Music domain ontology applications for intelligent web searching

María Clara Vallés y Pablo R. Fillottrani

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Baha Blanca, Argentina
mcvalles@gmail.com
prf@cs.uns.edu.ar

Abstract

The Semantic Web is an extension of the current Web that attempts to reach a state in the future where everything on the Web will no longer be only machine-readable, but also machine-understandable. Three important technologies for developing the Semantic Web are already in place: Extensible Markup Language (XML), the Resource Description Framework (RDF), and Web Ontology Language (OWL). An ontology language is a formal language used to encode ontologies. An ontology is an explicit specification of a conceptualization. Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields, and which can be used for reasoning about the objects within a particular domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. In this work, we develop an ontology of Argentinean music. Despite being highly specific, we illustrate how such an ontology can be expanded and used in applications that carry out complex music related searches. These applications can be embedded later on in electronic devices with some form of wireless networking capability, such as mobile phones, enriching their current functionality.

1 Introduction

Since its beginning, the World Wide Web has played an important role in our everyday life, transforming the world towards a knowledge society. As a result, the way computers are used has diversified, gaining popularity and users. Currently, one of their main utilities is related to information processing. Some few examples being applications such as text processing, data bases, and games. At present, the view of computers as an efficient

way to access information on practically any subject, has gained special attention.

Most of today's Web content is presented in a way that makes it suitable only for human consumption. In other words, information is expected to be consumed by individuals, not software programs. Typical uses of the Web involve people searching, sharing and making use of information, communicating with other people, shopping online, creating personal spaces, among others. These activities are human oriented and are thus, not well supported by software tools.

Apart from hypertext links, which allow the possibility of linking a document to any other document, keyword-based search engines have turned into an essential tool for information management on today's web. It has become almost natural to use these search engines to look up information for almost every possible topic, and society in general has come to rely on them. However, the use of these tools in this fashion has some disadvantages, given the fact that it is the person who must browse documents, extract the information he or she is looking for, and discard the rest. The next step to take in order to solve this problem, would be the automatization of this process.

Unfortunately, there is a mayor inconvenience we must solve before we can achieve this task, which has to do with the fact that Web content must allow a computer program to sort out the meaning (semantics) of the information it browses. In other words, it is easy for a person to distinguish the information that is meaningful, because humans can "understand" the meaning of the Web content they read. However, in order for a software tool to interpret sentences and extract useful information for users, Web content should be represented in a form that is more machine-processable and which allows intelligent techniques to take advantage of these representations. We refer to this future Web plan, as the Semantic Web. Therefore, the Semantic Web attempts to reach a state in the future where everything on the Web will not longer be only machine-readable, but also machine-understandable.

It is important to understand that the Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, enabling computers and people to work in closer cooperation. The first steps towards incorporating the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will cause the dawning of significant new functionality as machines become much better able to process and "understand"the data that they merely display at present.

The second section of this presentation focuses on the main topics that must be addressed in order for the semantic web to function. The third and fourth section centers in three important technologies for developing the Semantic Web: Extensible Markup Language (XML), the Resource Description Framework (RDF), and Web Ontology Language (OWL). Sections

five through seven present an ontology example, an application that benefits from it and several representative use cases. The last section corresponds to the final conclusion.

2 Knowledge Representation

For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. Artificial-intelligence researchers have studied such systems since long before the Web was developed. Traditional knowledge-representation systems typically have been centralized, requiring everyone to share exactly the same definition of common concepts such as "parent" or "vehicle". But central control is stifling, and increasing the size and scope of such a system rapidly becomes unmanageable. Moreover, these systems usually carefully limit the questions that can be asked so that the computer can answer reliably, or answer at all. To avoid such problems, traditional knowledge-representation systems generally each had their own narrow and idiosyncratic set of rules for making inferences about their data. However, even if the data could be transferred from one system to another, the rules, existing in a completely different form, usually could not [2].

Semantic Web researches, on the other hand, accept that unanswerable questions represent a small price to pay to achieve versatility. This can be compared to the initial observation detractors gave about the conventional Web, where the lack of a central database and tree structure makes it impossible to assure everything one seeks will actually be found. However, the expressive power of the system makes vast amounts of information available, and search engines now produce remarkably accurate results and offer a lot of material.

The task of adding logic to the Web encompasses a series of complex decisions, given the fact that the logic must be strong enough to describe object properties, but not to powerful so as to avoid agents tricking themselves into considering paradoxes.

3 Technologies for developing the Semantic Web

Two important technologies for developing the Semantic Web are already in place: Extensible Markup Language (XML) and the Resource Description Framework (RDF).

3.1 Extensible Markup Language (XML)

One of the fundamental contributions towards the Semantic Web to date has been the development of XML itself. Liberating data from opaque, in-

extensible formats as it does, XML provides an interoperable syntactical foundation upon which solutions to the larger issues of representing relationships and meaning can be built. It is an important center of agreement among individual developers and corporations [3].

XML, the Extensible Markup Language, is a W3C-endorsed standard for document markup.[1] XML owns its name to the fact that it allows users yo mark up data that can later be displayed on the web, with simple human-readable tags. Users can create their own tags, such as "address" or "career", that annotate Web pages or sections of text on a page. Scripts, or programs, can make use of these tags in sophisticated ways, but the script writer has to know what the page writer uses each tag for. Therefore, XML is purely syntactical, allowing users to add arbitrary structure to their documents but saying nothing about what the structures mean, i.e. its semantics [4].

3.1.1 The Benefits of XML

Data is included in XML documents as strings of text. The data is surrounded by text markup that describes the data. XML's basic unit of data and markup is called an element. The XML specification defines the exact syntax this markup must follow: how elements are delimited by tags, what a tag looks like, what names are acceptable for elements, where attributes are placed, and so forth. Superficially, the markup in an XML document looks a lot like the markup in an HTML document, but there are some crucial differences [4].

1. XML allows developers and writers to define the elements they need as they need them. Although XML is quite flexible in the elements it allows to be defined, it is quite strict in many other respects. It provides a grammar for XML documents that says where tags may be placed, what they must look like, which element names are legal, how attributes are attached to elements, and so forth. This grammar is specific enough to allow the development of XML parsers that can read any XML document. Documents that satisfy this grammar are said to be well-formed. Documents that are not well-formed are not allowed, thus XML processors will reject documents that contain well-formedness errors.

2. In well-designed XML applications, the markup says nothing about how the document should be displayed. That is, it does not say that an element is bold or italicized or a list item. Thus, XML is not a presentation language.

[1]The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding. For more information, visit www.w3.org .

3. The markup permitted in a particular XML application can be documented in a schema. Particular document instances can be compared to the schema. Documents that match the schema are said to be valid. Documents that do not match are invalid. Validity depends on the schema. That is, whether a document is valid or invalid depends on which schema you compare it to. Not all documents need to be valid. For many purposes it is enough that the document merely be well-formed.

