Valeriano Sandrucci, Marco Masera University of Florence

An ontology based semantic web portal

Un portale per il web semantico basato su ontologie

Technologies for ontological modelling and reasoning devise a new paradigm for the organization of software architectures with high degree of interoperability, maintainability and adaptability (Fayad 1996, ISO 2004). In fact: ontologies natively allow representation of explicit semantic models combining non ambiguity of technical specification with the un derstandability needed to fill the gap between technicians and stakeholders; ontologies enable the application of methods and tools off the shelf supporting reasoning for information search, model validation, and inference and deduction of new knowledge (Bozsak 2007); ontologies naturally fit into a distributed context, enabling creation of reusable models, composition and reconciliation of fragments developed in a concurrent and distributed manner (Tempich 2005, Tempich 2006); last, but not least, ontologies can model evolving domains, thus encouraging an incremental approach that can accompany the evolution towards shared models. In particular, this potential appears well suited for the organization of web portals, where ontologies provide a paradigm to consistently design, manage, and maintain information architecture, site structure, and page layout (Garzotto 1995). This has crucial relevance in the construction of portals with weblike organization (Lynch 2002), 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. In Schreiber (Schreiber et al., 2006), a semantic portal is presented which supports unified access to a va riety of cultural heritage resources classified ac cording to a public unifying ontology. The portal is developed using standard ontological technologies and SWI-prolog to support semantic search and presentation of retrieved data.

In Stojanovic (Stojanovic 2001), a portal based on ontology technologies is used as a basis to support the contribution of contents by a distributed community. The proposed architecture assumes that contribution is limited to individuals, and the investigation is focused on different techniques supporting the determination of a rank of relevance for the result-set of semantic queries. In Yuhui Jin (Yuhui Jin, 2001), a declarative approach to the construction of semantic portals is proposed, which relies on the defini-

Le tecnologie per la modellazione ed il ragionamento ontologico delineano un nuovo paradigma per l'organizzazione di architetture software con alto grado di interoperabilità, manutenibilità ed adattabilità (Fayad 1996), (ISO, 2004).

Infatti le ontologie permettono nativamente la rappresentazione di modelli semantici combinando la non ambiguità della specifica tecnica con la comprensibilità richiesta per colmare la distanza tra i tecnici e gli altri soggetti interessati; le ontologie abilitano l'applicazione di metodi e strumenti per il ragionamento per la ricerca di informazione, la validazione di modelli ed inferenza e deduzione di nuova conoscenza (Bozsak 2007); le ontologie naturalmente si adattano ad un contesto distribuito, abilitando creazione di modelli riusabili, composizione e riconciliazione di frammenti sviluppati in maniera concorrente e distribuita (Tempich 2005, Tempich 2006); ultimo, non per importanza, le ontologie possono modellare domini che evolvono, incoraggiando in questo modo un approccio incrementale che può condurre all'evoluzione verso modelli condivisi.

In particolare, questo potenziale appare adatto all'organizzazione di portali web, in cui le ontologie forniscono un paradigma per progettare, gestire e mantenere l'architettura dell'informazione, la struttura del sito ed il layout delle pagine (Garzotto 1995). Ciò ha una rilevanza cruciale nella costruzione di portali con una organizzazione di tipo web (Lynch 2002), in cui le pagine sono organizzate secondo il pattern del grafo generico e la navigazione guidata dalla semantica inerente il contesto più che da una classificazione gerarchica fatta a priori.

Schreiber (Schreiber 2006) presenta un portale che supporta un accesso unificato ad una varietà di risorse di cultural heritage classificate in accordo ad una ontologia pubblica unificante. Il portale è stato sviluppato usando tecnologie ontologiche standard e SWI-prolog per supportare la ricerca semantica e la presentazione dei dati recuperati.

In un portale basato su tecnologie ontologiche (Stojanovic 2001) viene utilizzato come base per supportare la contribuzione di contenuti da parte di una comunità distribuita. L'architettura proposta assume che la contribuzione sia limitata agli individui e l'investigazione centrata su differenti tecniche per la determinazione di un indice di rilevanza per i result-set di query semantiche.

In (Yuhui Jin 2001), viene proposto un approccio dichiarativo alla costruzione di portali semantici che si basa sulla definizione di un insieme di ontologie create dal programmatore del portale per definire i concetti e contenuti di dominio, struttura di navigazione e stile di presentazione.

In (Corcho 2006), l'approccio dichiarativo di (Yuhui Jin 2001) viene este-

tion of a suite of ontologies created by the portal programmer to define domain concepts and contents, navigation structure and presentation style.
In Corcho (Corcho 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 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 chapter, 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 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. We report on the usage of the architecture in the case of a cooperative portal on mudbrick construction practices, that we call Muddy. In so doing, we also highlight the application programming model that permits the construction of a portal using the proposed architecture.
The chapter is organized in 5 sections. 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).

Requirements analysis

The Ontological Paradigm

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 infor mation is sematically defined so as to enable automated processing. 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 Seman tic Data. The Ontology Schema is a conceptualization

so in un framework basato su ontologie che supportano la costruzione di applicazioni web combinando informazione e servizi. Il framework implementa il pattern architetturale Model View Controller (Schmidt 2000). Mentre il modello è dichiarato usando le ontologie, le viste sono implementate con tecnologie di presentazione esistenti ed in particolare con JSP, che principalmente si basa sul paradigma OO. Per colmare la distanza tra i due paradigmi (Woodfield 1997), allo sviluppatore viene fornito un insieme di componenti JSP predefiniti che assumono la responsabilità dell'adattamento.
In questo capitolo trattiamo lo sviluppo di un portale che abilita l'accesso semantico, l'interrogazione e la contribuzione sia di individui che di concetti.
A questo fine possiamo sviluppare un'architettura SW eseguibile basata su linguaggi e componenti standard, scelta che riconduce la creazione di un portale cooperativo alla definizione di un'ontologia del modello di dominio, architettura che viene implementata con componenti che possono essere efficientemente riusati in una varietà di applicazioni data intensive e basate su conoscenza.
Muddy è il nome dato al portale cooperativo sperimentale sulle pratiche costruttive in terra cruda per il quale è stata utilizzata tale architettura di cui si viene discusso qui il modello di programmazione che permette la costruzione di un portale.
A questo scopo, dopo una breve panoramica riguardo la nostra prospettiva sul paradigma ontologico nelle architetture software, viene introdotto il caso di Muddy attraverso l'analisi dei suoi requisiti identificando i ruoli astratti ed i casi d'uso (Sect. 2). Successivamente viene allargato il progetto architetturale ed i tratti salienti della sua implementazione e sono identificati i ruoli coinvolti nell'applicazione (Sect.3). Infine si descrive il portale Muddy (Sect. 4) e si riportano le conclusioni (Sect. 5).

