Developing a Repository of Knowledge for Virtual Communities with Semantic Web Technologies

M. Clara Casalini, Elsa Estevez and Pablo Fillottrani
Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur
Bahía Blanca, 8000, Buenos Aires, Argentina
{mcca, ece, prf}@cs.uns.edu.ar

Abstract

The appearance and continuous growth of virtual communities on the web imposes the challenge of coping with the large amount of information and knowledge these communities produce. The Semantic Web with its set of standards and technologies provides the basic means for implementing repositories of knowledge for virtual communities. However not every member of a virtual community knows these technologies and is ready to use them. We present an implementation of a repository of knowledge that follows the Resource Description Framework enabling a standard representation of knowledge in a community and providing the necessary functionality for members of the community to manage this repository introducing information and relating this information in a simple manner that also facilitates the interoperability with other repositories.

Keywords: Virtual Communities, Knowledge Management, Semantic Web, Interoperability, Web Standards

1 INTRODUCTION

A virtual community is defined in [12] as "an aggregation of individuals or business partners who interact around a shared interest, where the interaction is at least partially supported and/or mediated by technology". Whether oriented to business, entertainment or education, groups of people who gather to share and interact sometimes without even knowing each other and living very far from the rest are becoming more popular every day. Many of these communities share a set of assets which is developed by the members of the community and becomes larger and more vital for the community as it grows. The spread of virtual communities leads researchers to the task of finding the best way to represent these assets and to implement a virtual repository for their knowledge.

The Semantic Web is emerging as a second, enhanced version of the World Wide Web whose mission is not only to hold information, but to allow sharing and reusing that information, giving it a unique meaning and providing the means for automating the tasks related to processing that information. In this context, the work under development to define what the Semantic Web is and how it is going to be implemented seems like the ideal framework for the applications needed by virtual communities.

This paper presents an implementation of a repository of knowledge to support virtual communities. The underpinning technologies of its design and implementation are the technologies being developed for the Semantic Web. Section 2 presents and explains these technologies together with some examples of web sites which are already applying them. The development of the repository is presented in section 3. The complete process is explained from the requirements elicitation (section 3.1), through domain and use case modeling (sections 3.2 and 3.3), architectural and detailed design (sections 3.4 and 3.5) to the implementation (section 3.6). Section 4 is dedicated to showing

a concrete application of the system developed. An exploration of the related work and some the conclusions are given in section 5.

2 RDF AND SEMANTIC WEB

Currently, the World Wide Web is a place for sharing human-oriented information, like documents, pictures, or multimedia files. The Semantic Web [2, 4] is an evolving extension of the Web which provides a common framework that allows all kinds of data to be shared and reused across applications, enterprises and community boundaries. In this way, data would also be computer-understandable, enabling software agents to help humans with the tedious work of finding, sharing and combining information on the web, boosting B2B applications and e-commerce.

Semantic Web technologies are being developed by the World Wide Web Consortium (W3C) [1], mostly in the form of formal specifications. However, they are still very much in their infancies and their applications are limited. It comprises the standards of XML [11], XML Schema [20], RDF and RDF Schema [7, 3], and OWL [8], in a layered way (see Figure 1). XML provides an elemental syntax for content structure within documents, yet associates no semantics with the meaning of the content contained within. XML Schema is a language for restricting the structure and content of elements contained within XML documents, providing also standard datatype definitions. RDF is a simple language for expressing data models, which refers to objects (which are called resources and represented by URIs) and their relationships. An RDF-based model can be represented in XML syntax. RDF Schema is a vocabulary for describing properties and classes of RDF-based resources, with semantics for generalized-hierarchies of such properties and classes. Finally, 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. Thus, higher layers include languages with more expressivity but at the same time with more difficult inference procedures, belonging to harder complexity classes. The appropriate language for a given Semantic Web application is decided by making balance between being semantically more specific (and thus improving interoperability of data), and the time and space needed to evaluate the data.

RDF data model provides the elementary tools to model information, by defining ontologies which are shared conceptualizations of terms from a given domain. The RDF metadata model is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, called triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, one way to represent the notion “This document is in English” in RDF is as a triple of specially formatted strings: a subject denoting ”this document” a predicate denoting “has language”, and an object denoting “English”. Currently most known applications of RDF are RDF Site Summary [13], one of several RSS languages for publishing information about updates made to a web page, which is often used for disseminating news article summaries and sharing weblog content; FOAF (Friend of a Friend) [17] designed to describe people, their interests and interconnections; Music Brains [9] that publishes information about music albums, and SIOC (Semantically-Interlinked Online Communities) [14] designed to describe online communities and to create connections between Internet-based discussions from message boards, weblogs and mailing lists.