   There are many different XML schema languages, with different levels of expressivity. The most broadly supported schema language and the only one defined by the XML 1.0 specification itself is the document type definition (DTD). A DTD lists all the legal markup and specifies where and how it may be included in a document. DTDs are optional in XML. On the other hand, DTDs may not always be enough. The DTD syntax is quite limited and does not allow you to make many useful statements such as "This element contains a number" or "This string of text is a date between 1974 and 2032". The W3C XML Schema Language (which sometimes goes by the misleadingly generic label schemas) does allow you to express constraints of this nature.

3.1.2 What XML Is Not

First of all, XML is not a programming language. There is no such thing as an XML compiler that reads XML files and produces executable code. XML can be used as a format for instructions to programs that do make things happen, but in all cases it's the program taking action, not the XML document itself. An XML document by itself does not do anything [4].

XML is not a database. You're not going to replace an Oracle or MySQL server with XML. A database can contain XML data, but the database itself is not an XML document. You can store XML data into a database on a server or retrieve data from a database in an XML format, but to do this, you need to be running software written in a real programming language such as C or Java.

3.1.3 XML Schema

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntax constraints imposed by XML itself. An XML schema provides a view of the document type at a relatively high level of abstraction.

There are languages developed specifically to express XML schemas. The Document Type Definition (DTD) language, which is native to the XML specification, is a schema language that is of relatively limited capability,

but that also has other uses in XML aside from the expression of schemas. Two other very popular, more expressive XML schema languages are XML Schema (W3C) and RELAX NG. The mechanism for associating an XML document with a schema varies according to the schema language. The association may be achieved via markup within the XML document itself, or via some external means.

The process of checking to see if an XML document conforms to a schema is called validation, which is separate from XML's core concept of syntactic well-formedness. All XML documents must be well-formed, but it is not required that a document be valid unless the XML parser is "validating", in which case the document is also checked for conformance with its associated schema. DTD-validating parsers are most common, but some support W3C XML Schema or RELAX NG as well. Documents are only considered valid if they satisfy the requirements of the schema with which they have been associated.

XML Schema language was published as a W3C Recommendation in May 2001. It was the first separate schema language for XML to achieve Recommendation status by the W3C. Like all XML schema languages, XML Schema can be used to express a schema: a set of rules to which an XML document must conform in order to be considered "valid" according to that schema. However, unlike most other schema languages, XML Schema was also designed with the intent of validation resulting in a collection of information adhering to specific datatypes, which can be useful in the development of XML document processing software.

An XML Schema instance is an XML Schema Definition (XSD) and typically has the filename extension ".xsd". The language itself is sometimes informally referenced as XSD. XSD is also an initialism for XML Schema Datatypes, the datatype portion of XML Schema.

3.2 Resource Description Framework (RDF)

The World Wide Web affords unprecedented access to globally distributed information. Metadata, or structured data about data, improves discovery of and access to such information. The effective use of metadata among applications, however, requires common conventions about semantics, syntax, and structure. Individual resource description communities define the semantics, or meaning, of metadata that address their particular needs. Syntax, the systematic arrangement of data elements for machine-processing, facilitates the exchange and use of metadata among multiple applications. Structure can be thought of as a formal constraint on the syntax for the consistent representation of semantics [6].

The Resource Description Framework (RDF), developed under the auspices of the World Wide Web Consortium (W3C), is an infrastructure that enables the encoding, exchange, and reuse of structured metadata. This in-

frastructure enables metadata interoperability through the design of mechanisms that support common conventions of semantics, syntax, and structure. RDF does not stipulate semantics for each resource description community, but rather provides the ability for these communities to define metadata elements as needed. RDF uses XML as a common syntax for the exchange and processing of metadata. The XML syntax provides vendor independence, user extensibility, validation, human readability, and the ability to represent complex structures. By exploiting the features of XML, RDF imposes structure that provides for the unambiguous expression of semantics and, as such, enables consistent encoding, exchange, and machine-processing of standardized metadata.

RDF supports the use of conventions that will facilitate modular interoperability among separate metadata element sets. These conventions include standard mechanisms for representing semantics that are grounded in a simple, yet powerful, data model. RDF additionally provides a means for publishing both human-readable and machine-processable vocabularies. Vocabularies are the set of properties, or metadata elements, defined by resource description communities. The ability to standardize the declaration of vocabularies is anticipated to encourage the reuse and extension of semantics among disparate information communities. For example, the Dublin Core Initiative, an international resource description community focusing on simple resource description for discovery, has adopted RDF. Educom's IMS Instructional Metadata System, designed to provide access to educational materials, has adopted the Dublin Core and corresponding architecture and extended it with domain-specific semantics. RDF is designed to support this type of semantic modularity by creating an infrastructure that supports the combination of distributed attribute registries. Thus, a central registry is not required. This permits communities to declare vocabularies which may be reused, extended and/or refined to address application or domain specific descriptive requirements.The goals of RDF are broad, and the potential opportunities are enormous.

3.2.1 The RDF Data Model

RDF provides a model for describing resources. Resources have properties (attributes or characteristics). RDF defines a resource as any object that is uniquely identifiable by an Uniform Resource Identifier (URI). The properties associated with resources are identified by property-types, and property-types have corresponding values. Property-types express the relationships of values associated with resources. In RDF, values may be atomic in nature (text strings, numbers, etc.) or other resources, which in turn may have their own properties. A collection of these properties that refers to the same resource is called a description [6].

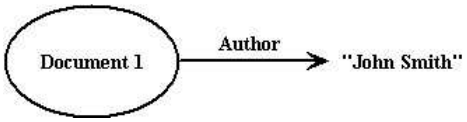
The application and use of the RDF data model can be illustrated by

concrete examples. Consider the following statements:

- "The author of Document 1 is John Smith"
- "John Smith is the author of Document 1"

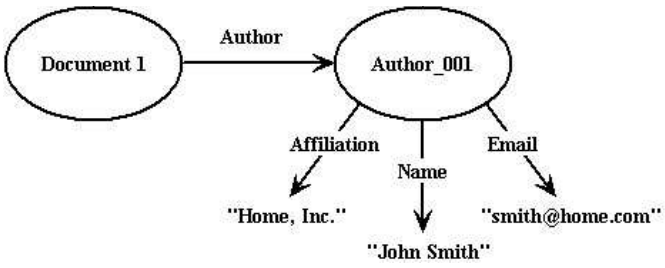
To humans, these statements convey the same meaning (that is, John Smith is the author of a particular document). To a machine, however, these are completely different strings. Whereas humans are extremely adept at extracting meaning from differing syntactic constructs, machines remain grossly inept. Using a triadic model of resources, property-types and corresponding values, RDF attempts to provide an unambiguous method of expressing semantics in a machine-readable encoding.