Analisi dei requisiti

Il Paradigma Ontologico

Le tecnologie ontologiche nascono principalmente con l'intento di contribuire alla realizzazione del Semantic Web (Berners-Lee 1998). Questo denota un'evoluzione del web attuale in cui l'informazione semanticamente definita così da abilitare elaborazioni automatiche.
Il paradigma del Semantic Web quindi enfatizza la relazione tra informazione e meta informazione, che ci permette di distinguere i concetti di Resource, Ontology Data ed Ontology Schema come illustrato in Fig.1. Le risorse sono qualunque oggetto di informazione sul web: un file, una pa-

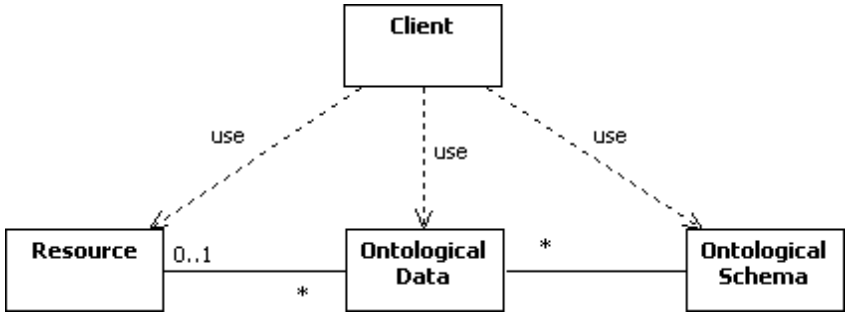


Fig. 1: Resources and meta-data relations.

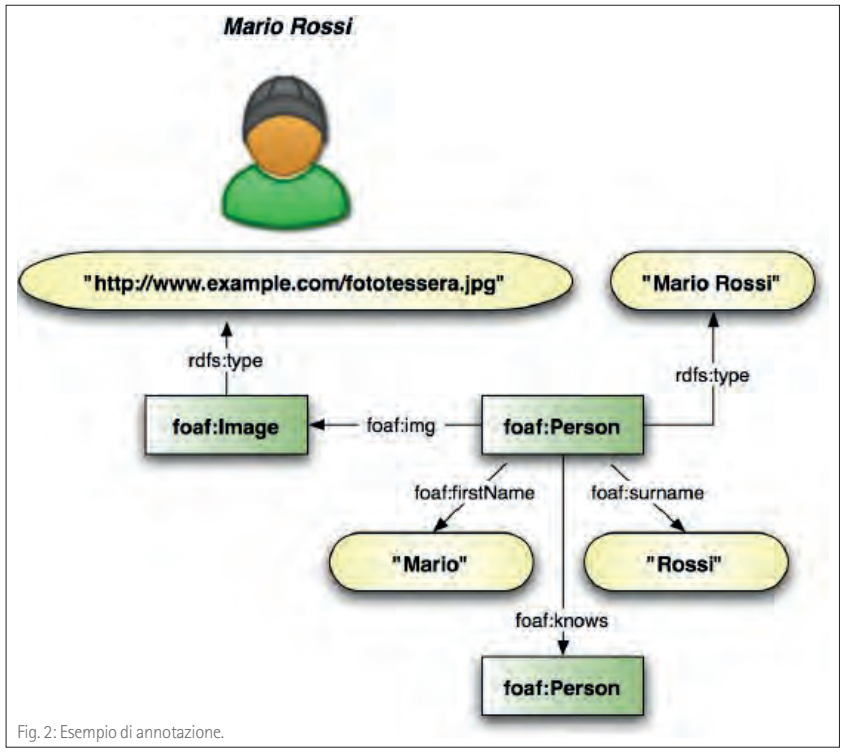


Fig. 2: Esempio di annotazione.

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 cas-

gina HTLM, un'immagine, un filmato. Nella prospettiva del Semantic Web ciascuna risorsa contiene i propri dati semantici. L'Ontology Schema è una concettualizzazione condivisa tra gli utenti che cattura la parte intensionale del modello, definendo i tipi delle entità e le loro possibili relazioni (concetti). Ontology Data (individui), sono la parte estensionale del modello, classificano le Resource come realizzazioni dei concetti in Ontology Schema.

Client è un utente del Semantic Web e può essere umano od un'applicazione. In entrambi i casi il Client è interessato ad accedere a Resource e ad i suoi dati semantici relati.

In Fig. 2 viene mostrato un esempio. Prendiamo come risorsa una immagine: in particolare una la fototessera del signor 'Mario Rossi'. A questa, tramite l'ontologia pubblica denominata FOAF, è possibile associare una serie di concetti: un agente software così in grado di interpretare che la foto (foaf:Image) associata (foaf:img) ad una persona (foaf:Person), il cui nome (foaf:firstName) 'Mario' e il cognome (foaf:surname) 'Rossi'. Inoltre, 'Mario Rossi' conosce (foaf:knows) altre persone: l'ontologia può essere estesa per esempio dall'insieme di conoscenti di 'Mario Rossi', corredati con le rispettive foto.

Le tecnologie ontologiche comprendono un framework ricco di paradigmi, linguaggi e componenti pronti che può servire oltre lo specifico intento del Semantic Web e può diventare un pattern per l'organizzazione di complesse architetture SW.

Un Portale Cooperativo riguardo le Pratiche di Costruzione in Terra Cruda

Il progetto Muddy mira a supportare la rappresentazione di conoscenza esplicita e condivisa riguardo le pratiche di costruzione basate sulla terra cruda. Ciò è motivato dal vasto uso di questa tecnologia costruttiva (stime indicano che il 30% della popolazione mondiale ancora vive in abitazioni di terra cruda), dalla ricca varietà di differenti pratiche sviluppate in diverse località e dall'alta efficienza energetica ed il basso impatto ambientare che la rende una pratica sostenibile.

