AN ONTOLOGICAL SW ARCHITECTURE FOR THE DEVELOPMENT OF COOPERATIVE WEB PORTALS

Giacomo Bucci, Valeriano Sandrucci, Enrico Vicario
Dipartimento di Sistemi ed Informatica, Università degli Studi di Firenze
{bucci,sandrucci,vicario}@dsi.unifi.it

Saverio Mecca
Dipartimento Tecnologie dell'Architettura e Design, Università degli Studi di Firenze
saverio.mecca@unifi.it

Keywords: SW Architectures, Semantic Web Technologies, Ontologies, Semantic Portals, SW Engineering.

Abstract: Ontological technologies comprise a rich framework of languages and components off the shelf, which devise a paradigm for the organization of SW architectures with high degree of interoperability, maintainability and adaptability. In particular, this fits the needs for the development of semantic web portals, where pages are organized as a generic graph, and navigation is driven by the inherent semantics of contents. We report on a pattern-oriented executable SW architecture for the construction of portals enabling semantic access, querying, and contribution of conceptual models and concrete elements of information. By relying on the automated configuration of an Object Oriented domain layer, the architecture reduces the creation of a cooperative portal to the definition of an ontological domain model.

1 INTRODUCTION

Technologies for ontological modelling and reasoning devise a new paradigm for the organization of software architectures with high degree of interoperability, maintainability and adaptability (M. Fayad, 1996), (ISO, 2004). This potential appears particularly well suited for consistent development and management of information architecture, site structure, and page layout of web portals with weblike organization, where pages are organized in the pattern of a generic graph, and navigation is driven by the inherent semantics of contents more than from a hierarchical upfront classification (P.J. Lynch, 2002) (Franca Garzotto, 1995). In (Schreiber et al., 2006), a semantic portal based on standard ontological technologies and SWI-prolog is proposed, which supports unified access to cultural heritage resources classified according to a public unifying ontology. The portal supports semantic search and presentation of retrieved data, while contribution of contents is not considered. Contribution of contents by a distributed community based on ontologies is addressed in (Stojanovic et al., 2001), as a part of a work mainly focused on the determination of a rank of relevance for the result-set of semantic queries. However, the proposed architecture only permits contribution of ontology individuals and does not enables evolution and reuse of the implementation when the ontology changes in its concepts. In (Yuhui Jin, 2001), a declarative approach to the construction of semantic portals is proposed, which relies on the definition of a suite of ontologies created by the portal programmer to define domain concepts and contents, navigation structure and presentation style. The work does not address the subject of content contribution neither the personalization of presentation style by the user. In (Corcho et al., 2006), the declarative approach of (Yuhui Jin, 2001) is enlarged into a framework based on ontologies supporting the construction of a web application combining information and services. The framework implements the Model View Controller architectural pattern (Schmidt et al., 2000). While the model is declared using ontologies, views are implemented with existing presentation technologies, and in particular JSP, which mainly rely on the OO paradigm. To fill the gap between the two paradigms (Woodfield, 1997), the developer is provided with a suite of predefined JSP components assuming the responsibility of the adaptation.

In this paper, we address the development of a portal enabling semantic access, querying, and contribution of both domain individuals and concepts. To this end, we propose an executable SW architecture (Kruchten, 2003) based on standard languages and components off the shelf, which reduces the creation of a cooperative portal to the definition of an ontological domain model, and which is implemented with components that can be efficiently reused in a variety of data intensive and knowledge based applications. The programming model that permits the construction of a portal using the proposed architecture is illustrated with reference to the case of a cooperative portal, that we call Muddy, supporting distributed contribution and usage of knowledge about mudbrick construction practices, that we call Muddy

In the rest of the paper, after a brief overview of our perspective on the ontological paradigm in SW architecture, we introduce the Muddy case and we analyze its requirements, identifying abstract roles and use cases (Sect.2). We then expound the architectural design and some salient traits of its implementation, and we identify the roles involved in its application (Sect.3). Finally we describe the Muddy portal (Sect.4) and draw conclusions (Sect.5).

2 REQUIREMENTS ANALYSIS

Ontological technologies mainly originate with the intent to contribute to the realization of the Semantic Web (Berners-Lee, 1998). This denotes an evolution of the current web, in which information is sematically defined so as to enable automated processing.