RDF provides a mechanism for associating properties with resources. So, before anything about Document 1 can be said, the data model requires the declaration of a resource representing Document 1. Thus, the data model corresponding to the statement "the author of Document 1 is John Smith" has a single resource `Document 1`, a property-type of `author` and a corresponding value of `John Smith`. To distinguish characteristics of the data model, the RDF Model and Syntax specification represents the relationships among resources, property-types, and values in a directed labeled graph. In this case, resources are identified as nodes, property-types are defined as directed label arcs, and string values are quoted. Given this representation, the data model corresponding to the statement is graphically expressed as:



If additional descriptive information regarding the author were desired, e.g., the author's email address and affiliation, an elaboration on the previous example would be required. In this case, descriptive information about John Smith is desired. As was discussed in the first example, before descriptive properties can be expressed about the person John Smith, there needs to be a unique identifiable resource representing him. Given the directed label graph notation in the previous example, the data model corresponding to this description is graphically represented as:

In this case, "John Smith" the string is replaced by a uniquely identified resource denoted by Author_001 with the associated property-types of name, email and affiliation. The use of unique identifiers for resources allows for the unambiguous association of properties. This is an important point, as the person John Smith may be the value of several different property-types. John Smith may be the author of Document 1, but also may be the value of a particular company describing the set of current employees. The



unambiguous identification of resources provides for the reuse of explicit, descriptive information.

In the previous example the unique identifiable resource for the author was created, but not for the author's name, email or affiliation. The RDF model allows for the creation of resources at multiple levels. Concerning the representation of personal names, for example, the creation of a resource representing the author's name could have additionally been described using "firstname", "middlename" and "surname" property-types. Clearly, this iterative descriptive process could continue down many levels. It is important to consider which could be the practical and logical limits of these iterations.

There is no one right answer to this question. The answer is dependent on the domain requirements. These issues must be addressed and decided upon in the standard practice of individual resource description communities. In short, experience and knowledge of the domain dictate which distinctions should be captured and reflected in the data model.

The RDF data model additionally provides for the description of other descriptions. For instance, often it is important to assess the credibility of a particular description (e.g., "The Library of Congress told us that John Smith is the author of Document 1"). In this case the description tells us something about the statement "John Smith is the author of Document 1", specifically, that the Library of Congress asserts this to be true. Similar constructs are additionally useful for the description of collections of resources. For instance, "John Smith is the author of Documents 1, 2, and 3". While these statements are significantly more complex, the same data model is applicable. A more detailed discussion of these issues is outside the scope of this overview, but more information is available in the RDF Model and Syntax Specification".[2]

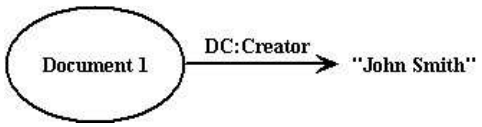
3.2.2 The RDF Syntax

RDF defines a simple, yet powerful model for describing resources. A syntax representing this model is required to store instances of the model into machine-readable files and to communicate these instances among appli-

[2]URL:http://www.w3.org/RDF/Group/WD-rdf-syntax/

cations. XML is this syntax. RDF imposes formal structure on XML to support the consistent representation of semantics [6].

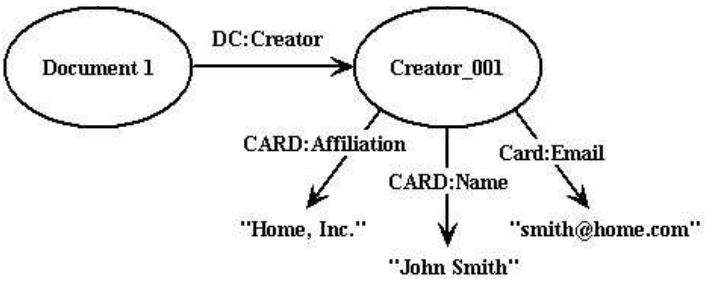
RDF provides the ability for resource description communities to define semantics. It is important, however, to disambiguate these semantics among communities. The property-type "author", for example, may have broader or narrower meaning depending on different community needs. As such, it is problematic if multiple communities use the same property-type to mean very different things. To prevent this, RDF uniquely identifies property-types by using the XML namespace mechanism. XML namespaces provide a method for unambiguously identifying the semantics and conventions governing the particular use of property-types by uniquely identifying the governing authority of the vocabulary. For example, the property-type "author" defined by the Dublin Core Initiative as the "person or organization responsible for the creation of the intellectual content of the resource" and is specified by the Dublin Core CREATOR element. An XML namespace is used to unambiguously identify the Schema for the Dublin Core vocabulary by pointing to the definitive Dublin Core resource that defines the corresponding semantics. If the Dublin Core RDF Schema, however, is abbreviated as "DC", the data model representation for this example would be:



This more explicit declaration identifies a resource Document 1 with the semantics of property-type Creator unambiguously defined in the context of DC (the Dublin Core vocabulary). The value of this property-type is John Smith.

In the more advanced example, where additional descriptive information regarding the author is required, similar syntactic constructs are used. In this case, while it may still be desirable to use the Dublin Core CREATOR property-type to represent the person responsible for the creation of the intellectual content, additional property-types "name", "email" and "affiliation" are required. For this case, since the semantics for these elements are not defined in Dublin Core, an additional resource description standard may be utilized. It is feasible to assume the creation of an RDF schema with the semantics similar to the vCard [3] specification designed to automate the exchange of personal information typically found on a traditional business card, could be introduced to describe the author of the document. The data model representation for this example with the corresponding business card schema defined as CARD would be:

[3]URL:http://www.imc.org/pdi



The structural constraints RDF imposes to support the consistent encoding and exchange of standardized metadata provides for the interchangeability of separate packages of metadata defined by different resource description communities.

3.2.3 The RDF Schema

RDF Schemas (RDF-S) are used to declare vocabularies, the sets of semantics property-types defined by a particular community. RDF schemas define the valid properties in a given RDF description, as well as any characteristics or restrictions of the property-type values themselves. The XML namespace mechanism serves to identify RDF Schemas [6].

A human and machine-processable description of an RDF schema may be accessed by de-referencing the schema URI. If the schema is machine-processable, it may be possible for an application to learn some of the semantics of the property-types named in the schema. To understand a particular RDF schema is to understand the semantics of each of the properties in that description. RDF schemas are structured based on the RDF data model. Therefore, an application that has no understanding of a particular schema will still be able to parse the description into the property-type and corresponding values and will be able to transport the description intact (e.g., to a cache or to another application).

The exact details of RDF schemas are currently being discussed in the W3C RDF Schema working group.[4] It is anticipated, however, that the ability to formalize human-readable and machine-processable vocabularies will encourage the exchange, use, and extension of metadata vocabularies among disparate information communities. RDF schemas are being designed to provide this type of formalization.

4 Ontologies

A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some

[4]URL:http://www.w3.org/TR/WE-RDF-Schema/

area of interest and the relationships that hold among them (Genesereth & Nilsson, 1987). A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly. An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence.

In recent years the development of ontologies has become common on the World-Wide Web, moving from the realm of Artificial-Intelligence laboratories to the desktops of domain experts. The ontologies on the Web range from large taxonomies categorizing Web sites (such as on Yahoo!) to categorizations of products for sale and their features (such as on Amazon.com). Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields, and which can be used for reasoning about the objects within a particular domain. An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.[7] Some of the reasons for developing ontologies are:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

Sharing common understanding of the structure of information among people or software agents is one of the most common goals in developing ontologies (Musen 1992; Gruber 1993). For example, suppose several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.

