally and especially in AI the concept of natural language is considered to be similar to logic. This view underestimates the expressiveness of natural languages and blurs the relationship between the natural language and the world. Bateman states that the world is a higher-order socio-semiotic construct constituted by meanings and we should not consider the world as an unproblematized construct. [1]

Walking the same paths as Bateman, Barry Smith expresses the doubt [34] that many ontological theories based on set-theoretic languages, such as KIF, are implicitly committed to the view that the world is fundamentally a set-theoretic construction. Smith asks if an ontological theory expressed in such a language conceives a theory of reality or is it just a collection of formulas together with a set-theoretic semantics. Many computer scientists building ontological theories avoid the objective of traditional philosophical ontology and rarely speak of a true model but more practically of a useful or an adequate model. Smith argues that this use of words draws a relation between the reality and the ontological theory modeling it. One should be able compare the ontological theory to the reality in order to conclude that the model is adequate [34].

What are the benefits of the ontological theories? Ontological theories are engineering artifacts and the motivation of such theories is often practical. An ontological theory is constructed for a certain purpose. Common to ontological theories is that they model a certain fragment of the real world and try to capture, not the state of affairs, but more abstract constrains of the world. Ontological theories are expected to be reusable decreasing the construction costs and improving the overall quality of the knowledge systems based on ontological theories. Application areas of ontological theories can be found today, for example, in medicine, physical sciences, linguistics, and in business.

5.2 Ontology Languages for the Semantic Web

In the knowledge representation perspective, an ontology is not just an informal vocabulary or thesaurus of word meanings. The requirement for algorithmic interpretability means that the ontology should be represented in a

formal language that gives machine-processible meaning to the concepts and words used in terms of mathematical constructs.

There are lots of ontology languages and ontologies developed [6]. Their differences are various but there are also some general, domain independent constructs commonly used, such as:

- Classes. The generic concepts of the vocabulary are often called classes (or frames). For example, class "Tiger" can represent the generic category of the species tiger.

- Superclass relationship. Classes are organized into conceptual hierarchies in the spirit of object-oriented programming. For example, the class of tigers, Tiger, is a subclass of the class of carnivores, Carnivore.

- Class properties (slots). Class may have properties, often called "slots". Properties may be inherited by subclasses, which leads to representational economy and makes some simple inferences possible. For example, since carnivores eat meat and have sharp teeth also tigers do, because tigers are carnivores.

- Property features (facets). The class properties may have features and constraints of their own. Such features are sometimes called slot "facets". For example, value type and cardinality facets may be attached to a slot "parents" stating it's values must be of class type Person and that the number of values is at most 2.

- Individuals. The individual objects that the generic ontological theory talks about are called instances or objects. Each object is an instance of one or more classes. For example, Tony the tiger could be an instance of the classes Tiger and CartoonCharacter. The class membership relation is often called "is-a" relation and it should not be confused with the superclass relation (subsumption) defined between two classes.

- Axioms and constraints. The ontology may be based of formal logic and have axioms and additional constraints or inference rules. They express new, implicit relationships and constraints between the elements of the ontology. Axioms and rules are typically constructed by using terms, functions, predicates, operators, and quantifiers of predicate logic. The underlying logic systems often have features higher than first order, such as the possibility of expressing belief and other propositional attitudes.

An ontological theory defines the concepts in an application domain in terms of an ontological vocabulary. The application typically makes use of the objects by using axioms and logical rules of some sort. Such rules are

not part of the general domain ontology but rather tell how to use the concepts for problem solving in particular application tasks. This distinction of expressed by saying that the ontological (terminological) part of the systems in the *t-box* and the other inferences in the *a-box* of the logical system. This distinction between ontological and other task-specific reasoning is of course often superficial and difficult to make.

5.2.1 RDFS, a Minimal Ontology Language

In the W3C Semantic Web Activity initiative, the simplest generic ontology scheme is RDF Schema (RDFS) [2, 21]. It complements the Resource Description Framework [26] RDF with a vocabulary definition mechanism for hierarchical classes, properties, and constraints on their usage. The general idea is that RDF is used for giving metadata about web resources in terms of concept ontologies defined by RDF Schemas.

RDFS introduces basic object-oriented concepts into mark-up languages with simple intended semantics based on class hierarchies and property inheritance. The system includes a set of three core classes:

- Class: the general notion of the class.
- Resource: the class of resources being described.
- Property: the class of RDF properties.

There is also a set of core properties including the following:

- type: indicates the class of a resource instance. Resource instances belong to one or more classes.
- subClassOf: indicates the superclass of a (sub)class

Properties are treated as classes and may form a class hierarchy with inheritance of their own.

RDFS also makes it possible to express constraints on class properties:

- Range constraint: the value range of a property can be set by a class restriction.
- Domain constraint: the class on whose members the property can be used can be set as a restriction.

The idea with constraints is to define properties in terms of classes on which they apply. This property-based approach makes it easy to create new properties for describing existing resources. Also simple semantic validation can be based on property constraints; in XML only syntactic validation is possible.

5.2.2 Description Logic Layer

RDFS provides on simple ontology representations language but has several shortcomings including the following [28]:

- Properties of properties (facets) cannot be defined.
- Necessary and sufficient conditions for class membership cannot be distinguished.
- Equivalence or disjointness of classes cannot be expressed.
- Only constraints on property domains and ranges can be used.
- The semantics of the language is underspecified.

In many applications, deeper semantical modeling beyond RDFS is needed. OIL (Ontology Inference Layer) [23] can be seen as the next semantical layer on top of RDFS. It's central idea is to be downward compatible with RDFS semantics but, at the same time, to extend the language towards formal description logics. An ontology description in OIL can be translated into RDFS with additional OIL-specific constructs. An RDFS-aware interpreter can make use of OIL descriptions w.r.t. the RDFS semantics part by simply skipping OIL-based markup tags. This is in contrast with another major ontology language, DAML[8] that has its own constructs for many RDFS-constructs, which makes the systems incompatible. At the moment, W3C is planning to merge OIL and DAML into a common ontology logic language recommendation on top of RDFS. In the following, OIL is briefly described as an example of a logical ontology description language. The presentation is based on [3].

OIL is a frame-based (object-oriented) description logic system[9] with well-defined first order semantics. This means that the language can be provided with a reasoning support system, such as the FaCT system [24]. Reasoning is useful, for example, when determining the logical subsumption relations between descriptions and for determining whether an ontological theory is consistent.

OIL ontologies can be written using an easy to read syntax. For example, the fact that Tiger is a subclass of classes Carnivore and Mammal can be expressed as:

```
class-def Tiger
   subclass-of Carnivore Mammal
```

Properties (slots) may have type restrictions (has-value) as in RDFS.:

[8] http://www.daml.org

[9] For material on description logic research, see http://dl.kr.org/.

```
class-def Carnivore
  slot-constraint eats
     has-value Animal
```

This syntax can easily be transformed into RDFS- and XML-notation by the machine and then parsed for internal algorithmic interpretation.

OIL extends RDFS, e.g., in the following ways:

- Slot (property) constraints. More detailed constraints on properties can be defined. For example, a property may have minimum and maximum cardinality telling the minimum and maximum number of instances that can be used in the property value:

  ```
  class-def Chair
     slot-constraint has-legs
       min-cardinality 3 Leg
       max-cardinality 4 Leg
  ```

- Axioms. Axioms are a mechanism for expressing additional facts about the ontology classes. Classes may be defined to be equivalent or mutually disjoint. One can also demand that an instance belongs to exactly one or at least one class in a list of classes. For example:

  ```
  class-def Wine
     slot-constraint has-color
       value-type (one-of Red White Rose)
  ```

  Boolean expressions may be formed and can be used as a kind of unnamed classes:

  ```
  (Publication and (Newspaper or (not Book)))
  ```

- Defined and primitive classes. By default, a class definition is primitive meaning that class properties are a necessary but not sufficient conditions for class membership. For example, if the primitive class Tiger has the value type Animal for the eats-slot, and we know that some animal instance x eats Animal, one cannot deduce that x is a Tiger. On the other hand, if the class Carnivore is a defined class with the same property, then one can infer that if x east Animal them x indeed must be a carnivore.

- Slot definitions. In addition to domain and range restrictions on properties (slots), also other constraints can be defined.

  – Multiple restrictions are allowed for domain and range restrictions (that must be satisfied at the same time).

- Class expressions and concrete type-expressions can be used in domain and range constraints. For example:

```
slot-def age
   domain (Elephant or Tiger)
   range (range 0 70)
```

- Property qualities. Slots may have "inverse" slots and have relational qualities such as "transitive" or "symmetric". For example:

```
slot-def has-part
   inverse is-part-of
   properties transitive
```

  By this definition, with the help of the inverse relation and the transitivity quality, one can conclude, for instance, that if Europe has-part Finland and Finland has-part Helsinki, then Helsinki is-part-of Europe.

In above, the class system of OIL called "Standard OIL" has been discussed. Standard OIL includes most of the RDFS semantics. The only RDFS feature falling outside of Standard OIL is the higher-order notion of reification in RDF(S); OIL is a first order system.

Standard OIL language can be extended into "Instance OIL" by two additional constructs. Firstly, in Instance OIL, class instances can be expressed by the "instance-of" statement in addition to classes and properties. Secondly, named relations between instances and data can be asserted by the "related" statement.

5.3 Classifying Ontologies

An ontological theory — an ontology for short — is a set of formulas intended to be always true according to a certain conceptualization. An ontology is an artifact implemented using methods of ontological engineering. These artifacts can be built using different methodologies and for different purposes, and can be classified in many ways.

Uschold classifies [37] ontologies using the dimensions of formality, purpose, and subject of matter:

- *Formality* depends on how formally ontology's vocabulary is specified. Uschold's investigation reveals four levels of formality: highly informal, structured informal, semi-formal, and rigorously formal. Highly informal ontology uses very loose expressions, such as natural language to specify the vocabulary and the relationships between terms. Numerous glossaries are highly informal ontologies. Structured informal ontologies may rely on informal specifications of natural language, too,

but expressions are structured and the form of the expressions is restricted. Semi-formal ontologies are expressed in a formally defined language. Rigorously formal ontologies are expressed in formally defined language, and ontologies are proved to be sound and complete.

- *Purpose* depends on the intended use of ontology. Uschold introduces three main categories of purpose for ontologies. Ontologies are used for communication between people, inter-operability among systems, and for systems engineering benefits. Ontologies improve re-usability, knowledge acquisition, reliability, and specification processes in systems engineering.

- *Subject matter* reflects the general theme or application domain of the ontology. Categories of the subject of matter are domain, task, and representation ontologies. Ontologies of medicine, geology, or finance are domain theories. Task ontologies model processes of problem solving. Task ontologies are also called method or problem solving ontologies. Representation ontologies model representation languages.

The classification proposed by van Heijst et al. [38, 11] is similar to Uschold's classification. Van Heijst et al. classifies ontologies along two dimensions:

- Subject of ontology. The dimension of subject is analogous to Uschold's subject of matter dimension and categorizes ontologies to application, domain, generic, and representation ontologies. Also problem-solving ontologies form a category on the axis of subject although this is not included in the original proposition.

- The structure of conceptualization. This dimension distinguishes ontologies to terminological, information, and knowledge modeling ontologies. Terminological ontologies, such as lexicons and taxonomies, are language-dependent, use words as conceptual primitives, and contain a fixed and limited set of relationships. Information ontologies are specifications of the record structure of a database. Knowledge modeling ontologies specify conceptualizations of knowledge. [11, 17]

Guarino [17] argues strongly against the information ontologies category represented in the paper of van Heijst, et al. Guarino states that record structure specifications are symbolic level information and ontologies should contain ontological level information. Ontological level expresses the vocabulary and constraints of possible interpretations of the vocabulary while the symbolic level expresses the factual information. Although the category of information ontologies is inadequate, the classification criteria of ontological complexity remains useful.

Noy and Hafner [31] set up eight classes of questions to classify and evaluate ontologies:

- **G**eneral
  - The intended purpose of the ontology? Especially is the ontology general or domain specific?
  - How easy it is to integrate more general ontology into the ontology?
  - What is the size of ontology? Especially what is the number of terms, rules, links, etc.
  - What formalism has been used to build the ontology?
  - What is the implementation platform and language of the ontology?
  - Has the ontology been published?
- **D**esign Process
  - How was the ontology built?
  - Was the ontology formally evaluated?
- **T**axonomy
  - What is the general organization of ontology's taxonomy?
  - How many taxonomies the ontology contains?
  - What is in the ontology: things, processes, relations, and properties?
  - What is the treatment of time?
  - What is the top-level division?
  - How tangled and dense is the taxonomy?
- **C**oncept structure, concept relationships
  - Do concepts have internal structures?
  - Do concepts have properties and roles?
  - Are there other kinds of relation between concepts?
  - How are part-whole relationships represented?
- **A**xioms
  - Are axioms explicit?

  - How are axioms expressed?

- **I**nference mechanism
  - Does the ontology support reasoning and how the reasoning is implemented?
  - What are some instances of going beyond first-order logic?
- **A**pplications
  - What is the retrieval mechanism?
  - What kind of user interface the ontology supports?
  - What are the applications the ontology has been used in?
- **C**ontributions
  - What are the major strengths and contributions?
  - What are the weaknesses of the ontology?

In the following, some major general categories of ontologies classified according to their content and conceptual level of description are listed.

Top-level ontologies Top-level ontologies, called also upper ontologies provide a basic, domain independent vocabulary and object specifications to be used as the basis of other, more domain specific ontologies. Top-level ontologies aim to provide ontology engineering with a solid and tested frameworks to start with.

Standard Upper Ontology (SUO) work group of IEEE[10] has been specifying a standard top-level ontology since 2001. The objective of the working group is to assist the development of ontologies and advance knowledge engineering in general by providing a common ground for more specific domain ontologies. [30]

Generic ontologies Generic ontology is also called meta-ontology and core-ontology. Generic ontologies are reusable across domains. An example of generic ontology is a mereology ontology defining the part-of relation and its properties. Such an ontological theory can be used to model, for example, devices containing components.

General and common ontologies General ontologies capture human commonsense knowledge about our everyday life. A famous example of a general ontology is CYC[11] that has been under development since

[10] http://suo.ieee.org
[11] http://www.cyc.com

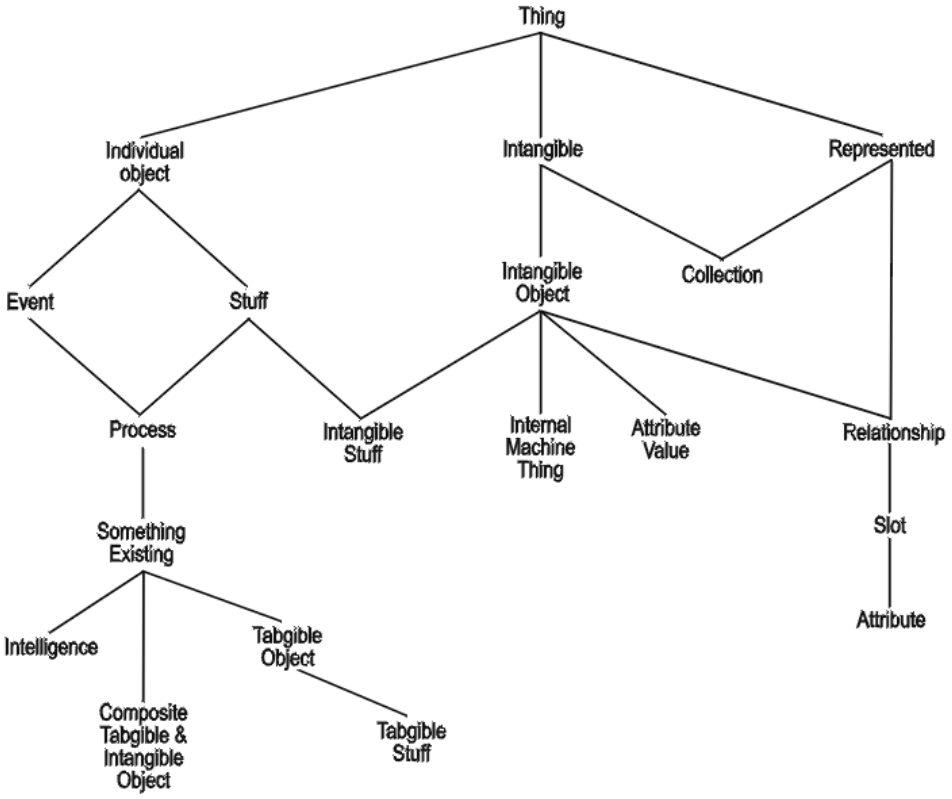


Figure 5.2: Top-level concepts of CYC ontology.

1984. It contains over 100 000 hand-coded assertions. Some sources classify CYC as a top-level ontology, too. Figure 5.2 illustrates the top-level concepts of CYC.

Domain specific ontologies Domain ontology describes a reusable vocabulary of a given domain. Top-level ontology can be used as the foundation of a domain specific ontology. For example, the mathematical modeling ontology EngMath is a domain specific ontology.

Task-specific and method ontologies A task-specific ontology provides vocabulary and knowledge used to solve problems associated to a certain task. Method ontologies provide terms and knowledge to particular problem solving methods. Method ontologies relate to task-specific ontologies.

Domain-task ontologies Domain-task ontology is a task specific ontology where the intended problem solving covers only problems in a given domain area.

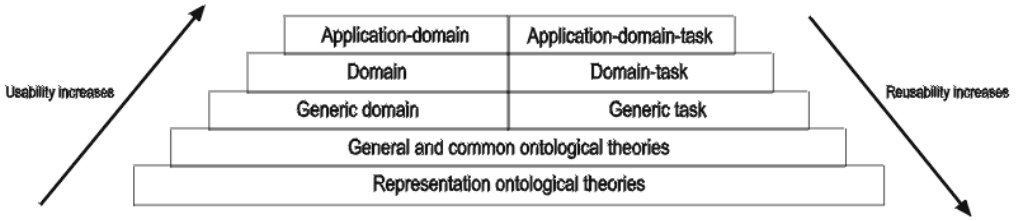


Figure 5.3: A pile of different ontology categories. More general ontologies are located under more specific ontologies, are less usable but more reusable. [13]

Application ontologies Application ontologies specify the vocabulary required to model a certain application.

Representation ontologies A representation ontology provides the representation primitives of knowledge representation paradigms without committing to any particular domain. The ontology does not express the exact purpose of the primitives but only offers a framework that enables the usage of the provided representation primitives. An example of this is Frame ontology [14] defining primitives such as class, subclass, attribute, relation, and axiom. Frame ontology is used as a basis for less general ontologies requiring these representation primitives.

5.4 Conclusions

"Ontology" is a central concept in the Semantic Web vision. An ontology gives shared meaning to a vocabulary in a machine-processible form, and can be used as the basis for implementing intelligent services and applications on the web. The term ontology itself has many meanings. In this paper, we have discussed the different perspectives to ontologies, overviewed two ontology languages for the Semantic Web as examples, and presented some classification schemes for ontologies.

The idea of ontology is not new but dates back to the times of Aristotle. Ontologies in one form another have been used in various applications already before the times of the computer and Internet. However, in Semantic Web research three interesting paths for ontology development seem to merge in a new, fruitful manner [9]:

1. Structured documents. Ideas and languages, such as SGML, HTML, and XML, developed in the structured document and web communities

form a syntactic basis for describing and communicating ontological information.

2. Object-oriented programming. Modeling and implementation ideas developed in the field of object-oriented programming now seem to be merging into document mark-up languages.

3. Description logics. Terminological logic systems developed in the field of artificial intelligence provide rich and well-defined semantics for ontological theories.

It seems that a new branch of engineering, "ontology engineering", for creating and maintaining ontological information systems is emerging. The new challenges in this field are not only theoretico-epistemological, but the actual business of creating ontologies involves lots of additional concerns, such as problems of merging ontologies, ontology maintenance, and agreeing about complicated semantic standards in the first place.

In remains to be seen how well these challenges will be met by the emerging Semantic Web research community.

Bibliography

[1] John A. Bateman. Ontology construction and natural language. Technical Report Internal Report 01/93, LADSEB CNR, Padova, Italy, 1993. In: N. Guarino, R. Poli (eds.), Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation.

[2] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[3] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An architecture for storing and querying RDF data and scheme information. Technical report, Free University of Ansterdam, 2001.

[4] Howard Burrows and Ramachandran Suresh. Digital library approaches to resource discovery in earth and space science. In *Proceedings of the Earth Observation and Geo-Spatial Web and Internet Workshop '98*. Instituts fur Geographie der Universität Salzburg, Salzburger Geographische Materialien, Volume 27, 1998. http://www.sbg.ac.at/geo/eogeo/authors/burrows/burrows.htm.

[5] Marc Cohen. Predication and ontology: the categories, January 2002. http://faculty.washington.edu/smcohen/320/cats320.htm.

[6] D. Fensel (ed.). The semantic web and its languages. *IEEE Intelligence Systems*, Nov/Dec 2000.

[7] Raul Corazzon (ed.). What is ontology? predication and ontology: the categories, November 2001. http://www.formalontology.it/section_4.htm.

[8] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, New York, 1989.

[9] D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce*. Springer-Verlag, 2001.

[10] D. J. Foskett. Thesaurus. In *Encyclopaedia of Library and Information Science, Volume 30*, pages 416–462. Marcel Dekker, New York, 1980.

[11] Johann Gamper, Wolfgang Nejdl, and Martin Wolpers. Combining ontologies and terminologies in information systems. Technical report, Institut für Rechnergestützte Wissensverarbeitung, University of Hannover, April 1999.

[12] A. Gómez-Pérez. Ontological engineering: a state of the art. Technical report, Facultad de Informatica, Universidad Politecnica de Madrid, 1999.

[13] A. Gómez-Pérez. Ontological engineering. tutorial on ontological engineering, 1999. IJCAI tutorial.

[14] T. R. Gruber. A translation approach to portable ontology spesifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[15] Nicola Guarino. The ontological level. In R. Casati, B. Smith, and G. White, editors, *Philosophy and the Cognitive Science*. Holder-Pichler-Temsky, Vienna, 1994.

[16] Nicola Guarino. Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human Computer Studies*, 43(5/6):625–640, 1995.

[17] Nicola Guarino. Understanding, building, and using ontologies. In *Proceedings of KAW'96, the 10th Knowledge Acquisition Workshop (Banff, Alberta, Canada)*, pages 9–14, November 1996. http://ksi.cpsc.ucalgary.ca/KAW/KAW96/guarino/guarino.html.

[18] Nicola Guarino. Formal ontology and information systems. In *Formal ontology in information systems. Proceedings of FOIS'98*, 1998.

[19] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases, towards a terminological clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*. IOS Press, 1995.

[20] F. Heylighen. Ontology, introduction. *Principia Cybernetica*, August 1993, modified 1995. http://pespmc1.vub.ac.be/ONTOLI.html.

[21] J. Hjelm. *Creating the Semantic Web with RDF*. John Wiley & Sons, New York, 2001.

[22] Thomas Hobbes (1588-1679). Great voyages. the history of western philosophy from 1492 to 1776, 2002. http://www.orst.edu/instruct/phl302/philosophers/hobbes.html, edited by Bill Uzgalis, Oregon State University.

[23] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, F. van Harmelen, M. Klein, S. Staab, and R. Studer. OIL: The ontology inference layer. Technical report, University of Manchester, England, 2000. http://www.ontoknowledge.org/oil/.

[24] I. Horrocks and P. Patel-Schneider. Optimizing description logic subsumbtion. *Journal of Logic and Computation*, 9(3):267–293, 1999.

[25] Eduard Hovy, Nancy Ide, Robert Frederking, Joseph Mariani, and Antonio Zampolli. Multilingual information management: current levels and future abilities. A Report Commissioned by the US National Science Foundation and also delivered to the European Commission's Language Engineering Office and the US Defense Advanced Research Projects Agency, April 1999. http://www-2.cs.cmu.edu/ ref/mlim/index.html.

[26] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[27] Paul J. Lewis. *Information system development*. Pitman Publishing, 1994.

[28] S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, pages 46–53, March/April 2001.

[29] Pentti Määttänen. *Filosofia, johdatus peruskysymyksiin (in Finnish)*. Gaudeamus, Helsinki, 1997.

[30] Ian Niles and Adam Pease. Origins of the ieee standard upper ontology. Technical report, Teknowledge, Palo Alto, 2001.

[31] N. Noy and C. Hafner. The state of the art in ontology design. *AI Magazine*, (3), 1997.

[32] P. Roget, J. Roget, and S. Roget. *Roget's thesaurus of synonyms and antonyms*. Pathmaker Publications, Newton Upper Falls, Massachusetts, 1975.

[33] S. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach*. Prentice-Hall, New Jersey, 1995.

[34] Barry Smith. Ontology and information systems, draft, November 2001.

[35] J. Sowa. *Knowledge representation. Logical, philosophical, and computational foundations*. Brooks/Cole, 2000.

[36] Rudi Studer, Richard Benjamins, and Dieter Fensel. Knowledge engineering: principles and methods. Technical report, Institute AIFB, University of Karlsruhe, 1998.

[37] Mike Uschold. Building ontologies: towards a unified methodology. Technical report, Artificial Intelligence Application Institute, University of Edinburgh, September 1996.

[38] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in kbs development. *International Journal of Human Computer Studies*, 46(2/3):183–292, February/March 1997.

[39] Jos Vasconcelos, Cris Kimble, and Feliz Ribeiro Gouveia. A design for a group memory system using ontologies. In *Proceeding of 5th UKAIS Conference*. University of Wales Institute, April 2000.

[40] Nico Weber. Semantic theory and the representation of meaning. In *Die Semantik von Bedeutungsexplikationen*. University of Bonn, 1999. English version of chapter 1, http://www.spr.fh-koeln.de/Personen/Weber/Tomsk%20Lectures/NW_HBL.pdf.

Chapter 6

Semantic Web Tools

Paula Silvonen and Eero Hyvönen

Semantic Web technologies are becoming more widely used. At the same time, the need for simple user-friendly tools for developing and managing metadescriptions and ontologies is growing. Construction of machine-processible semantic components is hardly a hoped-for task for the developers of web pages, portals, services, and other applications, if the only way to build them is to learn lots of cumbersome new languages, and to construct every new knowledge component from the scratch. Fortunately, many helpful tools have recently emerged. This paper describes and classifies some tools used for creating RDF and Topic Maps metadescriptions, and for ontology construction.

6.1 RDF Tools

Resource Description Framework (RDF) [9] and its schema extension RDF Schema (RDFS) [1] together form the metadata representation standard promoted by W3C. Although RDF(S) is a relatively new specification, there are a number of tools available for working with RDF(S)-based descriptions.

This section surveys such tools divided into following categories:

- RDF editors
- RDF parsers
- RDF database interfaces

Other RDF-related tools include libraries, inference engines and logic systems, and vocabularies for metadata. The RDF specifications provide a lightweight ontology system to support the sharing of knowledge on the Web. Most of the tools are free or even open source.

6.1.1 RDF Editors

RDF XML-syntax is complicated and verbose for humans to use. RDF editors have been developed for easier creation of RDF metadata. The general idea is to focus on extracting only content terms from the user and let the computer generate the actual XML-format. In addition to inserting information, the editors enable editing, viewing, and navigating in RDF data.

DC Dot

The DC Dot[1] service is able to automatically generate Dublin Core metadata from a Web page, either as HTML tags or as RDF/XML. The tool provides a form, where the generated metadata can be edited. The metadata can then be converted to various other formats, such as USMARC (U.S. Machine-Readable Cataloging format), SOIF (Summary Object Interchange Format), IAFA/ROADS (Internet Anonymous FTP Archives/Resource Organisation And Discovery in Subject-based services), TEI (Text Encoding Initiative) headers, GILS (Government Information Locator Service), IMS (Instructional Metadata System) or RDF.

GramToR

The GraMToR[2] is a graphical RDF editor developed as a part of the XWMF (eXtensible Web Modeling Framework) project at the University of Essen. The graphical representation can be serialized into XML-RDF syntax and, in addition, into RDF triples. GraMToR is able to store an RDF data model in three formats: files with the extension .rdf contain XML-RDF syntax, .fmr-files contain triple notation, and .brm-files contain triple notation with additional position information for the corresponding graphical RDF representation.

Metabrowser

Metabrowser[3] is a web browser for cataloging web pages using schemas, such as Dublin Core, GILS, and AGLS (Australian Government Locator Service). It also contains a metadata editor, Metadata View, which enables annotating pages with metadata. Metabrowser supports custom Metadata Templates for entering metadata.

---

[1] http://www.ukoln.ac.uk/metadata/dcdot
[2] http://nestroy.wi-inf.uni-essen.de/xwmf/
[3] http://metabrowser.spirit.net.au/

Mozilla

Mozilla[4] represents various kinds of structured data in RDF form. For example, bookmarks, history, file systems, document structures, and sitemaps can be represented in RDF, to name a few. The creation, access, and manipulation codes for them are completely independent: each has its own storage system, editing and viewing tools, and query and manipulation APIs.

RDFPic

RDFPic[5] is a data-entry program that allows easy editing of metadata for pictures. The system has been designed to enable quick entry of metadata about photos. Most metadata fields to be filled by the user show by default the value that was entered for the previous photo, and give quick access to the values entered for the last few photos. This means that only fields that have to be changed from the previous photo need to be edited and the amount of typing is minimised.

RDFPic stores the metadata in RDF form inside the comment blocks of the JPEG file. JPEG limits each comment block to 64K, but there can be as many blocks as necessary. Typically the descriptions generated by the program are only a few hundred bytes long.

RDF Schema Editor

The RDF Schema editor[6] is an experimental prototype for viewing, editing and navigating in RDF data, based on RDF Schemas. The development is currently continuing with the second generation prototype in the WRAF (Web Resource Application Framework) project.

The Reggie Metadata Editor

The Reggie Metadata Editor[7] aims at the easy creation of various forms of metadata with one program. The Reggie applet can create metadata using the HTML 3.2 standard, the HTML 4.0 standard, the RDF format, and the RDF abbreviated format. The Reggie Metadata Editor uses a schema file to read in the details of all the elements in a set, their characteristics and descriptions. To create metadata based on a different element set or different language, one has to simply create a new schema file.

---

[4] http://www.mozilla.org/rdf/doc
[5] http://jigsaw.w3.org/rdfpic/
[6] http://jonas.liljegren.org/perl/proj/rdf/schema_editor/
[7] http://metadata.net

6.1.2 RDF Database Interfaces

One of the alternatives to persistently store and manipulate RDF data is to utilize the relational database technology. The approach is natural because RDF triples can be stored easily in relational database tables.

Algae

Algae[8] is a stackable RDF database query and assertion interface based on s-expressions (a data structure based on LISP for representing complex data). It enables structured queries and assertions based on those queries. The work goes currently on with the generalized Algae functions to query multiple databases and to assert into any database.

GINF

GINF[9] (Generic Interoperability Framework) from Stanford University includes an SQL database interface for storing and accessing RDF models. It provides extensible lightweight technology for the integration of heterogeneous components.

rdfDB

rdfDB[10] attempts to be a simple, scalable database system for RDF. The goals of this project are to build a database that supports a graph oriented API via a textual query language in the spirit of SQL, and to be able to load into the database RDF files with a given URI. Support for RDF Schemas and basic forms of inferencing will also be provided.

Redland RDF Application Framework

Redland[11] is a library that provides a high-level interface for RDF allowing the model to be stored, queried, and manipulated. Redland implements all RDF model concepts in its own classes and provides an object-oriented API for using them. Some of the classes — e.g., those providing the parsers and storage mechanisms — are built as modules that can be added or removed as required. The software includes interfaces for C, Perl, Python, Tcl, and Java. Redland is free/open source.

---

[8] http://www.w3.org/1999/02/26-modules
[9] http://www-diglib.stanford.edu/diglib/ginf
[10] http://web1.guha.com/rdfdb/
[11] http://www.redland.opensource.ac.uk/

The Web Resource Application Framework

WRAF[12] implements an RDF API. Its purpose is to enable the construction of applications that fully use the RDF data model in order to realize the Semantic Web. All data is described in RDF. The user interface is specified in RDF, to. Data presentation dependents on the user profile, situation context, and just what information can be found from trusted sources.

All functions and program code are named, described, and handled as RDF literals. Running an application can result in method calls to services on other Internet servers. New functions could be transparently downladed and executed from trusted sources by reference. The actual code is written in Perl but the system could be extended to use other languages, too. The development of applications is done in the same system used to run the application. WRAF uses interfaces to other sources in order to integrate all data in one environment, regardless of the storage format. Information can be updated from configuration files, databases, XML files, LDAP repository [7], etc.

6.1.3 RDF Parsers

RDF parsers are needed to make the RDF data utilizable for other applications. An RDF parser takes over after the XML processor is executed; it translates the XML representation into the RDF triplet data model. An API for manipulating the RDF model is provided. Some systems also prove a database system and a query language for storing and retrieving RDF data.

There are several RDF parsers available, both free, open source, and commercial. Some of them are listed below in alphabetical order.

ICS-FORTH RDFSuite

The ICS-FORTH RDFSuite[13] suite includes a validating RDF parser, an RDF Schema based database system, and an RDF query language for it.

Jena

The Jena Java RDF API and toolkit[14] is a comprehensive Java system created by HP Labs, Bristol. In addition to an RDF/XML parser, Jena contains RDF model API, an RDF query language, DAML[15] support, and persistent storage.

[12] http://uxn.nu/wraf/
[13] http://139.91.183.30:9090/RDF/
[14] http://www.hpl.hp.com/semweb/
[15] http://www.daml.org

6.1.4 Profium SIR

Profium SIR [16] provides a commercial API product for inputting, outputting, and querying of RDF metadata in Java.

SWI-Prolog Parser

SWI-Prolog Parser[17] provides an RDF parser as a part of its SGML package. The parser reads an RDF document and produces a 3-tuple representation of the data model.

RDF XSLT transformer

RDF XSLT transformer[18] accepts only an URI as input.

6.1.5 Redland

Redland[19] is a C-library that provides a high-level interface for RDF allowing the model to be stored, queried, and manipulated. The system includes the Raptor RDF Parser Toolkit for handling RDF/XML and other syntaxes. Interfaces for Perl, Python, Tcl, and Java are provided.

repat

repat[20]] is a callback-based RDF parser built on James Clark's expat [21].

SiRPAC

The Generic Interoperability Framework[22] (GINF) has been developed to facilitate integration of heterogeneous components. One of the main principles it employs is the generic representation of protocols, languages, data, and interface descriptions. The current implementation of the framework is based on RDF and includes various RDF tools including an RDF parser based on SiRPAC[23].

---

[16] http://www.profium.com

[17] http://www.swi.psy.uva.nl/projects/SWI-Prolog/packages/sgml/online.html, online demo.