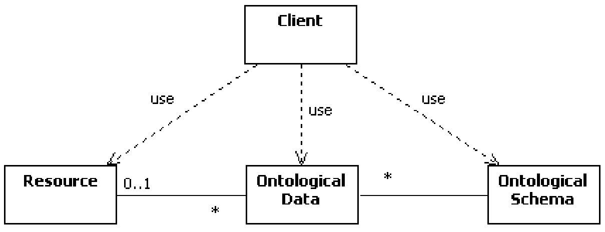

Figure 1: Resources and meta-data relations.

The Semantic Web paradigm thus emphasizes the relation between information and meta information, which distinguishes the concepts of Resource, Ontology Data, and Ontology Schema, as illustrated in Fig.1. Resources are any information object on the web: a file, an HTML page, an image, a movie. In the semantic web perspective, each Resource will contain its Semantic Data. The Ontology Schema is a conceptualization shared among users, which captures the intensional part of the model, defining types of entities and their possible relations (concepts). Ontology Data (individuals), are the extensional part of the model, classifying Resources as realizations of the concepts in the Ontology Schema. Client is a Semantic Web reader and can be a human or an application. In both the cases, Client is interested to access Resources and their related semantic data.

2.1 Abstract Roles in a semantic cooperative portal

The Muddy project aims at supporting explicit and shared representation of knowledge about construction practices based on mudbrick. In particular, the Muddy project aims at developing a web portal based on ontological models, enabling cooperation among subjects from different localities and different domains of expertise, in the development of a shared model and in the contribution of concrete contents for it.

In the light of the organization of information of Fig.1, this identifies roles, users' needs, and use cases generalized beyond the limits of the specific context of use (ISO, 1998), that are outlined in Fig.2

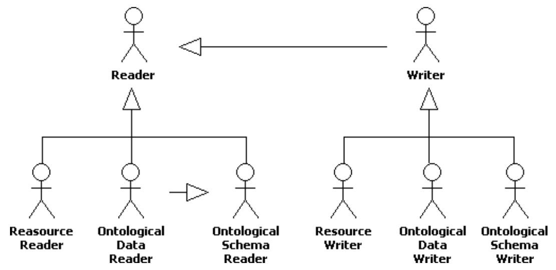

Figure 2: Abstract roles in a semantic portal.

Resource Readers correspond to readers in the conventional web. They are interested in accessing information, using meta-information to maintain context and to access resources, through direct links or search engines. Resource Writers are also enabled to insert, update and delete resources.

Ontological Schema Readers correspond to the second-level reader of (Eco, 1994). They are interested in understanding the organization of concepts more then their concrete realizations. They need to navigate in ordered manner and search classes and properties of an ontological model.

Ontological Schema Writers also modify models, taking part to the definition of the strategy of content organization. In particular, they may be interested in fixing errors, changing the model to follow some evolution, or extending the model by specialization and inheritance.

An Ontological Data Writer is a human or a SW indexing Resources with respect to the concepts of an Ontological Schema. Besides, an Ontological Data Reader is a human or more frequently a SW which exploits Ontological Data to access concrete resources in semantic querying (Bonino et al., 2003). Ontological data can also be formatted to be easily readable as well as a resource by human users (Dzbor M., 2004).

2.2 Use cases in the Muddy portal

In the specific context of the Muddy portal, the main goals of a Reader are browsing of pages derived from resources and semantic models, navigation of links corresponding to relations in the ontological model, execution of semantic queries on the knowledge base (Fig.3).

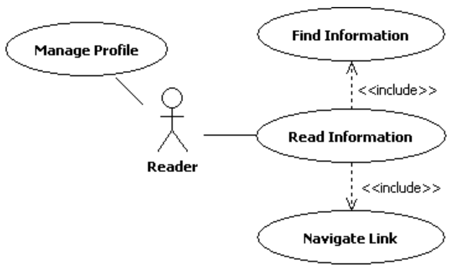

Figure 3: Web portal Reader use cases.

Besides, the Writer (Fig.4), extends the Reader capability to access information with the capability to contribute new knowledge in the form of a generic Ontological Schema or Data. Namely, the enabled user can contribute either the extensional or the intensional part of the ontological model for the web portal. Writers can also send/receive feedback and comments about the ontological model so as to encourage collaboration and cooperation among users. To contribute, writers will upload model files to ease the development of a portal prototype.