In particolare, il progetto Muddy mira a sviluppare un portale web basato su modelli ontologici che abiliti la cooperazione tra soggetti di differenti località e differenti domini di esperienza nello sviluppo di un modello condiviso e nella contribuzione di contenuti concreti per esso.

Ruoli Astratti

L'analisi dei requisiti funzionali del progetto Muddy nella prospettiva dell'organizzazione dell'informazione in accordo al paradigma ontologi-

es, Client is interested to access Resources and their related semantic data. Ontological technologies comprise a rich framework of paradigms, languages, and components off the shelf, which can serve beyond the specific intent of the Semantic Web, and may become an effective pattern for the organization of complex SW architectures.

A Cooperative Portal about Mudbrick Construction Practices

The Muddy project aims at supporting explicit and shared representation of knowledge about construction practices based on mudbrick. This is motivated by the widespread use of this building technology (estimates indicate that almost 30% of world population still live in mudbrick habitations), by the rich variety of different practices developed in different localities, and by the high energetic efficiency and low environmental impact which make it a sustainable practice.

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.

Abstract Roles

Analysis of functional requirements of the Muddy project in the light of the organization of information according to the ontological paradigm of Fig. 1, identifies roles, users needs, and use cases generalized beyond the limits of the specific context of use (ISO, 1998). These roles are outlined in Fig. 2

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

co di Fig. 1 identifica ruoli, bisogni degli utenti e casi d'uso generalizzati oltre i limiti dello specifico contesto d'uso (ISO 1998). Questi ruoli sono mostrati in Fig. 3

I Resource Readers corrispondono ai fruitori del web convenzionale. Essi sono interessati ad accedere all'informazione usando la meta informazione per mantenere il contesto ed accedere alle risorse attraverso collegamenti diretti o motori di ricerca. Resource Writers sono pure abilitati ad inserire, modificare e cancellare risorse.

I fruitori di Ontological Schema corrispondono ai fruitori di secondo livello di (Eco 1994). Essi sono interessati nel capire l'organizzazione dei concetti più che alle loro concrete realizzazioni. Essi necessitano di navigare in maniera ordinata e ricercare classi e proprietà di un modello ontologico.

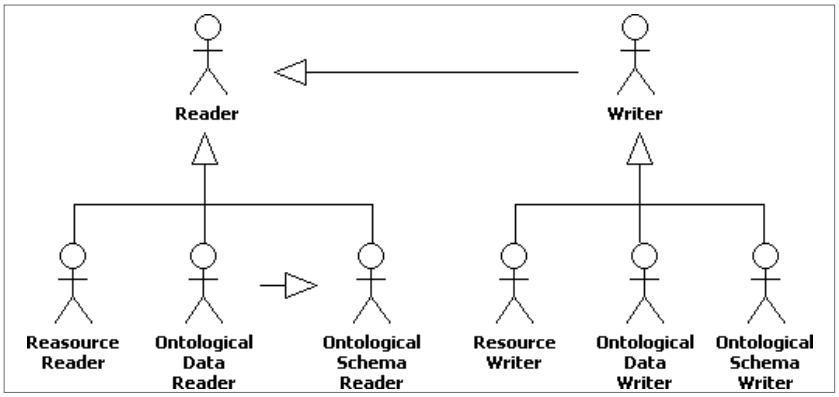
I contributori dello Schema Ontologico pure modificano i modelli, prendendo parte alla definizione della strategia dell'organizzazione del contenuto. In particolare essi sono interessati nel correggere gli errori e nel cambiare il modello perché segua una qualche evoluzione, estendendo il modello con specializzazioni ed ereditarietà.

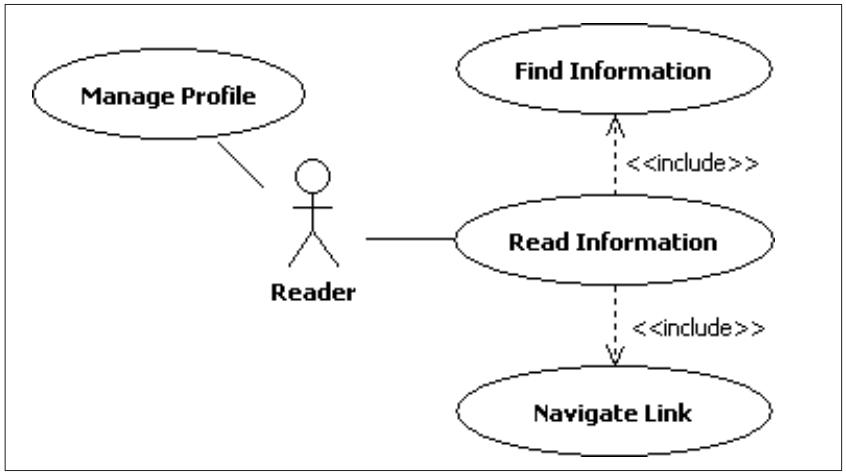
Un contributore di Ontological Data un umano o un SW che indicizza Resource rispetto

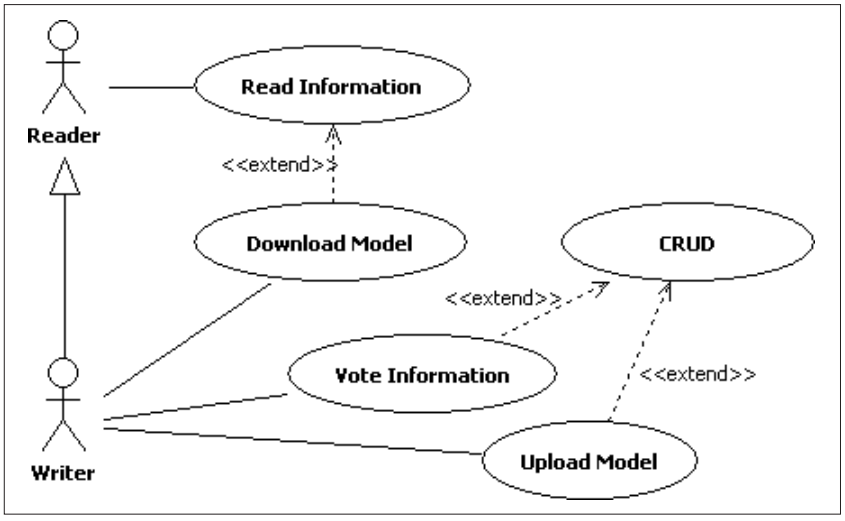
ai concetti di Ontological Schema. An Ontological Data Writer is a human or a SW indexing Resources with respect. Inoltre, un fruitore di Ontological Data un umano o, più frequentemente, un SW che sfrutta Ontological Data per accedere alle risorse concrete nelle interrogazioni semantiche (Bonino 2003). Ontological Data possono anche venire formattati per essere facilmente leggibili come le risorse per utenti umani (Dzbor 2004).