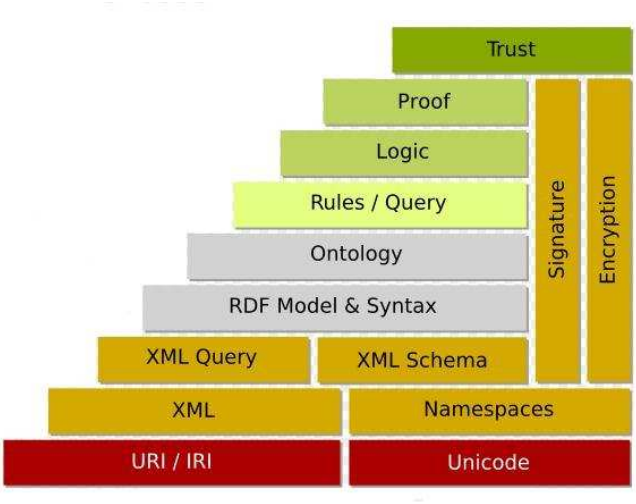


Figure 1: Semantic Web Technologies

3 REPOSITORY DEVELOPMENT

This section presents the development of the repository of knowledge for virtual communities. Six development stages are explained: Requirements Elicitation, Domain Modeling, Use Case Modeling, Architectural Design, Detailed Design, and Implementation. The repository is implemented using Semantic Web technologies, particularly the Resource Description Framework, the language for representing information about resources in the Web. The main advantage of the implementation approach is the representation of information in a standard way enabling interoperability with others repositories of knowledge through sharing and exchanging information.

3.1 Requirements Elicitation

The functional requirements for managing the repository are related with three main functions:

- *modifying the repository* - adding, removing and changing its components,
- *exploring the repository* - performing searches and browsing,
- *managing users* - administering user registrations and privileges.

Table 1 presents the requirements for modifying the repository. These requirements enable to create and remove type categories from the category hierarchy, register and remove resource types, and add and remove properties and statements. Type categories can be of two kinds: resource type categories - such as Conference, Book, Paper, etc; and data type categories - such as Integer, Text, Figure, etc. Users can add a type category by simply providing its name (CAT01), making the category a child of the *root* category. However, it is possible to create a category as a sub-category of an existing one (CAT02). In the latter case the user provides the name for the new category and the name of the parent category. Users can decide to remove categories (CAT03), although this operation can only be performed if no resources of the type category exist in the repository and no properties are defined with domain or range of that type category. In addition, properties can be defined by indicating its identifier name and the type categories for its domain and range - the domain must be a resource type category and the range may be both a resource or data type category. Two operations for defining

(PRO01) and removing (PRO02) properties are provided. A property can be removed only if there are no statements in the repository holding the property to be removed as predicate.

Table 1: Functional Requirements for Modifying the Repository

CAT01	Creating a type category
CAT02	Creating a type sub-category
CAT03	Removing an existing type category
PRO01	Defining a new property
PRO02	Removing an existing property
RES01	Registering a new resource to the repository
RES02	Removing an existing resource
STA01	Writing a new statement
STA02	Removing an existing statement

The repository is enriched by its users by incorporating new resources and discovering information and relationships about and among them. A new resource is registered by providing a name to identify it and the resource type category to which it belongs (RES01). It is possible to remove a resource (RES02) only in the case that no statements describing it or relating it to other resources exist in the repository. Knowledge acquired concerning a resource as well as relationships to other resources are added to the repository in the form of statements. A statement is added (STA01) providing the subject -resource about which the statement is reflecting information-, the predicate -property defined in the repository-, and the object -value of the property for the subject being described-. An operation for removing statements is also provided (STA02).