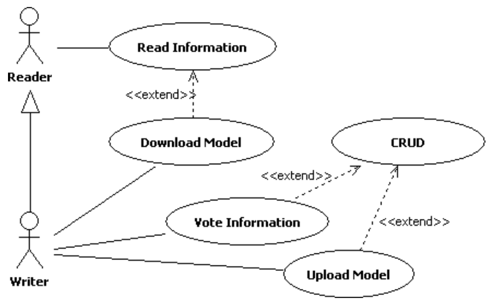

Figure 4: Web portal Writer use cases.

3 ARCHITECTURE AND DEVELOPMENT PROCESS

The conceptual architecture of our semantic portal composes a variety of partecipants that can be effectively assembled using W3C supported specifications and components.

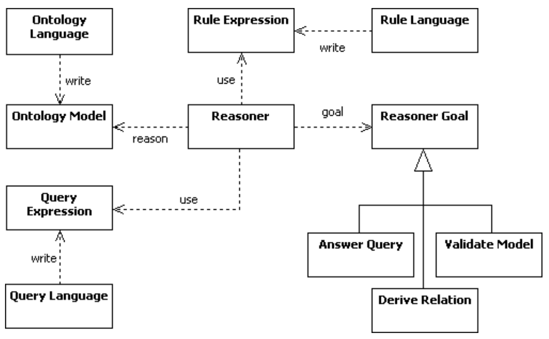

Figure 5: Architectural components.

3.1 Ontology Model

The Ontology Model (Fig.5) is the main component of the architecture, with the main responsibility of providing representation of the domain model of the application. It is implemented by composition of the Ontology Schema and Ontology Data (Fig.1), both encoded using the W3C standard Ontology Web Language (OWL) (W3C, 2004).

In a logic perspective (Fig.6), the ontology model can be decomposed in three main parts: classes, properties and individuals. Classes in the Ontology Model are part of the Ontological

Schema and play a role that can be compared to that of table definitions in a relational database. However, as opposed to relational tables, classes can be composed through delegation and inheritance. Properties are also part of the Ontological Schema, and they are used to define relations among classes. Individuals are realizations of concepts described by classes, and play a role that can be compared to that of records in a relational database.

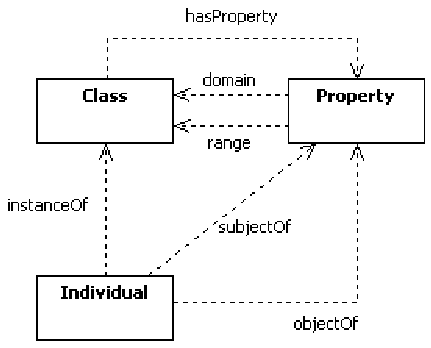

Figure 6: Logical compontents of the ontology model.

3.2 Rules

To form the overall knowledge base, the ontological model is complemented with Rules that extend the information explicitly represented with inference rules that let the system derive new knowledge. In our architecture, Rules are represented using the W3C supported Rule Language SWRL. To circumvent limitations affecting open source reasoners, we internally represent the language using Jena rules (Company, 2002).

3.3 Querying and reasoning

Query on the knowlegde base are expressed using the W3C supported SparQL. While the architecture supports the full expressivity of SPARQL, for the sake of usability, and in particular learnability, only a restricted fragment of the language is provided to the user.

The API Jena is used to drive reasoning and retrieve information by deciding SPARQL queries on the model. The API is also used to validate the ontology model and derive new knowledge using OWL axioms and inference rules. In general, any reasoner represents a trade-off between power and efficiency in computing. In particular, in the case of our architecture termination of reasoning tasks is guaranteed only if the model and the rules are encompassed within the boundaries of OWL-DL and SWRL, respectively.

3.4 Participants in the Development Process

The proposed SW architecture supports separation of concerns among four different roles of Domain Expert, Ontology Expert, Stakeholder, IT Expert. These naturally fit in a realistic social context of development (Cockburn, 1996) and basically correspond to the roles identified in (Tempich et al., 2005).

The Domain Expert knows about the domain of the portal and share partially formalized models among the community who belongs to. Domain Experts usually use specific tools to do their analysis and produce their research result. It is often the case that they don't know anything about ontologies and also they don't have opportunity (no time available) to learn about them.

The Ontology Expert is able to use sematic modelling tools and can describe knowledge contributed by Domain Experts with an Ontology Model. In this way the information, that was heterogeneous and sometimes also tacit or embedded, becomes formalized, explicit, homogeneous and consistent (Kryssanov et al., 1998).