[18] http://www.w3.org/XML/2000/04rdf-parse/

[19] http://www.redland.opensource.ac.uk/

[20] http://injektilo.org/rdf/repat.html

[21] http://www.jclark.com/xml/expat.html

[22] http://www-diglib.stanford.edu/diglib/ginf/

[23] http://www-db.stanford.edu/ melnik/rdf/api.html

6.2 Topic Map Tools

The Topic Maps[24] (TM) [13] is the metadata representation scheme for describing web resources standardized by ISO. At the moment, there are only few tools available for building Topic Maps.

An example is the Ontopia Knowledge Suite [25], which is planned to be a complete suite of tools for building and deploying topic map-based applications. It includes Topic Map Engine for accessing and manipulating the constructs found in topic maps, Topic Map Navigator for building web-based topic map delivery applications, and Full Text Search Integration software. The company Ontopia also provides Omnigator, a technology showcase and teaching aid. It is their only free Topic Map software available at the moment.

6.3 Ontology Modeling Tools

This section describes various tools developed for ontology construction and retrieval of ontological information. Knowledge-component reuse and sharing, and analysis of ontologies is made easier by different import and export languages. The tools differ by their expressive power: some tools support only certain predefined ontological primitives while others let the user extend the system by specifying new primitives. All tools do not support specification of instances for the ontology classes, so additional instance level tools for building knowledge bases may be needed. Another issue is the knowledge representation language and whether it can be used for inference. Some tools include inference engines and interfaces for building ontology-based queries. The choice of the right tool depends on the application. Many of the tools listed are still under development, so it is important to keep track of the new features. Also completely new tools are being created.

6.3.1 CODE4

CODE4 (Conceptually Oriented Description Environment)[26] is a general-purpose knowledge management tool developed in Artificial Intelligence Laboratory, University of Ottawa [17]. It can be used for analyzing, storing, and retrieving conceptual knowledge about some domain. The main features of CODE4 learnability and adaptability for various applications, such as natural language processing, software specification and design, and expert systems. Knowledge representation formalism used in CODE4 is called CODE4-KR. It is based on ideas adopted from frame-based inheritance systems, conceptual graphs, object-orientation and description logic systems. Motivation for

[24] http://www.topicmaps.net
[25] http://www.ontopia.net/ontopia/texts/product-wp.html
[26] http://www.csi.uottawa.ca/ doug/CODE4.html

CODE4-KR is increased expressiveness over the ability to perform complex inferencing, and suitability for non-AI specialist use.

6.3.2 CONE — COnceptual NEtwork Software

The CONE software [8] is a tool for creating, editing, and querying ontologies, developed by VTT Information technology[27]. It has been utilized mainly for product modeling in multi-lingual applications, and for modeling of business-related domains for text mining. CONE ontology consist of one or more conceptual models describing the concepts of some specific domain. These models can be seen as different viewpoints to the subject in the ontology or as individual parts of the ontology domain. Each model consists of concepts and relations between them and the models can be linked by using bridges as relations between the concepts in different models.

A concept in CONE has a type, referent, description, and properties. The user can freely define her/his own concept types with attached properties and give them values. The user decides what information is represented as properties, as new concepts, and as links between concepts. The user can assign a link with a type name; a relation link itself does not initially have any meaning. Bridges are special kind of links between concepts defined in different models (even between models defined in different ontologies). Bridges cannot be named, they can be seen as connections between concepts that share the same function in different models. Both concepts and relations can be instantiated by using the graphical user interface.

To support dynamic presentation, the models can be divided into discrete collections of concepts, called clusters. By default, each model contains only the default cluster into which all concepts are initially placed. Any number of additional clusters can be defined by the user, and each model has it's own unique set of clusters. Concepts may be moved from a cluster to another. One concept can be a part of only one specific cluster.

CONE models are currently exported either in Prolog or in (XML-)CARIN[10]. The approach is strongly based on the underlying relational database structure used in the CONE software. CONE software includes interfaces for inference and ontology querying in Prolog, and a Java API for instance data manipulation.

6.3.3 GKB-Editor

The GKB-Editor (Generic Knowledge Base Editor)[28] is a tool for browsing and editing knowledge bases across multiple Frame Representation Systems (FRS) in a uniform manner. It offers an intuitive, graph-based user interface,

[27] http://www.vtt.fi/tte/

[28] http://www.ai.sri.com/ gkb/

in which the user can edit a knowledge base through direct manipulation and selectively display the region of the knowledge base that is currently of interest. GKB-Editor has been developed at Artificial Intelligence Center (AIC) at SRI (Stanford Research Institute) International.

6.3.4 GrIT

The aim of the GRIP [2] group was to produce a user interface for conceptual graphs that supports the graphical manipulation of conceptual graphs. The result of this project was GrIT: a software package for managing conceptual graphs.

GrIT provides a graphical representation and a set of basic tools for managing conceptual graphs. Operations such as moving, deleting, resizing graphs are all handled using the mouse (drag and drop). Capabilities of GrIT correspond roughly to other ontological modeling tools in this paper. However, the GrIT paper [2] introduces some interesting ideas on the usage of three-dimensional user interface in conceptual graph interfaces. Complex conceptual graphs can be displayed as a three-dimensional model on overlapping planes, each filtering information being passed to the topmost plane. Different planes have their own color coding. They cannot be reorganized in order to get a different perspective to the graph.

6.3.5 JOE — Java Ontology Editor

Java Ontology Editor (JOE)[29] is an ontology construction tool developed in Center for Information Technology, University of South Carolina [11]. The basic idea behind JOE is to provide a knowledge management tool that supports multiple simultaneous users and distributed, heterogeneous operating environments. Ontologies, from JOE's point of view, are Entity-Relationship (ER) or Frame-Slot models of a given knowledge base. The ontology can be viewed in three different formats: as an entity-relationship (ER) diagram, as a hierarchy similar to the Microsoft Windows file manager, or as a graphical tree structure.

6.3.6 OilEd

OilEd[30] is a simple ontology editor which allows the user to build ontologies using OIL. OilEd is not intended as a full ontology development environment. It does not actively support the development of large-scale ontologies, the migration and integration of ontologies, versioning, argumentation and many other activities that are involved in ontology construction. Rather, it is meant

[29] http://www.engr.sc.edu/research/CIT/demos/java/joe/
[30] http://img.cs.man.ac.uk/oil/

to be the "NotePad" of ontology editors, offering just enough functionality to allow users to build ontologies and to demonstrate the use of the FaCT reasoner to check the consistency of those ontologies.

6.3.7 IKARUS

IKARUS[31] (Intelligent Knowledge Acquisition and Retrieval Universal System) is a web-based successor for the CODE4 knowledge management environment. IKARUS uses a hierarchical frame-based knowledge representation scheme. Frames store knowledge about the subject in the form of one or more statements, and they are organized hierarchically. Each subject may have multiple parents and any number of children, and they can be used to represent types (classes) as well as their instances. Statements contain the information about subjects either as predicates with well-defined syntax and semantics, or as unstructured fragments of information, such as text, URLs to web documents, etc.

6.3.8 OntoEdit

OntoEdit[32] is an ontology engineering environment developed at the Knowledge Management Group of Karlsruhe University. It enables inspecting, browsing, codifying, and modifying ontologies, and supports in this way the ontology development and maintenance tasks. Currently OntoEdit supports the following representation mechanisms: the serialization of the ontology model using OXML(XML-based storage format), DAML+OIL, Frame Logic and RDF-Schema.

6.3.9 Ontology Editor

The object-oriented Ontology Editor[33] by Steffen Schulze-Kremer enables exporting the data in either Prolog clauses or plain ASCII text files. Users can build hierarchical trees of entries in a graphical user interface. Concepts, which are presented in taxonomies, and instances are supported. Is-a and part-of relationships are supported by this tool.

6.3.10 OntoSaurus

OntoSaurus[34] was developed by the Information Sciences Institute at the University of South California. It consists of an ontology server that uses Loom

[31] http://www.site.uottawa.ca/ kavanagh/Ikarus/IkarusInfo.html
[32] http://ontoserver.aifb.uni-karlsruhe.de/ontoedit/
[33] http://igd.molgen.mpg.de/ www/prolog/oe.html
[34] http://www.isi.edu/isd/ontosaurus.html

as its knowledge representation system, and an ontology browser server that dynamically crates html pages to display the ontology hierarchy. The pages include also images and textual documentation. Translators from Loom to Ontolingua, KIF (Knowledge Interchange Format), KRSS (Knowledge Representation System Specification) and C++ are supported by Ontosaurus.

6.3.11 Protégé 2000

Protégé 2000[35] is probably the best- known ontology-development and knowledge-acquisition environment. It was developed by the Stanford Medical Informatics group of Stanford University. Protégé has three functions:

- Construction of domain ontologies.
- Customization of knowledge acquisition forms, together with extensions for graphical widgets for tables, diagrams and other components.
- A library which other applications can use to access and display knowledge bases.

The main idea behind the functions is to make the knowledge representation format adaptable for various ontology languages, whereas other ontology modeling tools tend to choose some specific languages to concentrate on.

6.3.12 Stanford KSL Ontology Editor

The Stanford KSL Ontology Editor[36] [3] is a set of tools and services that support distributed, collaborative editing, browsing and creation of Ontolingua ontologies. It was developed in the context of the ARPA Knowledge Sharing Effort by the Knowledge Systems Laboratory at Stanford University. The ontology server architecture provides access to a library of ontologies, translators to languages (Prolog, CORBA's IDL (Interactive Data Language), CLIPS (C Language Integrated Production System), Loom, KIF) and an editor to create and browse ontologies. There are three modes of interaction:

- Remote collaborators that are able to write and inspect ontologies.
- Remote applications that may query and modify ontologies stored at the server over the Internet using the generic frame protocol.
- Stand-alone applications.

[35] http://smi-web.stanford.edu/projects/protege/
[36] http://www-ksl-svc.stanford.edu:5915/

6.3.13 VOID

VOID[37] [15, 16], the KACTUS toolkit, is an interactive environment for browsing, editing, and managing (libraries of) ontologies. VOID supports the theoretical and application oriented work packages by providing an environment in which one can experiment with, e.g. Organisation of libraries of ontologies, translating between different ontology formalisms, and also perform practical work. For example, one can browse, edit, and query ontologies in various formalisms. The toolkit can handle various ontology formalisms, such as CML (Chemical Markup Language), EXPRESS[14] and Ontolingua[38], and can perform (partial) translations between these formalisms.

6.3.14 WebODE

WebODE[39] is a tool for modeling knowledge using ontologies developed by the Artificial Intelligence Department of the Computer Science Faculty of the Technical University of Madrid. It is based on the Methontology methodology [4, 6, 5] built in the Technical School of Computer Science in Madrid. WebODE ontologies can be integrated with other systems using its automatic exportation (WebODE's XML, RDF(S), OIL, XML-CARIN) and importation (WebODE's XML, XML-CARIN) services.

6.3.15 WebOnto

WebOnto[40] is a tool developed by the Knowledge Media Institute of the Open University (England). It supports the collaborative browsing, creation and editing of ontologies, which are represented in the knowledge modeling language OCML [12] through a graphical user interface. Its knowledge model has the same expressiveness as the OCML language.

6.4 Conclusions

In order to fully utilize the information available on the Web, we need to add machine-processible semantics to the Web contents. Various tools for creating metadata and ontologies for the Semantic Web have been developed in order to make knowledge exchange and sharing easier.

Important aspects to consider when choosing between them include:

- The knowledge representation scheme used.

[37] http://www.swi.psy.uva.nl/projects/Kactus/toolkit/about.html
[38] http://www.ksl.stanford.edu/software/ontolingua/
[39] http://delicias.dia.fi.upm.es/webODE/WebODE_Home.html
[40] http://webonto.open.ac.uk

- Exporting and importing capabilities, as there are various languages in use, and is quite impossible to say which of them are to stay in use.
- Scalability and adaptability.

Several issues need to be settled before standards could be developed.

Ontological modeling in general does not assign the developers to any language in particular.

Bibliography

[1] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[2] P. Eklund, J. Leane, and C. Nowak. GRIP, toward a standard GUI for conceptual structures. In *Proceedings of PEIRCE Workshop (ICCS'93)*, 1993.

[3] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: A tool for collaborative ontology construction. In *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1996.

[4] M. Fernández, A. Gómez-Pérez, and N. Juristo. METHONTOLOGY: From ontological art towards ontological engineering. In *Symposium on Ontological Engineering of AAAI*, 1997.

[5] M. Fernández-López, A. Gómez-Pérez, A. Pazos-Sierra, and J. Pazos-Sierra. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems and Their Applications*, January/February 1999.

[6] A. Gómez-Pérez. Knowledge sharing and reuse. In J. Liebowitz, editor, *Handbook of Expert Systems*. CRC, 1998.

[7] T. A. Howes, M. C. Smith, and G. S. Good. *Understanding and Deploying LDAP Directory Services*. Macmillan Technical Publishing, USA, 1999.

[8] T. Kankaanpää. Design and implementation of a conceptual network and ontology editor. Technical report, VTT Information Technology, Espoo, Finland, 1999.

[9] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[10] A. Y. Levy and M.-C. Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proceedings of the 12th European Conference on AI (ECAI-96)*, 1996.

[11] K. Mahalingam and M. Huhns. Ontology tools for semantic reconciliation in distributed heterogeneous information environments. *Intelligent Automation and Soft Computing: An International Journal on Distributed Intelligent Systems*, 1998.

[12] E. Motta. *Reusable Components for Knowledge Modelling*. IOS Press, 1999.

[13] Steve Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe 2000, Paris, France*, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.

[14] D. Schenck and P. Wilson. *Information Modeling: The EXPRESS Way*. Oxford University Press, 1994.

[15] A. Schreiber and P. Terpstra. Sisyphus-VT: A CommonKADS solution. Technical report, University of Amsterdam, 1995.

[16] A. Schreiber, B. Wielinga, and W. Jansweijer. The KACTUS view on the 'O' word. Technical report, University of Amsterdam, 1995.

[17] D. Skuce. A unified system for managing conceptual knowledge. *International Journal of Human-Computer Studies*, (42):413–451, 1995.

Chapter 7

Device, Document, and User Profiling on the Semantic Web

Sten Malmlund and Eero Hyvönen

Web content will be used more and more in different kind of devices, such as PDAs or cellular phones, whose computational capabilities and physical properties vary a lot. At the same time, also the needs of the users as well as the content and structure of the web content is becoming more and more diverse. These developments present a threat to web content providers who will have to meet the challenge that devices, services, and users may not match well with each other. This paper discusses how semantic web techniques can be used to solve the problem by providing metadata profiles for devices, documents, and users. We focus on standard proposals developed by W3C and FIPA.

7.1 Introduction

The advent of mobile devices and audiovisual media is currently changing the way people are interacting with the web. Web services are becoming accessible from a wide range of devices including cellular phones, TV, digital cameras, and in-car computers. The user base, usage situations, and ways in which a web service is used is becoming more versatile and dynamic depending, e.g., on the location of the device, networking environment, language used etc. As a result, content providers can no longer deliver only one static version of their content to the user, but need to adjust the layout and content depending on the capabilities of the viewing device, the network used, the service, and the user.

The heterogeneity of the devices used to access the information on the

web leads to problems. The clients may receive content that they cannot store or display, or it may take too long to convey the content over the network to the client device. A way to prevent difficulties is to describe the terminal to the server in such a way that the server can adapt the content to the client and make sure that the user gets the best possible presentation for her/his device. If one wants to adapt the presentation of the content to the device, one has to have a declaration of the properties of the device, i.e., device profiles. In the same way, if one wants to adapt the content according to what the user desires, one has to have a description of the user needs, i.e., a user profile. One also has to have a description of the content to match the content with the characteristics of the device and the needs of the user, i.e., document profiles. A solution approach to managing the complexity is to attach profiling metadata information to devices, documents, and users of the web. Services can the be adjusted dynamically based on the metadata descriptions.

This paper will discuss device profiles, annotations for document profiles, and user profiling. We first discuss CC/PP and FIPA Device ontology schemes for profiling devices. After this, document profiling and transcoding is in focus. As an example of user profiling, W3C Platform for Privacy Preferences (P3P) is then presented. In conclusion, practical usefulness of the profiling schemes is considered.

7.2 Device Profiles

Providing content for a single device or a browser may exclude large numbers of users but, on the other hand, re-authoring the layout or content manually for all different needs may be impractical as well. The threat we face is that new devices and services may not be interoperable with each other or with the existing web. This could cause fragmentation of the web space and make device independent authoring difficult.

7.2.1 Composite Capability/Preference Profiles

The Composite Capability/Preference Profiles (CC/PP)[1] is a user-side framework for content negotiation. It was issued as a W3C Note[2] on the 30th November, 1998.

Using profiles

A Composite Capability/Preference Profile is a collection of information which describes:

[1] http://www.w3c.org/Mobile/CCPP
[2] http://www.w3c.org/TR/NOTE-CCPP

- The capabilities that a particular device has when accessing the web. These include hardware, system software, browser, and applications used.

- The preferences that the user may have in addition to the default capabilities. For example, a device may be capable of rendering audio, but the user may turn this ability off.

When accessing a web page over the web the device sends, along with the HTTP request, a reference to the device's CC/PP profile and a set of user's preferences overriding the profile defaults. The profile is stored at some global repository since resubmitting the whole description with each request would be inefficient. By using the profile and user preferences the web server first resolves the actual profile to be applied and then customizes the presentation of the requested web page accordingly.

There are two main strategies for customizing the content.

- Different versions of the web content can be authored and stored, and the profile is only used by the server for selecting the right one.

- The content is written only once (e.g., an XML repository may be provided) and is transformed into customized pages by the server using the profile.

CC/PP Framework

The following review of CC/PP is largely based on Johan Hjelm's book [3], chapter "Device Descriptions and User Profiles", and on the papers and material W3C provides on CC/PP on their web site[3]. There are significant efforts to integrate web technologies into various devices. Mark H. Butler[4] has written a comprehensive report of current technologies related to the creation of device independent web content and web applications.

CC/PP is a framework; it does not mandate a particular set of components or attributes. The definition of the exact vocabulary is left for other standards. A CC/PP vocabulary is written in RDF [6] and consists of a set of classes expressed in RDF Schema [2]. The CC/PP core vocabulary has only seven entries, but is intended to be infinitely extensible through XML namespaces.

The capabilities and characteristics are referred to as attributes. Together they form a vocabulary whose semantics are given in a schema. A profile is an instance of the schema and contains one or more attributes from the

[3]http://www.w3c.org/Mobile/CCPP

[4]See http://www.hpl.hp.com/techreports/2001/HPL-2001-83.html for details.

vocabulary. The attributes in the schema are classified into one of several components, each of which represents a distinguished set of characteristics.

CC/PP profiles are structured into named components, each containing a collection of attribute-value pairs, or properties. The components may include

- the hardware characteristics (screen size, image capabilities, manufacturer, etc.),
- software characteristics (operating system vendor and version, a list of audio and video encoders, etc.),
- the application and user preferences (browser manufacturer and version, markup languages and versions supported, sound on/off, etc.),
- WAP characteristics (WML script libraries, WAP version, etc.), and
- network characteristics (device location, latency, reliability, etc.).

A CC/PP description is a collection of RDF statements [6]. The underlying data model therefore is a collection of RDF triplets. The profile consists of

- a number of *components* and
- attribute values attached to them.

Major components describe the client's

- hardware platform,
- software platform, and
- application software such as the browser or the email-system.

The default settings of a standard device profile may be overridden by persistent local changes and temporary changes made by user.

The following example taken from [7] illustrates the the idea of a CC/PP profile. The profile consists of the the following major components: the hardware platform (`HarwarePlatform`), the software platform (`SoftwarePlatform`), email (`EpocEmail1.0`), calendar (`EpocCalendar1.0`), and user preferences (`UserPreferences`). Each component is characterized by a set of named attribute values (RDF properties expressed as tag attributes). The particular vocabulary used is defined in the name space `prf` whose URI is given as an attribute value of the `rdf:RDF` tag. Some default values have been modified (overridden) by the user.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.w3.org/TR/WD-profile-vocabulary#">

<rdf:Description about="HardwarePlatform">
   <prf:Defaults
      Vendor="Nokia"
      Model="2160"
      Type="PDA"
      ScreenSize="800x600x24"
      CPU="PPC"
      Keyboard="Yes"
      Memory="16mB"
      Bluetooth="YES"
      Speaker="Yes" />
   <prf:Modifications
       Memory="32mB" />
</rdf:Description>

<rdf:Description about="SoftwarePlatform">
   <prf:Defaults
      OS="EPOC1.0"
      HTMLVersion="4.0"
      JavaScriptVersion="4.0"
      WAPVersion="1.0"
      WMLScript="1.0" />
   <prf:Modifications
      Sound="Off"
      Images="Off" />
</rdf:Description>

<rdf:Description about="EpocEmail1.0">
   <prf:Defaults
      HTMLVersion="4.0" />
</rdf:Description>

<rdf:Description about="EpocCalendar1.0">
   <prf:Defaults
      HTMLVersion="4.0" />
</rdf:Description>

<rdf:Description about="UserPreferences">
   <prf:Defaults
      Language="English"/>
</rdf:Description>

</rdf:RDF>
```

The profile information originates from multiple sources. The client settings are combined with the standard profile. If there are intermediate proxies between the content consumer and the provider, then the CC/PP information may need to be modified according to the proxy's behavior. Because different parts of the capabilities profile may be differently cached, the various components must be explicitly described in the network transaction. Each attribute is limited in its scope to the component it is describing.

UAProf Vocabulary

The WAPForum[5] has developed a vocabulary named UAProf[6] which is based on CC/PP. The UAProf service provides a mechanism for describing the capabilities of clients and the preferences of users to an application server. UAProf supports the client-server transaction model by sending client and user information to servers with the request. This information permits the servers to adapt their content accordingly in preparation for the response. This service model would also permit intervening proxies to provide value-added services by providing these adaptation services directly. Recognizing the importance of user's privacy, personal information submitted in these requests may be controlled by the user.

A CC/PP example is given below:

```
<?xml version="1.0"?>
<!-- Checked by SiRPAC 1.16, 18-Jan-2001 -->
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
     xmlns:prf="http://www.wapforum.org/UAPROF/ccppschema-20000405#">

  <Description about="http://example.com/MyProfile">
    <ccpp:component>
      <Description about="http://example.com/TerminalHardware">
        <type resource="http://example.com/Schema#HardwarePlatform" />
        <prf:cpu>PPC</prf:cpu>
        <prf:display>320x200</prf:display>
      </Description>
    </ccpp:component>

    <ccpp:component>
      <Description about="http://example.com/TerminalSoftware">
        <type resource="http://example.com/Schema#SoftwarePlatform" />
        <prf:name>EPOC</prf:name>
        <prf:vendor>Symbian</prf:vendor>
        <prf:version>2.0</prf:version>
      </Description>
    </ccpp:component>

    <ccpp:component>
      <Description about="http://example.com/Browser">
        <type resource="http://example.com/Schema#BrowserUA" />
        <prf:name>Mozilla</prf:name>
        <prf:vendor>Symbian</prf:vendor>
        <prf:version>5.0</prf:version>
        <prf:htmlVersionsSupported>
          <Bag>
            <li>3.0</li>
            <li>4.0</li>
          </Bag>
        </prf:htmlVersionsSupported>
      </Description>
    </ccpp:component>
  </Description>
```

[5] http://www.wapforum.org

[6] http://www.wapforum.org/what/technical.htm

```
</RDF>
```

This profile mixes the RDF namespace, the CC/PP namespace, and a UAProf namespace. `TerminalHardware` and `TerminalSoftware` are instances of the classes `HardwarePlatform` and `SoftwarePlatform` required in the CC/PP specification. The UAProf drafting committee has defined additional components:

BrowserUA A collection of properties that describe the browser application.

NetworkCharacteristics A collection of properties that describe the network related infrastructure and environment.

WapCharacteristics A collection of properties that describe the WAP capabilities supported on the device.

Additional components can be added by other groups simply by adding namespaces to the schema. To modify and override default attribute values without re-sending the whole profile (e.g., to turn sound from off to on) one would need to send the following data:

```
<?xml version="1.0"?>

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:prf="http://www.wapforum.org/UAPROF/ccppschema-20000405#">
     <Description ID="SoftwarePlatform" prf:Sound="On"/>
</RDF>
```

The CC/PP data structure architecture is protocol-independent regarding transportation of information between a client and a server. One way of transportation is to use the CC/PP Exchange Protocol[7], which enables CC/PP profiles to be transported over HTTP/1.1 using the HTTP Extension Framework[8]. It enables effective profile caching at web servers and proxies, in addition to overriding of URI values. In the WAP environment, the WAP Session Protocol[9] has been enabled to carry CC/PP.

Proxies can also cache both the information and the profile, as can origin servers. This can cause problems for transcoding proxies, which change the formatting of the information — the presentation — depending on the parameters of the terminal. The same URI will be returned using one set of presentation and filtering parameters to users with one parameterization;

[7] http://www.w3.org/TR/NOTE- CCPPexchange

[8] http://www.w3.org/Protocols/HTTP/ietf-http-ext/draft-frystyk-http-extensions-03.txt

[9] http://www.wapforum.org/what/technical.htm

a different presentation filtered in a different way will be returned to users with another set.

In a typical situation, the server or the proxy does content negotiation and personalization work, while the user's device (client) does presentation. This makes it possible to have very dumb terminals.

An HTTP proxy or origin server will deliver content from its cache only if the content has not expired and the profile information associated with the cached request exactly matches the profile information associated with the new request. There is one problem though: many caches do not actually use HTTP/1.1, so if you want to use CC/PP, it is better to ensure that all proxies and caches used support it.

The request from the client is intended for the origin server, e.g., by way of a WAP gateway and possibly other proxies in between. It is the responsibility of the origin server to receive the request and generate appropriate content, which is then delivered as an HTTP response to the WAP gateway and from there delivered to the client. As a request travels over the network from the client device to the origin server, each network element may, optionally, add additional profile information to the transmitted profile. These additions may provide information available solely to that particular network element. Alternatively, this information may override the capabilities exposed by the client, particularly in cases where that network element is capable of performing in-band content transformation to meet the capability requirements of the requesting client device.

The origin server extracts the profile information sent along with the HTTP request, resolves all indirect references to information stored at other repositories in the network, fetches the profile elements from them (if necessary), and uses that information to select or otherwise customize the content being delivered to the client.

The customization method is not specified anywhere and depends on the implementation in the origin server. It may be anything from traditional content negotiation, in which a variant that fits the user preferences is selected, to a full contextualization[10], in which information from a database is filtered depending on the user location.

7.2.2 FIPA Device Ontology

FIPA (Foundation for Intelligent Physical Agents) has also released a specification for device ontology[11]. Two devices may exchange device profiles, either directly or through a brokering agency, and acquire a list of services provided by the other device. The list of services may include both hardware and software services. The profile needs to support the identification

[10] See [3] for more details on contextualization.

[11] http://www.fipa.org/specs/fipa00091/

of services for various input and output capabilities, such as audio input and output.

There is an overlap between the definitions of CC/PP, UAProf documentation, and the FIPA specification.

The FIPA Device ontology can be used by agents when communicating about devices. Agents pass profiles of devices to each other and validate them against the FIPA Device ontology. Agent A1 can ask agent A2 whether device D has enough capabilities to handle some task A1 has in mind.

FIPA specification is a general framework which can be realized in different frameworks and languages. The other specifications rely on specific frameworks and languages, namely RDF and XML.

FIPA uses the general concept of the *frame* which corresponds to the notion of class used in RDF. Frames have *parameters* with values in the same way as attributes (RDF properties) are used to characterize components in CC/PP. For example, the FIPA frame `device` has the parameters `hw-properties` and `sw-properties` that are lists of properties describing the hardware and software features of the device in question. The underlying data models of the specifications are essentially compatible, which makes it easy to their usage. Furthermore, the frames in the FIPA specification use many similar concepts to the CC/PP specification, such as `hw-properties` that corresponds to the `HardwarePlatform` component in CC/PP.

There are, however, also differences, mainly due to the different goals of FIPA and W3C. For example, in CC/PP UAProf the ontology is divided into the following components at the highest level: Terminal Hardware, Terminal Software, and Terminal Browser. Of these only Terminal Hardware and Terminal Software were adopted by FIPA. Terminal Browser was left out because FIPA is not as focused to WWW as W3C is. On the other hand, in the FIPA specification the top level frame `device` has the parameter called "agent-compliancy" that is not found in the CC/PP specifications. The value of the agent-compliancy parameter tells whether the device in question is capable of hosting one or more FIPA agents or not.

Despite the differences between the approaches, the FIPA Device ontology can be used in a CC/PP profile. This can be accomplished in a similar fashion as with UAProf. For example, if the developer wants to express the fact that some device is FIPA-compliant, he can use a CC/PP profile in the following way:

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#"
     xmlns:fipa="http://www.fipa.org/profiles/device-20010202#">
     xmlns:uaprof="http://www.wapforum.org/UAPROF/ccppschema-19991014#">

  <Description about="http://www.foo.com/profiles/ProfileX">

    <ccpp:component>
```

```
      <Description about="http://www.foo.com/TerminalHardware">
        <type resource="http://www.foo.com/Schema#HardwarePlatform" />
        <ccpp:Defaults
           rdf:resource="http://www.foo.com/profiles/hwproperties" />
        <fipa:compliancy>true</fipa:compliancy>
      </Description>
    </ccpp:component>

    <ccpp:component>
      <Description about="http://www.foo.com/TerminalSoftware">
        <type resource="http://www.foo.com/Schema#SoftwarePlatform" />
        <ccpp:Defaults
           rdf:resource="http://www.foo.com/profiles/swproperties" />
        <fipa:ap-description>FIPA-OS v2.1.1</fipa:ap-description>
      </Description>
    </ccpp:component>

    <ccpp:component>
      <Description about="http://www.foo.com/Browser">
        <type resource="http://www.foo.com/Schema#BrowserUA" />
        <ccpp:Defaults
           rdf:resource="http://www.foo.com/profiles/browserproperties" />
        <uaprof:BrowserName>Internet Explorer</uaprof:BrowserName>
        <uaprof:BrowserVersion>5.0</uaprof:BrowserVersion>
      </Description>
    </ccpp:component>

  </Description>
</RDF>
```

Here the namespace `fipa` is used to tell that the device `ProfileX` described is FIPA-compliant and that the agent platform it has is `FIPA-OS v2.1.1`. The other CC/PP defined properties are (supposedly) found in the URIs declared by the `rdf:resource` attribute values of the `ccpp:Defaults` elements.

The `fipa` namespace declaration in the 4th row defines the URI that should contain a CC/PP schema:

```
http://www.fipa.org/profiles/device-20010202\#
```

The schema in this location corresponds to the ontology presented in the FIPA specification, but in CC/PP terms. More specifically, only those elements that are not found in the CC/PP schema itself are specified there. FIPA agent-compliancy is naturally an one of these.

7.3 Document Profiles

Annotations for Transcoding Web Content

In the W3C Note [5] and [4] an approach to annotating documents with metadata about their structure and semantics is proposed. These papers propose a framework of external annotations for HTML and XML documents,

and introduces a vocabulary for content adaptation. In the following, this approach will be shortly described.

The document profile of document specifies the expected capabilities of the browser in terms of, e.g., HTML support, style sheet support, and so on. During the process of matching the document with a rendering device, the document profile is compared with the device profile, the best fit between the two is determined, and a suitable document is generated or the best fitting variant is selected for viewing. This adaptation can be done at a content server, a proxy, or a client device. The original HTML document, authored for a specific client such as a PC, can be augmented with annotations, which provide hints for adapting the document to the other client devices. It is important to note that the result of applying an annotation to a target document depends on a particular transcoding policy and on knowing the particular needs of the target device.

An HTML document, which is provided for a desktop PC, is analyzed and annotated with a separate file by using an annotation tool. The annotated document must be viewable by an ordinary browser on a PC. Furthermore, such an annotated document can also be authored by using a stand-alone editor. Upon a request from a pervasive device, a proxy server may adapt the document on the basis of the attached annotations. The rendered document is then loaded down to the client device. In the process of document transcoding, it is also necessary to exploit user preferences and device capabilities for the content adaptation. Such information profiles can be described by using CC/PP.

The external annotation files contain hint information that is linked to the elements in the original document. RDF is used for external annotation files. In addition, XPath and XPointer is used for linking annotations with the annotated elements. An annotation file, which is an XML document, therefore contains a set of descriptions for annotating the subject HTML file. The way of adding a descriptive note is illustrated in the figure 7.1.

An annotation file refers to portions of a subject document. A reference may point to a single element (e.g., an IMG element), or a range of elements (e.g., an H2 element and the following paragraphs). XPath and XPointer allows for such addressing into the internal document structure.

When annotation files are stored in a repository, an appropriate annotation file for a subject HTML document is selected dynamically from the repository either implicitly by means of a structural analysis of the subject document, or explicitly by means of a reference contained in the subject document or some other association database

The following adaptation hints of rendering HTML/XML documents for pervasive computing can be used[12]:

[12]See http://www.w3c.org/TR/annot for details.

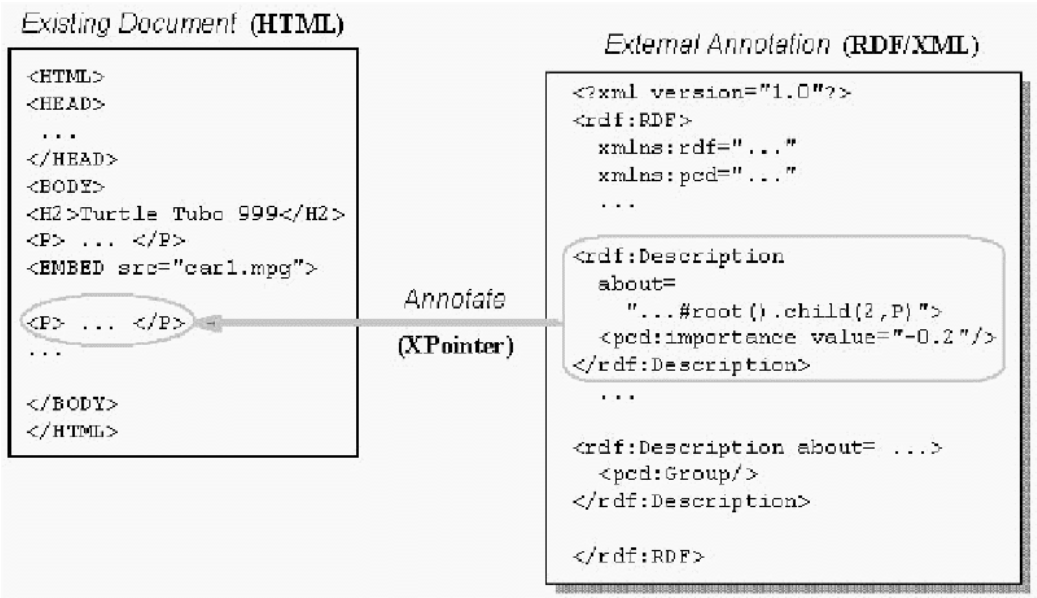


Figure 7.1: External Annotation of Web Content for Transcoding. The RDF description on the right side is attached to the paragraph (P) in the HTML page on the left side by using its URI as the "about" attribute value. Source: http://www.w3c.org/TR/Note.