Table 2: Functional Requirements for Navigating the Repository

BRO01	Browsing categories
BRO02	Browsing resources
BRO03	Browsing properties
BRO04	Browsing statements
SEA01	Searching resources from a category
SEA02	Searching statements about a resource
SEA03	Searching statements with a predicate
SEA04	Searching properties with a particular domain
SEA05	Searching properties with a particular range

Requirements for exploring the repository are listed in Table 2 including operations for two main types of exploration: browsing the content and performing predefined searches. Browsing the repository content consists in obtaining a list of all the elements currently available in the repository. Users can browse the defined categories (BRO01), the set of registered resources (BRO02), the defined properties (BRO03), and the set of existing statements (BRO04). A group of predefined search operations is provided enabling to perform specific searches. Resources can be searched according to their category (SEA01); statements according to a particular resource they may have as subject (SEA02) or to a particular property as predicate (SEA03). Searches over properties can be executed according to their domain (SEA04) or their range (SEA05).

Table 3 presents the requirements for managing users. Users are responsible for updating the repository, therefore some restrictions and privileges are defined for them. Casual visitors of the

website including a link to the repository are allowed to explore the repository. However, operations involving changing the repository state is reserved for registered users, called members. Moreover, the creation and removal of categories can only be done by a special user playing the role of administrator. Managing such privileges implies the need to provide functions for registering (USE01) and un-registering (USE02) users; and for enabling users to log -in (USE03) and -out (USE04) of the application.

Table 3: Functional Requirements for Managing Users

USE01	Registering user
USE02	Un-registering user
USE03	Logging in to the application
USE04	Logging-out from the application

3.2 Domain Modeling

The development aim is to follow the RDF recommendation for implementing the repository. RDF, as a language, was designed for representing information about web resources. In RDF, resources are referenced using Uniform Resource Identifiers (URIs) [19], and described in terms of properties and values through statements. RDF statements are triples consisting of a *subject*, a *predicate* and an *object*. The *subject* is the resource being described by the statement; the *predicate* represents the property, characteristic or relation that is being assigned to the subject; and the *object* is the value of the mentioned property. Properties are described by assigning each its own URI and providing the domain and range of the property. The property domain identifies the resource type that will be described by such property, and the property range represents the data type identifying the allowed values for the property. While the domain must always be a resource type, the range can be a resource as well as a data type, thus providing greater flexibility.

The concepts explained above are modeled as follows: a repository is formed by the resources added by the users and the information gathered about these resources. The information about resources is stored by defining properties and statements. For modeling resource and data types, an abstract category type is introduced for representing both sorts of types.

The conceptual model is shown in Figure 2 depicting the main concepts of the repository and their relations. Two type categories exist: *Resource Type* and *Data Type Categories*. A *Type Category* is a subcategory of another category. *Resource Type Categories* provide the types for resources, while *Data Type Categories* provide types for simple data. *Resources* and *Data* are generalized in the class *Value*. A *Property* is composed of a *Range* and a *Domain*, both of them are categories. The *Domain* must be a *Resource Type Category* however the *Range* can be any *Type Category*. *Statements* have three components: *Subject*, *Predicate* and *Object*. The *Subject* is an instance of a *Resource*; the *Predicate* is a *Property* in the repository and the *Object* is a *Value* instance, i.e. a resource or plain data (depending on the definition of the predicate). A *Statement* describes a *Resource*, and one *Resource* can be described by many *Statements*. A *Repository* holds a set of *Resources*, a set of *Properties* and a set of *Statements* as well as a hierarchy of *Type Categories*. *Users* interact with the *Repository* and can be *Registered* or *Unregistered*. There is one special user among the registered users known as *Administrator*.