The Stakeholder is interested in the domain logic but he/she is not necessarily expert. For this role, it is useful to have an ontology model that can be read and studied and that can be used to navigate through Domain Experts documents.

Finally, the IT Expert has to develop software tools needed by other roles so to let them read and write resources and ontology models (Fig.1 and 2).

3.5 Salient Aspects of the Implementation

3.5.1 Layering

The source code of the web portal (Fig.7) is organized in three layers (Schmidt et al., 2000), (Fowler, 2002). As usual, in SW architecture, layering separates presentation, domain logic and persistence. In this case, layering also helps in filling the gap between ontological and object oriented perspectives (Woodfield, 1997) (Guizzardi et al., 2001), which somehow reproduces the well known problem of Impedance Mismatch between Objects and relational databases (Ambler, 2003).

The presentation layer contains the logic to handle the interaction between the user and the software, relying on Java Server Faces (JSF) (Mann, 2004); the domain layer contains the application logic i.e. the domain model, implemented with Plain Old Java Objects (POJO); the persistent layer, is implemented as an ontology expressed as an OWL model and presently encoded as an RDF repository.

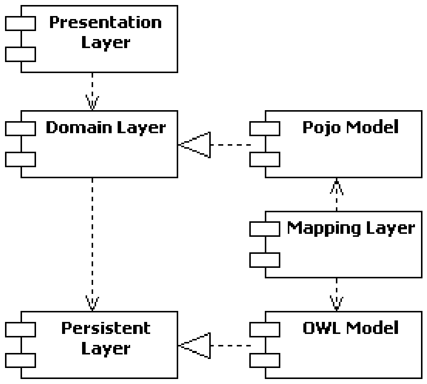

Figure 7: Web portal layering.

A Mapping layer is inserted between Domain layer and Persistent layer to improve decoupling and make easer testing and concurrent developing (Heumann, 2001), (Beck, 2002). The Mapping layer manages the mapping between models and meta-models, elaborates complex relations (reification), hides SPARQL embedded code and improves performances with methods like caching and proxying. Last but not least, only the mapping layer refers to the low level API Jena (Company, 2002).

3.5.2 Domain Model

The POJO Model in the domain layer is composed by two Java packages named Domain and UserProfile. The DataMapper components of the Mapping layer initialize objects of the POJO model with data contained in the OWL model which is in turn composed by three submodels named: Domain ontology, UserProfile ontology and Presentation ontology. The Domain package has responsibility to manage information contributed by users and is derived from Domain ontology according to the architectural pattern of reflection (Schmidt et al., 2000): classes of the Domain package are independent from the specific types defined in the Domain ontology thus enabling reuse of the OO layer, defining different evolutions of a portal or different portals insisting on different application domains. This is the feature that permits the cooperative portal to accommodate contributions not only in the individuals of the extensional part, but also in the concepts of the intensional part of the knowledgebase. Derivation of the Domain package is also affected by the Presentation ontology defining directives for the presentation of data in the page layout. The individuals of this ontology are used by the mapping layer to determine the presentation and filtering of concepts defined in the ontological Domain model. This accomplishes a responsibility which is much similar to that of ”site view graphs” in the OntoWebber framework (Yuhui Jin, 2001). The UserProfile package contains data about profiles, users and related information, and it is automatically derived from UserProfile ontology, so as to map ontology classes and properties to OO classes and attributes.

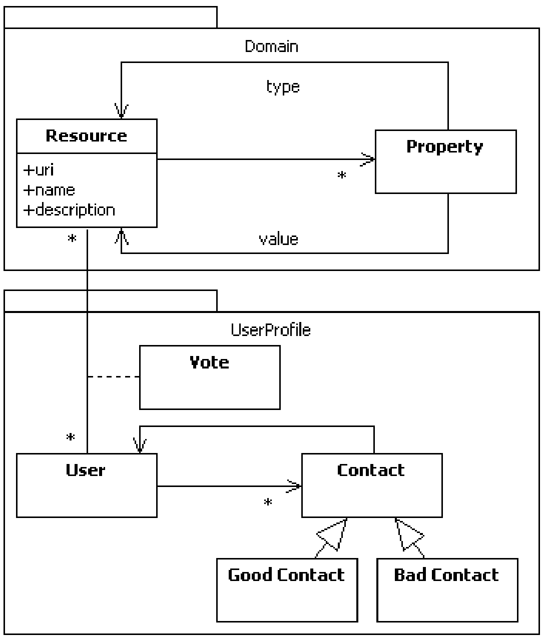

Figure 8: The POJO Model of the web portal.