- alternatives
- splitting hints, and
- selection criteria.

A document or any set of its elements can be provided with alternative representations. A transcoding proxy selects the alternative that best suits the capabilities of the requested client device. Elements in the subject document can then be altered either by replacement or on-demand conversion.

With splitting hints, an HTML/XML file, which can be shown as a single page on an ordinary desktop PC, may be divided into multiple pages in a client that has a smaller display screen.

The annotation may contain information that helps the transcoding proxy in selecting an optimal alternative representation for the client device. The annotation may indicate the client device capability expected for using the alternative resource, the resource requirements of the alternative, its fidelity to the original items, the semantic role of an element, and importance or priority.

An example of an external annotation file is given below. Here, the RDF `rdf:Description` tag specifies the importance `pcd:inportance` and

resource requirements `resourceRequirement` of an element in the resource file `catalog.html` together with a set of alternative representations.

```
<?xml version="1.0"?>

<rdf:RDF
   xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax"
   xmlns:pcd="http://www.ibm.com/annot/pcd"
   xmlns:prf="http://www.w3.org/TR/WD-profile-vocabulary">

   <rdf:Description
      about="http://foo.com/catalog.html#root().child(1,EMBED)">
      <pcd:importance value="0.8" />
      <pcd:resourceRequirement
         width="320" height="140" bpp="8" bw="1.4Mbps"
         color="color" media="video/mpeg" />

      <pcd:Alternatives>
         <rdf:Alt>
           <rdf:li>
              <pcd:Replace>
                 <pcd:fidelity value="0.8" />
                 <pcd:resourceRequirement
                       width="160" height="70" bpp="8" bw="28Kbps"
                       color="color" mime="video/h.263" />
                 <pcd:resourceToSubstitute>
                     <![CDATA[
                       <EMBED src="http://foo.com/car1.h263"
            width=170 height=80>
                     ]]>
                 </pcd:resourceToSubstitute>
              </pcd:Replace>
           </rdf:li>
           <rdf:li>
              <!-- a collection of an audio track and an image sequence -->
              <rdf:Bag>
                 <rdf:li> <!-- **First** item in the collection -->
                    <pcd:importance value="-0.4" />
                    <rdf:Alt>
                       <rdf:li>
                         <pcd:Replace>
                            <pcd:fidelity value="0.6" />
                            <pcd:resourceRequirement
                               bw="8Kbps" mime="audio/real" />
                           <pcd:resourceToSubstitute>
                              <![CDATA[
                                 <EMBED src="http://foo.com/car1_audio.ra"
                                        width=80 height=20>
                              ]]>
                           </pcd:resourceToSubstitute>
                         </pcd:Replace>
                       </rdf:li>
                    </rdf:Alt>
                 </rdf:li>
                <rdf:li> <!-- **Second** item in the collection -->
                  <pcd:importance value="0.6" />
                  <!-- in the sequence of two images,
                     each has a color and a b/w version -->
                  <rdf:Seq>
                    <rdf:li> <!-- **First** item in the sequence -->
                      <pcd:Replace>
                        <pcd:fidelity value="0.4" />
```

```
                    <pcd:resourceRequirement
                       width="320" height="240" bpp="8" size="18Kb"
                       color="color" mime="image/jpeg"/>
                    <pcd:resourceToSubstitute>
                          <![CDATA[
                             <IMG src="car1_1.jpeg" width=320 height=240>
                          ]]>
                    </pcd:resourceToSubstitute>
                  </pcd:Replace>
                </rdf:li>
                <rdf:li> <!-- **Second** item in the sequence -->
                  <pcd:Replace>
                  <pcd:fidelity value="0.4" />
                  <pcd:resourceRequirement
                          width="320" height="240" bpp="8" size="18Kb"
                          color="color" mime="image/jpeg" />
                  <pcd:resourceToSubstitute>
                          <![CDATA[
                             <IMG src="car1_2.jpeg" width=320 height=240>
                          ]]>
                   </pcd:resourceToSubstitute>
                   </pcd:Replace>
                 </rdf:li>
               </rdf:Seq>
             </rdf:li>
           </rdf:Bag>
         </rdf:li>
       </rdf:Alt>
     </pcd:Alternatives>
   </rdf:Description>
</rdf:RDF>
```

7.4 User Profiles: Personal Privacy Preferences

Personal information is often required when the user interacts with the various services on the WWW. Such information may be needed, for example, for customizing web content according to what the user desires. For example, the user may be especially interested in sports news concerning football and formula racing on a news server. To customize the service, the user must disclose personal preference information for the news server.

An important problem then is how to control the usage of one's personal data and how to give it out safely, i.e., how to protect one's personal privacy by personal preferences. W3C has prepared a standard called W3C Platform for Privacy Preferences (P3P)[13] for managing the user's privacy preferences. P3P is a format for a server to declare what information it will require from a user and how that information will be used.

In the following, we shortly overview this standard proposal as an example of using semantic web techniques for expressing metadata about user preferences.

[13] http://www.w3c.org/P3P

P3P contains a set of data elements that declare what information the server needs and what it will do with it. This dataset is written in XML and is extensible by using namespaces or by a built-in extension mechanism. The dataset defines a header to transport the information and messages back and forth between the client and the server. It does not define what the application should do internally. A mapping from this dataset to a dataset in RDF is planned to be released.

P3P specifies a vocabulary for expressing online privacy related to controlling the following aspects data usage:

- Who is collecting the data?
- What data is being collected?
- What is the purpose for collecting data?
- What data is being shared with others?
- Who are these data recipients?

In addition, P3P policies describe

- how the user can change the way in which her/his data is used,
- how disputes should be resolved,
- and the address of the site's human-readable privacy policy.

P3P declarations are positive: the sites state what they do. The vocabulary is designed to describe a site practice rather than indicate compliance with a particular law or code of conduct.

Security has two sides: the integrity of the information and the user's privacy. Encryption solves some of these problems but introduces a new one: it becomes impossible for the intermediaries to insert information into one's profile. One solution is to let information be transmitted in plain text, yet safeguard the user's privacy and information integrity. The standard P3P to be released has, however, dropped automatic data exchange protocol which was planned in the earlier drafts.

Company sites which support P3P include ATT[14], HP[15], Microsoft[16], Procter & Gamble[17], and Engage[18].

[14] http://www.att.com
[15] http://www.hp.com
[16] http://www.microsoft.com
[17] http://www.pg.com
[18] http://www.engage.com

7.5 Conclusions

In this paper, the W3C CC/PP proposal for characterizing the web content delivery context was presented. CC/PP is as framework that can be used as the basis for more specific vocabularies, such UAProf of WAPForum.

Today CC/PP and UAProf are not widely supported to convey delivery context. Existing user agents cannot rely on them being accepted by servers. Other, less flexible profiling methods include the use of the HTTP/1.1 request header, or the use of part of the URI, such as the URI query component. HTTP headers can convey a limited amount of delivery context information. URI extensions are also sometimes used for conveying some aspects of the delivery context.

External annotations can be used for profiling documents in order to transcode web content for different rendering devices. An RDF based technique was presented. Something similar could be done with XSL based techniques [1] XSLT and XSL-FO. However, it is harder to exploit semantic content information in these techniques than when using external semantic annotations.

P3P can be used to profile user preferences. A problem with P3P is that in order to be useful it should be widely used. Another problem is to know whether the service provider is reliable or not when requesting private user information. Regarding this matter, P3P is forced to rely on societal pressures and bodies. This might be a bigger problem in the United States because the legislation there doesn't regulate information collection as much as in Europe.

Bibliography

[1] N. Bradley. *The XML Companion*. Addison-Wesley, 2000.

[2] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[3] J. Hjelm. *Creating the Semantic Web with RDF*. John Wiley & Sons, New York, 2001.

[4] Masahiro Hori, Goh Kondoh, Kouichi Ono, Shin ichi Hirose, and Sandeep Singhal. Annotation-based web content, 1999. http://www9.org/w9cdrom/169/169.html.

[5] Masahiro Hori, Rakesh Mohan, Hiroshi Maruyama, and Sandeep Singhal. Annotation of web content for transcoding, July 10 1999. http://www.w3c.org/TR/annot.

[6] O. Lassila and R. R. Swick (editors). Resource description framework (rdf): Model and syntax specification. Technical report, W3C, 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[7] Ganesh Sivaraman. Composite capability/preference profile — an RDF based profiler for mobile devices, 2001. http://www.tml.hut.fi/Opinnot/Tik-11.590/2000/Papers/Rdf.html.

Chapter 8

eBusiness Standards for Web Services

Jyrki Haajanen, Petri Takala, and Eero Hyvönen

A major goal of the WWW is to create automated services. If machines are to help us on the web, then they must be provided with means on "understanding" the contents they are dealing with and the service processes they are conducting. This means that the idea of machine-processible semantics — as promoted by Semantic Web research — will be of central importance when developing web services. In this paper, we review some of the most prominent standards being developed for web services in eBusiness.

8.1 Introduction

Since the mid 1990's, the success of the World Wide Web (WWW) has lead to a movement from point-to-point inter-organizational communication towards many-to-many Internet-based solutions. This development has been boosted further by the rise of new formats in data representation, most notably the eXtensible Markup Language (XML) [3]. XML is a metalanguage that can be used to define new data formats for a wide range of applications. The same data in a single self-descriptive form is both human and machine readable.

We now have means to present the data in an expressive way but need methods and tools for using the data in practice by algorithmic means. From the eBusiness viewpoint, several problem areas are encountered, such as:

- *Service discovery.* The first problem is how to find the needed the data or service in the first place on the WWW.
- *Service description.* After finding the service the agent should be able

to know how to find out how to use the service provided. This means that some kind of explicit description of how services are used should be provided for the client.

- *Service communication.* Vocabulary and protocols for communicating complicated service requests and responses between the client and the service provider are needed.

The central goal of the Semantic Web [5, 2] is provide the web with machine-processible semantics. Semantic Web technologies should be useful when creating techniques for discovering data and services on the Internet. This can be accomplished by adding searchable descriptions, metadata [1], of the data and services on the web. The metadata is typically based on some commonly agreed conceptual model, an ontology [6], that is used for modeling the data and service content within the application area in focus. Semantic Web technologies are also useful in describing explicitly the content of services and how to use them according to given specifications. Finally, structured documents central in Semantic Web research also provide a conceptually higher level basis for communication than the traditional Hypertext Transfer Protocol HTTP.

In this paper, eBusiness standards for web services are discussed. The presentation is organized as follows. First, dimensions for categorizing eBusiness standardization work are identified. We then review some standard classification schemes and vocabularies for describing products and services, the targets of eBusiness activities. After this standards for exchanging structured messages on the web and explicit descriptions of service processes are discussed. Many of these standards are used as a part in the large generic eBusiness frameworks, such as XML/edi, ebXML, UDDI, and RosettaNet, to be reviewed next. In conclusion, major frameworks developed by individual major companies, ONE by Sun and .NET by Microsoft, are shortly discussed and the relationship of the "Web Service" -concept with "Semantic Web" analysed.

8.2 Dimensions of Standardization

The research and standardization work in the area of web services are advancing rapidly. There are already hundreds of standardization groups developing various kinds of vocabularies, languages, and frameworks for different aspects of electronic business. This work can be classified along various dimensions.

First, from the organizational view point there are three different development paths.

- *Academic research.* Lots of research is focusing on creating formal tools and methods for defining the underlying techniques for knowledge

representation [12], languages for ontologies [14], tools for creating, combining, and maintaining ontologies, and for defining ontologies and their interconnections [7, 6, 9].

- *Standardization consortia.* Various joint organizations of different companies and research institutions aim to develop universal or industrial standards for defining and presenting information of company business models and interaction capabilities.

- *Major software vendors.* Various software vendors provide solutions for web services. Most important actors in this scene include Microsoft with it's .NET framework[1], providing proprietary solutions for web services, and Sun Microsystems with the ONE framework[2] aiming at open standards.

Second, standardization work has horizontal and vertical dimensions:

- *Horizontal.* standards focus on generic business data and protocol.

- *Vertical.* standards focus on business data and protocols related to practical application fields, such as banking, electrical engineering etc.

Third, from the content type perspective, different focuses in the standards can be identified [8]:

- *Artifact-centric* standards focus on describing products, information, money, securities and similar items involved in business transactions.

- *Process and activity -centric* standards focus on modeling business processes.

- *Agent-centric* standards focus on people and organizations.

- *Other* kind of standards also exists, e.g., those describing generic categories such as time or location.

Yet another dimension to consider is the semantic depth of the standard [8]:

- *Vocabularies* are basically lists of agreed concept names and their definitions.

- *Taxonomies* add hierarchical specialization relation to vocabularies.

[1] http://www.microsoft.com/net
[2] http://www.sun.com/sunone

- *Thesauri* describe various additional conceptual relations, such as hyperonimy/hyponimy, synonimy, and partonimy.
- *Reference models* add still additional more complex relations in the definitions.

Finally, standardization work can be categorized according to the organization within which the work is carried out. In below a few major standardization bodies are listed.

W3C The World Wide Web Consortium[3] (W3C) is the public international consortium developing generic standards for the World Wide Web.

ISO The International Standards Organization[4] (ISO) develops global standards is all areas of standardization except electrical and electronic engineering. ISO is a federation of some 140 national standardization organizations.

OASIS The Organization for the Advancement of Structured Information Standards[5] (OASIS) is a "non-profit, international consortium that creates interoperable industry specifications based on public standards such as XML and SGML, as well as others that are related to structured information processing". OASIS is a major source for application specific XML standards.

IEC The International Engineering Consortium[6] (IEC) is a nonprofit organization dedicated to catalyzing positive change in the information industry and its university communities. IEC develops global standards in the fields of electrical and electronic engineering

CEN The European Committee for Standardization[7] (CEN) is a joint organization of European national standard bodies and develops standards for Europe. It has a special CEN/ISSS Electronic Commerce Workshop.

UN/CEFACT The United Nations Centre for Trade Facilitation and Electronic Business[8](UN/CEFACT) has both permanent and ad hoc working groups on content standards. For example, there is a working group developing the ebXML standard and another one for electronic commerce.

[3]http://www.w3.org
[4]http://www.iso.org
[5]http://www.oasis-open.org
[6]http://www.w3.org
[7]http://www.cenorm.be
[8]http://www.unece.org/cefact/

NIST The National Institute of Standards and Technology[9] (NIST) "is a non-regulatory federal agency within the U.S. Commerce Department's Technology Administration. NIST's mission is to develop and promote measurements, standards, and technology to enhance productivity, facilitate trade, and improve the quality of life".

A number of consortia have been established to develop and propose generic framework standards for eBusiness. Such proposals may include:

- Product and service description standards.
- Communications and messaging standards.
- Service description standards. These standards are needed for specifying what services are available and how they are used.
- Enterprise and business description standards for describing and discovering potential business partners.

In the following we review some prominent eBusiness standardization work along these categories. Enterprise and business description standards are considered only from the business discovery viewpoint as described in some generic eBusiness standardization frameworks, especially UDDI.

8.3 Product and Service Description Standards

The basis for eBusiness applications is a vocabulary or ontology by which the commodities provided to clients can be identified. In the following some horizontal classification systems covering different business areas are briefly introduced.

UN/SPSC The United Nations Standard Products and Services Code[10] (UN/SPSC) is a coding system that classifies products and services. USSPSC was created in 1998 through the merger of the United Nations Common Coding System (UNCCS) and Dun & Bradstreet standard Products and Services Classification. UN/SPSC is considered as open standard and is available free of charge. UN/SPSC coding system is organized as a five level hierarchy. The levels are:

- SEGMENT The logical aggregation of families for analytical purpose.

[9] http://www.nist.gov
[10] http://eccma.org/UN/SPSC/

- FAMILY A commonly recognized group of inter-related commodity categories.
- CLASS A group of commodities sharing a common use or function.
- COMMODITY A group of substitutable products or services.
- BUSINESS TYPE The function performed in support of the commodity. This value is vendor specific and is seldom used.

There are over 11,000 UN/SPSC codes, which are claimed to cover any product or service that can be bought or sold.

SCTG The Standard Classification of Transported Goods (SCTG) has been created jointly by agencies of the United States and Canadian governments to address statistical needs in regard to products transported. The SCTG is a five-digit code and its structure is hierarchical.

EAN The European Article Numbering (EAN) scheme was created in 1974 as a result of a council of manufacturers and distributors of 12 European countries . The actual EAN association was formed 1977. EAN International has 96 Member Organizations representing 98 countries. EAN's original goal was to develop uniform numbering system for identifying goods. Today EAN codes are used by 850,000 member companies worldwide.

Ecl@ss Ecl@ss[11] is a German standardized material and service classification system. It includes some 12,000 key words organized hierarchically. In addition to the hierarchy, Ecl@ss integrates attribute lists for the description of material and service specifications.

In addition to general horizontal classification systems, vertical dictionaries and ontologies are being developed for narrower business areas. For example, the RosettaNet consortium to be discussed later develops a dictionary focused on semiconductor, electrical, and IT industries.

8.4 Messaging Standards

The Hypertext Transfer Protocol (HTTP) (on top of the TCP/IP protocol) has been the key standard on the WWW for data exchange between clients and servers. The GET and POST methods are used for delivering the input data as a set of attribute value pairs from the client to the server. Named attribute values are not a flexible enough data model when representing more complicated message contents needed in intelligent web services. Since the

[11] http://www.eclass.de

data on the web will be more and more represented in XML languages, a natural choice is to start using XML-based messages in communications.

The Simple Object Access Protocol[12] (SOAP) is an open standard that supports the interoperability of autonomous systems. Since it can be run over HTTP, it helps to resolve the communication problems that occur when the applications have to operate through the fire walls of different organizations. Furthermore, SOAP supports remote procedure calls (RPC) for service invocation.

SOAP messages can contain data packed in XML format, so they provide an ideal platform for implementing a tailored content messaging service. Due to the XML approach, the SOAP messages are easy to understand when compared to their binary correspondents, such as CORBA, DCOM, and RMI. The character-based nature of the messages introduces the requirement for encoding and decoding messages at the ends of communication channel. However, this is a relatively light task when compared to the cost of the transmission itself.

SOAP is becoming popular as the communication layer for various XML based B2B Frameworks such as ebXML, UDDI, and BizTalk.

8.5 Service Description Standards

In this section we review two approaches for describing web services: WSDL and DAML-S. WSDL represents standardization work with immediate practical goals. This standard together with SOAP and UDDI (to be described later) complement each other well and can be used in as larger framework for creating web services. DAML-S is a more ambitious and academic approach for describing complicated services.

WSDL

The Web Services Description Language[13] (WSDL) is an XML based format for describing the details of a web service in a structured way. WSDL is based on XML-based message contents and SOAP protocol that wraps the messages and provides a reliable transportation layer.

WSDL is based on the following major elements that can be used to define a web service in full coverage [4]:

- Types, that provide the data type definitions to describe the messages that will be exchanged.

[12] http://www.w3.org/TR/SOAP/
[13] http://www.w3.org/TR/wsdl

- Message, an abstract definition of the data that is transmitted. It consists of logical parts that are associated with a definition within some type system. This element states what services are available in relation to the *portType* element (cf. below).

- A port type is a set of abstract operations, each of which referring to an input and output message. This element states what services are available in relation to the *message* element (cf. above).

- Binding, which specifies the concrete protocol and data format specifications for the operations and messages defined by a particular port type. This element states how the operations can be invoked.

- Port, which defines a single communication end point by specifying an address for binding.

- Service, which aggregates a set of related ports to a single service. This element states where the service can be accessed.

It is recommended to give the service definitions in separate files specifying the data types, the abstract service definitions, and the concrete binding of the services. This enables more efficient reuse of the existing services and also helps in managing the definitions. Figure 8.1 shows an example of this convention. In the figure, the data types TradePriceRequest and TradePrice are defined in the `stockquote.xsd` XML Schema file. These definitions are then used in the abstract service definition at the `stockquote.wsdl` WSDL file for defining the input and output parameters for the operation GetLastTradePrice. Finally this service is bound to a concrete implementation residing at http://example.com/GetLastTradePrice and given the service name StockQuoteService in the file `stockquoteservice.wsdl`. The arrows present how the files are used by each other. File names and usage are presented in bold font. In principle, WSDL provides means for platform independence through it's additional abstraction level at the message and transport protocol. However, this benefit is in practice partly hampered by the developer specific features in concrete SOAP and WSDL implementations. The situation will probably improve over time.

The process of using WSDL for presenting web services can use alternative paths depending on the situation where the technology is used. When creating a web service from the scratch, one should use WSDL in describing the data to be interchanged, the abstract service model ,and then the binding to the concrete implementation. If one is building on top of a legacy service, then it is possible to produce the WSDL description based on the legacy implementation. In both cases we can use WSDL to produce the stubs for the client side automatically.

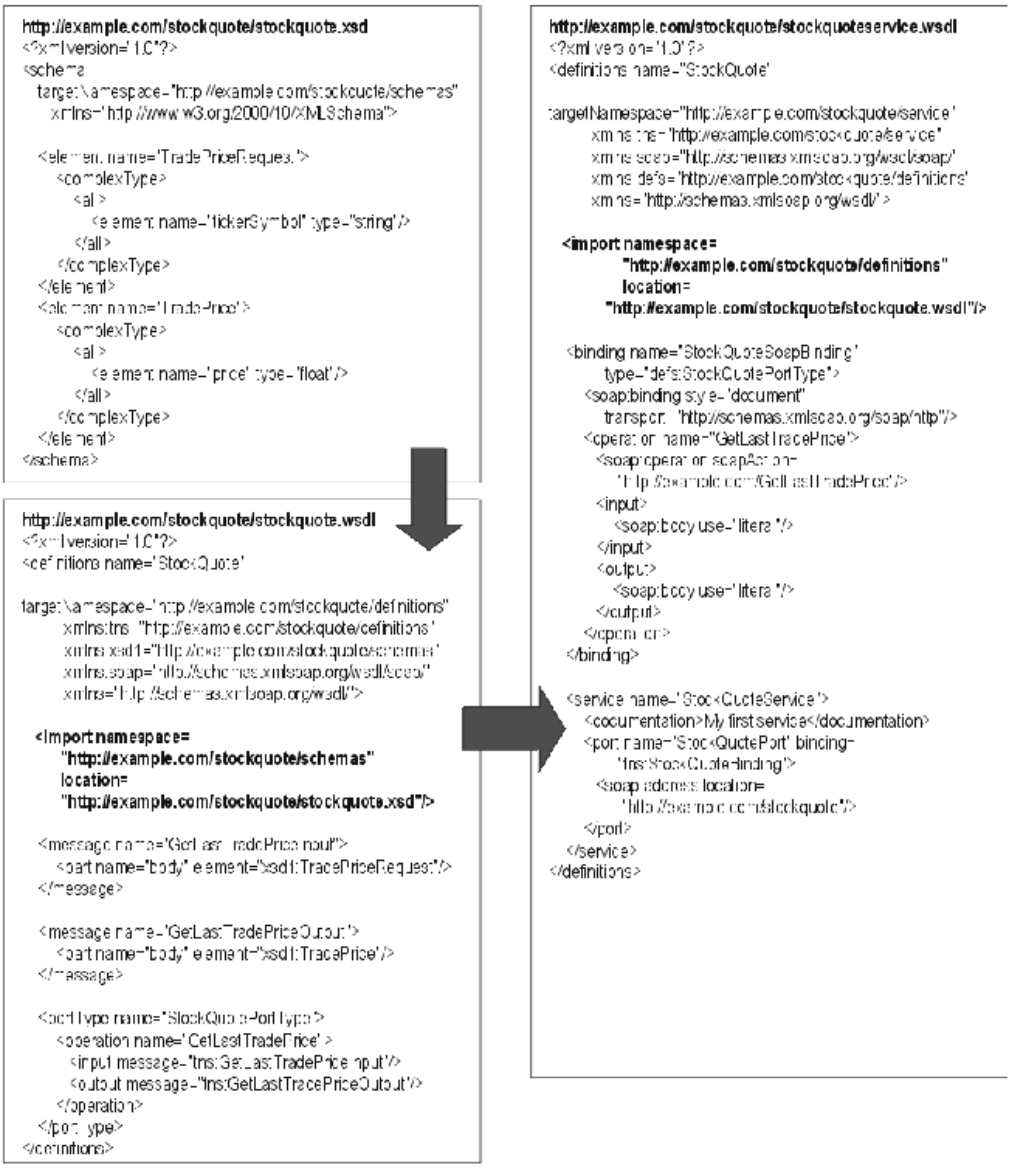


Figure 8.1: An example of using WSDL service.

WSDL's approach to web service definition has a declarative character. The providers describe their service and the intended means of use for the users. This differs quite a lot from, e.g., the negotiation approach used in ebXML. WSDL is a good solution for companies providing services for a large set of customers in a market situation resembling monopoly. It is also a good starting point for shifting to an open environment by bringing the legacy applications available on the web.

DAML-S

DARPA Agent Markup Language[14] (DAML) is a program aimed to develop languages and tools for the Semantic Web. As a part of the program, a web service description ontology called DAML Services or DAML-S is being developed. DAML-S provides a core set of language constructs for describing Web Service capabilities and properties in an automatically interpretable form. The language features include automated web service discovery, execution, interoperation, composition, and execution monitoring. All these aspects are relevant objectives for generic web service initiatives, such as UDDI.

DAML-S is based on two main modeling concepts: Service Profile and Service Process modeling. Service Profile specifies the following aspects: what service is provided, what preconditions must be met, and what are the results (postconditions) of the service, including also possible exceptions. Furthermore, the Profile also contains a human readable part that can be used in informing of the service on the web. Possible limitations of access are also described, such as geographic scope, language, etc. The Process Modeling is divided into two parts:

- *Process Model* allows two modes for processes: atomic and decomposable. Composite processes may include subprocesses that are executed conditionally. This supports composition of services from other services.
- *Process Control Model* is currently included only in the designs and will be included in the future versions of the language.

8.6 Generic Frameworks

8.6.1 XML/edi

The Electronic Data Interchange (EDI) messaging mechanism has gained some success as an eBusiness standard among relatively large companies.

[14] http://www.daml.org/services/

However, it has suffered from the need for bilateral customization between the partners of each business transaction and from the need for dedicated middle-ware arrangements.

The success of the Web and the raise of the new XML-based technologies have resulted to new tools for arranging eBusiness transactions and models with lower costs. XML/edi is an initiative that builds on the lessons learned from EDI and creates a new XML-based eBusiness communication platform.

The XML/edi is developed by XML/edi group that was found in 1997. The group includes a number of companies and governmental organizations from all over the world. In addition to the traditional EDI-like transaction processing, the XML/edi initiative approach provides means for automatically synchronizing the business rules and process control templates of the business partners.

The initiative is based on the analysis of the failure of the traditional EDI to create a broad acceptance in the eBusiness scene. The analysis resulted into the following "Business Top Ten" requirements list [15] for the standard:

1. Reduce the cost of doing business.
2. Reduce cost of entry into eBusiness.
3. Provide an easy to use tool set.
4. Improve data integrity and accessibility.
5. Provide appropriate security and control.
6. Provide extendable and controllable technology.
7. Integrate with today's systems.
8. Utilizes open standards.
9. Provide a successor to X12/EDIFACT and interoperability for XML syntaxes.
10. Globally deployable and maintainable.

The EDI approach is based on fixed message structures that the companies can exchange during the business process. The basic EDI framework contains a set of standard messages. In addition to this basic set, additional message structures have been defined for different industries. Typically, some standardization has been done, but at the application level the inflexibility of the standards has resulted into circumventing them by adding one's own data fields or by omitting some fields based on the particular business transaction being developed. [15]

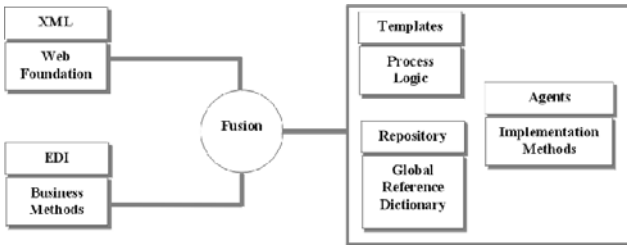


Figure 8.2: The Power of Five — the Components of XML/edi [15].

Due to the inflexibility of the EDI model, the developers of the XML/edi framework have added three features to their model in addition to the message structure. The new elements are

- Process Templates,
- Software Agents, and
- Global Entity Repositories.

Figure 8.2 depicts the XML/edi main elements and shows their relations. In parallel to these new elements, the use of XML in the message structures is introduced. This makes the messages more flexible and allows customization through extension. The Process Templates describe the data handling that occurs in the business process while a transaction is performed. They are presented in XML and are based on traditional process control language syntax. The Software Agents use the Process Templates and the transaction data definitions, and interact with the user applications to formulate new templates for each specific task. They can also use the Global Entity Repository to locate and deploy existing templates for already defined tasks. The Global Entity Repositories collect the meaning, contents, and semantics of various EDI elements, and provide their automatic search services for the Software Agents.

XML/edi is a vision for eBusiness after EDI. The guidelines it states are at least partially implemented in a number of eBusiness frameworks. It's most prominent successors are the ebXML and UDDI frameworks that will be described in more detail below.

8.6.2 ebXML

The ebXML framework[15] [10] is an implementation of XML/edi vision supported by UN/CEFACT, OASIS, various industry consortia, and numerous

[15]http://www.ebxml.org

significant hi-tech companies. Its architectural model is open and it aims to maximize reuse: the layers in the ebXML can be replaced with other corresponding implementations. Furthermore, the model can be partially implemented to satisfy the requirements of a specific system with minimum effort. The developers of the ebXML wish to see their model as a "plug-in" environment where other eBusiness frameworks can be attached to. The philosophical approach is to provide a horizontal service layer where industry specific vertical implementations can be integrated.

The framework follows a conversational model of business relationship creation — the partner candidates negotiate the details of interaction arrangements and when a compromise is formed, they start doing business together.

The ebXML initiative builds around a set of technical specifications elaborated by dedicated task forces. These specifications are:

1. Technical Architecture
2. Specification
3. Registry Information Model
4. Registry Services Specification
5. Requirements Specification
6. Collaboration Protocol Profile and Agreement Specification
7. Message Service Specification

Each specification responds to an identified requirement on eBusiness. Figure 8.3 shows the Business-to-Business collaboration process and its relation to the items defined in the ebXML specifications. The collaboration process is seen as a cyclical process having the following phases:

Process Definition At this phase the company that wishes to bring its services to eBusiness markets models its business process and associated documents and records them in a standard format to the Registry/Repository. The modeling method is not fixed although the Unified Modeling Methodology (UMM) of UN/CEFACT is recommended by ebXML. However, generally Unified Modeling Language (UML) [16] by Rational is used in ebXML.

Partner Discovery This phase consists of search actions for a potential business partner based on the profile descriptions deployed to Registry/Repository by them.

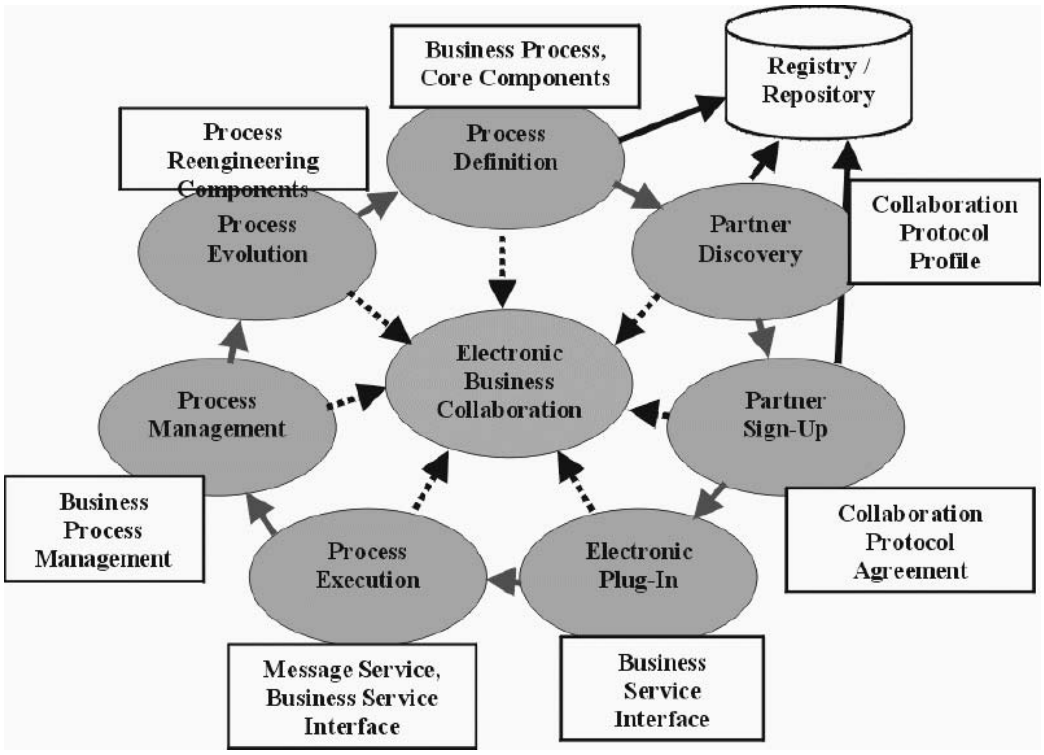


Figure 8.3: B2B Collaboration Process and ebXML Specifications [16].

Partner Sign-Up This phase is activated activated after a suitable partner is found. The business and technical details are negotiated and after making an agreement the partners sign up for a specific role in the business model.

Electronic Plug-In This phase consists of preparing the business application for co-operation.

Process Execution At this phase the interaction between the business partners begin using the environment set up in the previous stage.

Process Management This phase is active during the season while the co-operation agreement is valid and the process execution as agreed has to be ensured.

Process Evolution This stage occurs after the season for the agreed business collaboration has ended. At this stage the processes may be refined if a new contract is made.

The UMM model used for the Process Definition phase consists of four stages:

1. *Business Domain Modeling*, where the business domain is modeled to gain a generally accepted understanding of the domain and some high-level requirements.
2. *eBusiness Requirements*, where the work is further elaborated to formulate more detailed requirements for the eBusiness.
3. *Analysis*, where the requirements are transformed to implementation level specification.
4. *Design*, where the specification is used to construct the business scenarios and collaboration model specifications based on design patterns.