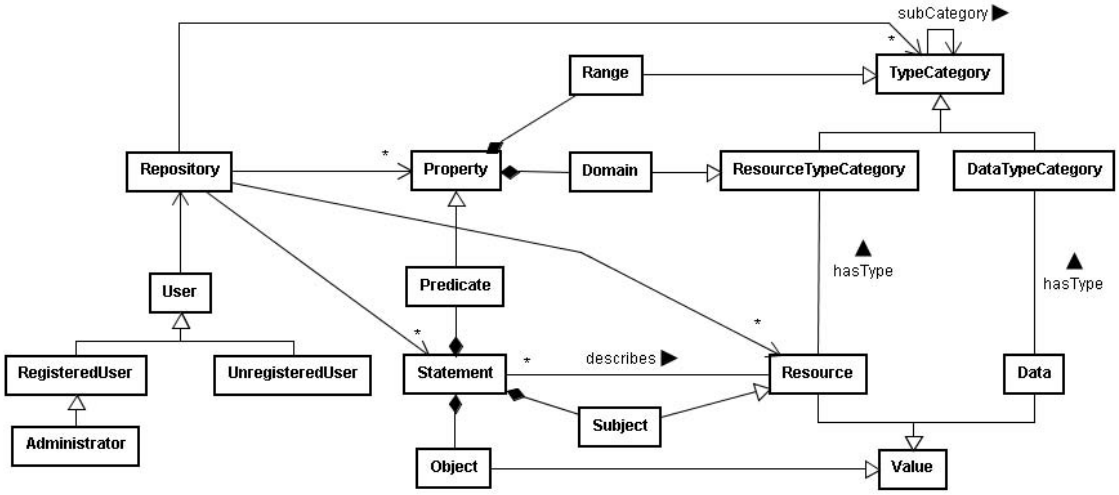
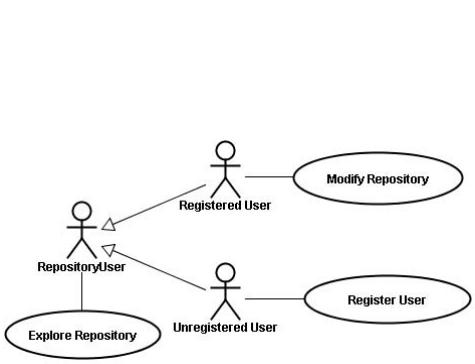


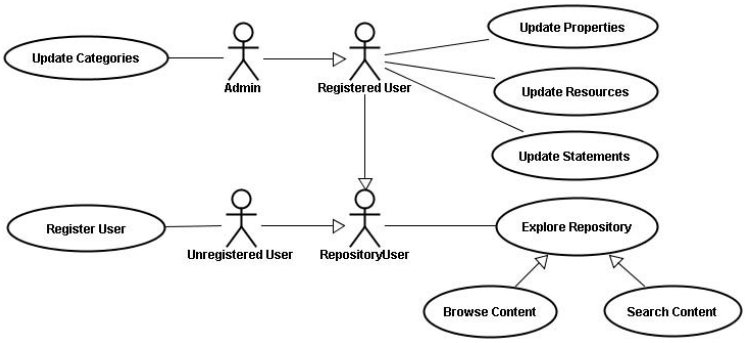
Figure 2: Conceptual Model

3.3 Use Case Modeling

This section presents the use case model which is depicted in Figure 3. Figure 3(a) presents the top level use case diagram showing the types of users interacting with the system and the available interactions. Every user (RepositoryUser) can use the repository to explore it (Explore Repository); not registered users (UnregisteredUser) can also make use of the registration feature (Register User); and every usage of the repository that implies changes over it (Modify Repository) can only be performed by registered users (Register User). Figure 3.3 shows Modify Repository and Explore Repository use cases in more details as well as the specialization of Registered User as the administrator (Admin) user.



(a) Top level use case



(b) Top level use case in more detail

Figure 3: Use Case Model

The Explore Repository use case involves two different use cases: Browse Content and Search Content. The Modify Repository use case has been broken down in four detailed functions: Update Categories, Update Properties, Update Resources and Update Statements. The first function can only be performed by the Admin user, while the others by any sort of Registered User.

3.4 Architectural Design

The application has a layered architecture composed of three tiers, as shown in Figure 4. The *Presentation Layer* includes the *JSP* pages used for displaying the repository content and supporting the interactions with the user. This layer requests services to the *Business Layer* who provides the corresponding functions and elements to populate the web pages. Thus, the *Business Layer* contains the components implementing the management of the repository and the management of users; providing functions for updating and searching the repository content and data about users. The *Persistency Layer* contains the component for managing the database. This component handles data persistency through a database, hence for storing and retrieving the repository content the *Business Layer* uses the *Persistency Layer*.