3.5.3 Mapping Layer

Mapping between ontological and object-oriented models of the architecture was implemented following a pattern-oriented design (Fowler, 2002), (Schmidt et al., 2000) aimed at building an extensible and reusable framework.

Mappers The mapping layer includes a mapper class for each element of the OO domain model (Fowler, 2002) (Fig.9). Each mapper class can read information form the ontological model to assign it to the object-oriented model and vice versa, and it is implemented as extension of the abstract DataMapper base class.

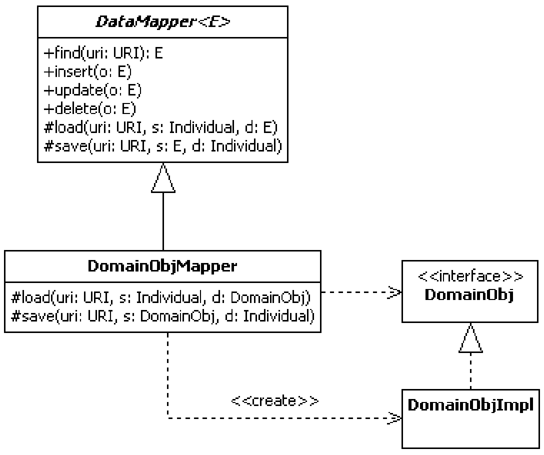
DataMapper<E>
+find(uri: URI): E
+insert(o: E)
+update(o: E)
+delete(o: E)
#load(uri: URI, s: Individual, d: E)
#save(uri: URI, s: E, d: Individual)
DomainObjMapper
#load(uri: URI, s: Individual, d: DomainObj)
#save(uri: URI, s: DomainObj, d: Individual)
<<interface>>
DomainObj
<<create>>
DomainObjImpl

Figure 9: Mappers.

Mappers decouple the object-oriented Domain Layer from the ontological Persistence Layer, so that a client can ignore how objects are persisted in the ontology. For the sake of performance and and behavioral abstraction, DataMappers implement some specific patterns (Fowler, 2002), (Gamma et al., 1995):

- **Identify Map (Cache):** mappers have a cache memory to speed up repeated access to the same object.
- **Proxy:** if possible, mappers substitute objects requested by the client with equivalent proxies. This delays the mapping operations until they are really needed.
- **LazyLoad:** mappers load objects of a list when they are used so the slow mapping operation is executed for useful objects only.
- **UnitOfWork:** unit of work class manages changes to objects that mappers have to persist.

Developers can use an instance of the class Session to access functions of the mapping framework (Fig.10).

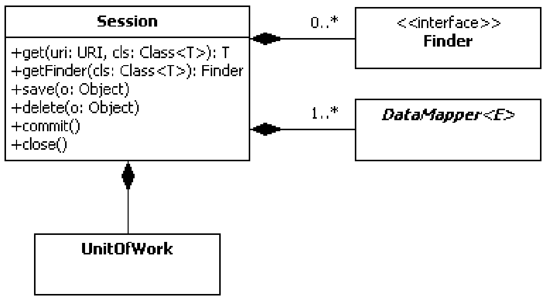

Figure 10: The Seession class.

4 THE MUDDY PORTAL

Muddy is a web-based application implemented as an instance of requirements and architecture described so far. It allows reading and writing of concrete and conceptual information according to an ontological paradigm, providing the following user functions: navigate information following semantic links; execute semantic queries on the knowledge base; contribute new knowledge uploading model files; read/write feedback about the knowledge base.

As characterizing traits: users know that data are organized according to an underlying ontological model; users cooperate in the construction of one or many ontological models, by creating, retrieving, updating and deleting not only individuals but also concepts.

4.1 The Portal Architecture

Fig.11 depicts the architecture of the portal managed by the application.

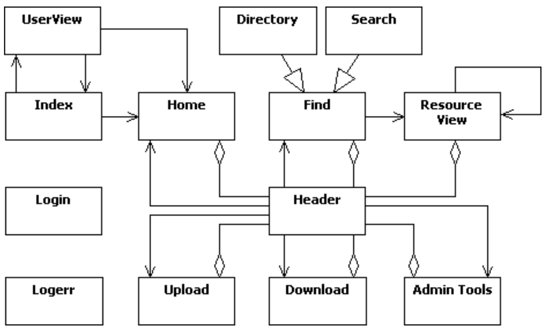

Figure 11: Structure of the Muddy Portal.