The ebXML architecture uses the Open-edi Reference model approach to distinguish between the operational and functional views of the business transactions [16]:

- Business Operational View (BOV) collects the business practices consisting of the data semantics of the business transactions along with the conventions and agreements related to them.
- The Functional Service View (FOV) states the technical details of the required IT infrastructure for these interactions, such as service capabilities, interfaces, protocols, and messaging services.

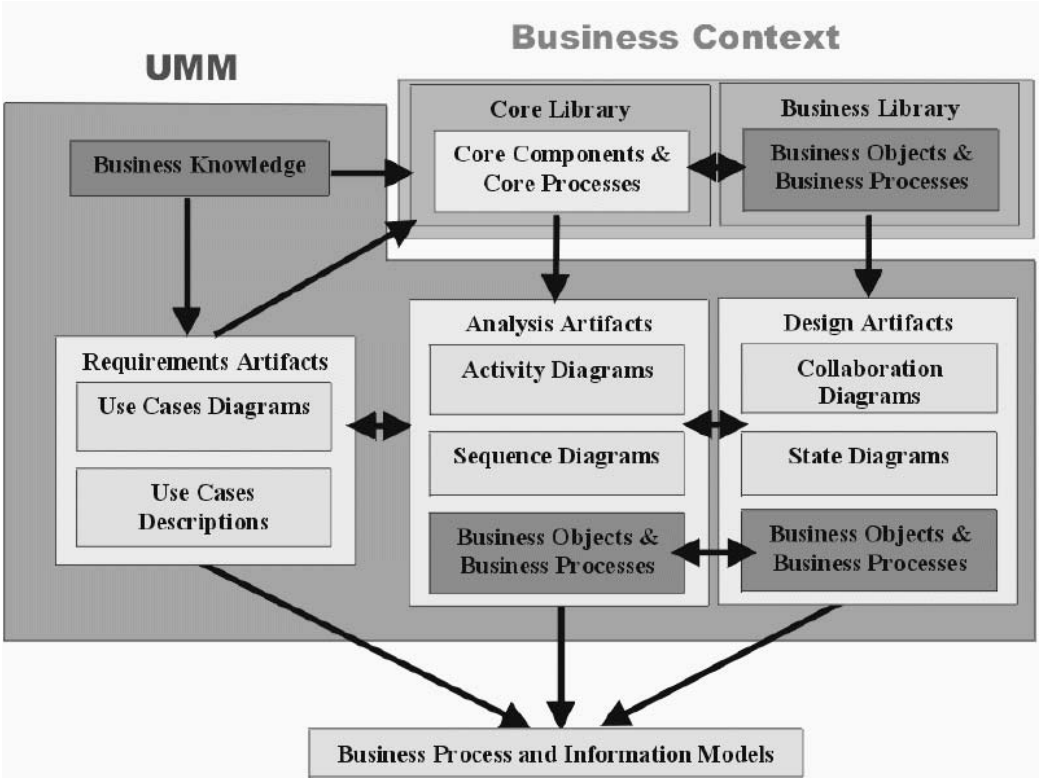


Figure 8.4: The ebXML Business Operational View [16].

The Business Operational View of ebXML is illustrated in the figure 8.4 [16]. The BOV is motivated by the fact that although the actual business routines applied by the business companies vary a lot, they still can be modeled using a meta level business process model that are not company dependent. These may in turn be generalized to reusable Business Process and Information models that can be used as templates in new business agreements.

The BOV definition takes advantage of the existing business knowledge that is presented by the Business Context area. In the beginning of the design process, this knowledge is used in addition to the information that is collected from the particular business case. These pieces of knowledge are presented by the Business Knowledge box in the figure 8.4. The next stage is to define the business requirements based on the Business Knowledge; these are expressed as the Use Cases shown in the Requirements Artifacts box in the figure. These requirements are then analyzed and activity, sequence, and conceptual class diagrams are formed as shown in the Analysis Artifacts box in the figure. Finally, at the design stage the acquired diagrams are further elaborated to collaboration, state, and final class diagrams, which are presented at the Design Artifacts box in the figure. Note that the resulting artifacts from each stage are stored to the Business Process and Information Models, and that they are fully implementation independent.

The ebXML Functional Services View (often referred as the ebXML Technical Architecture) binds the models acquired from the BOV design to the actual implementation. The BOV results and the Business Process and Information Models are the input for the Functional Service View; they are converted to XML format presentation and stored in the FSV registries (see the figure 8.5). The registries have a central role in ebXML because they enable the interaction of the companies in finding partners and forming business transaction processes. The registries are used in modeling the company's own business process for acquiring the existing business and transaction models. These are used as a basis for the creation of the company's Collaboration Protocol Profile (CPP). The information acquired from the registries is also used when the interface to company's internal business applications using CPP's called Business Service Interface is defined.

8.6.3 UDDI

UDDI[16], Universal Description, Discovery and Integration, is a project whose goal is to create a framework for describing services, discovering businesses, and obtaining information on how to connect with the web services of a business. UDDI information is replicated over several sites called Operator Services. Anyone can access an Operator Service free of charge and search for the information a business has made available about it. The information

[16]http://www.uddi.org

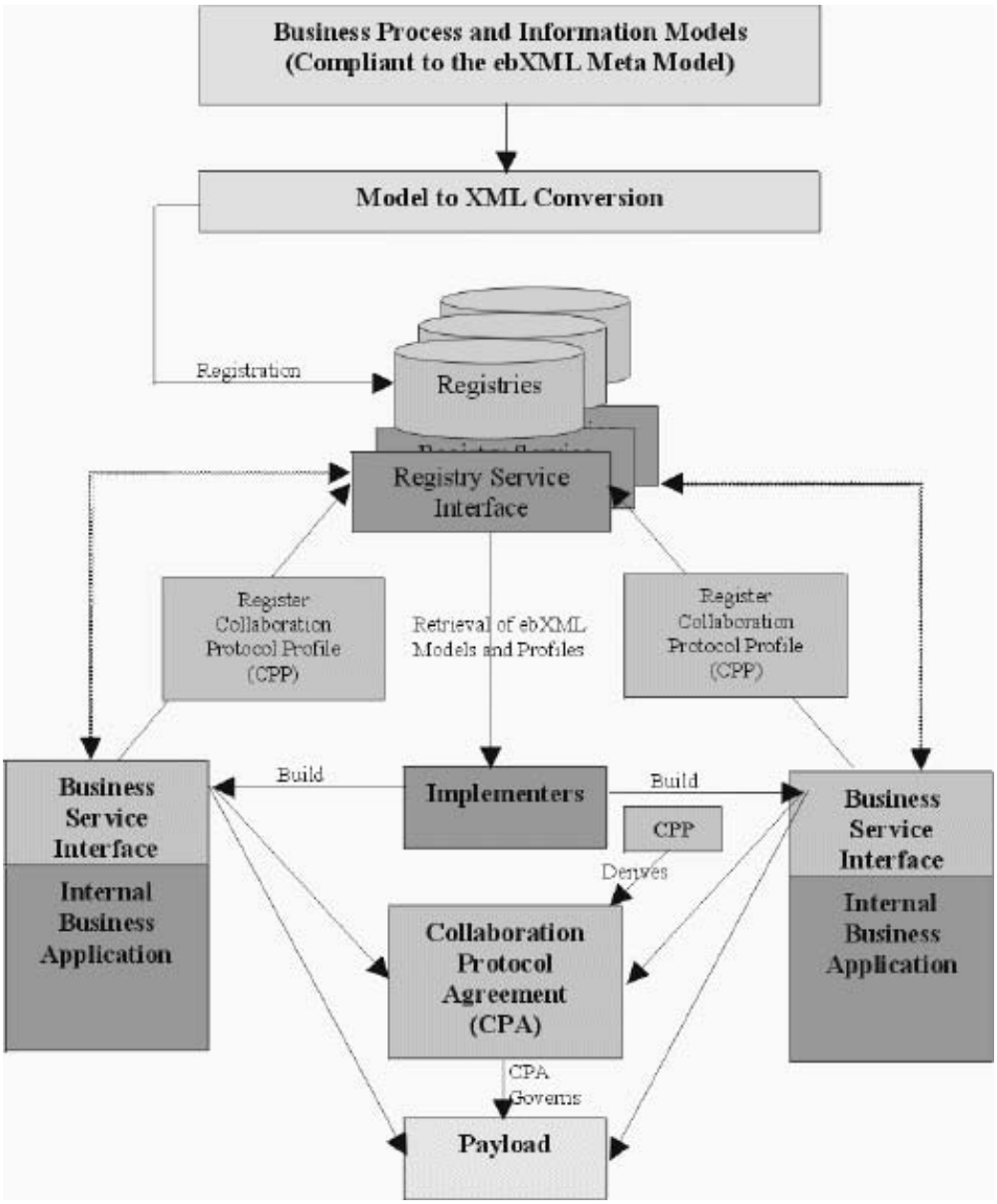


Figure 8.5: The ebXML Functional Service View [13].

that a business can register answers to simple questions such as who they are, what they do, where they are located (URL), and how to connect and interact with their web services.

The core information model used by the UDDI registries is defined in an XML schema. This schema defines four core types of information:

1. Business information
2. Service information
3. Binding information
4. Information about specifications for services

A structure called "businessEntity" contains the information for the UDDI business registration. BusinessEntity has substructures "businessService" and "bindingTemplate". The businessService structure is used to group a series of web services which are usually related to a business process or a category of services. One or more technical web service descriptions, bindingTemplates, exists within each businessService. These contain the information on how to connect to and communicate with a Web service.

BindingTemplates also include references for "tModels". A tModel is an abstract metadata definition which offers technical information about the interfaces and other aspects which have to be taken care of when connecting with the service.

UDDI describes the information provided in a business registration with three components (see figure 8.7):

White pages consists of company contact information.

Yellow pages uses standard taxonomies to categorize the business. Standard taxonomies used in yellow pages includes UN/SPSC, DUNS, ISO 3166, and NAICS.

Green pages is used for documenting the technical information about the services.

8.6.4 RosettaNet

RosettaNet[17] is a consortium of several companies acting the fields of Information Technology, Electronic Components, and Semiconductor Manufacturing. The goal of the consortium is to create open ebusiness standards for these relatively narrow vertical fields of business. RosettaNet is already used by many major companies such as Intel and Motorola.

RosettaNet framework consists of the

[17]http://www.rosettanet.org

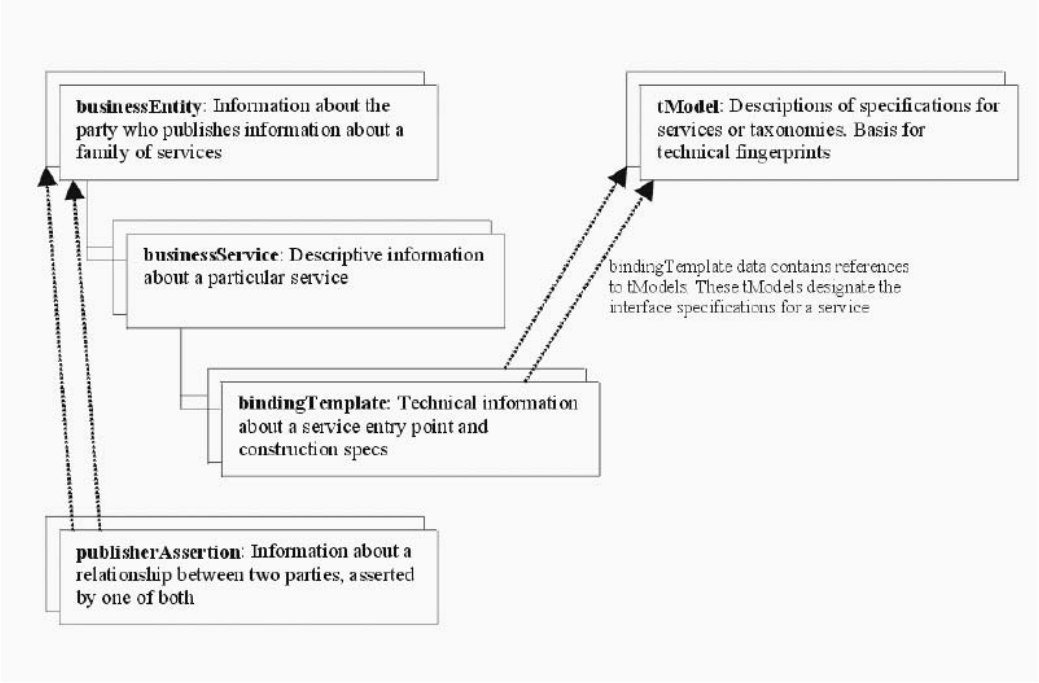


Figure 8.6: UDDI structures.

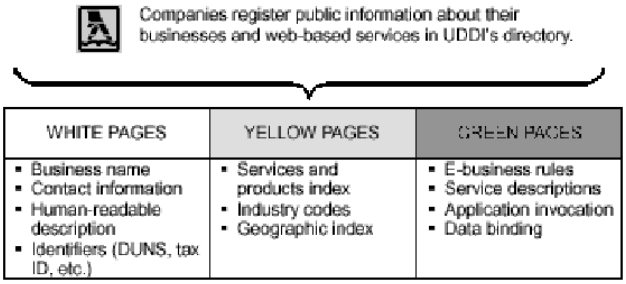


Figure 8.7: White, yellow, and green pages of UDDI.

- RosettaNet Business Dictionary,
- RosettaNet Technical Dictionary,
- RosettaNet Partner Interface Processes (PIP), and
- RosettaNet Implementation Framework (RNIF).

RosettaNet Dictionaries

RosettaNet dictionaries define the terms and their semantics used in PIP's. RosettaNet Business Dictionaries define common business terms while RosettaNet Technical Dictionaries define technical terms and their explanations. Currently there are two technical dictionaries, Electronic component technical dictionary and Information technology technical dictionary. The business dictionaries includes elements for the business actions messages used in PIPs while technical dictionaries are used to define products. In addition, RosettaNet uses DUNS numbering [18] to identify trading partners. For product specification GTIN product numbering [19] and UN/SPSC product classification [20] are used to supplement RosettaNet dictionaries.

Partner Interface Process

RosettaNet partners exchange information with RosettaNet Partner Interface Processes (PIP). PIP specification includes three parts: Business Operational View (BOV), Functional Service View (FSV) and Implementation Framewrok View (IFV). The BOV describes the roles of two business partners and the required interaction between them. These interactions translates into Business Actions and Signals that are decsribed in FSV. Finally IFV specifies the message formats and communication requirements as supported in RosettaNet Implementation Framework.

PIP's are divided to seven clusters by their business function, and these clusters are divided further to segments. Each segment comprises several PIPs. Each PIP contain at least one Activity, and Activities specify Actions. In addition, one cluster is reserved for RosettaNet administrative functions.

After selecting the PIP that will be used, trading partners select a communication model. For that purpose RosettaNet also includes an architecture for exchanging messages. This is called RosettaNet Implementation Framework.

[18] http://www.dnb.com/english/duns/default.asp
[19] http://www.ean-int.org/index800.html
[20] http://eccma.org/UN/SPSC/

RosettaNet Implementation Framework

RosettaNet Implementation Framework (RNIF) defines guidelines for exchanging messages based on PIP's. RNIF begin with business model that defines how companies interact with PIPs. The business model consists of five parts [10]:

1. Creation of PIP guidelines.
2. Distribution of these guidelines.
3. Validation of the exchanged message content
4. Extension of guidelines for special trading partner implementations. These cannot, however, override original RosettaNet specifications.
5. Exchange of extended guidelines for validations of these special messages.

The RosettaNet technical architecture follows the seven-layer ISO reference model with the exception that RosettaNet does not use presentation layer. OSI's session layer is matched by RosettaNet's security layer. RosettaNet business messages consist of a header and message body with content.

Multipart S/MIME is used for messaging. The message body and header are coded in XML.

8.7 Private Web Service Frameworks

In addition to standardization consortia, also major enterprises are developing frameworks for web services, both open and proprietary ones.

The best known initiative in the open standards' camp is probably the Sun Open Network Environment[21] (Sun ONE). It capsulates the Sun Server family into a single framework for implementing data, application, reporting, and transaction (DART) features for an enterprise. The servers include a Portal Server, a Directory Server, an Application and Integration Server, a Web Server, and Messaging and Calendar Servers. [11]

The Portal Server provides personalized access to the enterprise information systems with various client devices ranging from mobile devices, such as phones and PDAs, to portable and desktop computers. The Directory Server manages access to services and handles authentication of users, and balances the work load among the servers. The Application and Integration Servers operate on J2EE basis and form a bridge between the external applications and the business applications of the enterprise. The Web Server supports

[21] http://www.sun.com/sunone/

the reporting in intra-, extra-, and Internet. The person-to-person messaging required in daily business is supported with the Messaging and Calendar Servers. The fusion of the aforementioned servers is based on extensive use of XML as the data format and the J2EE as the means of capsulation for the functionality.

Microsoft .NET is the proprietary web services platform of Microsoft. The .NET Framework has two main components: the common language runtime and the .NET Framework class library. .Net runtime manages code at execution time while .Net class library offers object-oriented collection of reusable types. XML is used in communications within the framework.

8.8 Conclusions

The current WWW is basically used for publishing and delivering web pages and static documents from one place to another. It lacks processes and dynamicity and it's contents are difficult to use for the machines. The services are dynamic processes being performed by the computers and software agents. Web service standards, such as UDDI, WSDL, and SOAP hold the promise of making the web more "intelligent" and alive. According to the web service vision, the user will find what he needs through global service registries and publish his/her own services there for the others to use. Furthermore, the web and it's services can not only be used by humans but by machines as well.

In this paper, standardization work in the field of web services has been briefly reviewed. This survey is far from complete; there are hundreds of standardization projects going on in this economically important area. Current standardization work in web services focus on generic, conceptually low level standards and frameworks that can be used to model basically any business processes. However, in many web services, domain specific data, knowledge, and processes are needed in addition to general basic business transactions. It is likely that Semantic Web technologies will be ever more crucial in creating this next generation of web services, where more and more business domain specific content is embedded in ever more intelligent services.

Bibliography

[1] Kal Ahmed, Danny Ayers, Mark Birbeck, Jay Cousins, David Dodds, Joshua Lubell, Miloslav Nic Daniel Rivers-Moore, Andrew Watt, Robert Worden, and Ann Wrightson. *Professional XML Meta Data*. Wrox Press Inc., 2001.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[3] N. Bradley. *The XML Companion*. Addison-Wesley, 2000.

[4] Christensen, Curbera, Meredith, and Weerawarana. Web services description language (wsdl) 1.1, 2001. http://msdn.microsoft.com/xml/general/wsdl.asp.

[5] D. Fensel (ed.). The semantic web and its languages. *IEEE Intelligence Systems*, Nov/Dec 2000.

[6] D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce*. Springer-Verlag, 2001.

[7] W. Grosso, H. Eriksson, R. Ferguson, J. Gennari, S. Tu, and M. Musen. Knowledge modelling at the millenium (the design and evolution of protege-2000. In *Proceedings of 12th Workshop of on Knowledge Acquisition, Modeling and Management (KAW-1999), Banff, Alberta, Canada*, 1999.

[8] N. Guarino and M. Stefanova (eds.). State of the art in content standards. Technical report, LANDSEB-CNR, Padova, Italy and University of Madrid, Spain, 2002. http://www.ontoweb.org/deliverable.html.

[9] M. Klein. Combining and relating ontologies. In H. Stuckenschmidt and M. Uschold, editors, *Ontologies and information scharing*, Working notes, IJCAI-2001 workshops. AAAI, 2001.

[10] A. Kotok and D. Webber. *ebXML, the New Global Standard for Doing Business over the Internet*. New Riders, Indianapolis, Indiana, 2001.

[11] Sun Microsystems. Implementing services on demand with the Sun Open Net Environment — Sun ONE, 2001. A Sun Professional Services White Paper.

[12] J. Sowa. *Knowledge representation. Logical, philosophical, and computational foundations*. Brooks/Cole, 2000.

[13] ebXML Technical Architecture Team. Technical architecture specification v1.0.4. WWW publication, 2001. http://www.ebxml.org/specs/index.htm#technical_specifications.

[14] F. van Harmelen and I. Horrocks. FAQs on OIL: The ontology inference layer. *IEEE Intelligence Systems*, Nov/Dec 2000.

[15] David Webber and Anthony Dutton. Understanding ebXML, UDDI, XML/EDI. WWW publication, 2000. http://www.xml.org/feature_articles/2000_1107_miller.shtml.

[16] Frederik Willaert. XML based frameworks and standards for B2B ecommerce. Master's thesis, Katholieke Universiteit Leuven, 2001. http://www.ebxml.org/documents/ebxml-thesis.pdf.

Part III

Research and Applications

Chapter 9

Reality and Truth in the Semantic Web

Heikki Hyötyniemi

9.1 About Semantics

It has been claimed that (at least in Finland) the level of information technology is high as compared with the level of contents that is being delivered by that technology. Due to the technology enthusiasm, the services utilizing the infrastructure are asymmetrically underdeveloped. What comes to the Semantic Web Initiative, it seems that also in this case the techniques and formalisms are the driving force, and not very much has been thought of the actual applications. The role of this chapter is to show how it could really be the application itself that gives fresh blood in the already somewhat "institutionalized" Web arena.

9.1.1 Internet: Irregularity as a Rule

In what follows, a very special application area is being studied in depth. It turns out that the approaches and tools that can be utilized to achieve the objectives set forth by the Semantic Web initiative are also very special in this case. One could claim that as such a special case, the related discussions would be of interest only to a marginal group of people. However, this is not true: The Internet society consists of individuals, and the multitude of interests and needs is overwhelming. In fact, it is like with humans themselves: All are special, there exist no "averages". In different fields the best approaches differ from each other. In this sense subjective view becomes objective — concentrating on a single special field makes discussions more concrete, this way perhaps helping to see the essence of the multifaceted

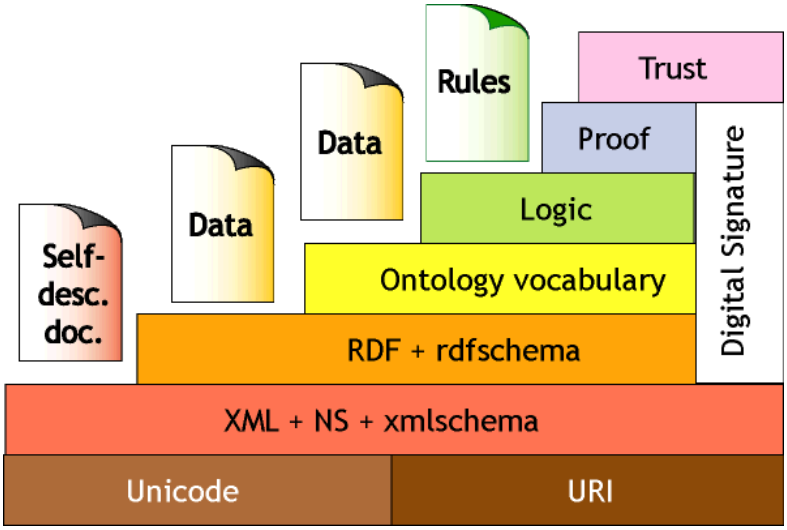


Figure 9.1: The normal view of the structure in the Semantic Web: Starting from the very low-level constructs, higher and higher conceptual levels can be defined. Today, only the very bottom levels have been reached — and getting forward will most probably be a struggle (courtesy of Marja-Riitta Koivunen)

general problem. So, starting from an application and putting it in a perspective, the real potential of the new paradigms can be visualized: What can be achieved if the heterogeneous nature of user interests is taken into account from the very beginning.

Indeed, when studying this one special case (research work on a mathematically oriented discipline) very interesting results can be reached — at least in principle. It turns out that one can directly leap on top of the very challenging hierarchy (see Fig. 9.1): It is now the *trust* itself that is being put in test.

9.1.2 Note on Artificial Intelligence

To be able to do anything clever with data, the meaning of the data and data structures, or semantics, needs to be known. When trying to capture the semantics of constructs, one is facing the eternal challenges that have been studied in the field of Artificial Intelligence (AI). If the computer should carry out such tasks as representing and processing of semantics-loaded information, the computer should somehow *understand* the data. Intuitively, true understanding can only exist in the human brain, and the experiences with *deep AI* should make us cautious when "semantics in the Web" is being developed. The enthusiasm on the Semantic Web resembles very much

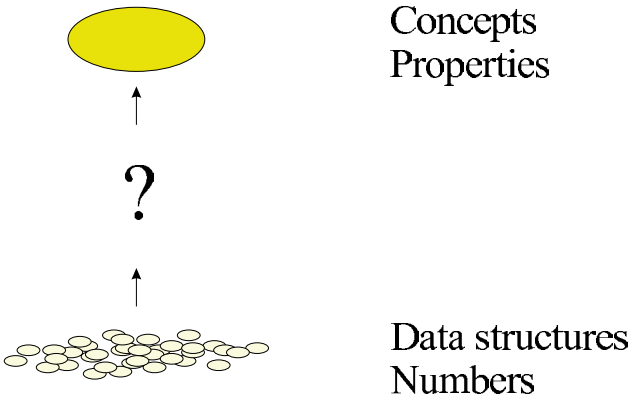


Figure 9.2: Grounding of semantics: An eternal mystery

the earlier euphoria in the AI community. Seemingly it takes one generation of researchers (some ten to twenty years) to forget the experiences of the previous AI boom ...

In the 1980's *knowledge engineering* with expert systems was seen as a key to capturing the semantics in different domains: The idea was to explicitly define knowledge in a rule form (or in the form of semantic *nets*!). As an ultimate example, the CYC project tried to formalize all common sense understanding. However, in retrospect, the expert system projects finally did not pay back. It was noticed that the symbolic representations are too clumsy; as a partial solution to this, *fuzzy* rule systems were introduced. And, after the symbolic disappointment, the other extreme was exploited: For example, in neural networks all symbolic knowledge is ignored, everything has to be presented in a numeric form.

Different branches of research seem to mature through similar routes — and there is probably no force that could resist the same natural dynamics of memetic development in the Semantic Web research[1]. Of course more streamlined evolution would take place if the lessons learned in the AI field were taken into account. Let us make a guess: After the symbolic approaches have been exploited in the Semantic Web field, and after their deficiency has been detected, perhaps some day also in that field connectionist approaches are reinvented, and some kind of "neural webs" are introduced!

Looking closer at the nature of semantics, it is clear that the structure among the semantic entities consists of a hierarchy, where higher-level con-

[1]As comparing the situation in Expert Systems research community twenty years ago to the situation in Semantic Web research today, there is one major difference, though: As the expert systems were constructed as stand-alone applications, all infrastructure had to be constructed from the beginning; now, all information already is in the electronic form, and new functionality can be implemented with not-so-huge efforts. Today, the extra development work may be worthwhile even if the gains were marginal.

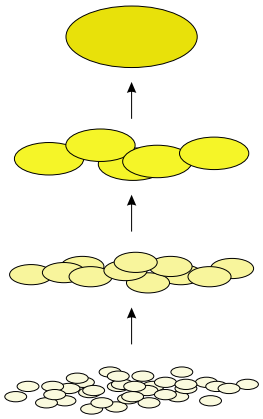


Figure 9.3: In special fields (for example, in systems engineering), the grounding of semantics can be explicitly defined

cepts are defined contextually, using the lower-level concepts. Still, the hierarchy of concepts is a delicate thing, and very little can be said about the mapping of lower-level constructs into higher-level concepts in general (see Fig. 9.2). The hierarchy tree is not only a directed graph of concepts; it seems that some kind of *qualitative steps* need to take place between the levels, so that the higher concepts somehow *emerge* from an appropriate combination of the lower-level ones. What seems to be true is that using symbolic representations only the definitions of concepts cannot be implemented in a natural way: The symbolic representations are too rigid to facilitate this emergence. In special cases, much more concrete results can be presented (see below).

9.1.3 Case: Research on Systems Engineering

Generally, semantics of a domain field cannot be explicitly expressed — but there are exceptions: In mathematical disciplines, typically, all concepts are unambiguous, and they can be reduced to lower-level constructs. One example of such mathematically oriented domain areas is *systems engineering*, where the mathematical machinery is applied to modeling of real-life (dynamic) phenomena. The model explains the object system in mathematical form; the model can be analyzed and used as a testbench for constructing controllers for the real system. Research in this field consists of defining mathematical methodologies and tools for different purposes.

In systems engineering environments, the holes in the semantic hierarchy tree can be fixed (see Fig. 9.3); it is the underlying numeric data structures and consistent mathematical procedureds that make this possible. What

this means in concrete terms, can perhaps best be illustrated using a simple example. Study the discrete-time system defined recursively as

$$y(k+1) = a \cdot y(k), \qquad y(0) = y_0, \tag{9.1}$$

where a is some parameter and y_0 is the initial value, k being the time index. Looking at this expression at the symbolic level, very little can be said about it (because of the self-referential structure, the symbolic process would perhaps only end in a *stack overflow*!). On the other hand, if one iterates the expression in the numeric form, it is easy to detect that minor changes in the numeric value of a result in major qualitative changes: Only if $-1 \leq a \leq 1$, the behavior remains bounded (no matter what is the numeric value of $y_0 \neq 0$).

The above example illustrates that such important high level concepts like stability, etc., cannot be attacked in the symbolic form; on the other hand, applying numeric calculations the emergence of conceptual phenomena can be reached in an intuitively plausible way. When the model dynamics is iterated indefinitely in numeric form, the truly symbolic concepts crystallize. It turns out that in special fields like in systems engineering the "Wittgensteinian trap" can be avoided: Even though some things cannot be expressed in a natural language, they still do not have to be automatically ignored. Even though there are "subsymbolic" constructs that cannot be attacked when using natural languages, mathematics is such a language where valid concepts can also be defined as *processes* or *algorithms,* so that the granularity and discontinuity of the symbolic representations can be avoided. Only after the processing, when the final results are to be interpreted in the human language, the data structures are granulated into symbolic form.

Mathematics is a powerful language for expressing semantics. Of course, semantics can be expressed explicitly in a math form only in special cases; however, these special cases are still extremely relevant at least for some people, and the perspective of such people is here elaborated on.

9.1.4 Science and Paradigms

Internet has already affected the scientific community a lot: The real-time communication, on-line databases and electronic journals, search machines and hyperdocuments ... these have all had a remarkable impact on how science is being done. But this far all these new possibilities have only been applied to support the old ways of doing science. What could perhaps be achieved if storing and processing of semantics were automated — in science the consequences would be revolutionary[2].

[2]It is like chasing the rainbow ... the target is so desperately far — today, many of us would be happy if only the simple mathematical formulas could be expressed in HTML!

In the scientific community there are different views whether the developments are good or bad. The purists still say that the only way to do science is to sit alone in a chamber with only a pencil and a piece of paper ... However, the theorems proven by computer have already changed this traditional view, and there is no return. In fact, these computer proofs do not yet represent any major shift in the scientific paradigm: The role of the computer is only to go through the routine derivations, whereas the theorem—proof structure of a theory still remains the same. A new view to the role of computer is proposed by, for example, Stephen Wolfram: In the *New Science* the reductionistic approach is substituted for holistic one [9]. The starting point in New Science is that the essential system-level phenomena cannot be seen when looking at the low-level constructs alone. When simple things cumulate, the final result can be qualitatively something quite different. An example of the emargence idea was given in the previous section — in systems engineering, stability is an emargent phenomenon. How to capture the emergent phenomena, then — here the computer with its huge iteration capacity has the central role.

There are many consequences what comes to the emergence of the New Science that can be used to reach new levels in science. Perhaps the most relevant result is that when the mathematical structures are analyzed in a numeric rather than in symbolic form, semantics can be incorporated in the mathematical structures themselves (as discussed in Sec. 9.1.2). Perhaps the new perspectives are best illustrated by an example:

> Remember the *Kuhnian theory* of development in science: First there is the *thesis,* standing for the mainstream view; then an *antithesis* pops up trying to refute the old view; finally, a *synthesis* is (hopefully) found, integrating the different views [7]. It could be assumed that finding syntheses would be easier today when the means of communication are so marvellous — however, it seems that no syntheses are aimed at today: It is easier for the researchers just to develop their own theories, without trying to find syntheses. New scientific schools emerge, trying to promote their own views. "Gurus" can efficiently advertise their ideas in the net, and the research becomes hype-driven. It is usually claimed that this explosion of information is an unevitable result of Internet; however, in the semantics-integrated environment one could have automatic tools for reaching understanding rather than misunderstanding. Web with semantics could serve as the *censor,* filtering out unsubstantiated claims!

The value of the scientific claims should be measured based on their *relevance,* not by the forum where they have been presented or who has presented them. This relevance issue will be elaborated on in the next sections.

9.1.5 Example: Parameter Identification

Assume that examples of a system input–output behavior are given, and one should find the model parameters best matching the observations. This kind of enterprise is called *parameter identification*, and it is a typical task to be carried out in systems engineering [8]. Parameter identification techniques have been developed very far and the mathematical basis is solid, but no doubt the above relevance considerations apply here as well. For example, a theorist could claim that

> "recursive identification methods give the same parameter estimates as the off-line methods".