Enabling reuse of domain knowledge was one of the driving forces behind recent surge in ontology research. If one group of researchers develops an ontology in detail, others can simply reuse it for their domains. Additionally, if we need to build a large ontology, we can integrate several existing ontologies describing portions of the large domain. We can also reuse a general ontology, and extend it to describe our domain of interest.

Making explicit domain assumptions underlying an implementation makes it possible to change these assumptions easily if our knowledge about the domain changes. Hard-coding assumptions about the world in programming-language code makes these assumptions not only hard to find and understand but also hard to change, in particular for someone without programming expertise. In addition, explicit specifications of domain knowledge are useful for new users who must learn what terms in the domain mean.

Separating the domain knowledge from the operational knowledge is another common use of ontologies. We can describe a task of configuring a product from its components according to a required specification and implement a program that does this configuration independent of the products and components themselves (McGuinness and Wright 1998).

Analyzing domain knowledge is possible once a declarative specification of the terms is available. Formal analysis of terms is extremely valuable when both attempting to reuse existing ontologies and extending them (McGuinness et al. 2000).

Often an ontology of the domain is not a goal in itself. Developing an ontology is akin to defining a set of data and their structure for other programs to use. Problem-solving methods, domain-independent applications, and software agents use ontologies and knowledge bases built from ontologies as data.

4.0.4 Web Ontology Language - OWL

An ontology language is a formal language used to encode the ontology. There are a number of such languages for ontologies, one of them being OWL. OWL is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL[5]. [8]

OWL has been designed to meet the need for a Web Ontology Language. OWL is part of the growing stack of W3C recommendations related to the Semantic Web:

- XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.

[5]http://www.w3.org/TR/daml+oil-reference

- XML Schema is a language for restricting the structure of XML documents and also extends XML with datatypes.
- RDF is a datamodel for objects ("resources") and relations between them, provides a simple semantics for this datamodel, and these datamodels can be represented in an XML syntax.
- RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.
- OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users.

- OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL, see the section on OWL Lite in the OWL Reference for further details.
- OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.
- OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL DL. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete OWL Full implementations do not currently exist.

OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF. Every OWL (Lite, DL, Full) document is an RDF document, and every RDF document is an OWL Full document, but only some RDF documents will be a legal OWL Lite or OWL DL document. Because of this, some care has to be taken when a user wants to migrate an RDF document to OWL. When the expressiveness of OWL DL or OWL Lite is deemed appropriate, some precautions have to be taken to ensure that the original RDF document complies with the additional constraints imposed by OWL DL and OWL Lite.

5 An Ontology Example

In order to fully understand the previous concepts that where mentioned, this section describes how to create an ontology of Argentinean music, dances that originate from it, and the provinces where they are played.

5.1 Domain of interest

There are several reasons that justify our election for ontology domain. Music provides us of many useful examples, but by itself it is too broad and complex a topic. Therefore, we restrain the domain by limiting music to a single country's traditional music. The country of choice in our case will be Argentina.

5.2 Ontology Language

The language we chose to use for the development this example was Protègè-OWL. Protègè is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protègè implements a rich set of knowledge-modeling structures and actions that support the creation,

visualization, and manipulation of ontologies in various representation formats. The Protègè-OWL editor is an extension of Protègè that supports the Web Ontology Language (OWL). OWL is the most recent development in standard ontology languages, endorsed by the World Wide Web Consortium (W3C) to promote the Semantic Web vision. A detailed tutorial on Protègè-OWL goes beyond the intended purpose of this essay. We refer the reader to [5] in order to gain a detailed explanation on the use of Protègè-OWL.

5.3 Creating our Ontology

The first step we must take after we decide to create an ontology, is to try to find existing ontologies that relate to our domain of interest, and reuse them. Protègè-OWL allows us to reuse existing ontologies, by importing them in our project. By taking this approach, we accomplish quality and simplicity in the ontology creation process. Creating an ontology from scratch is usually reserved for ad-hoc and new, less common domains.

For our ontology domain, there are already some ontologies developed that where considered in the creation process:

- The Music Ontology — http://purl.org/ontology/mo/
- The MusicBrainz Ontology — http://www.purl.org/net/MusicInstruments

However, we decided not to include the first one, given the fact that it surpassed our ontology's intended purpose. The Music Ontology covers a wide range of music related terms, but does not develop each term to its full extent. This is highly understandable, since the domain proves to be quite broad. In our case, we considered reusing the term "Musical Instrument" from an existing ontology. The music ontology's scope did not go beyond the inclusion of such a term. On the other hand, the MusicBrainz ontology did develop the term, so we imported and expanded it by including a number of traditional instruments that were not present in the original ontology.

5.3.1 The Onlotogy

We will now give a detailed explanation of each term that is included in our onlotogy, an how they relate to each other.

1. Classes

    - **Argentina**
      Represents a South American country by the same name. Argentina is constituted as a federation of twenty-three provinces and an autonomous city. This class contains a subclass "**Provinces**", which in turn contains an individual representing each Argentinean province.

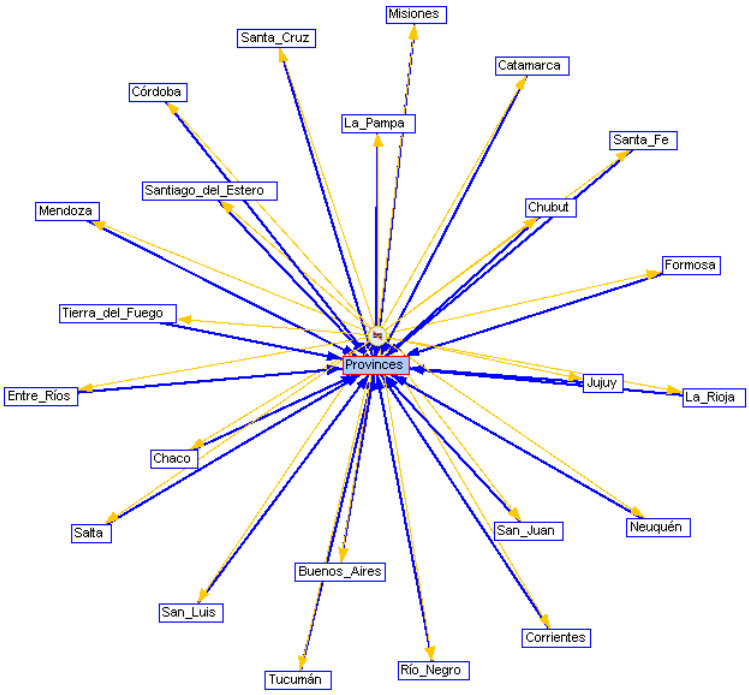


Figure 1: Provinces

- **Folklore Music**

  Represents every Argentinean traditional music genre and contains a subclass for each one.

- **Folklore Dances**

  Represents every Argentinean traditional dance genre, and contains a subclass for each one.

- **Instrument**

  Represents a set of musical instruments. Each instrument corresponds to an instrument individual from the MusicBrainz ontology. Such a correspondence is made by linking our Instrument class individuals to MusicBrainz instrument individuals by the use of annotation properties, in particular, the rdfs:isDefinedBy property. Thus the expression **S rdfs:isDefinedBy O** states that the resource O defines S.