Index is the first page of the portal with login and registration, giving access to the Home page and then, through Header page to the functions of the portal.

Users can be readers, writers and administrators and they are provided with different functions. A new user is always classified as reader and only administators can give users more privileges.

Find page is used to execute queryies on the knowledge base by users and it is specialized in Search and Directory pages. ResourceView page allows users to read information contained in the knowledge base. Upload and Download pages allow users to contribute new knowledge. Admintools page is for the administrator.

4.2 Find Pages

The Directory page (Fig.12) is used to execute pro-active search. The system shows to the user a list of categories that correspond to root classes of the ontological model managed by the portal. The user can select a category to get a page containing the list of instances of the category and the list of its direct subclasses. The user can navigate toward more specific classes or can inspect one of the instances found.

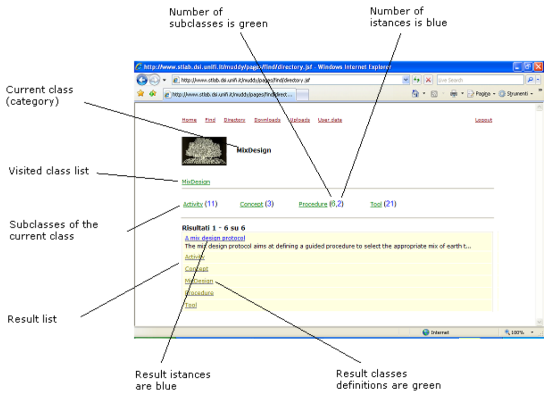

Figure 12: The Directory search page.

The Search page (Fig.13) implements an extension of full-text search methods. Users can specify one or more words that must be contained in desired resources. They can also specify the kind of relations that link desired resources to specified words. For instance, the expression "neededTools = sieve" lets a user require all resources that has a "sieve" among needed "tools"

4.3 Resource View Page

This page shows information about a resource (Fig.14), and allows users to speed up navigation towards semantic related resources.

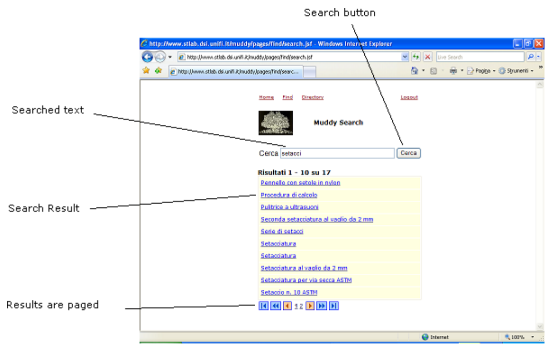

Figure 13: The Search page.

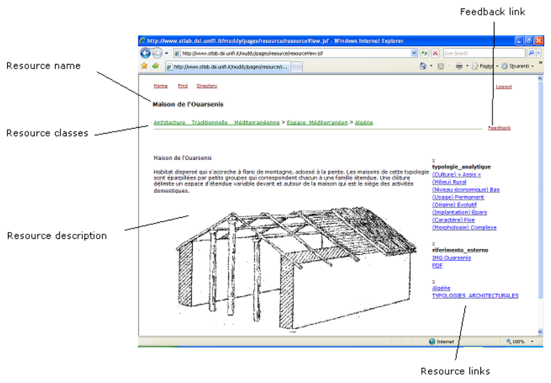

Figure 14: The ResourceView page.

The portal also allows users to give feedback about accessed resources which is used to calculate appreciation indexes about resources.

5 CONCLUSIONS

We are further developing the portal and its underlying architecture, facing various interrelated issues, that are crucial to tackle the transition phase towards the context of use:

- a usability cycle has been planned, to evaluate the capability of the portal to support the user in maintaining context in the navigation through the weblike structure of portal contents. This step should be largely facilitated by the orientation towards change in the overall architecture, and in particular by the presentation ontology implemented in the mapper;
- preliminary performance profiling indicates that performance can be largely improved by the integration of a more elaborated RDF repository;

- functional extensions are being developed to implement the automated derivation of an editor of individuals based on the structure of ontology concepts and a tool for annotation of existing web resources with reference to the concepts of an ontological model.