Casi d'uso per il portale Muddy

Nel contesto specifico del portale Muddy, gli obiettivi principali di un fruitore sono: consultare le pagine derivate dalle risorse ed i modelli semantici, navigare sui collegamenti dovuti alle relazioni nel modello ontologico, l'esecuzione di interrogazioni semantiche sulla base di conoscenza (Fig. 4). Inoltre, il contributore (Fig. 5), estende le capacità di accesso all'informazione del fruitore con la capacità di contribuire nuova conoscenza nella forma di un generico Ontological Schema o Data. Cioè l'utente abilitato può contribuire sia alla parte estensionale che intensionale del modello ontologico per il portale web. I contributori possono anche spedire e ricevere feeback e commenti circa il modello ontologico così da incoraggiare la collaborazione e cooperazione tra utenti. Per contribuire, gli utenti caricano i file di modello per semplificare lo sviluppo di un prototipo del portale.







Un ruolo strumentale aggiuntivo è l'amministratore la cui principale responsabilità è l'assegnazione dei privilegi di fruitore e contributore.

Architettura del portale e processo di sviluppo

Componenti Architetturali

L'architettura concettuale del portale semantico si compone di una varietà di moduli che possono essere efficacemente assemblati usando componenti e specifiche supportate dal W3C.

Modello Ontologico

Il Modello Ontologico (Fig. 6) è il componente principale dell'architettura, con la principale responsabilità di fornire la rappresentazione del modello di dominio dell'applicazione. Viene implementato attraverso la composizione dell'Ontology Schema con l'Ontology Data (Fig. 1), entrambi codificati usando lo standard W3C Ontology Web Language (OWL) (W3C 2004). In linea di principio l'intera espressività di OWL-full può essere impiegata ma il buon fine di portare a termine i processi di ragionamento inclusi nei servizi del portale è garantito solo dall'assunzione del modello incluso nello schema OWL-DL.

In una prospettiva logica (Fig. 7), il modello ontologico può essere scomposto in tre parti principali: classi, proprietà e individui. Le classi del Modello Ontologico sono parte dell'Ontological Schema e giocano un ruolo che può essere comparato a quello della definizione delle tabelle in un database relazionale. Comunque, a differenze delle tabelle relazionali, le classi possono essere composte per delega ed ereditarietà. Le proprietà sono anch'esse parte di Ontological Schema e sono usate per definire relazioni tra classi. Gli individui sono realizzazioni dei concetti descritti dalle classi e giocano il ruolo che può essere paragonato a quello dei record in un database relazionale. È bene notare che, in questa architettura, la logica di dominio modellata e specificata usando ontologie piuttosto che i diagrammi delle classi UML della comune pratica di sviluppo OO. In una prospettiva pratica, un modello UML può essere facilmente mappato in un'ontologia garantendo che alcune differenze maggiori tra diagrammi delle classi ed ontologie sono stati presi in considerazione (Brockmans 2004): nei modelli ontologici le proprietà esistono indipendentemente dalle classi cui si applicano e possono essere organizzate in modo gerarchico; una relazione tra due classi UML è codificata come una pro-

Fig. 3: Abstract roles.
Fig. 4: Web portal Writer use cases.
Fig. 5: Architectural components of the web portal.

semantic querying (Bonino 2003). Ontological data can also be formatted to be easily readable as well as a resource by human users (Dzbor M., 2004).

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 due to relations in the ontologi cl model, execution of semantic queries on the knowledge base (see Fig. 3).

Besides, the Writer (see 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. A further instrumental role is the Administrator, whose main responsibility is the assignment of readers and writers privileges.

Architecture and development process

Architectural Components

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

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 (see Fig. 1), both encoded using the W3C standard Ontology Web Language (OWL) (W3C 2004). In principle, the entire expressivity of OWL-full can be exploited, but successful termination of reasoning tasks included in portal services is guaranteed only under the assumption that the model is encompassed within OWL-DL reasoning.

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

prietà il cui dominio e codominio sono le due classi stesse; in un modello ontologico, un individuo può appartenere a più classi che possono cambiare durante il suo ciclo di vita come se il tipo fosse una proprietà essa stessa dell'individuo; nei modelli ontologici le classi possono essere definite usando costrutti di tipo insiemistico.

Possiamo immaginare facilmente un semplice esempio di modello semantico. Possiamo definire Book ed Author come due classi dell'ontologia tra le quali esiste una relazione orientata definita dalla proprietà hasAuthor e rappresentata nel diagramma come classe di associazione. Book, Author e hasAuthor costituiscono la parte intesionale del modello ontologico. Il Barone Rampante istanza della classe Book mentre Italo Calvino istanza della classe Author. I due oggetti sono in relazione infatti Il Barone Rampante ha un autore (hasAuthor) che Italo Calvino. Il Barone Rampante ed Italo Calvino sono individui ed assieme alle loro relazioni formano la parte estensionale del modello ontologico.

Regole

Per formare la base di conoscenza complessiva, il modello ontologico utilizza regole complementari che estendono l'informazione esplicitamente rappresentata con regole di inferenza che permettono al sistema di derivare nuova conoscenza. Una regola ha la forma:

antecedent =¿ consequent

per la quale sia l'antecendente che la conseguente sono in congiunzione di atomi scritti nella forma atomo1 and atomo2 and ... and atomo ennesimo. Le variabili sono indicate utilizzando la convenzione standard di utilizzare come prefisso un punto interrogativo (ad es. ?x). Utilizzando questa sintassi una regola che afferma che la composizione della proprietà di 'genitore' e 'fratello' implica dedurra la proprietà 'zio' in certe condizioni:

Person(?x) and Person(?y) and Person(?z) and parent(?x,?y) and brother(?y,?z) =¿ uncle(?x,?z) Da notare che Person è una classe del modello ontologico mentre parent, brother ed uncle sono proprietà.