Of course, this is true, it has been rigorously proven, and arguing against this claim is useless — however, a practicing expert could also say that

> "This is *true*, but ... *so what?*"

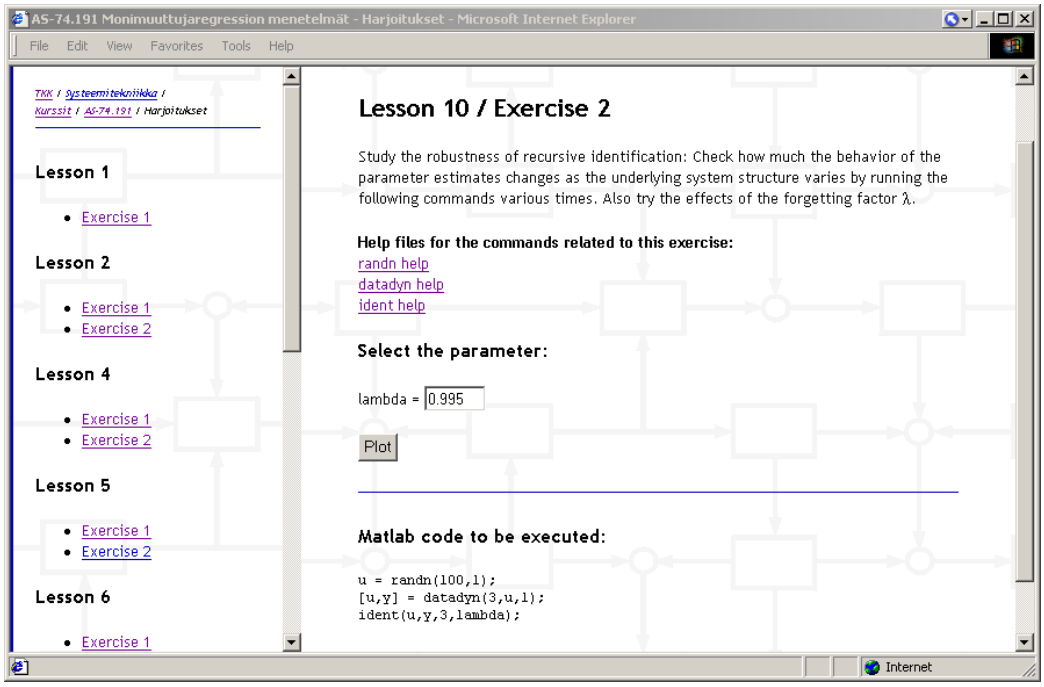


Figure 9.4: User interface: Testing the robustness of on-line vs. off-line identification approaches (see Fig. 9.5)

A domain area expert can question the relevance of academic theories. It is difficult to fight against proven theorems, but from the practical point of view, there really exist problems: First, the theorems only assure convergence after

infinite time; second, the robustness of the identification process can be poor, the dynamics of the parameter estimates perhaps being very unforeseeable — after a long time of stable behavior, very abrupt changes can take place [3].

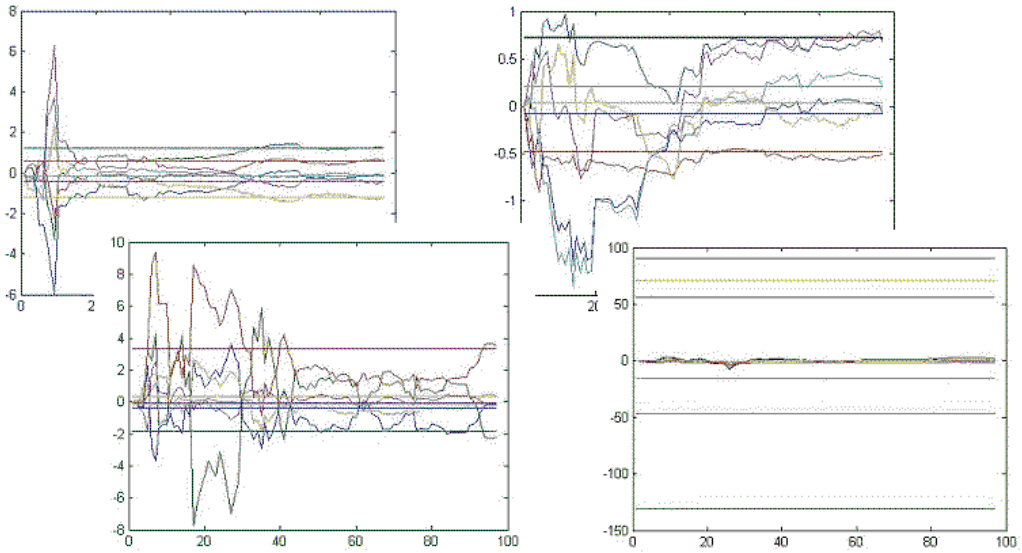


Figure 9.5: Qualitatively very different on-line identification results, showing that the robustness of recursive identification is very much dependent of the actual process parameters, and also of the signal realizations. Four different models of the same order have been defined, the parameters varying randomly; The models have been simulated, and the resulting data has been used for parameter estimation. The off-line parameters, explicitly minimizing the cost criterion, are shown as straight lines, whereas the evolution of the recursively identified on-line parameters are shown as functions of time. After seeing this kind of examples, one is perhaps better capable of assessing the true practical value of different approaches — note that according to the theory, on-line and off-line approaches should be equally applicable.

To emphasize the relevance of relevance (!) in practice, the related course at the Control Engineering Laboratory of Helsinki University of Technology also contain Web-based exercises where the theories can be tested against practice [4]. The calculations are carried out in the server, whereas the parameters are delivered by the end-user; this way, intuition of the robustness of the presented methods can be tested. The algorithms were implemented using the extensible Matlab formalism, and Matlab WebServer was utilized to realize the interaction over the Web (see Figs. 9.4, 9.5).

What comes to the New Science, its emergence can also be seen in the field of systems modeling. Take two extremes: Recursive identification has been

proven to work; however, in the applications (adaptive control, for example) it seems not be an absolute success story. On the other hand, take the self-organizing Kohonen networks [6]: Their convergence cannot generally be proven, but they still seem to give good results in practice. The memetic evolution cannot be stopped — no matter whether something can be proven or not; if it works in practice, it can be used as a basis for further research. And there already exist thousands of applications of Kohonen networks. (Of course, something that is mathematically proven *is* true, assuming that the underlying assumptions hold, and in this sense it is on a solid basis. But often the assumptions about the noise properties, etc., do not hold. Not fulfilling the assumptions that are needed to apply the mathematical results — and still applying them — sounds questionable: Is it not more honest to admit straight in the beginning that the analyses are heuristic, rather than hiding the uncertainties under a convincing-looking, dense mathematical jargon?)

9.2 "A New Kind of Research"

Perhaps the view of the future New Kind of Science as proposed by Stephen Wolfram [9] is too high-spirited — still, the objective of science has traditionally been to explain the principles of nature in explicit terms, not through obscure procedures. On the other hand, in *applied science,* for example in systems engineering research, we have more freedom: The pragmatic points of view dominate over the strictly theoretical ones ("If it helps us it will be used!").

9.2.1 Theory vs. Practice

In engineering sciences, compromises between pure scientists and practicing engineers need to be made: It is not only the scientific results themselves that are important, but also their applicability in practice. The theories are (or should be) developed to attack real-life problems. The key question of a practicing engineer is the following:

> "Is this method applicable to my problem?"

In the field of systems engineering, for example, there exist dozens of alternative approaches and algorithms for attacking almost any imaginable problem, each of them having special properties and special requirements for the application. In many cases the theoretical requirements cannot be checked, or they may be too conservative and cannot really be fulfilled. What is truly essential and what is not, can the method be applied after all — this cannot be explicitly formalized, and expert knowledge (or some other source of semantics in a functional form!) is needed. Yet another problem is that there

usually exist various parameters that can be tuned to enhance the algorithm performance. Which are the appropriate parameter values? Experimenting with novel methods, there is always the same uneasy feeling: Is the algorithm being evaluated fairly, or could the behavior of the algorithm be optimized for the special application somehow?

One additional problem is caused by the fact that the scientific papers, etc., where the new algorithms are presented, do not make this task of determining the validity of different methods any easier. In a typical paper, the theory is presented, the algorithm derived, and a few examples are shown. Surprisingly, according to the research papers, all methods seem to be successfull! There is a very fine line between extreme honesty and slight "streamlining" what comes to documenting of the results ... having the modern computing power available, it is always possible to find such a realization of data that the expected results are found.

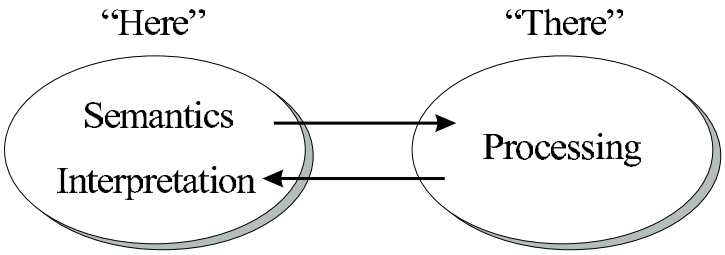


Figure 9.6: Data and interpretations (semantics) coming from the end-user, while mathematical manipulations are carried out elsewhere

It is clear that no "universal problem solvers" exist in research work — but the practicing engineers life would become much easier if there existed some tool that could give a comment like:

> "In your case, the relevance of the proposed method is only 0.1".

9.2.2 Role of Relevance

When doing research in the traditional way, facts are either true or false, there are no shades of gray — as the fuzzy theorists would say. The fuzzy enthusiasts have made a good job in introducing the creeping doubt in the modern man's mind about the validity of the age-old two-valued dichotomies. However, in our case the interpretation of fuzziness needs to be sharpened: At least in some applications *relevance* is intuitively better suited interpretation for fuzziness than what *possibility* is. As Manfred Eigen (Nobel Prize in Chemistry, 1967) has put it:

> "... A *theory* has only the alternative of being wrong. A *model* has a third possibility — it might be right but irrelevant.

How is this relevance issue related to the above discussions? As was noticed, in a mathematical domain field the concepts can explicitly be determined using lower-level concepts. But where do the lowest-level concepts derive their semantics? Mathematics itself consists of rather formal manipulations, and, finally, all semantics has to come from outside the purely mathematical environment. The lowest-level constructs in the hierarchy are the elementary data units that are characteristic to the actual application, the semantics being buried in the dependencies between data units. On the other hand, the relevance of the highest-level concepts is dictated by the user's preferences and design specifications.

Now it is time to collect the above discussions together: In the Semantic Web schemes presented earlier in this book, the semantics is preprogrammed and hidden inside the net. In our case, however, if a mathematical "smart" environment is implemented in the Web, it turns out that the grounding of semantics must come from the end-user: He delivers the problem case, data describing the system, and it is very much dependent of the properties of this data what kind of conceptual results are found during the mathematical manipulations. And after the manipulations, the relevance of the results still have to be evaluated by the end-user. The "semantic flow" between the end-user and the Semantic Web is visualized in Fig. 9.6.

9.2.3 Towards Semantic Libraries

It was assumed that the user delivers data and evaluates the final results — would it also be possible to automate the user's burden? This means that the "interfaces" should be made more general in both ends: First, the delivery of data needs to be redesigned.

To generalize the view of semantics, the starting point is the *model*. The role of the model is to describe how the signals are modified in the system: In this sense, the model also defines the system semantics. The naturalistic semantics of a model can be paraphrased as: How is the system connected to outside world, what kind of effects does it have when it is running? This is explicitly revealed by *simulation*. When a functional model is defined instead of one fixed set of data, much more functionality can be reached.

When a functional model is available, new information can be extracted from it when needed. For example, the problem of tuning the parameters can now be solved by the Semantic Web automatically: Stochastic optimization can take place, applying some Markov Chain Monte Carlo method, trying a set of parameters, running simulations, and adjusting the parameters appropriately.

What is more, if the semantic machinery has understanding also on the design objectives, etc., the whole design process can be automated, and the end-user can be eliminated from the design loop altogether. The semantic

machinery can evaluate intermediate results and make the decisions whether to continue iterating the design loop or not Fig. 9.7. Various libraries developed for different purposes can be combined for special design tasks. What is needed is a general-purpose simulation engine for running algorithms and procedures. This kind of ideas were presented already in [2], but with the new tools this target is much nearer (see also [5]).

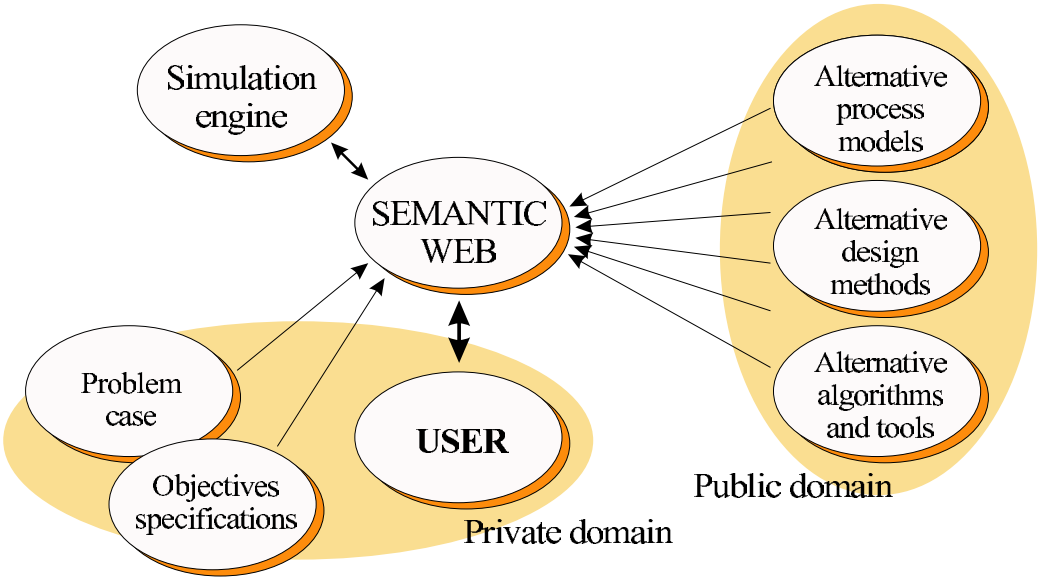


Figure 9.7: Semantic Web in real use, as seen by the end user: Loops of analysis and synthesis have been automated in design

No single person can be aware of all developments even in his own field. When the Semantic Web is there, expertise is not dependent of individual humans. The research community becomes self-organized by the system: Researchers send their algorithms, etc., in the Web[3], and the system automatically organizes the contributions, based on the relevance of that research work ... What a frightening (?) view: the Web truly is more competent expert in special fields than any human. But this is a natural extension — we have already got acquainted with the computer taking care of our information manipulation tasks; the next step is that the same happens with knowledge processing, and even with the tasks necessitating know-how!

It is easy to find applications for this kind of mathematical Semantic Web applications: Design systems, interactive publications, functional databases, etc. — and also in the field of education there exist big promises [1].

[3]What is needed is a standardized interface between semantic functional entities. If such a standard interface is someday introduced in the Web, there will be most probably be a boom like there was with HTML: When the HTML language just became available, there was an explosion of different kinds of Web pages.

Bibliography

[1] H. Hyötyniemi. Hypertechniques in control engineering education. In *Preprints of the IFAC Symposium on Advances in Control Education (ACE'94), Tokyo, Japan, August 1–2*, pages 185–188, 1994.

[2] H. Hyötyniemi. Towards portability of knowledge in control systems design. In *Proceedings of the First Asian Control Conference (ASCC'94), Tokyo, Japan, Vol. 3*, pages 721–724, July 27–30 1994.

[3] H. Hyötyniemi. On structural identifiability of dynamic models. In *Preprints of the IFAC Symposium on System Identification (SYSID'97), Fukuoka, Japan, July 8–11, Vol. 1*, pages 243–248, 1997.

[4] H. Hyötyniemi. Multivariate regression — techniques and tools. Technical Report Report 125, Helsinki University of Technology, Control Engineering Laboratory, 2001.

[5] H. Hyötyniemi and A. Nissinen. New Simulation Tools: Towards Mastering the Explosion of Information. In *Proceedings of the EUROSIM'98 Simulation Congress (ed. K. Juslin), Espoo, Finland, April 14–15, Vol. 1.*

[6] T. Kohonen. *Self-Organizing Maps*. Springer–Verlag, Berlin, 1995.

[7] T. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, 1962.

[8] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. The MIT Press, Cambridge, Massachusetts, 1983.

[9] S. Wolfram. *A New Kind of Science*. Wolfram Media, Champaign, Illinois, March 2002 (to appear).

Chapter 10

Annotea: Applying Semantic Web Technologies to Annotations

Marja-Riitta Koivunen

Annotea is a shared annotation system in the Web that uses the Semantic Web technologies. It is built on top of a general-purpose open RDF infrastructure and it models annotations as RDF metadata. Annotations can be attached to any XML-based document, such as XHTML and SVG document, or to annotations themselves, without having to modify the original documents. The annotations are shared by storing them to annotation servers which can be queried by a community of users who retrieve the annotations by using annotation capable clients, such as the W3C Amaya editor/browser.

Unlike other Web annotation systems, Annotea is completely open and is built on top of standard W3C technologies. We use RDF to model the annotations, XPointer and XPath to attach annotations to documents, and HTTP for all the data exchange between client and servers.

The Annotea annotation model can be easily extended to support other similar collaborative applications that share metadata about Web pages. Some of these scenarios are described here.

10.1 Introduction

Users of the World Wide Web collaborate by sharing content through Web pages. This kind of collaboration is limited as readers can seldom write

Figure 10.1: A user shares a comment with another user by annotating a Web page.

back to the pages to share annotations, such as comments or questions, even when they are members of a closed collaborative group. Instead, much effort is spent on forming and trying to understand different e-mail conventions for commenting or annotating Web documents.

Annotea [6] uses Semantic Web technologies to create annotations that can support very rich communications about the Web pages without requiring write access to the annotated page. A user can attach an annotation to a Web page for a collaborator who sees the annotation when he or she retrieves the same Web page (see Figure 10.1) as long as they share the same annotation server.

Annotea annotations are metadata about the Web pages or parts of Web pages. They use metadata vocabularies grounded in semantically rich ontologies that are themselves published in the Web. This metadata infrastructure opens many possibilities that can be extended beyond the basic annotation capabilities [7].

Section 2 in this paper describes a simple scenario showing how annotations can support collaboration and then broadens the scope of the annotations in a couple of additional scenarios. Section 3 explains the basic Annotea metadata infrastructure in more detail, and then explains the features that are needed to support the additional scenarios in Section 2.

10.2 Scenarios

The following subsections describe three annotation scenarios. The first scenario explains the use of annotations for basic collaboration, the second one shows an interpretation of shared bookmarks as annotations, and the last scenario examines the use of annotations for communicating evaluation results. Currently, Annotea offers implementations for the annotations and replies discussed in the basic collaboration scenario; the other scenarios describe possible extensions.

10.2.1 Scenario: Using Annotations for Collaboration

A group of remote education students are writing a report on the communication of whales. They collaborate by using the Web to publish new material, to search and share hypertext links to references and to annotate the material they uncover. By using annotations to conduct their commentary on their reading, the group avoids contention for write access to a single shared document and potential loss of data from conflicting updates.

The group uses an annotation (metadata) server dedicated to this seminar to store their annotations. Only users subscribing to the seminar annotation server can see the annotations. Figure 10.2 presents one of the annotations made by the group. It is presented to other students with a pencil icon, which can be opened to an annotation window containing the content of the annotation.

As the students read the reference papers they find from the Web, they mark each paper as interesting or uninteresting by attaching annotations to it. They use annotations also to mark or question unclear text, point out interesting perspectives, add keywords or categories, and share other general comments with each other.

At times, the students may disagree on the comments made by others or they may want to add some information to the comments. When this happens, they use replies to create a discussion thread and attach it to an annotation. Figure 10.3 presents a sample reply to the annotation in Figure 10.2. The original annotation now presents the thread of the replies at the bottom of the window. The replies can be opened from that thread by double-clicking on them.

Later in their process the students dedicate one person to write more detailed replies to selected research questions pointed out in the annotations. Occasionally, this starts fruitful discussions in the context of the reference document. The results are gathered into summary pages that again can be annotated.

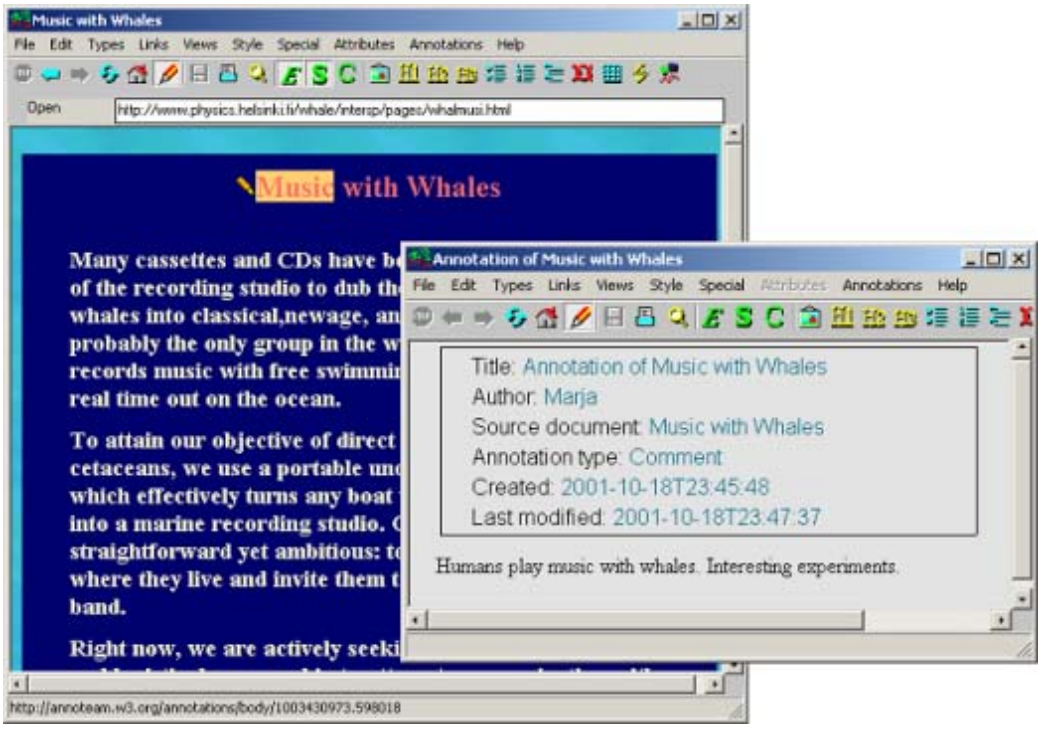


Figure 10.2: A selected annotation is opened in an annotation window.

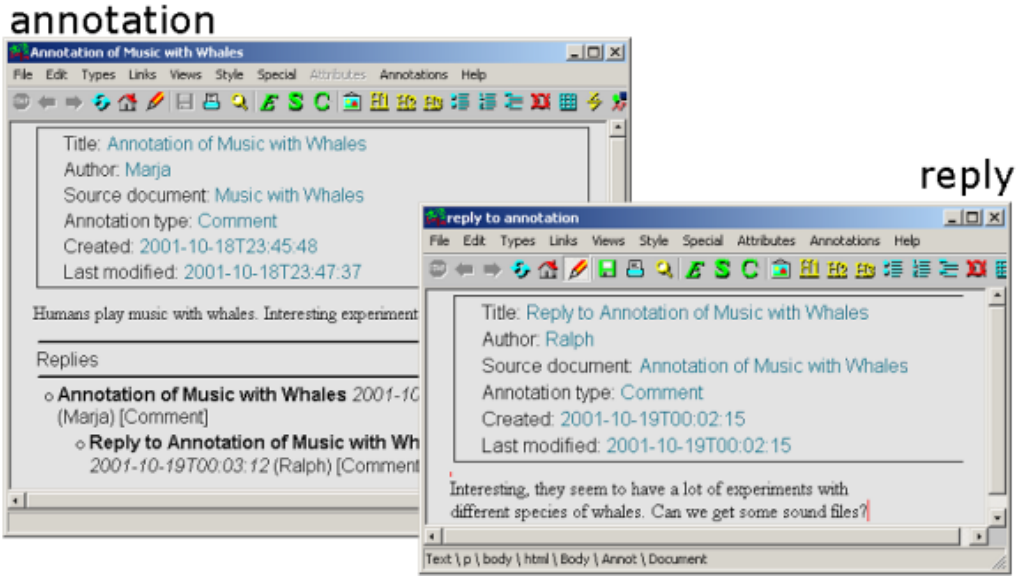


Figure 10.3: Annotation window with a reply thread at the bottom and an opened reply window.

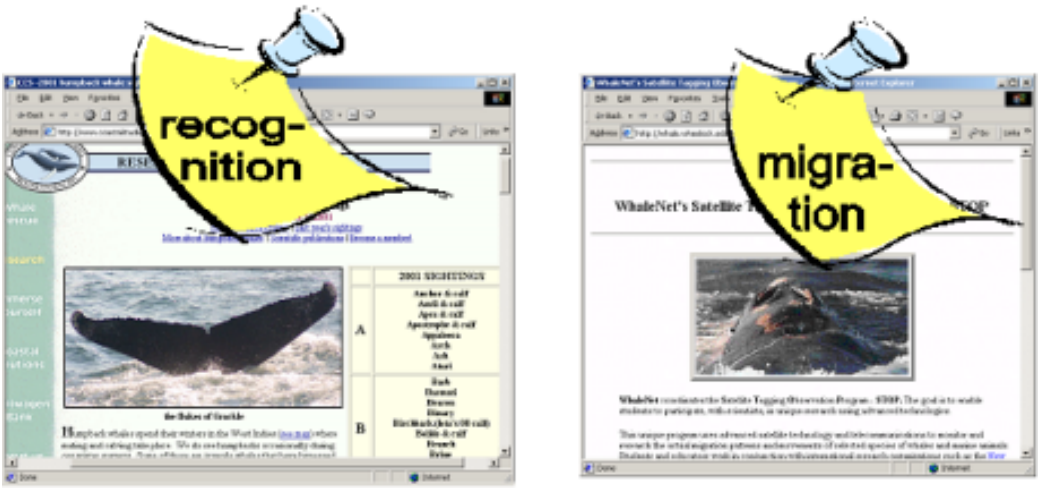


Figure 10.4: Annotations can attach categories to Web pages.

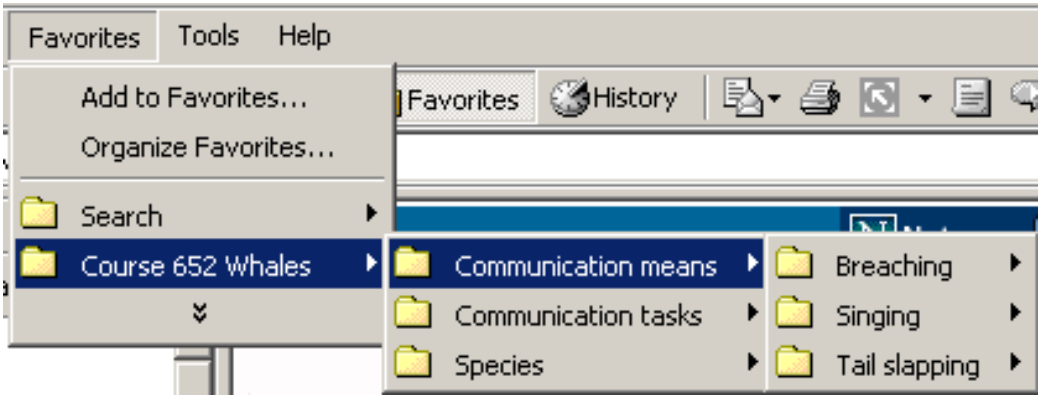


Figure 10.5: A sample hierarchical presentation of bookmark categories.

10.2.2 Scenario: Using Annotations for Shared Bookmarking

The student group in the earlier scenario uses traditional Web search tools to locate references on the Web. They want to group the papers under categories and list them on a Web page. Instead of each of them editing manually a shared page they attach special annotations with category information to interesting pages (see Figure 10.4). In a Semantic Web sense the category is just one of a variety of such extensions that the group can store with their annotation metadata.

These annotations can then be presented in a category hierarchy similar to the bookmark or favorites hierarchies in today's browsers (see Figure 10.5). Or they can be presented as icons on a bookmarked page. In a sense, the annotations with categories are bookmarks.

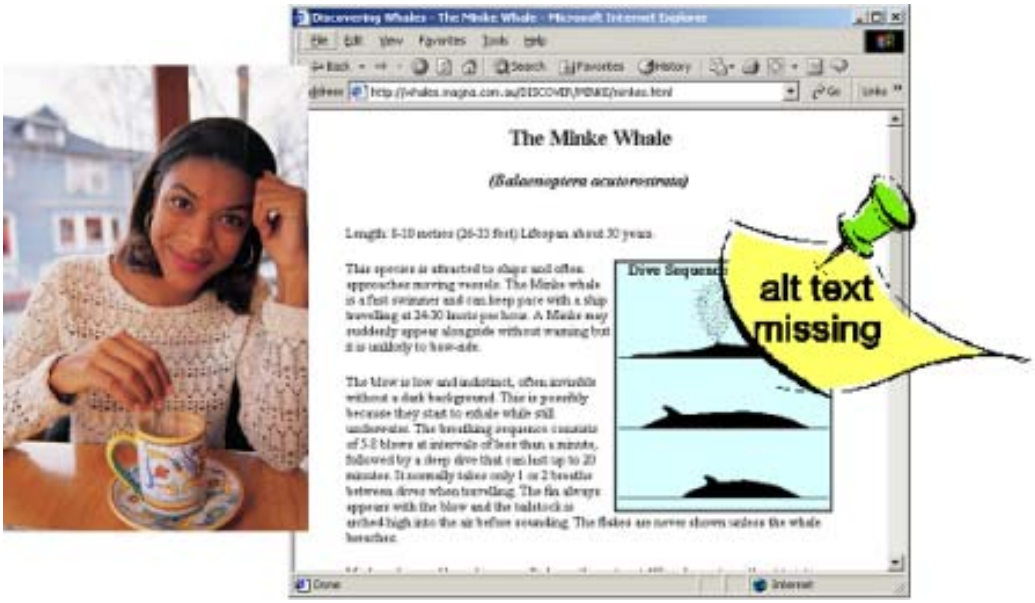


Figure 10.6: The teaching assistant marks the accessibility defects in students' pages by using annotations.

The students may filter the list of bookmarks in different ways, for instance, show all categories in alphabetic order, or show categories grouped under different users. Students can also define the categories themselves or select the categories from an existing ontology.

10.2.3 Scenario: Using Annotations to Present Evaluation Results

Annotations can also be used to attach either automatic or manual evaluations to a page. For instance, Kim, the teaching assistant for the seminar, uses annotations to remind the students that the readers of their documents may have different physical or cognitive abilities in receiving and interacting with the information and that their documents need to be accessible.

Kim uses the Web Accessibility Initiative [3] guidelines and some automatic tools for assessing the markup used within Web pages the students create. These accessibility assessment tools rely on EARL [5], a metadata language expressing what is or may be wrong in a page, citing by URI the specific guideline that describes the accessibility issue.

Kim stores the EARL analysis of each document in the same annotation server that holds the seminar's other annotations. Kim also adds to the server some inferencing rules that represent a transformation from the EARL vocabulary to the annotation vocabulary. The EARL vocabulary is a superset of the annotation vocabulary, so Kim includes some style rules that

instruct presentation clients in the rendering of the extra properties of the EARL metadata.

When students view their pages, they see the EARL report items as annotations on the pages as a result of processing the inferencing rules. For instance, in Figure 10.6 Kim has attached an annotation to the image of Minke whales stating that it does not have alternative text and therefore is not accessible for user's who cannot see the image. The students can address the accessibility issues in the context of the page and add additional metadata to the annotations, for instance by replying to them, to note them as fixed or to request help from Kim. When Kim helps the group, she sends a mail to the discussion list explaining the problem and adds a link to the EARL annotation so that others in the group can benefit from the example.

When the corrections are done, the group can run the accessibility evaluation tools again. The document author can choose to delete the earlier report annotations at this time or she may just mark them as obsolete. The group may also freeze a copy of the evaluated page with the original annotations.

10.3 Annotea Metadata Infrastructure

The basic Annotea infrastructure is presented in Figure 10.7. When a user attaches an annotation to a Web page the annotation metadata is stored on one or more annotation servers. When the annotated Web page is visited by a user using an Annotea capable client the client queries annotations related to the document from the annotations servers to which the user subscribes. For the queries, Annotea has a special query language Algae in addition to a simple URI request for 'all' annotations of a page. The retrieved annotations can be presented to the user in several ways. In our Amaya [1] client they appear in the context of the Web page as icons, but it is possible to query and present the metadata information in many other ways also.

Annotea uses W3C technologies when possible. The HTTP protocol is used to store and retrieve the RDF/XML [8] metadata describing annotations from the Annotea servers. Each Annotea server is a generic RDF store. The XPointer standard is used to refer to the part of the document being annotated and Xlink is used to present the annotations on a Web page.

The metadata infrastructure of the Annotea project makes it easy to support the annotation scenarios presented above. The basic annotation schema and the extensions needed for the previous scenarios are discussed in the following sections.

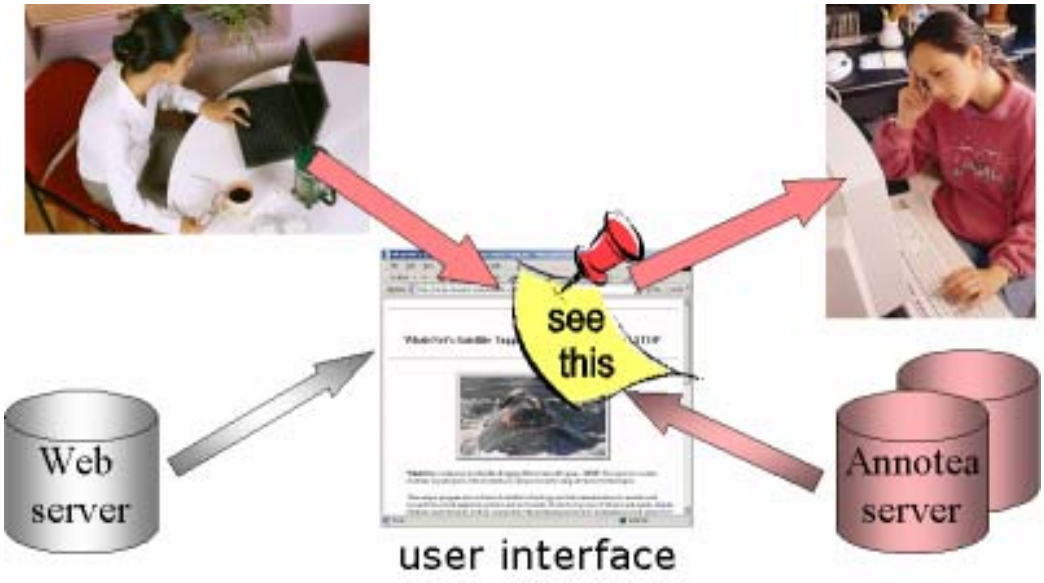


Figure 10.7: Annotea infrastucture.

10.3.1 Basic Annotea Annotations

In the first scenario, the students annotate Web pages and use the reply threads as supported by the Annotea infrastructure. The annotations and the replies in these scenarios are metadata described with RDF.

Figure 10.8 presents an instance of a basic annotation schema. It uses properties from multiple RDF schemas [4] e.g. Dublin Core (**dc:**) [2] to define the annotations. Annotations can apply to a whole document or just a part of it. The ***annotates*** property refers to the annotated document and the ***context*** property refers to the actual location of the annotation within the annotated document. The annotation content written by the user is stored in the **body** property and a descriptive annotation title is stored in the **dc:*title*** property. The other properties further describe the annotation.

10.3.2 Extending the Annotation Schema for Reply Threads

Annotea has also a reply concept that can be attached to an annotation or to another reply. Replies form discussion threads that start from an annotation.