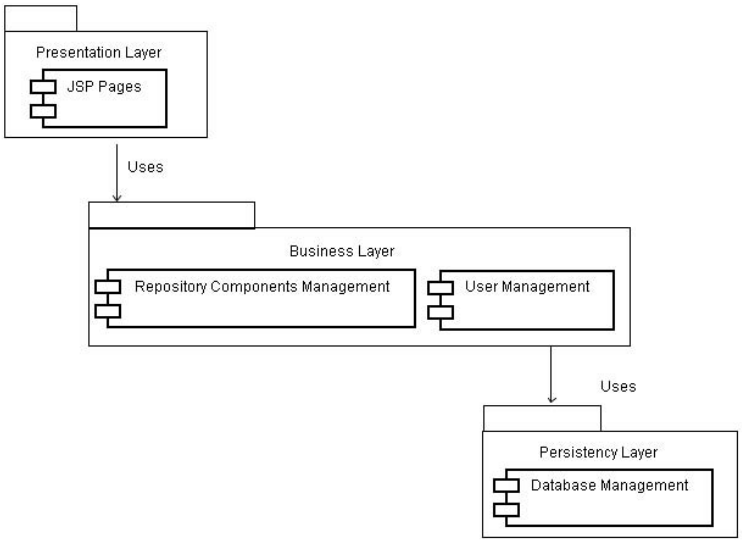


Figure 4: Architectural Static View

3.5 Detailed Design

The system design is shown in Figure 5. A static view of the system is provided by a design class diagram showing the classes implementing the main concepts described in the previous sections. A *Property* object can be identified by its name, and maintains the relationships with the the domain and range categories. A *Statement* keeps the triple relationship between the subject (*Resource*), predicate (*Property*) and object (*Value*). A *Resource* object only contains a name and the relationship to its own *Resource Type Category*. The *Repository* keeps one collection for each of these types of element: properties, statements and resources; plus two hierarchies of categories for resource type and for data type categories. The hierarchies are presented as a mapping from a category to the set of child categories. The root categories of both hierarchies are represented by the attributes resRoot and dataRoot.

3.6 Implementation

An initial implementation of this repository of knowledge is in place providing the repository services application for the virtual community of practice UNeGov.Net - Community of Practice for Electronic Governance [18]. The application is implemented in Java following the Apache Struts [16]

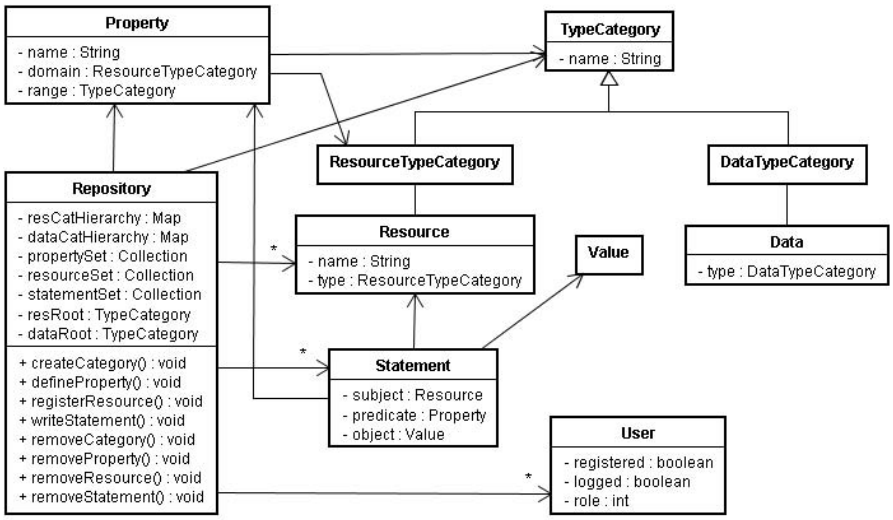


Figure 5: Design Class Diagram

framework. Persistence is obtained using MySQL [10] database, and Hibernate [6] is used for the object-relational mapping. The software is implemented as a Java Web Application and it needs to be deployed in a J2EE application server with a MySQL server. For the registration features to work as expected, a mail server is required for sending confirmation e-mails to the registered users.

4 APPLICATION

The presented repository of knowledge can be deployed as part of any portal or website. As mentioned before, a prototype has been deployed and is running as the repository of knowledge of UNeGov.Net - Community of Practice for Electronic Governance [18]. This version has been implemented including the following features:

a) User registration

b) User login and logout

c) Administration features

d) Browse content

e) Update content

Visitors arriving to the site can register themselves as members of the community, this process results in the user becoming a registered user of the community as well as becoming a resource of the repository under a special category created for members. Members is a sub-category of a more general category existing in the repository for general resource people. Once a person has registered as a member, he/she can log in into the system to update content, for instance defining new properties, registering new resources, modifying stored information about existing resources, writing statements, etc. Administrators of the site can also login into the site to perform operations reserved for them, like creating new categories, approving or denying the content updates requested by registered members, and assigning the administrator role to a registered user of the site. Browsing content allows any user

visiting the site to list the complete set of resources of the repository, or to filter the list by category. Properties can also be browsed, as well as the information regarding a particular resource, including the statements written about it. The left window in Figure 6 shows some of the currently available resources in the repository, while the right window shows how properties are displayed when browsed, together with the statements existing for a particular resource of category Paper.