Nella nostra architettura le regole sono rappresentate usando il linguaggio supportato dal W3C SWRL. Per aggirare le limitazioni che affliggono i reasoner open source, si può utilizzare il linguaggio Jena rules (Company 2002) per rappresentare le regole internamente al sistema.

Querying e reasoning

Le interrogazioni sulla base di conoscenza sono espresse adoperando il linguaggio SPARQL del W3C. Mentre l'architettura supporta la completa espressività di SPARQL, per la ragione dell'usabilità ed in particolare

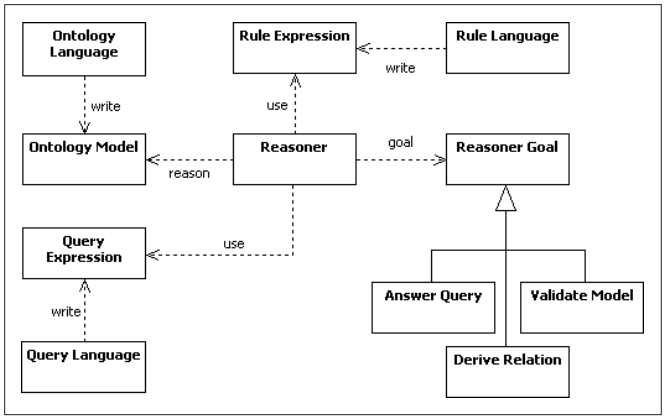


Fig. 6: Logical compontents of the ontology model.

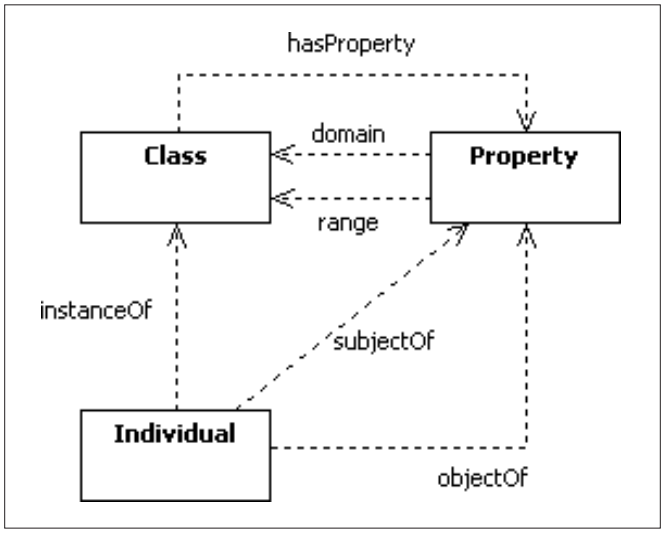


Fig. 7: Web portal layering.

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. It is worth noting that, in this architecture, the domain logic is modelled and specified using ontologies rather than UML class diagrams of the common practice of OO development. In a practical perspective, an UML model can be easily mapped to an ontology, provided that some major differences among class diagrams and ontologies are taken into account (Brockmans 2004): in ontological models, properties exist independently from classes to which they apply, and they can thus be organized in a hierarchy; a relation between two UML classes is encoded as a property, whose domain and range are the two classes themselves; in an ontological model, an individual can belong to mul-

dell'apprendibilità, solo un frammento ristretto del linguaggio è fornito all'utente. L'espressione di query che segue serve a recuperare il titolo, più precisamente il valore della proprietà http://purl.org/dc/elements/1.1/title, dell'individuo identificato dall'URI http://example.org/book/book1. Anche in SPARQL le variabili vengono rappresentate facendo precedere il loro nome da un punto interrogativo.

SELECT ?title WHERE ¡http://example.org/book/book1¿ ¡http://purl.org/dc/elements/1.1/title¿

?title

Utilizzando SPARQL si possono esprimere query anche molto complesse in maniera piuttosto semplice e compatta. La query che segue serve ad estrarre i titoli dei libri che contengono la parola Rampante

PREFIX dc: ¡http://purl.org/dc/elements/1.1/¿ SELECT ?title WHERE ?x dc:title ?title FILTER regex(?title, "Rampante", "i")

La API Jena è adoperata per guidare i ragionamenti e recuperare l'informazione decidendo le interrogazioni SPARQL sul modello. La API è anche usata per validare il modello ontologico e derivare nuova conoscenza usando gli assiomi OWL e le regole di inferenza. In generale ogni reasoner rappresenta un compromesso tra potenza ed efficienza nella computazione. In particolare, nel caso della nostra architettura la conclusione dei processi di ragionamento è garantita solo se il modello e le regole rispettano i limiti di OWL-DL e SWRL, rispettivamente.

Partecipanti del Processo di Sviluppo

L'architettura SW proposta supporta la separazione di interessi tra quattro differenti ruoli: Domain Expert, Ontology Expert, Stakeholder ed IT Expert. Questi naturalmente si adattano ad un contesto realistico di sviluppo (Cockburn 1996) e fondamentalmente corrispondo con i ruoli identificati in (Tempich 2005).

Domain Expert è l'esperto nel dominio del portale e condivide modelli parzialmente formalizzati con la comunità cui appartiene. I Domain Expert usualmente usano strumenti specifici per fare le loro analisi e produrre i loro risultati di ricerca. Frequentemente non sanno niente di ontologie e non hanno nemmeno l'opportunità (tempo non disponibile) di imparare qualcosa su di esse.

Un Ontology Expert è capace di usare gli strumenti di modellazione semantica e può descrivere la conoscenza contribuita dai Domain Expert con un Modello Ontologico. In questo modo l'informazione che era eterogenea e qualche volta anche tacita o embedded diventa formalizzata, esplicita, omogenea e consistente (Kryssanov 1998).

tiple classes which can change during its lifecycle, as the type is a property itself of the individual; in ontological models, classes can be defined using set-theoretical constructs.

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).

Querying and reasoning

Query on the knowlegde base are expressed using the W3C supported the 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.