2. Disjoint Classes

   Having added the previously mentioned classes to our ontology, we must specify that these classes are *disjoint*, so that an individual cannot be an instance of more than one of them.

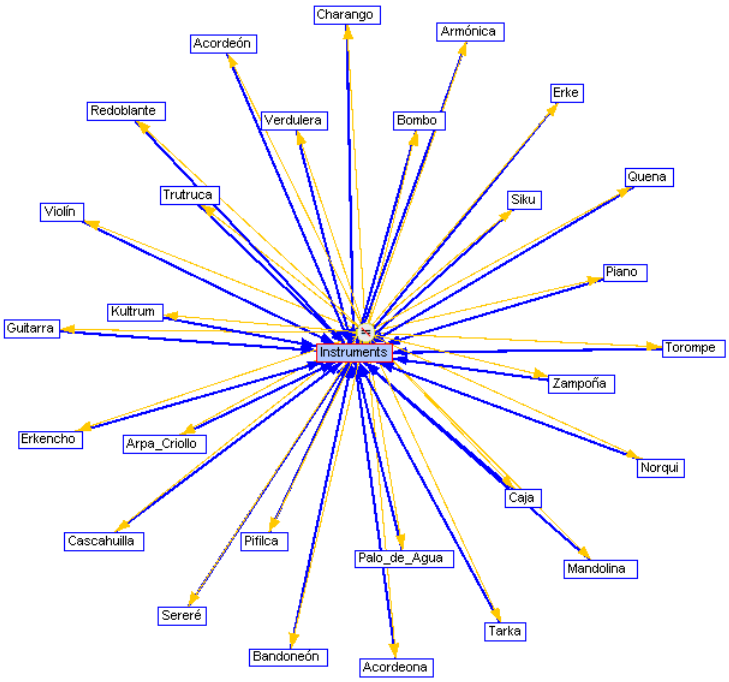


Figure 2: Instruments

3. Properties

   - **has_instrument**
     Determines whether a specific musical or dance genre, requires a particular musical instrument.
   - **played_in**
     Determines whether a specific musical genre, is played in a particular Argentinean province.
   - **danced_in**
     Determines whether a specific dance genre, is played in a particular Argentinean province.
   - **has_province**
     Determines whether a country, in our case "Argentina", has a particular province.
   - **is_partOf**
     Determines whether a province, is part of a particular country.

4. Restrictions

   In order to make sure that properties are used correctly, we include restrictions in order to restrict the individuals that belong to a class.

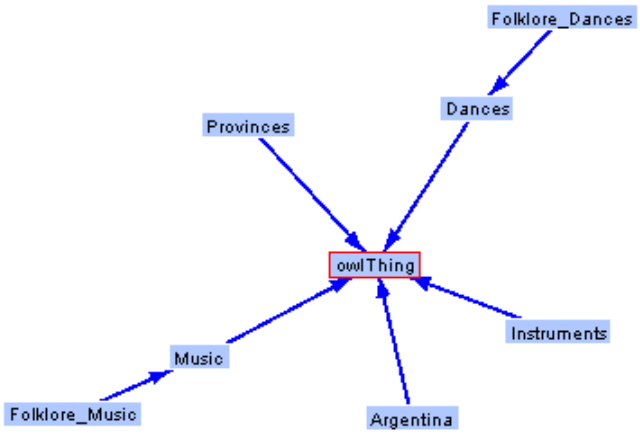


Figure 3: Argentinean music Ontology Classes

Quantifier Restrictions

Universal restrictions are given the symbol ∀. They *constrain* the relationships along a given property to individuals that are members of a specific class.

- ∀ **played_in only Provinces**
  In this way we establish that a certain individual can only have a relationship along the "played_in" property to a member of the Provinces class.
- ∀ **danced_in only Provinces**
  In this way we establish that a certain individual can only have a relationship along the "danced_in" property to a member of the Provinces class.
- ∀ **has_instrument only Instrument**
  In this way we establish that a certain individual can only have a relationship along the "has_instrument" property to a member of the Instrument class.
- ∀ **has_province only Provinces**
  In this way we establish that a certain individual can only have a relationship along the "has_province" property to a member of the Provinces class.
- ∀ **is_partOf only Argentina**
  In this way we establish that a certain individual can only have a relationship along the "is_partOf" property to a member of the Argentina class.

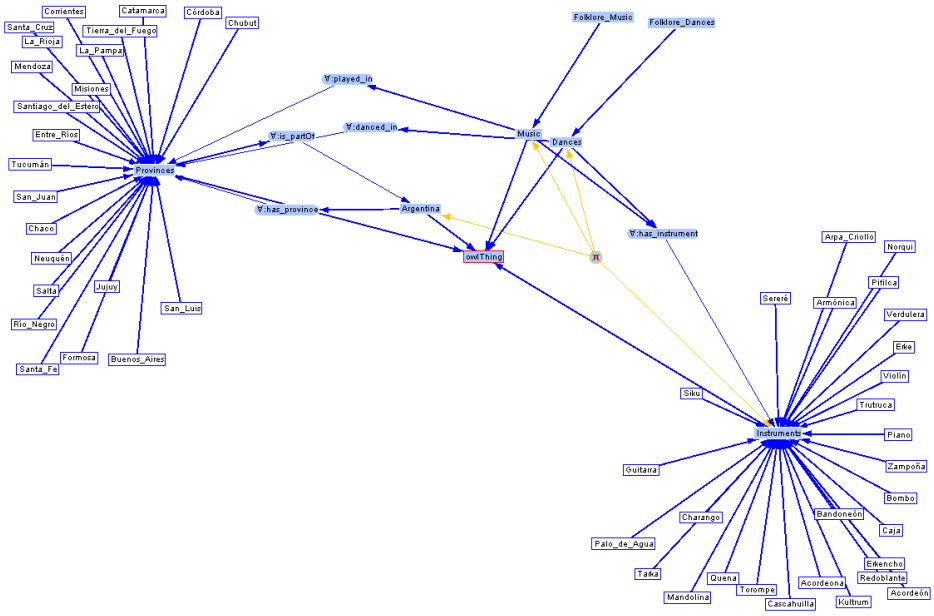


Figure 4: Argentinean music Ontology where Individuals are shown in white whilst classes are shown in blue

Has Value Restrictions

A hasValue restriction, denoted by the symbol ∋, describes the set of individuals that have *at least one* relationship along a specified property to a *specific individual*.

- ∋ **played_in has <a province>**
  where <a province> represents an individual from the Province class In this way we establish that a certain individual has the property of "being played" in a specified province. We must explicitly include such a restriction for each province individual where a musical genre is played.
- ∋ **danced_in has <a province>**
  where <a province> represents an individual from the Province class In this way we establish that a certain individual has the property of "being danced" in a specified province. We must explicitly include such a restriction for each province individual where a dance is danced.
- ∋ **has_instrument has <an instrument>**
  where <an instrument> represents an individual from the Instrument class In this way we establish that a certain individual has

the property of "having" a specified musical instrument. We must explicitly include such a restriction for each instrument individual that is used in a musical genre or dance.