Figure 10.9 presents an instance of a reply schema. It looks very similar to an annotation schema. It has two new properties, the ***inReplyTo*** property, which defines which annotation or reply was the previous one in the thread, and the **root**, which always points to the first annotation in the thread. The **root** is used for performance optimization, it permits us to find many replies more easily without having to do more chaining

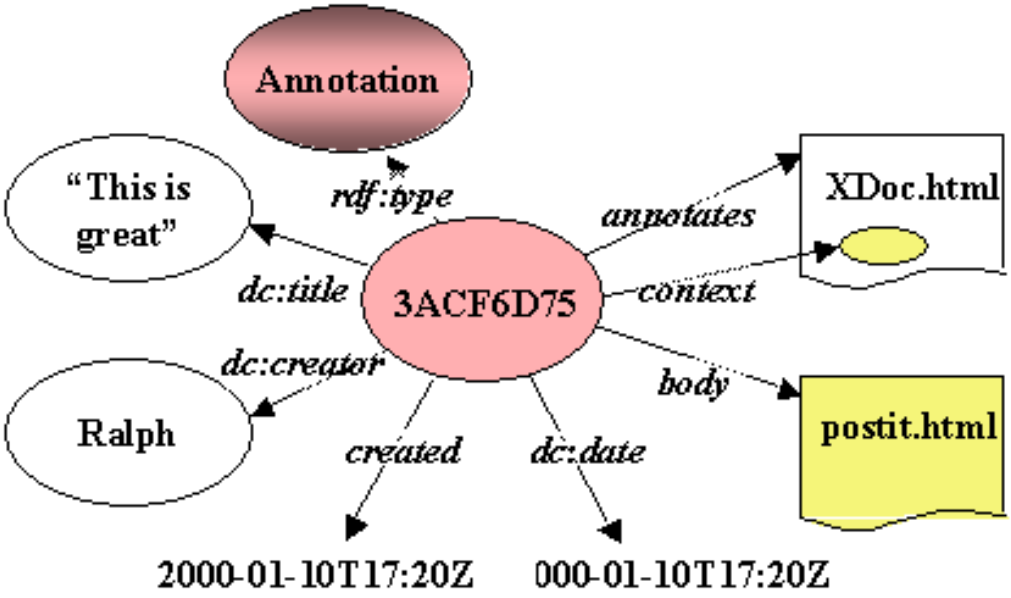


Figure 10.8: An instance of the basic annotation schema.

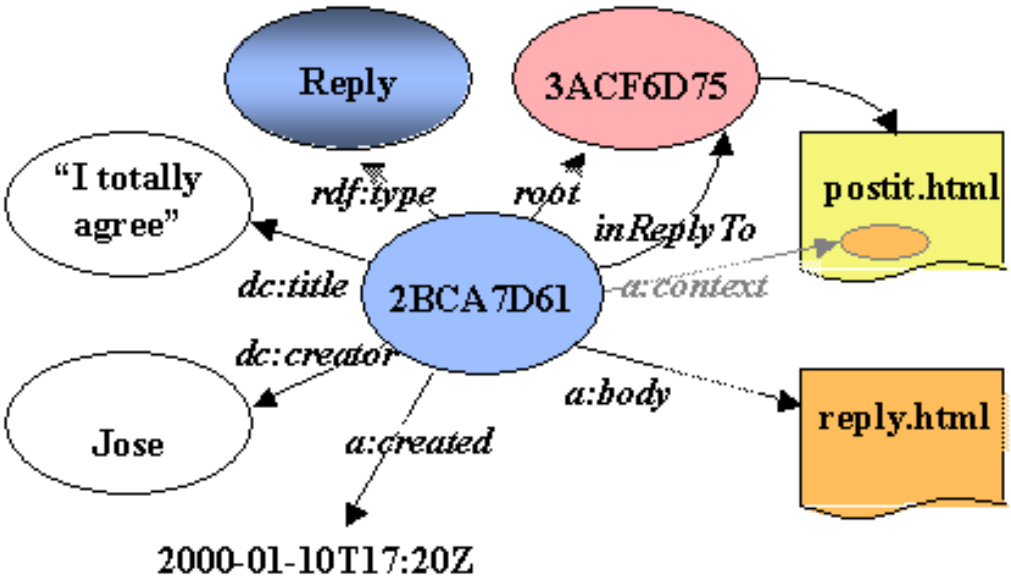


Figure 10.9: An instance of the reply schema.

The generic metadata-based design of our annotation server and the query language made it easy to incorporate the additional properties; the bulk of the work was in extending the user interface capabilities.

10.3.3 Using Annotea for Shared Bookmark Annotations

When we add a **category** property to a schema very similar to an annotation schema we can create bookmarks. These may be seen as annotations of type **bookmark** or as a separate bookmark concept. With RDF it is easy to add the category property. The DAML+OIL ontology construction vocabulary [9] provides a framework for describing new properties with precise semantics and placing those semantics in the Web.

The generic metadata approach naturally lends itself to supporting a variety of views of the bookmarks. We can write new queries so that no changes are needed for our annotation server. However, the user interface needs some work. The bookmarks need a special icon to visually differentiate them from other kinds of more conventional annotations when the bookmarks are presented on a visited page. Also we would like to be able to present all the bookmarks in a category hierarchy.

In addition, the client needs to be able to present the bookmark properties in a window in a similar way as we present the annotation properties. As new properties can be easily added to bookmarks or annotations it would be nice to be able to simply define a presentation style for each new property in the same metadata framework as properties of properties are defined.

10.3.4 Accessibility Evaluation Report Items as Annotea Annotations

Annotations can be used to present automatically generated report items, such as accessibility evaluation items or markup validation items. If the report items are described in the metadata format it is straight-forward to map them to an annotation schema. For instance, the EARL report item reporting an accessibility problem has semantics that map easily into an annotation of a part or the whole of the evaluated Web page. This mapping can be expressed as a collection of inference rules over the properties produced by the EARL tools.

The generic metadata framework provides the necessary flexibility to decide on a case by case basis whether to archive, delete, or revise annotations when a document is reprocessed through the evaluation tool. The tool can maintain state information for successive runs in the same metadata store.

10.4 Conclusions

Annotea is a metadata based annotation infrastructure for sharing annotations on Web pages. It uses standard W3C technologies and can support a broad range of different annotation needs. The generic property mechanism of RDF allows us also to construct ontology-neutral data stores. Applications can use several ontologies simultaneously to describe different aspects of their annotations.

Currently Annotea implements the basic annotations and replies for creating discussion threads. The new scenarios need extensions to the basic annotation schemas but the main work is in customizing the user interfaces. More research is needed to ease the presentation of the metadata, especially new properties from ontologies the application (or user) may not have previously seen. More work is also needed to develop client-side or server-side inferencing for mapping between ontologies.

Acknowledgments

Annotea is developed by a team of people. This paper is based on the innovative and hard work of Ralph Swick, Jose Kahan, Eric Prud'hommeaux, and Art Barstow. In addition, Eric Miller, Charles McCathieNevile and other W3C staff have contributed many ideas to Annotea. I also want to thank Elisa Communications for supporting this work.

Partial funding for the development of Annotea was provided by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0593.

Bibliography

[1] Amaya browser/editor home page. http://www.w3.org/Amaya/.

[2] Dublin core metadata initiative, Dublin Core metadata element set, version 1.1. http://purl.org/dc/documents/rec-dces-19990702.

[3] Web accessibility initiative home page. http://www.w3.org/WAI/.

[4] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[5] Sean Palmer (ed.). EARL 1.0 specification. WAI ER WG Note 09, 2001. http://infomesh.net/2001/earl1.0/.

[6] José Kahan, Marja-Riitta Koivunen, Eric Prud'Hommeaux, and Ralph R. Swick. Annotea: An open RDF infrastructure for shared web annotations. In *Proceedings of the WWW10 International Conference, Hong Kong*, May 2001. http://www10.org/cdrom/papers/488/index.html.

[7] Marja-Riitta Koivunen and Ralph Swick. Metadata based annotation infrastructure offers flexibility and extensibility for collaborative applications and beyond. In *Proceedings of the KCAP 2001 workshop on knowledge markup & semantic annotation*, 2001.

[8] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[9] Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks(eds.). Reference description of the DAML+OIL (march 2001) ontology markup language. Joint United States / European Union ad hoc Agent Markup Language Committee. http://www.daml.org/2001/03/reference.html.

Chapter 11

Semantic Web and Software Agents Meet Wireless World

Heimo Laamanen, Heikki Helin, and Mikko Laukkanen

The next generation Web is currently a very active and interesting research topic. One of the future trends is pervasive (ubiquitous) computing. There are significant challenges of designing and implementing pervasive computing environment. Firstly, the environment of wireless data communications is disperse and very complex. The use of wireless data communications in the Semantic Web requires adaptation to different kinds of wireless networks and to highly varying QoS including disconnections. Secondly, the amount of information in the Web is already beyond the manageable extent for a human being, and it is increasing all the time. This requires automatic, intelligent processing of information. This paper will discuss the Semantic Web and software agent technology from the viewpoint of wireless data communications. We will present features of wireless data communications and FIPA's approach to support agent-based Web services to adapt themselves. We will also discuss research challenges related to the Semantic Web and software agent technology in the context of wireless data communications.

11.1 Introduction

Despites its huge success, today's Internet lacks several features that enable easy, user friendly use of services everywhere and any time when needed by nomadic users. With nomadic users we mean all the people who may benefit of using of Internet services when moving one location to another. Moving may take place inside a home from the study to the kitchen or from one country to another. One of the future directions in the Web is pervasive

computing. Pervasive computing is the trend towards computing that takes place anywhere, any time connecting different kinds of computing devices using different kinds of data communications. In particularly, wireless data communications will play a significant role in pervasive computing. Pervasive computing devices range from tiny computing devices, which are embedded in almost any type of usable equipment to high power computers. In the future the pervasive computing environment comprises computing environment such as wearable computers, smart homes, and smart buildings. There are several challenges of designing and implementing pervasive computing, such as how to design interactions between different actors in the environment of pervasive computing [45].

The amount of data, or information, in the Web is already beyond the manageable extend for a human being, and it is increasing all the time. We can say that in the Web knowledge is already hidden in the pollution of unnecessary information. For example, searching for knowledge from the Web is often a frustrating experience resulting anything else but needed knowledge. And the frustration increases, when unnecessary information is transferred with high cost over low throughput wireless link. Therefore, we need tools to automatically manage, process, and transfer knowledge. The Semantic Web [5, 48] is planned to allow information systems to reason about data, data sources, and functionalities of other information systems. The Semantic Web will extend the data in the Web with well-defined meaning. This will be achieved by defining ontologies for each domain of knowledge and by specifying logical rules to process the knowledge. The languages used in the Semantic Web are based on XML [8], such as RDF [39], RDFS [9], DAML+OIL [46], and OWL [49]. These tools will enable automatic, intelligent processing of information, thus allowing filtering the knowledge out of polluted information. The protocols used in the Semantic Web to enable exchange of knowledge between producers and consumers are mainly SOAP [41] and HTTP [12].

Actors in the Semantic Web are called agents. Software agent technology has already been an active research topic for several decades—long before the birth of the Web. There are several roots of software agent technology starting from artificial intelligence to information retrieval and user interfaces. Mainly because of the several roots, there is no single, exact definition of the term agent. However, there is a general understanding about the main attributes of the software agent. Attributes, such as autonomy, proactiveness, goal-orientedness, and social ability are often mentioned as the main attributes. In multi-agent systems end-to-end interworking between peer agents is one of the key issues. Foundation for Intelligent Physical Agents (FIPA) [30], which is a non-commercial standardization body, has defined a set of specifications, that enable agents interact with each other. Software agents have already been implemented, for example, in industrial,

commercial, and educational application domains.

Wireless data communications is in the phase of rapid enhancements, even though the highest hype about 3G systems [33] has passed away. There are several trends: In the wide-area networks, there is a shift from 2nd generation systems, such as GSM, to 2.5 generation systems, such as GPRS [35, 36] and to 3G systems, such as UMTS [47]. In the local area networks, IEEE 802.11b [3] is gaining wider and wider popularity, and currently we think that WLAN will offer so called "hot spot" access point to the Internet. In the personal area networks, Bluetooth [7] is entering in the market offering technology for ad-hoc networks.

We can say that research in the field of the next generation Web includes three very interesting topics: wireless data communications, the Semantic Web, and software agent technology. In this paper we discuss the Semantic Web and software agent technology from the viewpoint of wireless data communications. We will present briefly features of wireless data communications and FIPA's approach to support agent-based Web services across wireless links. In addition, we will introduce research challenges related to the Semantic Web and software agent technology. This paper is organized as follows: Section 2 presents wireless data communications and its features. Section 3 discusses briefly software agent technology, and Section 4 discusses the Semantic Web. In Section 5 we introduce research challenges related to coalition of the Semantic Web, software agent technology, and wireless data communications. Section 6 summarizes this paper.

11.2 Wireless Data Communications

Wireless technology has been one of the hottest topics in data communications for the last five years. During these years there have been several improvements in the technology such as the shift from analog systems to digital systems, better availability and reliability, and higher throughput. And the rate of improvements is expected to increase during the next few years.

There are three domains in the wireless data communications: wide-area, local-area, and personal-area data communications. In each of these domains technology is expected to improve significantly and will offer more bandwidth, security, and reliability. These improvements will enable data communications that are required by pervasive computing. In wireless wide-area communications the next important step is 3G technology (such as UMTS [47]), which is currently entering into the market. UMTS offers transmission speeds from 64 Kbits/s up to 2 Mbits/s. UMTS and GPRS networks have a joint core network infrastructure, which enables users to seamlessly roam between them. In addition, operators can more easily to build nation-wide cover-

age areas. In wireless local-area networks, IEEE 802.11b is gaining more and more popularity in both intranet networks and hot-spot Internet access. The transmission speed will be up to 11 Mbits/s. The coverage area of a WLAN network is limited covering usually a building or a campus area. In personal-area networks Bluetooth technology is entering into the market. One of the main visions of Bluetooth is to network all devices in a small, limited area. The transmission speed varies from 108.8 Kbits/s up to 433.9 Kbits/s. The coverage area of Bluetooth is very limited and the cell radius is usually about 10 meters.

Each of the above wireless domains offers data transmission services to the Semantic Web. However, there are two significant issues that need to be taken into account before service deployment can really benefit the users. Firstly, wireless data communications is by its nature different compared to wireline data communications. In the environment of wireless data communications the Quality of Service (QoS) (such as line rate, delay, throughput, round-trip time, and error rate) may change dramatically when a user moves from one location to another. For example, when the user roams from a UMTS cell to a GPRS cell, the throughput may drop from 1 Mbits/s down to 24 Kbits/s. This highly variable environment of Web-based services creates a need for adaptability. Users will demand services that will automatically and transparently adjust to the changes mentioned above. Secondly, currently these three domains are very much isolated from each other, and there is little interoperability between them. Therefore, seamless roaming between these domains is not yet possible or it is awkward. However, it is foreseen that seamless roaming between different network technologies (e.g., between UMTS and WLAN) will be needed in the near future.

11.3 Software Agent Technology

The term agent in the context of software was first introduced in the mid-1950's at the Massachusetts Institute of Technology. Agent research can be split into two main phases: The first one began at the latter half of 1970's and the second one began around 1990. The roots of the first phase are mainly in distributed artificial intelligence (DAI). In the second phase the research on software agents is much broader studying different kinds of agents and agent systems starting from single, intelligent agents—e.g., BDI (Belief, Desire, and Intention) [43] agent—to multi-agent systems (MAS) consisting of numerous simple, co-operating agents [50]. Despite of the long research activity there is not a single, widely accepted definition of the term intelligent agent. For the purpose of this paper we can say that an intelligent agent is one that is capable of autonomous actions in order to achieve its goals, which are defined in their design objectives. This requires at least three

following things: 1) Reactivity: intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in their environment in order to achieve their goals; 2) Pro-activeness: intelligent agents are able to exhibit goal-directed behaviour by taking an initiative in order to achieve their goals; 3) Social ability: intelligent agents are capable to of interacting with other agents in order to achieve their goals. There are also other attributes that intelligent agents are mentioned to have. Temporal continuity means persistence of identity and state over long periods of time. Adaptability means capability to learn and improve with experience. Mobility means capability to migrate in a self-directed way from one host platform to another. To summarize, we can say that an intelligent agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future [32].

There are several application domains, where software agents are already been implemented and will be implemented in the future. Just to illustrate the wide scale of application domains, we list here a few of them. Industrial applications of agent technology were the first ones that were developed. Process control (e.g., ARCHON [38]), manufacturing (e.g., Holonic systems [37]), and air traffic control (e.g., OASIS [44]) are examples of agent based industrial applications. Software agent technology has also been implemented in several kinds of commercial applications, such as information management including information gathering and filtering, electronic commerce including electronic market places and auctions, and business process management [1]. We expect software agents to play a significant role in the Semantic Web.

Agent-to-agent communication is one of the key issues in the multi-agent systems. There are two approaches: either to standardize the communication between agents or to rely on propriety solutions. As the Semantic Web will be mostly based on standardized solutions (W3C and IETF), agent-to-agent communication should also be standardized. The Foundation for Intelligent Physical Agents (FIPA) [30] was formed in 1996 as a non-profit organisation with the remit of producing software standards for heterogeneous and interacting agents and agent-based systems across multiple vendors' platforms. This is expressed more formally in FIPA's official mission statement: "*The promotion of technologies and interoperability specifications that facilitate the end-to-end interworking of intelligent agent systems in modern commercial and industrial settings*". The emphasis here is on the practical commercial and industrial uses of agent systems. The aim is to bring together the latest advances in agent research with industry best practice in software, networks and business systems. FIPA has produced specifications for five categories: Applications, Abstract Architecture, Agent Communication, Agent Management and Agent Message Transport. The purpose of the FIPA Abstract Architecture [14] is to foster interoperability and reusability, and it leads

to the identification of architectural abstractions linked by their relationships. The FIPA Abstract Architecture specifies the elements that can easily be defined in an abstract manner, such as agent message transport, FIPA-ACL, directory services and content languages. The FIPA Agent Message Transport Specifications [25, 31, 15, 16, 17, 21, 20, 23, 24, 22] specifies the delivery and representation of agent messages over different network transport protocols, including both wireline and wireless data communications. A message consists of a message envelope and a message body at the message transport level. The envelope contains specific transport requirements and information. The message body is the payload and is usually expressed in FIPA-ACL [18, 26] and in a content language such as FIPA-SL [29]. The FIPA Agent Management Specification [19] specifies the framework for FIPA agents to exist and to operate. It specifies the logical reference model for the creation, registration, location, communication, migration and retirement of agents. The FIPA specifications for agent communication address the structure of agent interactions. The specifications comprise the communication language (FIPA-ACL), along with libraries of predefined communicative act types—such as INFORM and REQUEST—and interaction protocols [28], and content languages [27]. FIPA has also developed specifications for four agent-based applications: Personal Travel Assistance, Audio-Visual Entertainment and Broadcasting, Network Management and Provisioning, and Personal Assistant [13].

The Semantic Web utilising wireless data communications is an interesting domain for software agent technology. Software agents by being autonomous, goal oriented, proactive, and collaborative form a promising base to design and develop adaptive service environments. Autonomous working enables decision-making and actions based on decisions on required adaptation activities without end-users' intervention. Goal oriented working enables a fast and efficient selection of optimal adaptation mechanism. Proactive working enables to carry out adaptation activities in timely fashion—not too late or too earlier. Collaborative work enables to form a network of adaptation agents to provide efficient distributed information collection and information distribution. This domain has been addressed, for example, by FIPA and EU sponsored research projects CRUMPET [10, 42] and LEAP [40].

11.4 The Semantic Web

One of the biggest challenges in implementing intelligent agents in today's Web is that there are no tools that enable automatic reasoning about the data in the Web; in other words, there are not yet generally accepted methods to represent knowledge in the Web. Research in software agent technology and artificial intelligent has resulted in several methods to represent

knowledge [4]. However, these methods have not achieved general acceptance and wide-scale usage in the Web. The Semantic Web is an approach to be a widely accept method to enable reasoning and thus to enable the creation of intelligent Web applications. The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning [48]. Knowledge representation—structured collection of information and sets of rules to process the information—is one of the key issues when enabling reasoning about data. Some of the technologies for developing the Semantic Web are the following:

- Structure of data: XML [8], XSD [11, 6], XLS [2],
- Semantics of data: RDF [39] and RDF(S) [9],
- Ontologies: DAML+OIL [46] and OWL [49], and
- Protocols: SOAP [12].

In addition, there are requirements for proof and trust.

11.5 Research Challenges

Now, the coalition of wireless data communications, software agent technology, and the Semantic Web creates an interesting environment for pervasive/ubiquitous computing. There are several research challenges. Firstly, the knowledge representation in the Semantic Web is based on XML-based languages, and those are highly verbose. Therefore, the number of bits to be transmitted across wireless links will be significant affecting seriously to the transmission time. Thus, the challenge is to achieve bit-efficient transfer of knowledge; in other words, how to minimize the amount of bits transmitted across wireless links so that the knowledge is still valid after the transfer. There are following two challenges involved: The representation of knowledge should be such that the amount of bits is minimal, and interactions between peer agents should minimize the amount of bits transmitted across wireless links. Secondly, as wireless data communications act as a knowledge transfer service, about which software agents may need to reason, the service should be modelled so that reasoning can be well supported. Until now, we have thought data transmission as a single service, which does not have many alternatives, and therefore there is no need to reason about it. However, recent improvements and future developments in the wireless data communications will change this. In the Semantic Web agents need to reason about which is the best possible wireless transmission media for agent-to-agent knowledge exchange at each occasion, what can be transferred, and what is the representation of knowledge to be transferred. There are several attributes,

which may be used in the reasoning process: capabilities of mobile terminal, geographical location, available networks, throughput, delay, security, cost, etc. Thus, there is a following challenge involved: How to specify ontologies describing the domain of wireless data communications (the terms and relationships between the terms) so that the reasoning can be done in a just-on-time manner in the volatile environment of wireless data communications. In addition, we may need ontologies in the following domains: mobile terminals, geographical locations, and price.

FIPA has already addressed these issues to some extent in the context of FIPA Nomadic Application Support (NAS). The objective of FIPA NAS is to specify an agent-based framework, which supports the building of interoperable and adaptive nomadic applications.

11.5.1 Efficient Messaging

FIPA specifies a bit-efficient encoding [15] that reduces the number of bits of ACL messages between communicating peers. In the bit-efficient ACL, there are two primary ways to reduce the transfer volume over the wireless link: data reduction/compression and intelligent caching. The ACL message is encoded using a tokenised syntax, which itself is not a significant improvement compared to a simple string-based coding. The true power of the bit-efficient ACL lies in the intelligent caching, meaning that similar parts of subsequent messages are not transmitted multiple times over the communication path, as subsequent occurrences are replaced by short codes.

The communicative behaviours of interacting agents also affect the amount of bits transmitted over wireless links. Therefore, interactions between communicating agents need to be well designed in order to minimize the amount of bits to be transmitted over a wireless link. The challenge can be split into two domains: Firstly, when there is a common, often used interaction pattern, for example for electronic auctions, it is possible to specify a standardized interaction protocol. Now, the interaction protocol should be designed so that the number of round-trips is minimized. Secondly, when agents' interactions do not follow any common, often used pattern, agents themselves need to decide what to communicate, to whom, when, and how. There are many research challenges, such as how to evaluate the value of communication [34], how to obtain 'just-on-time' information about the QoS of wireless data communications, and how to make the decision process fast enough. These issues remain for future research.

11.5.2 Reasoning about Wireless Data Communications

In order to enable reasoning about wireless data communications in the Semantic Web, there must be ontology of the domain of wireless data communications, logical rules, and a framework that supports monitoring the services of wireless data communications. FIPA has addressed a couple of these issues in the FIPA NAS. The FIPA NAS framework comprises the following functions: monitoring the QoS of data transmission and controlling data transmission and data transmission equipment. In addition, FIPA specifies a nomadic application ontology that defines terms of QoS of wireless data communications and terms to access the services of monitoring and controlling wireless links. The purpose of the ontology is to provide agents and agent platforms with the means to communicate using both QoS terms and terms related to the communication channels and message transports.

11.6 Conclusions

We discussed the Semantic Web and software agent technology from the viewpoint of wireless data communications. We presented briefly the features of wireless data communications, software agent technology, and the Semantic Web. The high variability of QoS of wireless data communications creates a need for adaptability of the services of the Semantic Web. Software agent technology is seen as a good method to design and implement adaptable services. The Semantic Web will enable reasoning about information in the Web by giving information a well-defined meaning. Then we introduced several research challenges in this environment. We also briefly discussed about FIPA's NAS specifications, which have addressed some of the challenges.

Bibliography

[1] http://agents.umbc.edu/Applications_and_Software/index.shtml.

[2] S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, and S. Zilles. *Extensible Stylesheet Language (XSL) Version 1.0*. The World Wide Web Consortium, October 2001. W3C Recommendation.

[3] G. Anastasi and L. Lenzini. QoS provided by the IEEE 802.11 wireless LAN to advanced data applications: a simulation analysis. *Wireless Networks*, 6(2), 1999.

[4] ARPA Knowledge Sharing Effort. Knowledge sharing effort public library. http://www.ksl.stanford.edu/knowledge-sharing/index.html.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[6] P. V. Biron and A. Malhotra. *XML Schema Part 2: Datatypes*. The World Wide Web Consortium, 2001. W3C Proposed Recommendation.

[7] Bluetooth Special Interest Group. The Official Bluetooth SIG Website, 2001. http://www.bluetooth.com.

[8] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and Eve Maler, editors. *Extensible Markup Language (XML) 1.0 (Second Edition)*. The World Wide Web Consortium, October 2000. W3C Recommendation.

[9] D. Brickley and R. V. Guha. *Resource description framework (RDF) schema specification*. The World Wide Web Consortium, March 2000. W3C Candidate Recommendation.

[10] CRUMPET Project. CRUMPET Home Page, 2001. www.ist-crumpet.org.

[11] D. C. Fallside. *XML Schema Part 0: Primer*. The World Wide Web Consortium, 2001. W3C Proposed Recommendation.

[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol — HTTP/1.1. Request for Comments 2616, June 1999.

[13] Foundation for Intelligent Physical Agents. FIPA specifications, applications. http://www.fipa.org/repository/applicationspecs.html.

[14] Foundation for Intelligent Physical Agents. *FIPA Abstract Architecture Specification*. Geneva, Switzerland, November 2000. Specification number XC00001.

[15] Foundation for Intelligent Physical Agents. *FIPA ACL Message Representation in Bit-Efficient Specification*. Geneva, Switzerland, October 2000. Specification number XC00069.

[16] Foundation for Intelligent Physical Agents. *FIPA ACL Message Representation in String Specification*. Geneva, Switzerland, November 2000. Specification number XC00070.

[17] Foundation for Intelligent Physical Agents. *FIPA ACL Message Representation in XML Specification*. Geneva, Switzerland, October 2000. Specification number XC00071.

[18] Foundation for Intelligent Physical Agents. *FIPA ACL Message Structure Specification*. Geneva, Switzerland, October 2000. Specification number XC00061.

[19] Foundation for Intelligent Physical Agents. *FIPA Agent Management Specification*. Geneva, Switzerland, November 2000. Specification number XC00023.

[20] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification*. Geneva, Switzerland, November 2000. Specification number XC00088.

[21] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Envelope Representation in XML Specification*. Geneva, Switzerland, November 2000. Specification number XC00085.

[22] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Protocol for HTTP Specification*. Geneva, Switzerland, October 2000. Specification number XC00084.

[23] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Protocol for IIOP Specification*. Geneva, Switzerland, November 2000. Specification number XC00075.

[24] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Protocol for WAP Specification*. Geneva, Switzerland, October 2000. Specification number XC00076.

[25] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Service Specification*. Geneva, Switzerland, October 2000. Specification number XC00067.

[26] Foundation for Intelligent Physical Agents. *FIPA Communicative Act Library Specification*. Geneva, Switzerland, November 2000. Specification number XC00037.

[27] Foundation for Intelligent Physical Agents. *FIPA Content Languages Specification*. Geneva, Switzerland, October 2000. Specification number XC00007.

[28] Foundation for Intelligent Physical Agents. *FIPA Interaction Protocol Library Specification*. Geneva, Switzerland, October 2000. Specification number XC00025.

[29] Foundation for Intelligent Physical Agents. *FIPA SL Content Language Specification*. Geneva, Switzerland, November 2000. Specification number XC00008.

[30] Foundation for Intelligent Physical Agents. FIPA Home Page, 2001. http://www.fipa.org.

[31] Foundation for Intelligent Physical Agents. *FIPA Messaging Interoperability Service Specification*. Geneva, Switzerland, October 2001. Specification number PC00093.

[32] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III*, volume 1193 of *LNAI*, pages 21–36. Springer-Verlag: Heidelberg, Germany, August 1997.

[33] L. Gerber. Will 3G really be the next big wireless technology? *IEEE Computer*, January 2002.

[34] P. Gmytrasiewicz and E. Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, (4):233–272, 2001.

[35] GSM Technical Specification, GSM 03.60. Digital cellular telecommunications system (phase 2+), general packet radio service (GPRS) service description, stage 2, 2000. Version 7.4.0.

[36] GSM Technical Specification, GSM 03.64. Digital cellular telecommunications system (phase 2+), overall description of the GPRS radio interface, stage 2, 2000. Version 7.1.0.

[37] Holonic Manufacturing Systems Consortium. Holonic Manufacturing Systems home page, 2002. http://hms.ncms.org/.

[38] N. R. Jennings, J. M. Corera, and I. Laresgoiti. Developing industrial multi-agent systems. In *First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 423–430, San Francisco, CA, USA, June 1995. AAAI Press.

[39] Ora Lassila and Ralph R. Swick, editors. *Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium, February 1999. W3C Recommendation.

[40] LEAP Project. LEAP home page. http://leap.crm-paris.com/.

[41] N. Mitra, editor. *SOAP Version 1.2 Part 0: Primer*. The World Wide Web Consortium, December 2001. W3C Working Draft.

[42] S. Poslad, H. Laamanen, R. Malaka, A. Nick, P. Buckle, and A. Zipf. CRUMPET: Creation of user-friendly mobile services personalised for tourism. In *IEE 3G 2001 conference*, London, UK, March 2001.

[43] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, April 1991.

[44] A. S. Rao and M. P. Georgeff. BDI agents: from theory to practice. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 312–319, San Francisco, CA, USA, 1995. The MIT Press.

[45] J. Thackara. The design challenge of pervasive computing. *Interactions*, 8(3), 2001.

[46] The DARPA Agent Markup Language Program. DAML+OIL specification, March 2001. http://www.daml.org/2001/03/daml+oil-index.html.

[47] Third generation partnership project web site, 2002. http://www.3gpp.org.

[48] W3C. W3C semantic web activity. http://www.w3.org/2001/sw/.

[49] W3C. W3C Web-Ontology (WebOnt) Working Group. http://www.w3.org/2001/sw/WebOnt/.

[50] G. Weiss, editor. *Multiagent Systems — A Modern Approach to Distributed Artifical Intelligence*. The MIT Press, Cambridge, Massachusetts, 1999.

Chapter 12

Serendipitous Interoperability

Ora Lassila

This article will discuss issues of interoperability of Web Services. Semantic Web technologies, when applied to current Web Service architectures, will enable true automation and interoperation, and will "futureproof" systems unlike a priori standardization approaches. We will extend the Semantic Web approach to Web Services to Ubiquitous Computing: By abstracting device functionality via software agents, and using Semantic Web technologies to describe agent services and capabilities, we can achieve a rich and deep form of service discovery. By discovering partially matching services, and piecing these together into "virtual value chains", we can automatically form device coalitions which operate within a dynamically changing environment.

12.1 Introduction

"Web Services" – functionality that can be invoked remotely over the Web – is a strong recent trend in the development of World Wide Web -based systems. Several industry standards have emerged in this area, including SOAP [6], an invocation protocol, and WSDL [10], a formalism for describing the interfaces of Web Services. In addition, mechanisms have been proposed to facilitate the discovery of Web Services – one of these is UDDI [3]. All these specifications are being offered with the promise of greater opportunity for automation of tasks and improved interoperability of information systems.

In this paper we will argue that albeit we see progress in the right direction, these attempts will fall short of the goals of improved automation and interoperability, because they are based on heavy *a priori* standardization and they ultimately retain humans in the loop. The maintenance and management of all the emerging vocabularies will result in a phenomenon we can

only compare to the biblical story of the "Tower of Babble" [1]. We believe that our true goal should be "serendipitous interoperability", the ability of software systems to discover and utilize services they have not seen before, and that were not considered when the systems were designed. To realize this, qualitatively stronger means of representing the service semantics are required, enabling fully automated discovery and invocation, and complete removal of unnecessary interaction with human users.

The Semantic Web [5] offers means of advancing beyond the current proposed architectures for Web Services. Characterized by the exposure of declarative formal semantics of information and services, the Semantic Web allows information systems to *reason* about data sources and the functionality of other systems, and consequently allows them to better take advantage of these. The Semantic Web will also enable – finally – the emergence of *intelligent agents*, systems based on autonomously operating goal-oriented software entities.

In the context of Web Services, the application of the Semantic Web to representing information and its semantics will enable the following:

- Description of semantics of services to allow their automatic discovery, even if the services offered only partially match the needs of the requester.
- Automatic composition of multiple services – possibly partially matching ones – into a "super-service" satisfying the needs of the requester (whether the requester be a human or an artificial agent).

12.2 About Representation and Ontologies

The *ontological approach* characteristic of the Semantic Web is predicated on the existence and use of *ontologies*: documents or files that formally define the relationships between terms for any particular domain of discourse – a widely cited definition of an ontology is Gruber's "a specification of a conceptualization" [11], and in that sense we depart from the abstract philosophical notion of ontology defined as "a branch of metaphysics concerned with the nature and relations of being" [2].

People, as well as artificial agents, typically have a notion or conceptualization of the meaning of terms. Just as the specification inputs and outputs of a software program could be used as a specification of the program itself, ontologies can be used to provide a concrete specification of term names and meanings. If we consider ontologies as specifications of the conceptualizations of terms, there is much room for variation, and the spectrum of Web ontologies typically range from simple controlled vocabularies through informal concept hierarchies to something where arbitrarily complex logical

relationships can be specified between defined concepts. In practical terms, we would expect the following properties to hold in order to consider something an ontology [18]:

1. Finite controlled (extensible) vocabulary
2. Unambiguous interpretation of classes and term relationships
3. Strict hierarchical subclass relationships between classes

The following properties for ontologies are typical but not mandatory:

4. Property specification on a per-class basis
5. Inclusion of individuals (i.e., instances) in the ontology
6. Value restriction specification on a per-class basis

12.2.1 Representing Semantics of Web Services

In the context and environment of the World Wide Web, "Web-friendly" knowledge representation formalisms DAML+OIL[1] [12] and its foundation, W3C's RDF [15, 17, 8] will be used[2] to describe services for the purposes of discovery. Just as the success of the deployment of the Semantic Web will largely depend on whether useful ontologies will emerge, so will discovery services benefit from mechanisms that allow shared agreements about vocabularies for knowledge representation. Sharing vocabularies allows *automated* interoperability; given a base ontology shared by agents, each agent can extend this ontology while achieving partial understanding of the others; this is analogous to OOP systems, where a base class defines "common" functionality.