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 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 1998).The Stakeholder is interested in the domain logic but he/she is not necessarUno Stakeholder è interessato alla logica di dominio ma non necessariamente esperto. Per questo ruolo è utile avere un modello ontologico che può essere letto e studiato e che può essere usato per navigare attraverso i documenti dei Domain Expert. Infine, IT Expert deve sviluppare gli strumenti software necessari agli altri ruoli così da permettere loro di leggere e scrivere risorse e modelli ontologici (Fig. 1 e 3).

Aspetti Salienti dell'Implementazione

Layering

Il codice sorgente del portale web (Fig. 9) è organizzato in tre livelli (Schmidt 2000), (Fowler 2002). Come consueto, nelle architetture software, i livelli separano: presentazione, logica di dominio e persistenza. In questo caso la scomposizione in livelli aiuta anche a colmare la distanza tra prospettiva ontologica e ad oggetti (Woodfield 1997) (Guizzardi 2001) che in qualche modo riproduce il ben noto problema di Impedance Mismatch between Objects and relational databases (Ambler 2003). Specificatamente: il presentation layer contiene la logica per gestire l'interazione tra l'utente ed il software e si basa su Java Server Faces (JSF) (Mann 2004); il domain layer contiene la logica applicativa che consiste nel modello di dominio implementato con Plain Old Java Objects (POJO); il persisten layer è implementato con un'ontologia ed è usato per rendere persistenti gli oggetti del modello di dominio.

Gli sviluppatori possono riferirsi a pattern come Active Record o Mapper (Fowler 2002) ed anche adoperare strumenti (Stanford University 2006), (Kalyanpur 2004) per fare corrispondere gli oggetti del modello di dominio agli individui del modello ontologico. Per il portale web viene adoperato un Mapping layer tra domain e persistent layer perché esso permette un miglior disaccoppiamento, un test più semplice e lo sviluppo concorrente (Heumann 2001, Beck 2002).

In particolare si può notare che il Domain Layer contiene più sottocomponenti. Il Domain Model mostrato più avanti, il Domain Finder contenente un insieme di classi per l'interrogazione delle base di conoscenza al fine di recuperare gli oggetti di Domain Model ed infine il Service Sublayer che costituisce una faade di Domain Model e Domain Finder per gli strati superiori dell'applicazione. È interessante notare che il Domain Layer riesce ad accedere a tutte le funzionalità del Mapping Layer sottostante semplicemente attraverso l'interfaccia Session.

Il Mapping Layer gestisce la mappatura tra modelli e meta-modelli elabora relazioni complesse come la reificazione, nasconde i codice SPARQL embedded e migliora le prestazioni con metodi di caching e proxing. In ul-

ily 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).

Salient Aspects of the Implementation

Layering

The source code of the web portal (Fig.7) is organized in three layers (Schmidt 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 2001), which somehow reproduces the well known problem of Impedance Mismatch between Objects and relational databases (Ambler 2003).
Specifically: 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 used to persist objects of the domain model.
Developers can refer to patterns such as Active Record or Mapper (Fowler 2002) and also use tools (Stanford-University 2006), (Kalyanpur 2004) to let objects of the domain model correspond to individuals of the ontology model.
For the web portal, a Mapping layer was used between domain and persistent layers because it allows better decoupling, easer testing and concurrent developing (Heumann 2001, Beck 2002). The Mapping layer manages the mapping between models and meta-models, elaborates complex relations i.e. 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 i.e. Jena (Company 2002) so that is easer to change the used library (Aduna 2007) and that could useful for a rapidly changing domain such as ontologies.

Domain Model

The POJO Model in the domain layer is composed by two Java packages which are derived from three ontological models. The User package contains data about profiles, users and related information, and it is automatically derived from an ontology User model, so as to map ontology classes and properties to OO classes and attributes. The Domain package has responsibility to manage information contributed by users and is derived from an ontological Domain model according to the architectural pattimo e non per importanza solo il Mapping Layer si riferisce alla API di basso livello che Jena (Company 2002) cosicché pi facile cambiare la libreria usata (Aduna 2007) e questo potrebbe essere utile in un dominio in così rapido mutamento come quello delle ontologie
Per meglio comprendere la struttura del Mapping Layer conveniente osservare che è costituito da tre componenti principali:
• la libreria Loom (da noi realizzata) che offre tutte funzionalità di base necessarie per realizzare comodamente uno strato di mapping.
• i mapper e finder specifici per il dominio realizzati come classi che estendono alcune classi di base contenute in Loom Api.
• infine altre librerie aggiuntive per l'accesso a basso livello di modelli ontologici OWL, Jena API.

Modello di Dominio

Il Modello POJO nel domain layer composto da due package Java che sono derivati da tre modelli ontologici. Lo User package contiene i dati relativi ai profili, agli utenti ed i dati relati ed è automaticamente derivato dal modello ontologico User, in modo da mappare classi e proprietà dell'ontologia in classi ed attributi OO. Il Domain package ha la responsabilità di gestire l'informazione contribuita dagli utenti ed è derivato dal modello ontologico Domain in accordo al pattern architetturale reflection (Schmidt 2000): tutti i tipi del modello ontologico sono mappati in una singola classe OO generica (che è una meta-level-class nel pattern reflection); questa classe generica ha la responsabilità di gestire le relazioni tra i tipi definiti dagli utenti nel modello ontologico (che comprende le classi base-level-classe del pattern reflection).
Questo disaccoppia la struttura del domain layer OO dagli specifici tipi definiti nell'ontologia Domain, abilitando quindi al riuso del layer OO per una varietà di differenti modelli ontologici di dominio, definendo differenti evoluzioni di un portale o differenti portali insistenti su differenti domini applicativi. Questa è anche la funzionalità che permette ai portali cooperativi di gestire la contribuzione non solo degli individui della parte estensionale ma anche dei concetti della parte intensionale della base di conoscenza.
La derivazione di Domain package è influenzata da un'ontologia subordinata che definisce le direttive per la presentazione dei dati nella struttura della pagina. Gli individui di questa ontologia sono usati nel Mapping Layer per determinare la presentazione ed il filtraggio di concetti definiti nel modello ontologico Domain. Questo assolve ad una responsabilità che è molto simile a quella dei 'site view graph' nel framework OntoWebber (Yuhui Jin 2001).

tern of reflection (Schmidt 2000): all types in the ontological model are mapped into a single generic OO class (that would be a meta-level-class in the reflection pattern); this generic class has responsibility to manage relations among types defined by the users in the ontological model (that would comprise base level-classes in the reflection pattern). This decouples the structure of the OO domain layer from the specific types defined in the Domain Ontology, thus enabling reuse of the OO layer for a variety of different ontological Domain models, defining different evolutions of a portal or different portals insisting on different application domains. Also, 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 knowledge-base. Derivation of the Domain package is also affected by an additional ancillary 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).