REFERENCES

Ambler, S. (2003). *Agile Database Techniques: Effective Strategies for the Agile Software Developer.* John Wiley & Sons, Inc., New York, NY, USA.

Beck, K. (2002). *Test Driven Development: By Example.* Addison-Wesley Professional.

Berners-Lee, T. (1998). Semantic web roadmap. http:// www.w3.org/2001/sw/.

Bonino, D., Corno, F., and Farinetti, L. (2003). Dose: a distributed open semantic elaboration platform. ICTAI 2003, The 15th IEEE International Conference on Tools with Artificial Intelligence, November 3-5, 2003, Sacramento, California.

Cockburn, A. (1996). The interaction of social issues and software architecture. *Commun. ACM*, 39(10):40–46.

Company, H.-P. D. (2002). Jena a semantic web framework for java. http://jena.sourceforge.net/.

Corcho, O., Lpez-Cima, A., and Gomez-Prez, A. (2006). A platform for the development of semantic web portals. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, pages 145–152, New York, NY, USA. ACM Press.

Dzbor M., Motta E., D. J. B. (2004). Opening up magpie via semantic services. In Proc. of the 3rd Intl. Semantic Web Conference, November 2004, Japan.

Eco, U. (1994). Six walks in the fictional woods. Harvard University Press.

Fowler, M. (2002). *Patterns of Enterprise Application Architecture.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Franca Garzotto, Luca Mainetti, P. P. (1995). Hypermedia design analysis and evaluation issues. incomm. of the acm. Communications of the ACM.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns.* Addison-Wesley Professional.

Guizzardi, G., Falbo, R., and Filho, J. (2001). Using objects and patterns to implement domain ontologies. 15th Brazilian Symposium on Software Engineering, Rio de Janeiro, Brazil.

Heumann, J. (2001). Generating test cases from use cases. The Rational Edge.

ISO (1998). Iso 9241-11 "guidance on usability". http://www.iso.org/iso/en/ISOOnline.frontpage.

ISO (2004). Iso 9126 "software engineering – product quality". http://www.iso.org/iso/en/ISOOnline.frontpage.

Kruchten, P. (2003). *The Rational Unified Process: An Introduction, Third Edition.* Addison-Wesley Professional.

Kryssanov, V. V., Abramov, V. A., Fukuda, Y., and Konishi, K. (1998). The meaning of manufacturing know -how. In *PROLAMAT '98: Proceedings of the Tenth International IFIP WG5.2/WG5.3 Conference on Globalization of Manufacturing in the Digital Communications Era of the 21st Century*, pages 375–388, Deventer, The Netherlands, The Netherlands. Kluwer, B.V.

M. Fayad, M. C. (1996). Aspects of software adaptability. COMMUNICATIONS OF THE ACM.

Mann, K. D. (2004). *JavaServer Faces in Action (In Action series).* Manning Publications Co., Greenwich, CT, USA.

P.J. Lynch, S. H. (2002). Web style guide: Basic design principles for creating web sites. Yale University Press.

Schmidt, D. C., Rohnert, H., Stal, M., and Schultz, D. (2000). *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects.* John Wiley & Sons, Inc., New York, NY, USA.

Schreiber, G., Amin, A., van Assem, M., de Boer, V., Hardman, L., Hildebrand, M., Hollink, L., Huang, Z., van Kersen, J., de Niet, M., Omelayenko, B., van Ossenbruggen, J., Siebes, R., Taekema, J., Wielemaker, J., and Wielinga, B. J. (2006). Multimedian e-culture demonstrator. In *International Semantic Web Conference*, pages 951–958.

Stojanovic, N., Maedche, A., Staab, S., Studer, R., and Sure, Y. (2001). Seal: a framework for developing semantic portals.

Tempich, C., Pinto, H. S., Sure, Y., and Staab, S. (2005). An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (diligent). In Gmez-Prez, A. and Euzenat, J., editors, *Second European Semantic Web Conference, (ESWC 2005)*, volume 3532 of *LNCS*, pages 241–256, Heraklion, Crete, Greece. Springer.

W3C (2004). Owl web ontology language. http://www.w3.org/TR/owl-features/.

Woodfield, S. N. (1997). The impedance mismatch between conceptual models and implementation environments. ER'97 Workshop 4 Proceedings.

Yuhui Jin, Stefan Decker, G. W. (2001). Ontowebber: Model-driven ontology-based web site management. 1st International Semantic Web Working Symposium, Stanford University, Stanford.