One attempt to represent service semantics is DAML-S [4, 9], a Web Service ontology expressed in DAML+OIL. It gives Web Service providers a core set of markup language constructs for describing the properties and capabilities of their Web Services in an unambiguous, computer-interpretable form. DAML-S markup of Web Services will facilitate the automation of the following Web Service tasks:

1. **Automatic discovery** involves the automatic location of services that provide a particular function and adhere to requested constraints.

[1] http://www.daml.org/2001/03/daml+oil-index.html

[2] DAML+OIL will be succeeded by the emerging Web ontology language OWL, see http://www.w3.org/2001/sw/WebOnt/webont.

2. **Automatic invocation** involves the execution of a discovered service by a computer program or agent. Execution of a service can be thought of as a collection of function calls. DAML-S markup of Web Services provides a declarative, computer-interpretable interface for executing these function calls. A software agent should be able to interpret the markup to understand what input is necessary to the service call, what information will be returned, and how to execute the service automatically.

3. **Automatic composition and interoperation** involves the automatic construction of a plan to use a number of services to perform some task, given a high-level description of an objective. With DAML-S markup of Web Services, the information necessary to select and compose services will be encoded at the service Web sites. Software can be written to manipulate these representations, together with a specification of the objectives of the task, to achieve the task automatically.

4. **Automatic execution monitoring**: Individual services and, especially,compositions of services, will often require some time to execute completely. Users (human or artificial ones) may want to know during this period what the status of their request is. Their plans may also have changed requiring alterations in the actions the service provider takes.

In comparison with *service profiles* – the DAML-S characterization of services – the other industry standards are limited, first and foremost, in that they cannot express logical statements. Input and output types are supported to varying extents. From the DAML-S standpoint, services can be simple (or primitive); they invoke only a single Web-accessible computer program that does not rely upon another Web Service, and there is no ongoing interaction between the user and the service, beyond a simple response. Alternately, services can be complex, composed of multiple primitive services, often requiring an interaction or dialogue between the consumer and the provider of the services, so that choices can be made or information provided conditionally. DAML-S is meant to support both categories of services, but complex services have provided the primary motivation for the features of the language.

DAML-S provides an upper ontology for services, including the properties normally associated with all kinds of services. The upper ontology does not address what the particular subclasses of the base service class should be, or even the conceptual basis for structuring this taxonomy – this may take place according to functional and domain differences, or market needs. A service profile provides a high-level description of a service and its provider,

and is used as a request, or as an advertisement, within a service discovery process.

Service profiles consist of three types of information: a human readable description of the service, a specification of the functionalities that are provided by the service (represented as a transformation from the inputs required by the service to the outputs produced), as well as a host of functional attributes which provide additional information and requirements about the service that assist when reasoning about several services with similar capabilities (these include guarantees of response time or accuracy, as well as the cost of the service). Furthermore, a more detailed perspective on services is to view them as processes. DAML-S representation of processes draws upon established work in a variety of fields, such as automated planning and workflow automation, and will support the representational needs of a very broad array of services on the Web.

12.3 Semantic Web Meets Ubiquitous Computing

Ubiquitous Computing is an emerging paradigm of personal computing, characterized by the shift from dedicated computing machinery (that requires the user's attention – e.g., PCs) to pervasive computing capabilities embedded in our everyday environments [21, 22]. Characteristic to Ubiquitous Computing are small, handheld, wireless computing devices. The pervasiveness and the wireless nature of devices require network architectures to support automatic, *ad hoc* configuration [13].

A key functionality of true *ad hoc* networks is *service discovery*, by which functions offered by various devices on the network can be described, advertised, and discovered by others. Several frameworks and formalisms for this type of service discovery and capability description have already emerged – examples include Sun's "Jini"[3] and Microsoft's Universal Plug and Play (UPnP)[4] [19] as means of describing services and invoking them, as well as World Wide Web Consortium's (W3C) Composite Capability/Preference Profile (CC/PP) [14] as a means of describing device characteristics. The current service discovery mechanisms are based on *ad hoc* representation schemes and rely heavily on standardization (i.e., on *a priori* identification of all those things one would want to communicate or discuss). "Jini" also relies on the Java object system and instance serialization, and UPnP uses its own flavor of HTTP.

Our goal is to represent the functionality of Ubiquitous Computing devices as services in a multiagent framework, and apply the Semantic Web

[3] http://www.sun.com/jini/

[4] http://www.upnp.org/

-based Web Service techniques to facilitate the interoperation of these devices – more specifically, we aim at enabling the discovery and utilization of services by other agents without human guidance or intervention, thus enabling the automatic formation of device coalitions through this mechanism. We call these devices, capable of semantic discovery and coalition formation, *semantic gadgets* [16, 5].

12.3.1 Semantic Discovery

Semantic gadget technology begins with the discovery of functionality and services. As mentioned before, a number of mechanisms for low-level service discovery have emerged; these mechanisms attack the problem at a syntactic level, and rely heavily on the standardization of a predetermined set of functionality descriptions. Standardization, unfortunately, can only take us halfway toward our goal of intelligent automated behavior *vis-a-vis* discovery, as our ability to anticipate all possible future needs is limited. By elevating the mechanisms of service discovery to a "semantic" level, a more sophisticated description of functionality is possible, and the shared understanding between the consumer and the provider can be reached via the exchange of ontologies which provide the necessary vocabulary for a dialogue.

Semantic discovery mechanisms will undoubtedly be layered on top of existing, lower-level services. These services involve *ad hoc* networking technologies and other mechanisms that are beyond the scope of this article. In our approach, physical devices and their functionality will be abstracted as software agents. These agents will advertise services and/or will query for services they need. Both the advertisements and the queries will be abstract descriptions of functionality – in fact, there is no difference between the two, as Sycara has pointed out [20]. Through a matchmaking process (either by the provider, by the consumer, or by a third party "matchmaker") we are able to associate compatible advertisements with queries. The match might be perfect – in which case a service will exactly meet the need of the consumer – or partial – in which case the consumer might have to combine the service with some additional functionality (it can do this by itself, or – as we will demonstrate later – continue to discover the "missing pieces" and finally compose a service that meets its need).

The Semantic Web plays two key roles in the discovery process. First, Semantic Web techniques provide a rich mechanism for describing functionality: ontologies will describe the concepts and vocabulary needed to discuss functionality and services. Second, the Semantic Web provides a unifying layer of naming and distribution, an addressing mechanism that can encompass virtual and physical entities, and a way for various pieces of the "puzzle" to reside on various servers, devices etc. For example, a device description can refer to an ontology elsewhere, which in turn can be a specialization or

extension of another ontology, again somewhere else. The polymorphic nature of ontology extension will allow broader interoperation through partial understanding and agreement.

12.3.2 Services and Contracting

Once the services we want to use have been discovered and identified, there are several rather "bureaucratic" issues to be dealt with, related to "contracting" the use of the services. These include (but are not limited to):

- Assuring security, privacy, and trust
- Compensating the service provider
- Determining execution locus

The Semantic Web promises to be useful when it comes to matters of security, privacy and trust, as these are largely issues of representation. Generally, the Semantic Web rests heavily on a framework of trust partially constructed using digital signatures and other types of cryptographic certificates. Any agent can expose its reasoning about trust using the same mechanisms by which everything else is represented on the Semantic Web. These representations of reasoning – we might call them "proofs" – can themselves be exchanged between agents as persuasive communication. We anticipate the emergence of services specific to assisting agents with their security, privacy, and trust issues (e.g., there might be a service that rates the "reputation" of other services: reliability, trustworthiness, or some other relevant metric). Again, these services themselves can be discovered and their use contracted.

Given a functional security and trust framework, we can introduce the notion of payments. This is required for the purpose of compensating providers for services rendered. Generally, we anticipate third-party services (which, again, are discoverable) to facilitate payments or other types of compensation. We are not trying to imply that everything on the Semantic Web will cost something, but some services will emerge that are not free. Advertising (of products and services for humans this time) will no longer be a viable revenue model once automated agents take care of a large part of the information exchange, reasoning and service utilization. Furthermore, some type of compensation mechanism might be used to provide stability to the operation of a system of self-interested agents [7].

12.3.3 Composition of Services

The discovery of services based on some description of a requirement of new functionality might result in a partial match [20]. In this case the requester can attempt to provide the missing parts itself or continue the discovery

process and identify other services. They can then be pieced together to form an aggregate service that fulfills the original requirement.

Composition of exact required functionality from partially matching services should be viewed as a process of goal-driven assembly, achievable by the use of automated planning and/or configuration techniques. In the context of Ubiquitous Computing and semantic gadgets, contracting the use of services should always be viewed as a goal-driven activity because of the "volatile" nature of Ubiquitous Computing environments (not only can any device or service fail or be removed from the environment at any time, but new ones can be added to it; opportunistic exploitation of services might thus be beneficial).

The goal-driven approach takes information system interoperability beyond what mere standardization or simple interface sharing enables, since it is based on "deeper" descriptions of service functionality and can be performed *ad hoc* and on demand. In fact, the dynamically composed aggregate services are the Semantic Web's virtual equivalent of "real-life" value chains: large quantities of information (i.e., "raw material") may be obtained, and at each step the value of information increases and its volume decreases [5].

Given that the services discovered represent actual physical functionality of devices, aggregate services could be seen as device coalitions. Not only can individual devices extend their functionality, but the device coalitions effectively form task-specific "super-devices".

12.4 Conclusions

We have presented how Semantic Web technologies can be used in the context of Web Services to ensure "serendipitous" forms of interoperation. The Semantic Web encompasses efforts to populate the Web with content that has formal semantics; thus the Semantic Web will enable automated agents to reason about Web content, and produce an intelligent response to unforeseen situations. Our vision is to overlay the Semantic Web on a Ubiquitous Computing environment, making it possible to represent and interlink devices, their capabilities, and the functionality they offer. By abstracting device functionality as software agents, and then using Semantic Web technologies todescribe agent services, we can build a "semantic" discovery service capable of function beyond *a priori* standardization. Through the composition of discovered, partially matching services into virtual value chains we are able to form device coalitions which opportunistically exploit a dynamically changing Ubiquitous Computing environment.

The interoperability of physical devices is an objective worth pursuing: it is unlikely that a single company will be engaged in the manufacture of all the different types of devices that will make up the "web of semantic gad-

gets" (for example, Nokia manufactures wireless communication devices but it does not make thermostats; yet consumers might benefit from wireless devices that communicates with thermostats). In fact, applying Semantic Web technologies to Web Services in the Ubiquitous Computing context may have more compelling business models than the application of the same technology elsewhere, since the device manufacturers are not primarily in the Semantic Web business.

Acknowledgements

The author owes deep gratitude to the following individuals for discussions related to this article and its topics: Mark Adler (Nokia Research Center), Michael Mahan (Nokia Research Center), Deborah McGuinness (Stanford University), Katia Sycara (Carnegie Mellon University), the entire DAML-S coalition, and the ATMOS and PHEAR project teams of the Agent Technology Group at the Nokia Research Center. The research described in this paper was supported in part by Nokia Research Center, Nokia Ventures Organization, Nokia Mobile Phones, and Nokia Venture Partners.

Bibliography

[1] The dispersion of the nations at babel. *Genesis*, 11:1–9.

[2] *Merriam-Webster's Collegiate Dictionary*. Merriam-Webster, 1998.

[3] UDDI technical white paper. UDDI.org, September 2000. http://www.uddi.org/whitepapers.html.

[4] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, S. A. McIllraith, S. Narayanan, M. Paolucci, T. Payne, T. C. Son, K. Sycara, and H. Zeng. DAML-S: A semantic markup language for web services. In *Proceedings of the First Semantic Web Working Symposium (SWWS'01), Stanford University*, July 2001.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[6] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1, May 2000. W3C Note, World Wide Web Consortium, Cambridge, Massachusetts, available as http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

[7] S. Brainov and T. Sandholm. Power, dependence and stability in multiagent plans. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), Orlando, Florida*. AAAI Press, 1999.

[8] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[9] M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, S. A. McIllraith, S. Narayanan, M. Paolucci, T. Payne, T. C. Son, K. Sycara, and H. Zeng. DAML-S 0.6 draft release, draft document of the DARPA Agent Markup Language Program, 2001. http://www.daml.org/services/daml-s/2001/06/.

[10] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. W3C Note, World Wide Web Consortium, Cambridge, Massachusetts, March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[11] T. R. Gruber. A translation approach to portable ontology spesifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[12] J. Hendler and D. L. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent Systems*, 15(6):67–73, November/December 2000.

[13] C. Huitema. *Plug and Play, in IPv6: the New Internet Protocol*. Prentice-Hall, Upper Saddle River, New Jersey, 1998.

[14] G. Klyne, F. Reynolds, C. Woodrow, and H. Ohto. Composite capability/preference profiles (CC/PP): Structure and vocabularies. W3C Working Draft 15 March 2001, World Wide Web Consortium, Cambridge, Massachusetts. http://www.w3.org/TR/CCPP-struct-vocab/.

[15] O. Lassila. Web metadata: A matter of semantics. *IEEE Internet Computing*, 2(4):30–37, 1998.

[16] O. Lassila and M. Adler. Semantic gadgets – ubiquitous computing meets the Semantic Web. In D. Fensel et al., editor, *Spinning the Semantic Web*. The MIT Press, Cambridge, Massachusetts, to appear.

[17] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[18] O. Lassila and D. L. McGuinness. The role of frame-based representation on the Semantic Web. *Electronic Transactions on AI*, 2002 (to appear). Available as a Stanford KSL technical report KSL-01-02.

[19] G. G. Richard. Service advertisement and discovery: Enabling universal device cooperation. *IEEE Internet Computing*, 4(5):18–26, September/October 2000.

[20] K. Sycara, M. Klusch, J. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *ACM SIGMOD Record*, 28(1):47–53, March 1999.

[21] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, September 1991.

[22] M. Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, July 1993.

Chapter 13

Semantic Information Router (SIR)

Janne Saarela

Semantic Content Management is an emerging approach to Content Management. The use of semantic technologies such as the Resource Description Framework (RDF) to content management enables unambiguous processing of different types of file formats. In addition, these technologies enable a distributed management framework where the content to be managed need not be replicated to a central storage but only the descriptive data. This paper presents Profium SIR product [7] as a case example of semantic content management. The paper outlines the benefits of the approach and some of the technical issues related to its implementation.

13.1 Introduction

Content management marketplace is full of offerings ranging from highly sophisticated document management systems to Web content management systems. These systems provide support for managing the content value chain from authoring of content, to its management in a workflow and finally its delivery to actual presentation whether on paper or networked media such as the World Wide Web.

The richness of content types and their re-use in these publishing processes is often limited to the use of keywords or simple classification schemes. The quality of content management processes can often be improved by requiring as little human intervention as possible by letting the software do most of the processing. This approach, however, requires the computers are able to unambiguously determine whether an image is e.g. about a company

called Nokia, authored by a company called Nokia or about a city called Nokia.

13.2 Semantic Content Management

Semantic technologies such as Resource Description Framework (RDF) [4] from the World Wide Web Consortium (W3C) address the problem of resource description. RDF thus provides a sound technology for building semantic content management solutions where the processing of content can be made dependent on its semantic descriptions.

13.2.1 Resource Description Framework

Resource Description Framework (RDF) technology has been developed at the World Wide Web Consortium in 1998 and it was finalised in early 1999. RDF enables a metadata infrastructure through a data model which is a directed labelled graph. This graph enables the description of semantics about objects that can be addressed with a Uniform Resource Identifier (URI). The data model level specification is associated with RDF Schemas specification [2] that enables the construction of a type system which can be used to validate RDF data models. In addition, RDF Schemas enable the description of natural language semantics about the formal properties of the data model. At the writing of this paper RDF Schemas have not yet reached the final specification status at the W3C but require more practical feedback and coordination with Web Ontologies work recently started at the W3C.

13.3 Semantic Information Router

Profium Ltd. is a Finnish-French software company who develops semantic content management solutions. In April 2001, Profium launched the version 2.0 of its Semantic Information Router (SIR) product.

Profium SIR provides native support for RDF data model. In practice SIR stores RDF data model in a relational database and provides its own query language, RDFQL, to query the directed graph. RDFQL hides away the relational schema used by SIR internally and allows application developers take advantage of the expressive power of RDF. There's previous history of RDF querying research presented e.g. by [8].

Profium SIR can be configured to support multiple metadata vocabularies i.e. schemas. SIR configuration file can be set to point to RDF schema files that determine what properties and classes are applicable for building semantic content management solutions. Due to the simplicity of RDF schemas,

Profium SIR extends the expressivity of RDF schemas specification by the following features:

1. Basic data types for literal nodes – RDF schemas specification makes a forward reference to XML Schemas [9, 6] work that was finalized in April 2001. The binding of literal nodes to basic data types is not yet supported in the specifications but Profium SIR supports some data types including

   (a) Unicode strings

   (b) integer numbers

   (c) double precision real numbers

   (d) boolean values

   (e) URI references

   (f) timestamps that conform to ISO 8601

   (g) enumerated lists

2. Arity constraints – DAML and OIL specifications provide support for arity constraints while RDF schemas has left them out. Profium SIR includes support using minOccurs and maxOccurs constructs similar to XML Schemas.

These additional features can be embedded to the RDF schema document using a namespace specific to Profium SIR.

13.3.1 Conceptual Operation

Profium SIR is a best-of-breed component implemented using Java programming language. Profium SIR is typically combined with other best-of-breed components into a complete semantic content management solution.

Profium SIR provides rich APIs for inputting, outputting and querying the metadata. In fact, the query interface can be used in three different ways:

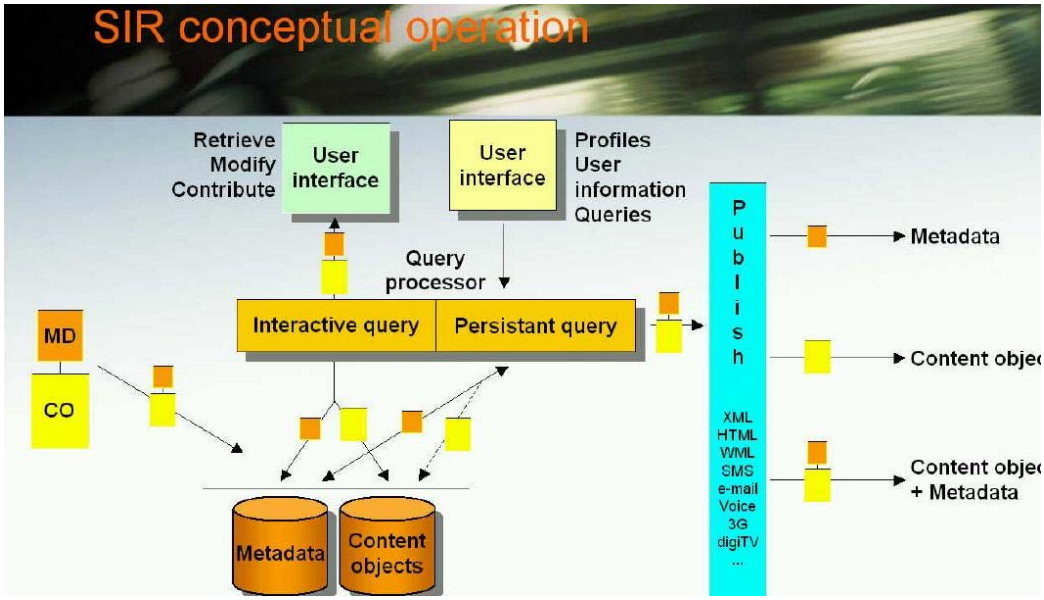
1. interactive queries – Profium SIR can be used as the query evaluator that receives queries activated by user's active interaction and returns result set in RDF. The user application can then decide how to present the result set to the user.

2. persistent queries – Profium SIR can be used to store queries with functionalities. These queries remain in the system independently of user's interaction are thus considered persistent. All incoming metadata is automatically evaluated against all persistent queries of the Profium

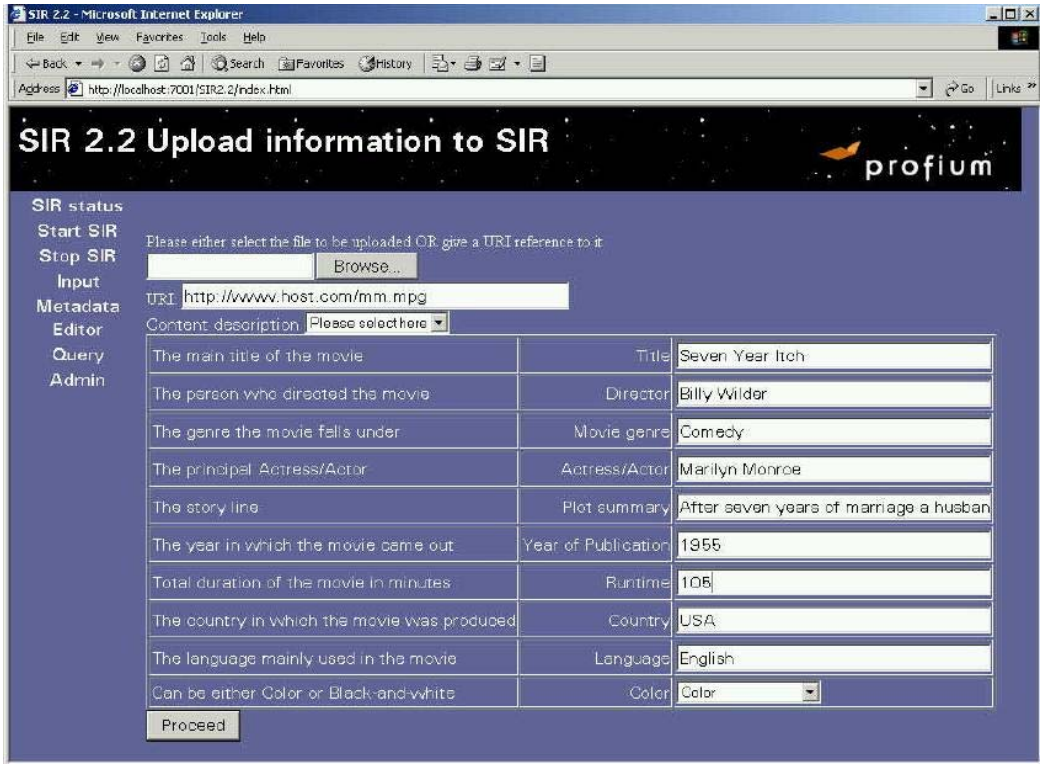
SIR installation and should any of the queries match, the corresponding functionality such as Web page publishing or email or SMS/GSM delivery can be activated with the query result set as a parameter. This feature accounts most to the Router word in the product's name as Profium SIR can be configured to automate content management workflows.

3. timed queries – Timed queries extend the persistent queries by associating a query schedule with each query. A query can be configured to run e.g. every hour, every Friday or every 1st day of the month. Again, if the query has a non-empty result set, the corresponding functionality will be activated.

Profium SIR installation is typically configured with one or more adapters that conform different types of metadata schemes to the one of RDF. Profium SIR can, for example, be configured to map document properties of Word, Excel, and PowerPoint files to RDF. The construction of adapters is, of course, dependent on the availability of metadata in the input files. Whereas structured content such as Reuters or Dow Jones news in XML format is easy to map to RDF, digital images provide little metadata for SIR to take advantage of.



In case metadata is completely missing, Profium SIR does provide highly adaptive user interfaces that assist the users to enter metadata. In fact, these user interfaces are completely adaptive to the configured RDF Schemas of the Profium SIR installation.



13.4 Market Indicators

Adobe announced at the Seybold conference in September 2001 the support for Extensible Metadata Platform (XMP) technology [1]. XMP allows embedding of document metadata inside application files using RDF technology. XMP is supported in Adobe's products such as Acrobat 5.0, InDesign 2.0, and Illustrator 10.

XMP technology is not only a read-only technology. This means 3rd party software vendors can change the XMP metadata inside application files in addition to reading and understanding these descriptions in an unambiguous way.

Other initiatives like the Publishing Requirements for Industry Specific Metadata (PRISM) [5] and Dublin Core [3] present shared vocabularies for users of metadata. The joint development of such standards assists metadata adopters to have quicker implementation times as consensus development may not be necessary.

13.5 Summary

In this paper we have briefly presented the need for semantic content management. We have presented RDF technology and its implementation in

a commercial software product, Profium SIR. Profium SIR provides native support for RDF technology and introduces a new query language RDFQL for querying RDF data.

Bibliography

[1] Adobe. *Extensible Metadata Platform (XMP). http://www.adobe.com/products/xmp/main.html*, 2001.

[2] D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[3] Dublin Core Initiative. *Dublin Core. http://www.dublincore.org.*

[4] O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.

[5] OASIS. *Publishing Requirements for Industry Specific Metadata (PRISM). http://www.prismstandard.org*, 2001.

[6] Ashok Malhotra Paul V. Biron. *XML Schemas: Data Types, W3C Recommendation*, 2001. http://www.w3.org/TR/xmlschema-2.

[7] Profium. *Profium SIR datasheet. http://www.profium.com/gb/products/sir*, 2001.

[8] Janne Saarela. *The role of metadata in electronic publishing.* PhD thesis, Helsinki University of Technology, 1999.

[9] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. *XML Schemas: Structures, W3C Recommendation*, 2001. http://www.w3.org/TR/xmlschema-1.

Chapter 14

Semantics of Mathematics on the Web

Mika Seppälä and Jouko Väänänen

14.1 Introduction

Almost any scientific information contains mathematical formulae. For many disciplines the mathematics that is needed is rather limited, but even areas like law need to be able to speak of lengths, areas, weights and volumes. Hence they need to be able to express information involving mathematical concepts. Sometimes these concepts have to be translated into equivalent other concepts, for example US units, inches, feet, miles, pounds and gallons, into equivalent metric concepts, or vice versa. This is a task that could well be done automatically given all the computing power that we have in the Internet.

But even such a modest task is not being done automatically. Manual conversions are used, and sometimes this may cause troubles if not done diligently. CNN's News item entitled "NASA's metric confusion caused Mars orbiter loss" (September 30, 1999) reported about the accident in which NASA lost a $ 125 million Mars orbiter. According to this CNN's news item:

> "... the spacecraft was lost because one engineering = team used metric units while another used English units for a key spacecraft operation. For that reason, information failed to transfer between the Mars Climate Orbiter spacecraft team at Lockheed Martin in Colorado and the mission navigation team in California. Lockheed Martin built the spacecraft."

For a layman this appears to be a trivial error in such a complex under-

taking. Had the conversion of the units taken place automatically one could have avoided this error.

The Ariane 5 failure in 1996 was another example of a very costly mistake that could have been avoided provided that the conversion of mathematical objects had been done correctly. The destruction of Ariane 5 in June 1996 was caused by a failure to convert a 64-bit floating point number to a 16-bit signed integer. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. This caused an operand error in the navigation system of the rocket, and hence initiated self-destruction of the spacecraft. For details see [1].

These examples underline the need to embed mathematical formulae into documents and programs so that the correct meaning of the object can be automatically understood.

Mathematics is unique among areas of knowledge in that the semantics of a correctly rendered electronically communicated expression can be understood, not only by another user, but in a sense even by the computer itself. The mere fact that computers are able to perform numerical calculations which were earlier totally beyond the reach of human mathematicians, has changed dramatically both theoretical mathematics and applications of mathematics in natural sciences, medical science, = etc. But this is only part of the story. The modern mathematical programs (like Maple and Mathematica) are about as good in many symbolic computations as human mathematicians. This is true, for example, of symbolic integration. This means, that scientists can really meaningfully electronically interact on the mathematical = level, not only with each other, but also with computers. Another consequence = is the possibility to search the Internet on the basis of mathematical content. For example, semantic search concerning the equation $x^n + y^n = 3Dz^n$ finds also the equation $u^k + v^k = 3Dw^k$, and searching for $\int e^{-x^2} dx$ finds also $\int e^{-y^2} dy$.

14.2 Languages for Mathematical Information

14.2.1 MathML

The Mathematics Markup Language, MathML, of the W3 Consortium allows the visual representation of mathematical formulae on the web. MathML can also express the content of simple formulae contained typically in the K12 (or, after the last revisions, perhaps K14) curriculum.

MathML has two components: the presentation MathML that allows one to write mathematical formulae that can be rendered correctly in the web browsers, and the content MathML that can be used to understand the

meaning of simple formulae automatically.

Presentation MathML is a slight improvement to expressing mathematical information as gif pictures. The improvement is in the fact that formulae behave nicely when increasing or decreasing the font sizes, or when the window in which the formulae are being viewed, is resized. Presentation MathML also results more satisfying printed documents.

One can argue that presentation MathML is not very useful, since other solutions, like pdf files, can be used to produce a similar result: high quality screen documents that yield also high quality printed versions. However, already the presentation MathML provides a platform for later addition of content tags.

The power of MathML lies also in the fact that mathematical software packages, like Maple or Mathematica, can now be used to *import* and to *export* MathML encoded formulae automatically. Thus these powerful mathematical programs can automatically interact with each other without any human intervention. The emerging *mathematical broker systems* automatically search for the best platform for the solution of a given mathematical problem.

14.2.2 OpenMath

OpenMath extends the power of content MathML so that, in principle, any scientific information can be embedded in documents in such a way that the meaning of the formulae can be automatically understood. This has been achieved through extensible Content Dictionaries that are used to define the meaning of formulae. These CD's are publicly available at the OpenMath web site (www.openmath.org, [5]).

MathML can be viewed as OpenMath Lite, a simplified fragment of OpenMath. The capabilities of MathML to define semantics are limited, but MathML certainly suffices in situations like the formula $\sin(x^2+1)$ discussed below.

14.2.3 Examples of Mathematics Embedded in Documents

This article has been typeset with LaTeX. Mathematical formulae are embedded in the document using the typesetting commands provided by LaTeX. These typesetting commands will produce very high quality printed documents. The typesetting commands do, however, not contain any semantic information about the formula in question. The mathematical expression $\sin(x^2+1)$ is embedded in the document as `$\sin(x^2+1)$`. These commands define the visual presentation of this mathematical expression, but a human is needed in order to correctly interpret its meaning.

The above mentioned languages OpenMath and MathML will provide means to attach semantics to formulae so that this semantics can be automatically understood.

The following is self–explanatory.

Presentation MathML Encoding of the Expression $\sin(x^2+1)$

```
<math xmlns=3D'http://www.w3.org/1998/Math/MathML'>
<mrow>
    <mi>sin</mi>
    <mo>&ApplyFunction;</mo>
    <mfenced>
    <mrow>
        <msup>
            <mi>x</mi><mn>2</mn>
        </msup>
        <mo>+</mo>
        <mn>1</mn>
    </mrow>
    </mfenced>
</mrow>
</math>
```

Content MathML Encoding of the Expression $\sin(x^2+1)$

```
<math xmlns=3D'http://www.w3.org/1998/Math/MathML'>
    <apply id=3D'id7'><sin id=3D'id1'/>
        <apply id=3D'id6'><plus/>
            <apply id=3D'id4'> <power/>
            <ci id=3D'id2'>x</ci>
            <cn id=3D'id3' type=3D'integer'>2</cn>
            </apply>
        <cn id=3D'id5' type=3D'integer'>1</cn>
        </apply>
    </apply>
    </math>
```

OpenMath Encoding of the Expression $\sin(x^2+1)$

```
<OMOBJ>
    <OMA>
        <OMS cd=3D"transc1" name=3D"sin"/>
        <OMATTR>
        <OMATP>
```

```
            <OMS cd=3D"presentation" name=3D"left"/>
                <OMSTR>(</OMSTR>
            <OMS cd=3D"presentation" name=3D"right"/>
                <OMSTR>)</OMSTR>
        </OMATP>
        <OMA>
        <OMS cd =3D"arith1" name=3D"plus"/>
            <OMA> <OMS cd=3D"arith1" name=3D"power"/>
            <OMV name=3D"x"/>
            <OMI>2</OMI>
        </OMA>
        <OMI>1</OMI>
        </OMA>
        </OMATTR>
        </OMA>
</OMOBJ>
```

Comments

Presentation MathML does not define any semantic information of the mathematics it is displaying. That can be done by Content MathML. The above example shows how content MathML embeds the formula in question in an XML document. Several commercial products that allow the creation, editing and rendering of MathML (both Presentation and Content) exists already now (March 2002). Hence the inclusion of mathematical formulae into XML documents with semantics that can automatically understood is already possible. This will make XML preferred mode for scientific publishing very soon.

The semantics supported by MathML is limited, but that can partly be circumvented by the use of annotations which allow one to add a program specific representation of a mathematical object as an annotation to a MathML encoding of a formula.

A better solution has been provided by OpenMath. The core of the OpenMath language for mathematics is the collection of Content Dictionaries (see www.openmath.org) in which mathematical objects are defined using the OpenMath language. The above OpenMath encoding of the formula $\sin(x^2+1)$ illustrates the use of the OpenMath CD's. To define the function sin one refers to the respective CD in which this function is defined. In fact, the name sin is overloaded. There are two different kind of sin functions: one with real arguments and values in the interval $[-1, 1]$, and one with complex arguments and complex values.

In the same way the operation '+' is defined by pointing to a particular CD. This may look like an overkill, but one should observe that the symbol '+' is heavily overloaded: the same symbol may mean anything from usual

addition to an operation in an vector space or a group. Giving the CD in which the meaning of this symbol is defined, the semantics of this formula can be automatically understood.

One may wonder what is the purpose of Presentation MathML since LaTeX typesetting produces excellent results as such. One reason for Presentation MathML is in the desire to be able to express typesetting information in XML documents using XML itself to do this rather than embedding LaTeX formulae in XML documents. Presentation MathML renders XML such typesetting abilities.

The following is a quote from an editorial in the journal Scientific Computing World [6]:

> Murray-Rust and Rzepa have just published the first ever peer-reviewed journal article - about the CML language - completely in XML (see references). And he has hopes for much greater things.
>
> "I pay great tribute to the W3C," he says. "They have built an infrastructure out of XML that covers the whole of networked computing - metadata, machine communications, etc. - all which is moving towards a number of advanced IT application goals. However, Tim Berners-Lee's vision of a 'semantic Web', a true information infrastructure, is where I think the significance really lies. I think it will be universal, and XML will be the transport mechanism."