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 (Fowler 2002) includes a mapper class for each element of the OO domain model (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.

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 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.

Mapping Layer

La mappatura tra modelli ontologici e ad oggetti dell'architettura è stata implementata seguendo l'approccio a pattern (Fowler 2002, Schmidt 2000) con l'intento di costruire un framework riusabile.

Mappers

Il Mapping Layer include una classe mapper per ogni elemento del modello di dominio OO (Fowler 2002) (Fig. 14). Ciascuna classe mapper può leggere l'informazione dal modello ontologico ed assegnarla al modello ad oggetti e vice versa ed implementata come un'estensione classe di base astratta DataMapper

I mapper disaccoppiano il Domain Layer orientato agli oggeti dal Persistence Layer ontologico, in modo che un client può ignorare come gli oggetti sono resi persistenti nell'ontologia. Per la ragione delle prestazioni e l'astrazione del comportamento i DataMapper implementano alcuni pattern specifici (Fowler 2002, Gamma 1995):

• Identify Map (Cache): i mapper hanno una memoria cache per velocizzare l'accesso ripetuto allo stesso oggetto.

• Proxy: se possibile i mapper sostituiscono gli oggetti richiesti dal client con proxy equivalenti. Questo ritarda le operazioni di mappatura finché non sono veramente richieste.

• LazyLoad: i mapper caricano gli oggetti di una lista quando sono usati cosicché le lente operazioni di mappatura sono eseguite solo per gli oggetti utili.

• UnitOfWork: la classe UnitOfWork gestisce i cambiamenti agli oggetti che i mapper devono rendere persistenti.

Gli sviluppatori possono usare un'istanza della classe Session per accedere alle funzionalità del framework di mapping (Fig. 15).

Il portale muddy

Muddy è un'applicazione di tipo web implementata come istanza dei requisiti e dell'architettura finora descritta. Essa permette la lettura e scrittura di informazioni concettuali e concrete in accordo al paradigma ontologico, fornendo le seguenti funzionalità:

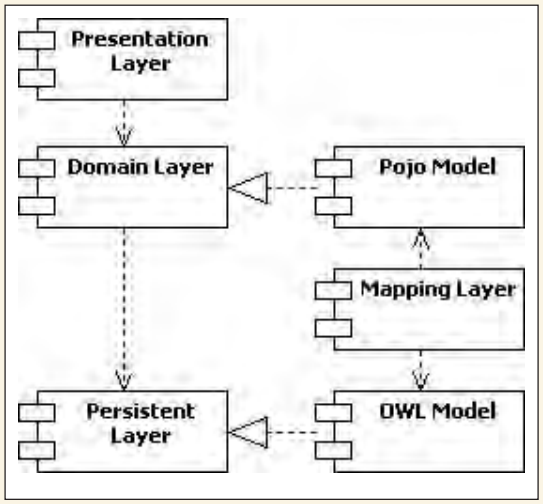
• navigare l'informazione seguendo link semantici;

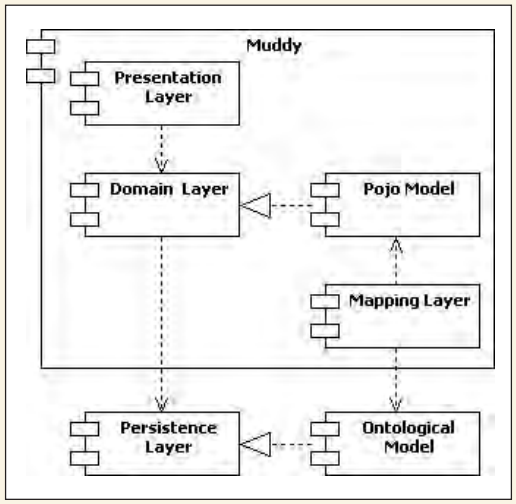
• eseguire interrogazioni semantiche sulla base di conoscenza;

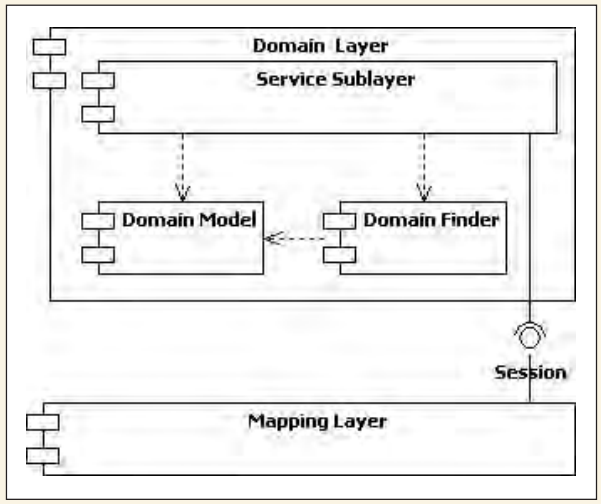
• contribuire nuova conoscenza caricando file di modello;

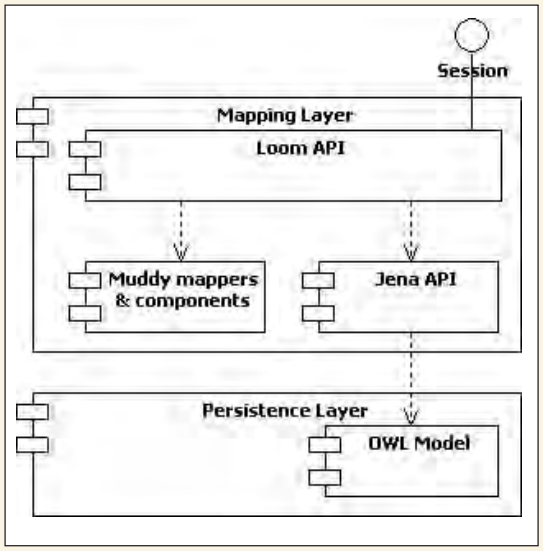
• leggere/scrivere feedback circa la base di conoscenza.