Complex concepts can therefore be built up in definitions out of simpler concepts. Furthermore, the logical model we build with Protègè-OWL allows the use of a reasoner which can check whether or not all of the statements and definitions in the ontology are mutually consistent and can also recognize which concepts fit under which definitions. The reasoner can therefore help to maintain the hierarchy correctly. This is particularly useful when dealing with cases where classes can have more than one parent.

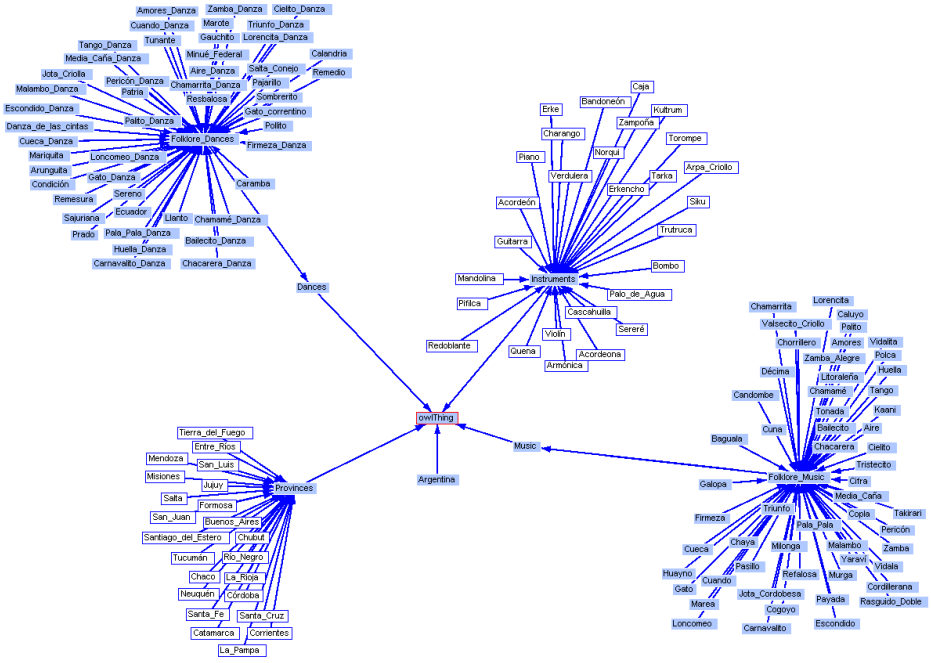


Figure 5: Complete Argentinean music ontology

6 Use cases of web ontologies

On February 10th 2004 W3C emitted a recommendation addressing OWL's requirements along with several use cases[1]. This section quotes six representative use cases of web ontologies as described in the previously mentioned document. Note that this is not an exhaustive list, but instead a cross-section of interesting use cases.

6.1 Web portals

A web portal is a web site that provides information content on a common topic, for example a specific city or domain of interest. A web portal allows individuals that are interested in the topic to receive news, find and talk to one another, build a community, and find links to other web resources of common interest.

In order for a portal to be successful, it must be a starting place for locating interesting content. Typically this content is submitted by members of the community, who often index it under some subtopic. Another means of collecting content relies on the content providers tagging the content with information that can be used in syndicating it. Typically, this takes the form of simple metatags that identify the topic of the content, etc.

However, a simple index of subject areas may not provide the community with sufficient ability to search for the content that its members require. In order to allow more intelligent syndication, web portals can define an ontology for the community. This ontology can provide a terminology for describing content and axioms that define terms using other terms from the ontology. For example, an ontology might include terminology such as "journal paper," "publication," "person," and "author." This ontology could include definitions that state things such as "all journal papers are publications" or "the authors of all publications are people." When combined with facts, these definitions allow other facts that are necessarily true to be inferred. These inferences can, in turn, allow users to obtain search results from the portal that are impossible to obtain from conventional retrieval systems. Such a technique relies on content providers using the web ontology language to capture high-quality ontology relationships, and an objective of OWL is to enable sufficiently rich and useful metadata content to motivate the necessary effort. It is a separate challenge to minimize this effort and an ontology language will likely have a greater impact if it can facilitate metadata capture as an nonintrusive part of any information creation process.

One example of an ontology based portal is OntoWeb. This portal serves the academic and industry community that is interested in ontology research. Another example of a portal that uses Semantic Web technologies and could benefit from an ontology language is The Open Directory Project; a large, comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors. RDF dumps of the Open Directory database are available for download.

6.2 Multimedia collections

Ontologies can be used to provide semantic annotations for collections of images, audio, or other non-textual objects. It is even more difficult for

machines to extract meaningful semantics from multimedia than it is to extract semantics from natural language text. Thus, these types of resources are typically indexed by captions or metatags. However, since different people can describe these non-textual objects in different ways, it is important that the search facilities go beyond simple keyword matching. Ideally, the ontologies would capture additional knowledge about the domain that can be used to improve retrieval of images. Multimedia ontologies can be of two types: media-specific and content-specific. Media specific ontologies could have taxonomies of different media types and describe properties of different media. For example, video may include properties to identify length of the clip and scene breaks. Content-specific ontologies could describe the subject of the resource, such as the setting or participants. Since such ontologies are not specific to the media, they could be reused by other documents that deal with the same domain. Such reuse would enhance search that was simply looking for information on a particular subject, regardless of the format of the resource. Searches where media type was important could combine the media-specific and content-specific ontologies. As an example of a multimedia collection, consider an archive of images of antique furniture. An ontology of antique furniture would be of great use in searching such an archive. A taxonomy can be used to classify the different types of furniture. It would also be useful if the ontology could express definitional knowledge. For example, if an indexer selects the value "Late Georgian" for the style/period of (say) an antique chest of drawers, it should be possible to infer that the data element "date.created" should have a value between 1760 and 1811 A.D. and that the "culture" is British. Availability of this type of background knowledge significantly increases the support that can be given for indexing as well as for search. Another feature that could be useful is support for the representation of default knowledge. An example of such knowledge would be that a "Late Georgian chest of drawers," in the absence of other information, would be assumed to be made of mahogany. This knowledge is crucial for real semantic queries, e.g. a user query for "antique mahogany storage furniture" could match with images of Late Georgian chests of drawers, even if nothing is said about wood type in the image annotation.

6.3 Corporate web site management

Large corporations typically have numerous web pages concerning things like press releases, product offerings and case studies, corporate procedures, internal product briefings and comparisons, white papers, and process descriptions. Ontologies can be used to index these documents and provide better means of retrieval. Although many large organizations have a taxonomy for organizing their information, this is often insufficient. A single ontology is often limiting because the constituent categories are likely con-

strained to those representing one view and one granularity of a domain; the ability to simultaneously work with multiple ontologies would increase the richness of description. Furthermore, the ability to search on values for different parameters is often more useful than a keyword search with taxonomies.

An ontology-enabled web site may be used by:

- A salesperson looking for sales collateral relevant to a sales pursuit.
- A technical person looking for pockets of specific technical expertise and detailed past experience.
- A project leader looking for past experience and templates to support a complex, multi-phase project, both during the proposal phase and during execution.