Figure 6: Browsing the UneGov.Net Repository

5 CONCLUSIONS AND FUTURE WORK

The aim of this paper was to present a model and an implementation of a repository of knowledge for virtual communities based on Semantic Web technologies. This idea was motivated by the shortage of semantic web support for collaborative work on the web. The repository presented follows the Resource Description Framework standard and implements its main concepts. A detailed explanation of the design and development of the repository was presented and a concrete application of the system produced was shown: the Community of Practice for Electronic Governance (UNeGov.Net). The main advantage of this implementation is that it allows interoperability with other repositories or systems implementing Semantic Web technologies while keeping the design simple and clear for community users.

Semantic Web technologies are a subject of extensive interest among researchers, particularly in the area of knowledge management. Many applications and frameworks are being designed and built to use RDF and ontology languages like OWL, proving that the benefits from using these technologies are widely acknowledged. In [5] the author presents an ontology-based repository dedicated to software patterns, aiming to solve issues like establishing relationships between patterns and the dissemination of the patterns as well as the discovered relations among them. The work described in [15] pursues the introduction of ontologies for knowledge management in the field of web based learning to enable efficient reuse and sharing of knowledge. These approaches to knowledge management in the web, as well as our proposal, see ontologies as the backbone of the repositories being provided to the users to exploit and to rely on. They provide a solid tool to the automatization of knowledge discovery however such approaches do not provide the users the opportunity to be part of the creation of the vocabularies. The application presented in this work provides users with the mechanisms to enrich the repository through the addition and creation of new resources as well as through the creation of new ways to relate them and annotate them. It aims to reduce the shortage of support for these

kinds of procedures. As far as we know, there is no other application with such a general objective, promoting interoperability of resources through different communities.

Future work on this subject include considering the usage of ontologies to introduce the possibility of making inferences over the available information. In this sense, it will be necessary to represent, validate and extend our ontology using some OWL dialect.

Acknowledgments.

We wish to thank Tomasz Janowski for leadership, support and useful comments for developing this work. This work was partly supported by United Nations University - International Institute for Software Technology (UNU-IIST), Macao S.A.R. China, and Agencia Nacional de Promoción Científica, Argentina, through Project PAV-076/2003 "Intelligent Systems Supporting Productive Processes" sub-project: "Web Services and Intelligence for the WEB"

REFERENCES

[1] W3C. World Wide Web Consortium. http://www.w3.org/.

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

[3] Dan Brickley and Ramanathan V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[4] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel C. A. Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.

[5] Scott Henninger. Using the semantic web to construct an ontology-based repository for software patterns. *Workshop on the State of the Art in Automated Software Engineering*, pages 18–22, June 2002.

[6] Hibernate. relational persistence. http://www.hibernate.org/.

[7] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[8] Deborah L. McGuinness, Michael K. Smith, and Chris Welty. OWL web ontology language guide. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[9] MusicBrainz. http://musicbrainz.org/.

[10] Mysql open source database. http://www.mysql.com/.

[11] Jean Paoli, C. M. Sperberg-McQueen, and Tim Bray. XML 1.0 recommendation. first edition of a recommendation, W3C, February 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

[12] Constance Elise Porter. A typology of virtual communities: A multi-disciplinary foundation for future research. *Journal of Computer-Mediated Communication*, 10(1), 2004.

[13] RDF site summary (RSS) 1.0, December 2000. http://web.resource.org/rss/1.0/.

[14] SIOC project. semantically-interlinked online communities. http://sioc-project.org/.

[15] Bhavani Sridharan, Alexei Tretiakov, and Kinshuk. Application of ontology to knowledge management in web based learning. *Proceedings of IEEE International Conference on Advanced Learning Technologies*, pages 663–665, 2004.

[16] Struts. the apache software foundation. http://struts.apache.org/.

[17] The friend of a friend (FOAF) project. http://www.foaf-project.org/.

[18] UNeGov.Net. community of practice for electronic governance. http://www.unegov.net.

[19] Uniform resource identifier (URI): Generic syntax. http://gbiv.com/protocols/uri/rfc/rfc3986.html.

[20] Priscilla Walmsley and David C. Fallside. XML schema part 0: Primer second edition. W3C recommendation, W3C, October 2004. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.