This is a strong statement about the role of XML in developing semantic web. Among the community developing tools and content for the semantic web, nobody challenges this statement. For the prominent role of XML in this undertaking it is necessary to develop XML so that it can deal with scientific information. MathML was the first meta-language developed for XML, and there are others too. OpenMath provides a mechanism to harness the power of XML in any area of mathematics.

14.3 OpenMath Architecture

The extensible OpenMath CD's (see [5]) define mathematical objects so that they can be embedded in the documents retaining their semantics. In order to be able to understand these objects, one needs auxiliary programs that translate the OpenMath encoding of mathematical objects to encodings pertinent to the particular program.

For example,

The OpenMath definition for $\int \sin(x)dx$

```
<OMOBJ>
 <OMA>
   <OMS cd=3D"calculus1" name=3D"int"/>
   <OMS cd=3D"transc1" name=3D"sin"/>
 </OMA>
</OMOBJ>
```

will result to the nicely displayed formula $\int \sin$ (the name of the argument is actually not defined) when placed on a web page.

When glued in an OpenMath compatible mathematics program, the same expression will result into the formula $-\cos$.

So how the mathematical object is being displayed in a program – or rather what a program does to a mathematical object – will depend on the program and, of course, on the object. The phrase books take care of the necessary translations between program specific representations and the OpenMath encoding.

14.4 Multiple Encodings of Mathematics

OpenMath provides a flexible way to embed semantic information in web pages and other documents. These methods can be applied to any discipline.

The following is a quote from A. M. Odlyzko's important paper ([2]). He wrote, in 1996:

> "Standards, or lack of them, can be a significant impediment to the adoption of new technologies.
> ...
> ...
> Mathematics, computer science, and physics all seem to have settled on TEX and its various dialects as the de facto typesetting standards. This makes it easier for these disciplines to move into electronic publishing than it is for others that have not converged on a solution. However, while TEX is adequate for almost all current papers, it may not suffice in the future as we move into a multimedia world. It is also possible that commercial packages such as Microsoft Word will be enhanced with addition of modules to handle scientific material, and may become the prevalent tools. (The era when scientists dominated electronic communications is coming to an end, and systems such as TEX, developed by scholars for scholars, might soon be eclipsed by general purpose packages.) We should not become too committed to any particular standard, as it may be transitional."

Now 8 years later mathematics still continues to be dominated by TEX and LATEX but the transition to other ways of publishing mathematics is clearly happening. One can cut mathematical formulae rendered by IBM's techexplorer in PowerPoint slides and paste them to Maple so that the semantics will be automatically understood. The techexplorer display quality of the mathematics in PowerPoint is as good as what can be produced by LATEX but the encoding is not anymore LATEX nor TEX.

Almost all of the current mathematics is, however, encoded in LATEX. The new emerging ways of embedding mathematics into documents have to accommodate also TEX and LATEX encodings. The above mentioned IBM's techexplorer program does just that, and there are other ways too. For several years now we are going to see scientific documents where a combination of LATEX and MathML and OpenMath is used. The reasons for this are

- the necessity of being able to use existing materials encoded in TEX or LATEX,
- the production of MathML or OpenMath encoding is still very expensive because of the lack of good editors.

The best way today to create MathML or OpenMath encoded mathematics is to express the formula in question in a mathematics program like Maple or Mathematica and then export the formula into the desired format. There are tools for the LATEX to OpenMath conversion, but the conversion is a non-trivial task and cannot possibly be done automatically beyond a certain level.

14.5 Example of the Usages of the OpenMath Concepts: the MAMMA Project and the Helsinki Learning System

Mathematics with the Aid of MultiMedia (MAMMA) project directed by the authors of this paper has developed an adaptive interactive learning system, the Helsinki Learning System (HLS, see [4]), that currently supports a mixed LATEX- MathML encoding for mathematics in the system. The first application area of the Helsinki Learning System is calculus, but the system is in no way limited to mathematics only. It can be used in any discipline.

The core of the HLS is formed by the problem and exercise database and the Course Content Dictionaries (CCDs), which define the subject areas of problems. Through the CCDs one can associate the problem database with any text-book.

HLS takes advantage of the possibility of interpreting mathematical semantic content submitted by the student. On the basis of this interpretation

the system drafts a profile of the student's strong areas and weak areas and interacts with the student accordingly. Part of the idea of HLS is automated grading of student performance. While multiple-choice questions make such grading possible in any area of learning, a special feature of mathematics is that also strings submitted by the student can be graded on the basis of their semantics.

14.6 History of the OpenMath Project

It was proposed, at the International Conference of Mathematicians in Warsaw, 1983, that a database for mathematical facts should be developed. Consequently the European Mathematical Council, chaired by Sir Michael Atiyah, set up the Database working group.

Principal leaders of that working group were Michael Demazure (former N. Bourbaki) and Flemming Topsøe. Practically all West European countries were represented in this project which later became known as the Euromath Project. Mika Seppälä was the Finnish representative.

The Euromath Project started before any of the people involved in the project had personal computers. Most of them had had no experience with TEX either. So it was no wonder that the Euromath working groups could not fully understand the difficulties in creating the Database of Mathematical Facts, which was the original charge of the project.

By 1990 it became apparent that the development of the Database of Mathematical Facts was not possible. The project redefined itself and aimed at producing a method to include mathematics in SGML documents. The project was successful in doing this in the sense that it did produce a DTD for mathematical SGML documents, and an editor, the Euromath editor, to create and to manipulate such documents.

In the Euromath DTD mathematics was embedded in SGML documents using LATEX encoding. The Euromath editor allowed one to write mathematics so that the mathematical expressions are entered using their LATEX encoding. By pressing a suitable sequence of keys, this encoding was then shown as a traditional mathematical expression in the document. All this was fine, but in 1991 it still required too much computing power. Only a handful of mathematicians had, at that time, access to computers powerful enough to run the Euromath editor. Also the Euromath DTD in which mathematics was embedded as TEX expressions in SGML documents was not a satisfactory solution.

The need of developing this standard further became apparent (see the Editorial of the Euromath Bulletin Vol. 1, N. 1 by M. Seppälä [3]). In November 1993 M. Seppälä submitted a proposal, to the European Community, to establish a European network to develop a standard for mathematics in the

Internet. Shortly thereafter (December 1993) the first OpenMath workshop was organized by Gaston Gonnet at ETH in Zürich. This initial OpenMath project was entitled "Editing and Computing" and it laid foundations for the current MathML and OpenMath languages. The Mathematics Department of the University of Helsinki is a partner in a current EU funded Openmath Thematic Network project. During 1997-1999 Jouko Väänänen led a TEKES-project in Finland which developed resources for OpenMath.

The service mark "OpenMath" belongs to the international OpenMath Society[1], which is a registered society in Helsinki.

[1] http://www.openmath.org/

Bibliography

[1] European Space Agency. Ariane 501 - Presentation of Inquiry Board report. Published in the www, `http://www.esa.int/export/esaCP/Pr_33_1996_p_EN.html`, July 1996.

[2] A.M. Odlyzko. Tragic loss or good riddance? The impending demise of traditional scholarly journals. *Intern. J. Human-Computer Studies*, 42:71 – 122, 1995. Original paper dated Nov. 6, 1994, URL `http://www-mathdoc.ujf-grenoble.fr/textes/Odlyzko/amo94/amo94.html`.

[3] Mika Seppälä. Standarization of mathematical documents. *Euromath Bulletin*, 1(1):5–6, August 1992.

[4] Mika Seppälä and Jouko Väänänen. The Helsinki Learning System. URL `http://mark.math.helsinki.fi/hls/`, 2001.

[5] OpenMath Society. Content Dictionaries. Published in the web. URL `http://www.openmath.org/cd/`.

[6] Vanessa Spedding. XML to take Science by Storm. Available on the wwww. SECIS ONLINE, URL `http://www.secis.rl.ac.uk/index.html?rhs=collaboration/scw_xml_science_storm.html`.

Chapter 15

Using RDF(S) for Multiple Views into a Single Ontology

Santtu Toivonen

This paper deals with RDF (Resource Description Framework). The main point is to present a general model describing when and how to exploit RDF technology. It is suggested that RDF(S)[1]*. functions best as a means to provide mechanisms for expressing contextual and case-specific information. In other words, RDF(S) is suitable for providing different views into a single extensive ontology, rather than specifying the actual ontology. The ontology "behind" the case-specific RDF(S) is preferably to be expressed using some other mechanism than RDF(S).*

15.1 Introduction

15.1.1 Nature and Scope of the Paper

This paper is theoretical and methodological in its nature. It is theoretical since applications and implementation-specific details are excluded. It is methodological since it concentrates on the proper usage of RDF(S).

This paper introduces a simple model on how to exploit RDF(S) in large and heterogeneous environments that include several different applications. The opinion is that RDF(S) has a lot of useful features in describing resources, but also some drawbacks. After the model is presented, the possibilities as well as limitations of RDF(S) are discussed.

[1]RDF(S) refers to combined technologies of RDF and RDFSchema. Cf. [18]

15.1.2 Technologies with Significance to the Proposed Model

RDF(S) technology aims at describing web resources. It is under development and standardization in the World Wide Web Consortium. RDF is specified in two separate documents, one about model and syntax of RDF [13] and the other about RDF schemas [3].

XML is one proposed representation format for RDF statements. Of the large amount of technologies in the XML family at least XML Namespaces [2] and XML Schema [15, 8, 19] are relevant with respect to RDF. Namespaces are needed in RDF(S) because they help identifying the particular domains and modeling layers [18]. Furthermore, the particular RDF schema that is used for validating different RDF documents is identified using namespace notation. XML schema technology is needed for syntactic validation of RDF documents that are in XML format.

There are some differences between validation in XML and RDF [4]. Validation through RDF schemas grounds mainly on semantics, i.e. the meaning-based hierarchy and relations among the concepts to be defined. XML schemas perform syntactic validation instead; they concentrate on the grammar of the XML documents [6]. There is some semantics in XML schema technology, like the usage of datatypes, but compared with RDF schemas it is best thought of as a syntactic validation mechanism.

15.2 Overview of the Model

This chapter presents the general structure of the proposed model. The motivation is to familiarize the reader with different parts of the model.

Figure 15.1 depicts the overview of the model and illustrates the role of RDF(S). Unlike in [4, 6, 18], RDF(S) is not intended to cover the whole semantic categorization in the environment[2]. It is rather intended as a mechanism to provide domain-specific data related to small-scale tasks. An individual RDF document as well as an RDF schema document consists of a set of concepts that is likely to be subset of the concepts in the ontology.

Figure 15.1 is now examined from right to left. The rightmost section of the picture denotes ontology, the most general description of the environment in question. The next two sections are in the core focus of this paper. RDF schemas are seen as domain-specific validating filters. RDF documents are relatively small pieces of information that are validated against RDF

[2]Note that in [4, 6, 18] basic RDF(S) is extended with a language called OIL (Ontology and Inference Language). Also DAML (DARPA Agent Markup Language) [12] extends RDF(S). What is proposed here, is different. Here the basic mechanisms of RDF(S) are thought to be such that RDF(S) (or any system with similar internal structure) is not suitable for describing a potentially large ontology.

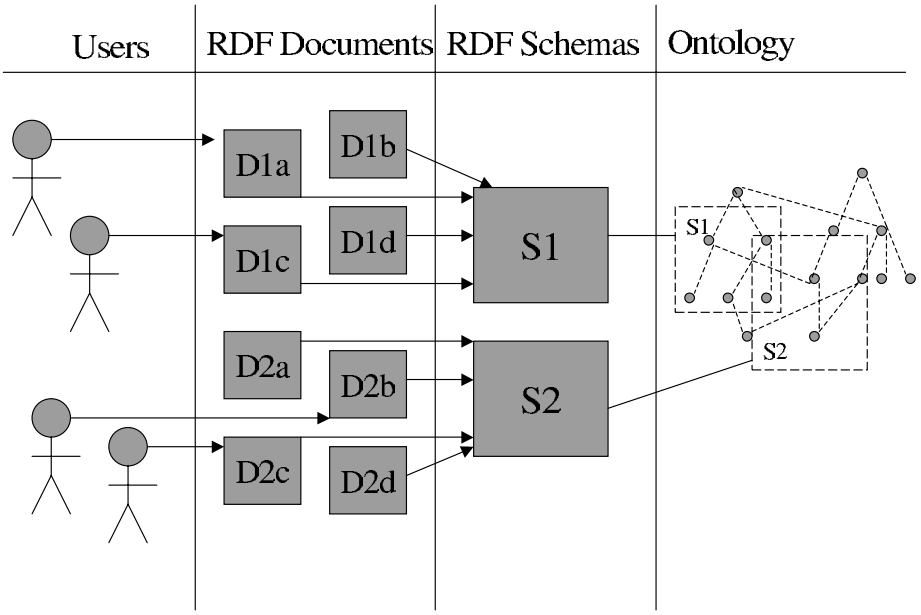


Figure 15.1: Overview of the proposed model.

schemas. And finally, users are the ones that utilize RDF(S) as their source of knowledge when working in the environment. Users can be software agents as well as human beings.

15.2.1 Ontology

Details of the ontology are outside the scope of this particular paper; ontology is treated here as a "black box". It could be implemented for example as a semantic network or a tree structure. The approach in this paper favors the adoption of one big shared ontology as opposed to several smaller ones. Initiatives like SUO[3] and Cyc[4] influence this paper and the approach based on large ontologies. SUO aims at defining a set of general concepts that could be specialized in smaller domain ontologies. Cyc is a large knowledge base trying to capture and formalize common sense.

It is acknowledged that this shared ontology might expand and become too slow and complicated to use. Additional limitations include the complexity and slowness of defining standards needed for one large heterogeneous ontology [5]. First one of the problems of the shared ontology approach, the slowness of using a large ontology, can be eliminated with RDF schemas. With RDF schemas it is possible to specialize the users of the ontology to be task-specific experts; they do not have to know every bit of information about

[3]Standard Upper Ontology, http://suo.ieee.org
[4]Cycorp, http://www.cyc.com

the environment. Problems with defining standards for large ontologies are outside the scope of this paper.

The size and magnitude of the environment is a relevant question within the limits of this paper; how large and heterogeneous is the environment supposed to be? Instead of concentrating on domains, disciplines, business branches, etc., the concept of environment is used here along the following guideline: if two applications share one or more concepts, they belong in the same environment. And there should be only one general ontology in one environment.

One important property of an ontology is extensibility [9]. It should be possible to introduce new concepts into an existing ontology so that the applications utilizing the ontology stay unbroken. In this model it is entirely possible to extend the ontology with new concepts since the users of the ontology operate using the RDF(S) that they themselves have defined. In the ontology all the concepts have similar ontological statuses[5]. RDF schemas and RDF documents together provide different views into the ontology.

15.2.2 RDF Schemas

RDF schemas are intended to function in a roughly similar role than DTD's function for XML documents. Individual RDF documents are validated against some RDF schema. RDF Schema specification [3] has defined a number of worthwhile concepts to be used when validating RDF documents. They are now presented briefly, since understanding their hierarchy and interrelations is important for the model presented in this paper.

At the topmost level the concepts are divided into three categories: `rdfs:Resource`, `rdfs:Class`, and `rdf:Property`. Two important properties, `rdf:type` and `rdfs:subClassOf`, are needed in order to express the relationships among these concepts. Resource is the topmost class of the RDF system. Everything else is describable as a subclass of resource. Type-property is needed in order to express that each resource is a member of a class. A property is a specific aspect, characteristic, attribute, or relation used to describe a resource [13]. With respect to this paper, the division between properties and other resources is crucial[6].

The RDF schema specification [3] defines two important constraint properties: `rdfs:range` and `rdfs:domain`. These constraints are used only within RDF schemas; they do not appear in other RDF documents. The domain

[5]This does not necessarily mean that the ontology is totally flat; there can naturally be some very general hierarchies among the concepts in the ontology. For example subclass-superclass -relation is something that can be said to hold between certain concepts regardless of the case-specific details.

[6]Every property is a resource and also a member of some class. In this paper properties are nevertheless often contrasted with classes and resources. The reason for this is to differentiate the concepts that get defined from those that participate in defining them.

constraint indicates that a property may be used along with the resources of a certain class. For example, author is a property that could originate from a resource that i s an instance of class book. A property may have zero, one, or more than one class as its domain.

Range, on the other hand, is something more rigorous; it specifies the class that the value of the property in question should be a resource of [3]. For example, a range constraint applying to the author property might express that the value of an author must be a resource of class person. A property can have at most one range property.

15.2.3 RDF Documents

Individual RDF documents are validated against RDF schemas. RDF documents consist of descriptions, which in turn consist of statements. Each description represents some resource. Each statement represents some feature of the resource that is being described. In RDF schemas the interrelations among selected resources and properties are defined. RDF documents contain naturally more specific and case-related data than RDF schemas; in RDF documents properties and resources are given values and thereby the ontological system is tied to actual instances of the resources. Following is a simple example of RDF in an XML syntax:

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf- syntax-ns#">
  <Description about="http://www.santtusvideos.com/taxidriver.mpg">
    <director xmlns="http://www.santtusvideos.com/schema/">
      Martin Scorsese
    </director>
    <starring xmlns="http://www.santtusvideos.com/schema/">
      Robert De Niro
    </starring>
    <length xmlns="http://www.santtusvideos.com/schema/">
      114
    </length>
  </Description>
</RDF>
```

Here the mpg-version of Taxi Driver is presented as an RDF resource. It has three properties: director, starring and length. These are specified in individual statements. This document is validated against the RDF schema of an imaginary web video service called Santtu's Videos. Here all the statements refer to the same schema but this is not necessary. Features of a given resource could be defined in separate schema documents.

The shared ontology approach is favored in this paper over the multiple ontologies approach [5]. Nonetheless the usage of RDF(S) adopts some features from the multiple ontologies point of view. RDF documents and schemas are often organized into a hierarchy and descriptions might specialize other descriptions defined in other RDF(S)'s using `rdfs:subClassOf` and `rdfs:subPropertyOf` properties. It is good to keep in mind, however, that

RDF(S) is treated as a means to provide views into ontologies, rather than specifying the actual ontologies.

15.2.4 Users

RDF(S) is intended to provide metadata about web resources that is both human-readable and machine-understandable [13]. Hence the users of the model proposed here can consist of software agents in addition to human beings. The semantic information that software agents use should be mainly external to the agents themselves [20]. Agents should have access to an external ontology describing the general structure of the environment. The ontology would constitute an independent repository of information. This way the agents themselves would not become "walking encyclopaedias" (cf. [7]) but remain relatively simple.

This semantic information external to the agents is distributed to all the other parts of the proposed model: RDF documents, RDF schemas, and the ontology. The agents should be designed so that they understand one or more RDF schemas. The agents can utilize individual RDF documents as pieces of case-specific information. They can do different things with the documents (and applications/services bound to the documents) according to their internal inference rules and the RDF schema/schemas they are committed to.

Also human users of the environment benefit from this model. The model helps people to understand different parts of the environment and applications appearing in it. For example, if someone decides to introduce a new service or resource into the environment, he can examine the schemas of the existing resources (assuming that the schemas are publicly available for examination).

Should he find a suitable schema, he can utilize it as a means to exploit the ontology. If no schemas as such work for the developer, there still might be some guidelines or pieces of information in existing schemas that help the developer to get started. Either way, people working in an environment with shared ontology can reduce the amount of work with public RDF schemas and this way avoid re-inventing the wheel.

15.3 Usage of RDF(S)

15.3.1 RDF and Web Resources

RDF is intended to provide metadata about web resources. Different web resources naturally have different means of categorization. For example libraries, video stores, and digital phone books use different concepts as metadata [1]. There are nevertheless some common aspects among all of these.

First, they all have resources, be they books, movies or phone book entries. Second, they all have properties characterizing the resources. Movies, for example, have actors, directors, length of the movie, etc. Third, the resources may be grouped into classes. There might be a class called movies and it might have a subclass called horror movies.

15.3.2 Properties in RDF(S)

RDF(S) is in this paper proposed as a means to provide case-specific information rather than means to constitute the whole ontology. The reason for this reduces to the question concerning the ontological status of properties. In [13] properties are described the following way: "*A property is a specific aspect, characteristic, attribute, or relation used to describe a resource*".

RDF(S) properties are thereby qualifiers that characterize some resources. They have a clearly different ontological status than classes, for example. Classes are something that are defined (definienda, sing. definiendum), properties are something that participate in defining them (definientia, sing. definiens). This is fully acceptable as long as the case-specificity and contextuality of the model is kept in mind. Depending on the context, concept c1 can be an attribute of c2 and vice versa [16].

In other words: Concepts that are definienda in some case or application form the basic level of concepts for that particular case. In RDF(S) terminology these would be classes to be defined. There are two other levels in addition: subordinate and superordinate level. Definiens (property in RDF(S) terminology) is at the subordinate level when compared with definiendum. Depending on the domain, however, relations between these levels vary (cf. [17, 14]).

15.3.3 Characterizing the Case-Specificity of RDF(S)

From the electronic video store's point of view director is a property that characterizes the resources of a class called movie. This is fine as long as it is clear that somewhere else director could appear also as a class that gets defined by some other properties. An electronic catalog of artists might have director as a class[7]. Now directors could have movies that they have directed as their properties. Just the other way around than in the video store[8]. This is illustrated in Figure 15.2.

[7]Artist catalog would probably have artist as a basic level concept and director as subordinate level concept. However, in RDF(S) terminology these would both be classes, not properties.

[8]In principle even two different video stores could interpret the hierarchy of some set of concepts variously.

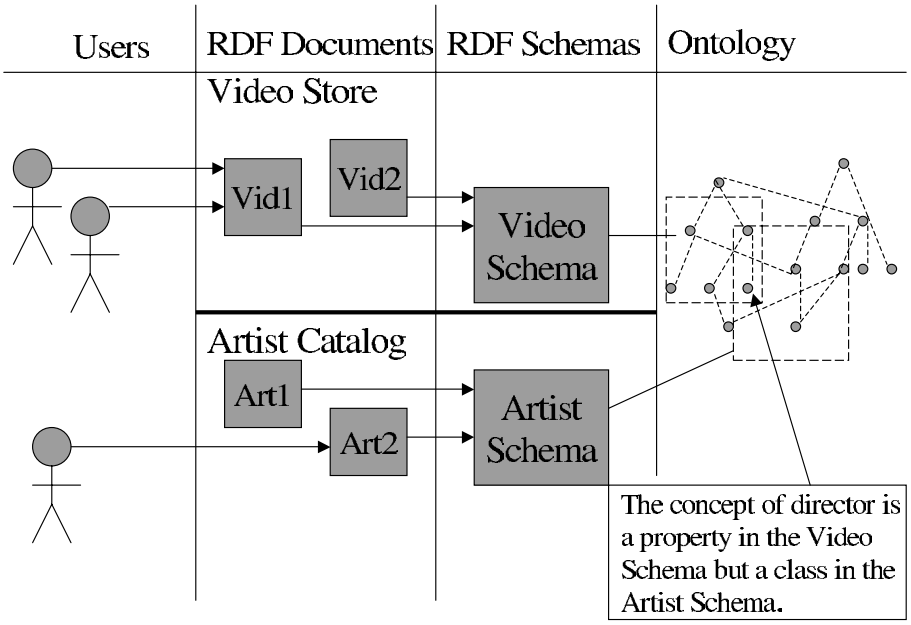


Figure 15.2: An example about the domain-specificity.

When representing the world, the structure of concepts should be as analogous to reality as possible [16]. And there is no a priori way to declare that some concept functions always as definiens while some other is always definiendum. That is why concepts should not be universally placed in either of these categories. In the end all concepts are similar with respect to their ontological statuses. RDF(S) is a technology with no good conventions that help coping with this matter.

Of course it is possible to introduce all (or at least majority of) concepts twice; once with the status of definiendum and again with the status of definiens. However, this is not a desirable solution. It leads to compatibility problems and violates the simplicity principle of ontologies. The principle states that there should be as few ontological commitments in an ontology as possible [9]. Introducing all concepts twice would cause a situation found in Figure 15.3.

First the RDF documents column of the Figure 15.3 is examined. It tells us that in video store the movie Taxi Driver has a property named director with the value "Scorsese". Artist catalog, on the other hand, has the director Martin Scorsese with a property named director that has the value "Taxi Driver". So Taxi Driver and Scorsese both appear once as resources to be defined and once as properties. The same thing concerns the RDF schemas column of the picture. In video store director is a property that belongs in

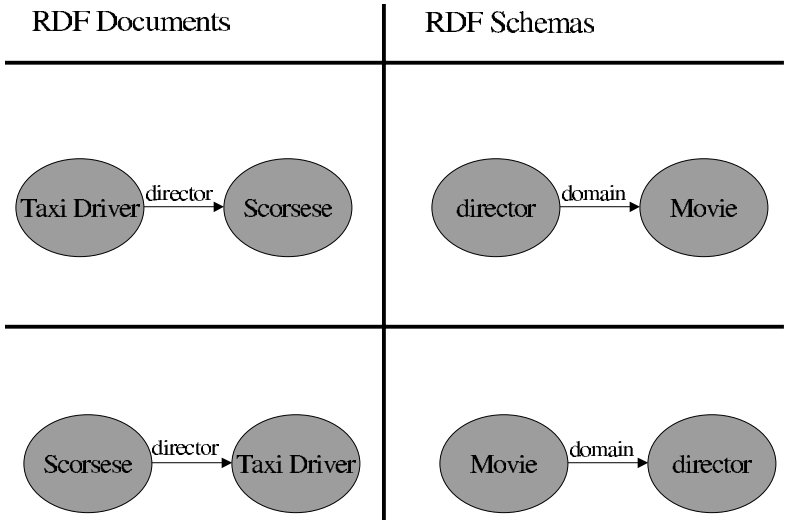


Figure 15.3: Usage of concepts in different cases.

the domain[9] of movies. In artist catalog the situation is contrary.

A phenomenon closely related to this is observed in [11]; different RDF schemas can specialize some class defined in another schema with `rdfs:subClassOf` and `rdfs:subPropertyOf` properties. They can use the same name but different definitions for that class in their own specializations. So there could be an upper RDF schema that has movies, directors, etc. all as classes. However, when electronic video store and electronic artist catalog specialize the classes in their own unique ways, the system as a whole becomes incoherent. This is one of the basic drawbacks of the multiple ontologies approach.

One remark here could be that since RDF is intended for describing web resources, the movie Taxi Driver (at least in mpg-format as in the code example presented earlier) is more appropriate candidate for a web resource than the director Martin Scorsese. That is because Martin Scorsese can not appear in a format distributed in the Internet unlike Taxi Driver.

Ontologically speaking, however, the movie Taxi Driver is not the same entity as the mpg-version of it distributed in the net. It is rather an abstract thing that has different instances. Compared with object-oriented programming, the movie Taxi Driver would be a class and the copies of that movie

[9]For the sake of simplicity only one constraint property is presented here. Besides domain, al so range is a useful property to be exploited in RDF schemas. In the schema of video store, for example, there could be a range constraint property named person attached to the director property. This would mean that the value of the director property is always a member of the class person. Furthermore, only one domain for each property is presented. If necessary, though, director could have other domains besides movies. It could be attached to TV-series, theatre plays, etc.

(for example the mpg-version distributed in the electronic video store) in turn instances of the class.

15.4 Conclusions and Discussion

The expressive power of RDF(S) does not necessarily complete all the parts that are needed for expressing a semantic description of some system. An ontology independent of the domain-specific details of its usage is needed. There should be an "isolated basic backbone" of ontology that is independent of any case-specific details [10]. And based on the arguments and examples presented here, it should be clear that RDF(S) alone does not fit together with this requirement.

What RDF(S) technology can do, however, is to provide means to access an ontology characterizing some environment — no matter how large or heterogeneous — in many ways.

15.4.1 Rethinking the Properties

The main problem of RDF(S) presented in this paper is the division of the concepts into properties and classes. The answer proposed to this problem is the usage of an external ontology in addition to the RDF(S). From the ontology's point of view the usage of concepts in different RDF(S)'s is based on roles [10]; `rdfs:Resource`, `rdfs:Class` and `rdf:Property` are different roles of some concept defined in the ontology.

Another attempt to resolve this would be to reformulate the properties. Earlier an example of using director as a property belonging in the domain of movie in one place and movie as a property belonging in the domain of director in another was presented. Why not use directors and movies always as classes? Movie would have a property directed_by that would have a director as its value. Director would have has_directed property that would have a movie as its value.

At first sight this might seem wise. More carefully examined, however, this leads to a situation not preferable to defining every concept twice; once as a property and once as a class. The number of properties would be doubled as shown in Figure 15.4; has_directed and directed_by would both exist between a movie and its director even though they have the same information content. This would again violate the simplicity principle of ontologies [9].

Yet another attempt to overcome the problem of classes versus properties is reacting to it at levels residing on top of RDF. OIL (see [4, 6, 18]) and DAML (see [12]) are examples of languages that are on a higher level than RDF. However, introducing rules and restrictions that cope with limitations of RDF at a higher level does not seem feasible. For one thing, this again

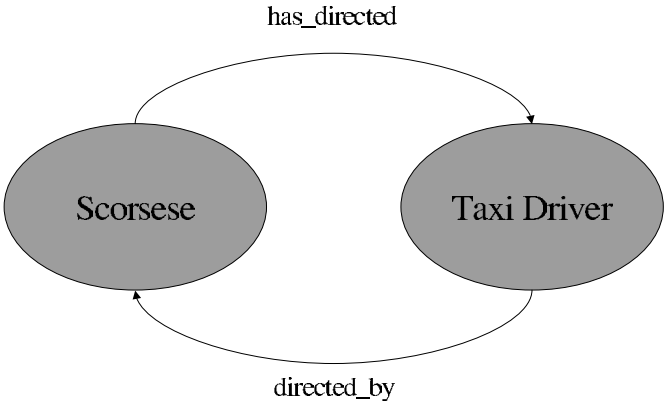


Figure 15.4: Duplicating the properties.

violates the simplicity principle of ontologies [9]; for each ambiguous class-property distinction at the RDF level there would exist a fixing principle at a higher level. Secondly, the whole idea of coping with problems of some level at another is not desirable; each level should be clear enough not to require fixing or configuring at other levels.

15.4.2 Deducing the Ontology from RDF(S)

In this paper the general ontology "behind" the RDF(S) is treated as a "black box". Its detailed structure is not discussed. It could however be possible (even in the model proposed in this paper) that the whole ontology is deducable from the total amount of RDF documents and schemas in a given environment. This depends on the interpretation of domain-specificity.

If all the concepts and their interrelations are such that they are found in the ontology it could be possible to make that deduction. Possible conflicts should however try to be avoided. If some concept is a property (definiens) in one schema and a resource to be defined (definiendum) in another, does that have any impact on the ontology? If it does, which one of the schemas determines the "ontological location" of the concept in question. If it does not, how it is possible to construct any hierarchy in the ontology (since there would be nothing in addition to the schemas)? On the other hand, if there are some general relations or attributes at the ontological level that are not visible in RDF(S), the deduction is not possible.

Clearly a deduction in the other direction is not possible. There is no way of knowing how the concepts in the ontologies are used and grouped in different RDF(S). This means that it is not possible to deduce all imaginable RDF(S) just by examining the ontology. And this is due to the proposed case-specific nature of RDF(S).

Acknowledgements

Thanks to Joose Niemistö and Johannes Gröhn for their help on this article.

Bibliography

[1] T. Bray. RDF and metadata, 1998. http://www.xml.com/xml/pub/98/06/rdf.html.

[2] T. Bray, D. Hollander, and A. Layman. *Namespaces in XML*, 1999. W3C Recommendation. http://www.w3.org/TR/REC-xml-names.

[3] D. Brickley and R. V. Guha. *Resource description framework (RDF) schema specification*, 2000. W3C Candidate Recommendation http://www.w3.org/TR/rdf-schema.

[4] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the web by extending RDF Schema. In *Proceedings of the tenth World Wide Web conference WWWW'10*, Hong Kong, May 2001.

[5] Z. Cui, V. Tamma, and F. Bellifemine. Ontology management in enterprises. *British Telecommunications Technology Journal*, October 1999.

[6] S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, and S. Melnik. The Semantic Web — on the roles of XML and RDF. *IEEE Internet Computing*, September/October 2000.

[7] D. C. Dennett. When philosophers encounter artificial intelligence. In *Daedalus, Proceedings of the American Academy of Arts and Sciences*, number 117, pages 283–295, 1988.

[8] D. C. Fallside. *XML Schema Part 0: Primer*, 2001. W3C Proposed Recommendation. http://www.w3.org/TR/xmlschema-0/.

[9] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In Nicola Guarino and Roberto Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Padova, Italy, 1993. Kluwer Academic Publishers.

[10] N. Guarino. Some ontological principles for designing upper level lexical resources. In *Proceedings of First International Conference on Language*

Resources and Evaluation, pages 527–534, Granada, Spain, 1998. ELRA - European Language Resources Association.

[11] J. Heflin and J. Hendler. Semantic interoperability on the web. In *Proceedings of Extreme Markup Languages*, 2000.

[12] J. Hendler and D. L. McGuinness. The DARPA agent markup language. *IEEE Intelligent Systems*, 15(6):67–73, November/December 2000.

[13] Ora Lassila and Ralph R. Swick, editors. *Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium, February 1999. W3C Recommendation.

[14] G. L. Murphy and M. E. Lassaline. Hierarchical structure in concepts and the basic level of categorization. In Koen Lamberts and David Shanks, editors, *Knowledge, Concepts, and Categories*, pages 93–131. Psychology Press, Hove, 1997.

[15] P. V. Biron and A. Malhotra. *XML Schema Part 2: Datatypes*, 2001. W3C Proposed Recommendation. http://www.w3.org/TR/xmlschema-2/.

[16] P. Saariluoma. *Foundational analysis: Presuppositions in experimental psychology*. Routledge, London, 1997.

[17] J. I. Saeed. *Semantics*. Blackwell Publishers Ltd, Oxford, 1997.

[18] S. Staab, M. Erdmann, A. Maedche, and S. Decker. An extensible approach for modeling ontologies in RDF(S). In *First Workshop on the Semantic Web at the Fourth European Conference on Digital Libraries*, Lisbon, Portugal, 2000.

[19] H. S. Thompson, D. Beech, M. Maloney, and M. Mendelsohn. *XML Schema Part 1: Structure*, 2001. W3C Proposed Recommendation. http://www.w3.org/TR/xmlschema-1/.

[20] S. Toivonen. Definition and usage of a software agent. *Arpakannus*, (2):9–13, 2000.