A typical problem for each of these types of users is that they may not share terminology with the authors of the desired content. The salesperson may not know the technical name for a desired feature or technical people in different fields might use different terms for the same concept. For such problems, it would be useful for each class of user to have different ontologies of terms, but have each ontology interrelated so translations can be performed automatically.

Another problem is framing queries at the right level of abstraction. A project leader looking for someone with expertise in operating systems should be able to locate an employee who is an expert with both Unix and Windows. One aspect of a large service organization is that it may have a very broad set of capabilities. But when pursuing large contracts these capabilities sometimes need to be assembled in new ways. There will often be no previous single matching project. A challenge is to reason about how past templates and documents can be reassembled in new configurations, while satisfying a diverse set of preconditions.

6.4 Design documentation

This use case is for a large body of engineering documentation, such as that used by the aerospace industry. This documentation can be of several different types, including design documentation, manufacturing documentation, and testing documentation. These document sets each have a hierarchical structure, but the structures differ between the sets. There is also a set of implied axes which cross-link the documentation sets: for example, in aerospace design documents, an item such as a wing spar might appear in each. Ontologies can be used to build an information model which allows the exploration of the information space in terms of the items which are represented, the associations between the items, the properties of the items,

and the links to documentation which describes and defines them (i.e., the external justification for the existence of the item in the model). That is to say that the ontology and taxonomy are not independent of the physical items they represent, but may be developed/explored in tandem.

A concrete example of this use case is design documentation for the aerospace domain, where typical users include:

- Maintenance engineer looking for all information relating to a particular part (e.g., "wing-spar").
- Design engineer looking at constraints on re-use of a particular subassembly.

To support this kind of usage, it is important that constraints can be defined. These constraints may be used to enhance search or check consistency. An example of a constraint might be:

```
biplane(X) => CardinalityOf(wing(X)) = 2
wingspar(X) AND wing(Y) AND isComponentOf(X,Y) => length(X) < length(Y)
```

Another common use of this kind of ontology is to support the visualization and editing of charts which show snapshots of the information space centered on a particular concept (e.g., a class or instance). These are typically activity/rule diagrams or entity-relationship diagrams.

6.5 Agents and services

The Semantic Web can provide agents with the capability to understand and integrate diverse information resources. A specific example is that of a social activities planner, which can take the preferences of a user (such as what kinds of films they like, what kind of food they like to eat, etc.) and use this information to plan the user's activities for an evening. The task of planning these activities will depend upon the richness of the service environment being offered and the needs of the user. During the service determination / matching process, ratings and review services may also be consulted to find closer matches to user preferences (for example, consulting reviews and rating of films and restaurants to find the "best"). This type of agent requires domain ontologies that represent the terms for restaurants, hotels, etc. and service ontologies to represent the terms used in the actual services. These ontologies will enable the capture of information necessary for applications to discriminate and balance among user preferences. Such information may be provided by a number of sources, such as portals, service-specific sites, reservation sites and the general Web. Agentcities is an example of an initiative that is exploring the use of agents in a distributed service environment across the Internet. This will involve building a network of agent platforms

that represent real or virtual cities, such as San Francisco or the Bay Area, and populating them with the services of those cities. Initially, these services will be oriented towards business to consumer services, such as hotels, restaurants, entertainment, etc., but eventually, they will be expanded to include business to business services, such as payroll, and business to enterprize services. This will require a number of different domain and service ontologies: Key issues include:

- Use and integration of multiple separate ontologies across different domains and services
- Distributed location of ontologies across the Internet
- Potentially different ontologies for each domain or service (ontology translation/cross-referencing)
- Simple ontology representation to make the task of defining and using ontologies easier

6.6 Ubiquitous computing

Ubiquitous computing is an emerging paradigm of personal computing, characterized by the shift from dedicated computing machinery to pervasive computing capabilities embedded in our everyday environments. 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. An additional reason for development of automatic configuration is that this technology is aimed at ordinary consumers. A key technology of true ad hoc networks is service discovery, functionality by which "services" (i.e., functions offered by various devices such as cell phones, printers, sensors, etc.) can be described, advertised, and discovered by others. All of the current service discovery and capability description mechanisms (e.g., Sun's JINI, Microsoft's UPnP) 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). The key issue (and goal) of ubiquitous computing is "serendipitous interoperability," interoperability under "unchoreographed" conditions, i.e., devices which weren't necessarily designed to work together (such as ones built for different purposes, by different manufacturers, at a different time, etc.) should be able to discover each others' functionality and be able to take advantage of it. Being able to "understand" other devices, and reason about their services/functionality is necessary, since full-blown ubiquitous computing scenarios will involve dozens if not hundreds of devices, and a priori standardizing the usage scenarios is an unmanageable task. The interoperation scenarios are dynamic in nature (i.e., devices appear and disappear at any moment as their owners carry them from one room or building

to another) and do not involve humans in the loop as far as (re-)configuration is concerned. The tasks involved in the utilization of services involve discovery, contracting, and composition. The contracting of services may involve representing information about security, privacy and trust, as well as about compensation-related details (the provider of a service may have to be compensated for services rendered). In particular, it is a goal that corporate or organizational security policies be expressed in application-neutral form, thus enabling constraint representation across the diversity of enforcement mechanisms (e.g., firewalls, filters/scanners, traffic monitors, application-level routers and load-balancers). Thus, an ontology language will be used to describe the characteristics of devices, the means of access to such devices, the policy established by the owner for use of a device, and other technical constraints and requirements that affect incorporating a device into a ubiquitous computing network. The needs established for DAML-S (particularly the issues surrounding the expressiveness of the language) and the RDF-based schemes for representing information about device characteristics (namely, W3C's Composite Capability/Preference Profile (CC/PP) and WAP Forum's User Agent Profile or UAProf) directly relate to this use case and the resource infrastructure which will support applications which will negotiate and dynamically configure ad hoc networks.

6.7 Music Ontology related use cases

Until now we have described the basic steps involved in the development of our Argentinean music ontology. Despite being highly specific, such an ontology can be expanded and used in applications that carry out complex music related searches.

Shazam Entertainment, an English company that specializes in music-recognition software, has created a music-recognition system, that allows people to identify tunes using their cell phones. When you hear a song – on the radio, in a bar or on television – you punch in the Shazam service's four-digit code number on the handset, point the phone at the source of the sound and hold it there for 15 seconds- Within a few minutes, the service should return a text message giving the name of the song and the artist. When a user with a mobile phone inputs 15 seconds of music, the system creates a digital signature for that snippet and then looks for a matching pattern in the database. To minimize the retrieval speed, all the data are kept in active memory (rather than on hard disks) on a distributed computer system consisting of about 70 PCs. The searches involved in systems such as Shazam focus mainly on retrieving a song's name and artist, and could benefit greatly of a music ontology similar to the one we have developed in order to expand their current data and search complexity. Consequently a user could be able to retrieve not only a song's name and artist, but every artist that has made a version of the song, whether they have scheduled

concerts near the user's current location, or any other related information of interest. An example scenario in which a user is interested in upcoming performances of the unknown artist que is listening to at the moment within a 5 mile radius is depicted in figure 6.

Thus, by embedding these applications in electronic devices with some form of wireless networking capability, such as mobile phones, we can significantly enrich their current functionality.

7 Agents

The real power of the Semantic Web will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine-readable Web content and automated services (including other agents) become available. The Semantic Web promotes this synergy: even agents that were not expressly designed to work together can transfer data among themselves when the data comes with semantics [2].

An important facet of agents functioning will be the exchange of "proofs" written in the Semantic Web's unifying language (the language that expresses logical inferences made using rules and information such as those specified by ontologies). Another vital feature will be digital signatures, which are encrypted blocks of data that computers and agents can use to verify that the attached information has been provided by a specific trusted source. Agents should be skeptical of assertions that they read on the Semantic Web until they have checked the sources of information.

In the Semantic Web, the consumer and producer agents can reach a shared understanding by exchanging ontologies, which provide the vocabulary needed for discussion. Agents can even "bootstrap" new reasoning capabilities when they discover new ontologies. Semantics also makes it easier to take advantage of a service that only partially matches a request.

A typical process will involve the creation of a "value chain" in which subassemblies of information are passed from one agent to another, each one "adding value" to construct the final product requested by the end user. To create complicated value chains automatically on demand, some agents will exploit artificial-intelligence technologies in addition to the Semantic Web. But the Semantic Web will provide the foundations and the framework to make such technologies more feasible.

In the next step, the Semantic Web will break out of the virtual realm and extend into our physical world. URIs can point to anything, including physical entities, which means we can use the RDF language to describe devices such as cell phones and TVs. Such devices can advertise their functionality, what they can do and how they are controlled, much like software

agents. Such a semantic approach opens up a world of exciting possibilities.

8 Conclusion

The potential implications of widespread adoption of semantic web technologies, promises a knowledge revolution. If properly designed, it would not only become a tool for conducting individual tasks, but it would also assist the evolution of human knowledge as a whole, gaining access to our everyday life through personal computers and handsets. Once the web has been sufficiently populated with rich metadata, searching on the web will become easier as search engines have more information available, and thus searching can be more focused. Also, searching the web will be less time-consuming, given the fact that software agents will do it instead. The web of today, the vast unstructured mass of information, may in the future be transformed into something more manageable - and thus something far more useful, allowing agents and users to work and learn together.

References

[1] Owl web ontology language use cases and requirements. *W3C Recommendation*, February 2004.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 2001.

[3] Edd Dumbill. The semantic web: A primer. *www.XML.com*, November 2000.

[4] W. Scott Means Elliotte Rusty Harold. *XML in a Nutshell*. O'Reilly, June 2002.

[5] Alan Rector Robert Stevens Chris Wroe Matthew Horridge, Holger Knublauch. A practical guide to building owl ontologies using the protege-owl plugin and co-ode tools. August 2004.

[6] Eric Miller. An introduction to the resource description framework. *D-Lib Magazine*, May 1998.

[7] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology, September 2005.

[8] W3C Recommendation. Owl web ontology language, February 2004.

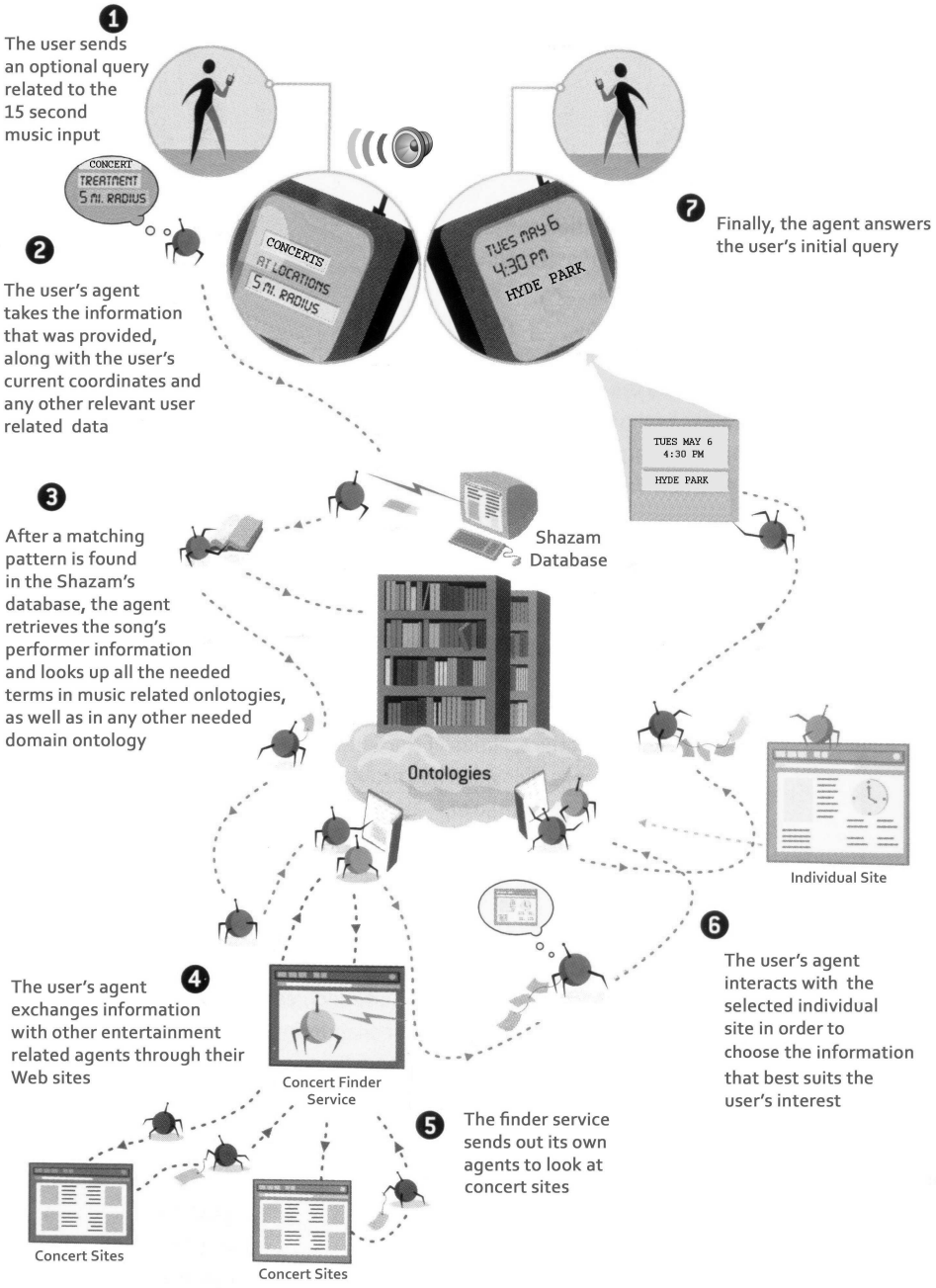


Figure 6: Upcoming concerts within a 5 mile radius query.