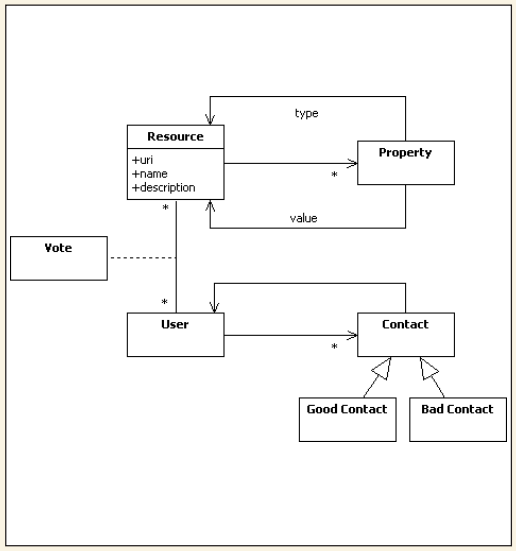
Come tratti caratteristici si sottolineano i seguenti: gli utenti conoscono il tipo di tecnologia impiegata per modellare i dati a differenza dei sistemi

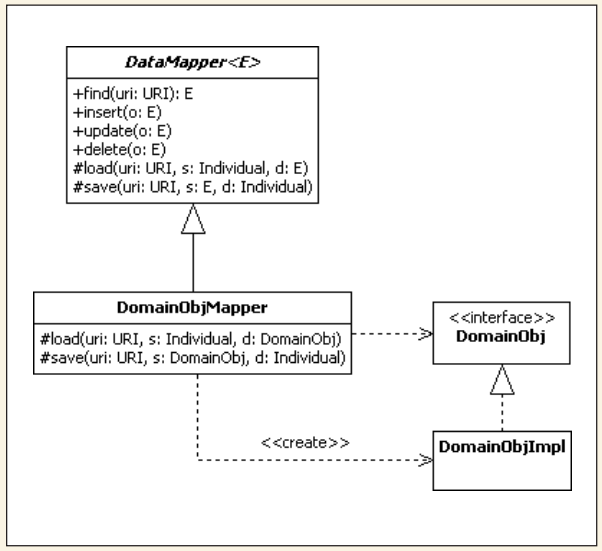


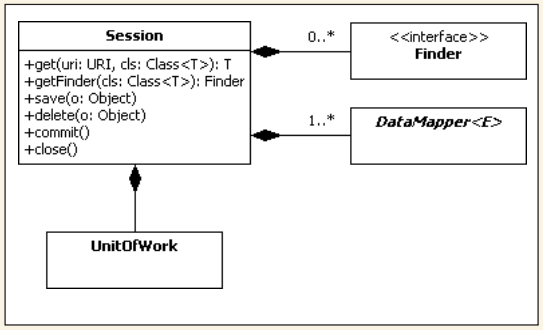












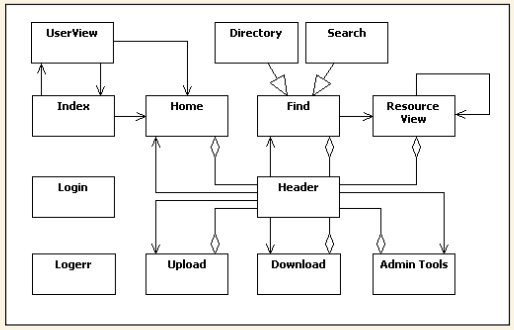


Fig. 9: Web portal layering.
Fig. 10: Muddy Web portal layering.
Fig. 11: Structure of the Muddy Portal.
Fig. 12: The Directory search page.
Fig. 13: The Search page.
Fig. 14: The ResourceView page

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

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 the kind of technology employed to model data, as opposed to systems where a service is offered to users who ignore the underlying technology; users can share arbitrary ontological models, as opposed to applications where users interact with predefined conceptual models by creating, retrieving, updating and deleting individuals only.

The Portal Architecture

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 aministators 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.

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.

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".

This page tries to simplify the use of SPARQL to users.

dove un servizio offerto agli utenti che ignorano la tecnologia sottostante; gli utenti possono condividere modelli ontologici arbitrari a differenza delle applicazioni dove gli utenti interagiscono con un modello concettuale predefinito creando, recuperando modificando e cancellando solamente gli individui o istanze della base di conoscenza.

L'Architettura del Portale

La Fig.16 mostra l'architettura del portale gestito dall'applicazione. Index è la prima pagina del portale con login e registrazione che danno accesso alla pagina Home e poi attraverso la pagina Header alle funzioni del portale. Gli utenti possono essere fruitori, contributori ed amministratori e sono provvisti di differenti funzionalità. Un nuovo utente è sempre classificato come fruitore e solo gli amministratori possono dare agli utenti maggiori privilegi. La pagina Find è usata per eseguire interrogazioni sulla base di conoscenza da parte degli utenti ed è specializzata nelle pagine Search e Directory. La pagina ResourceView permette agli utenti di leggere l'informazione contenuta nella base di conoscenza. Upload e Download sono le pagine che permettono agli utenti di contribuire nuova conoscenza. La pagina Admintools svolge le funzioni per l'amministratore.

Find Pages

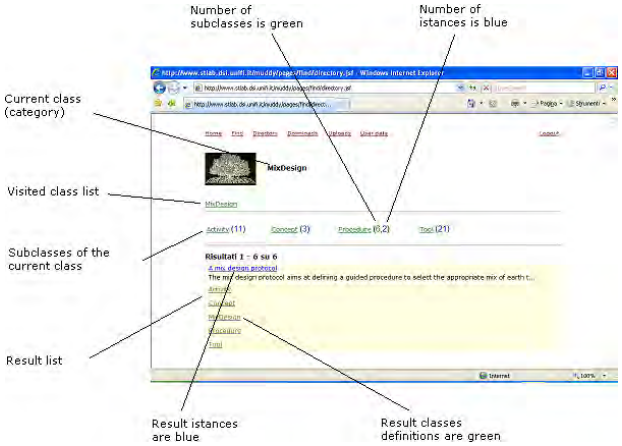
La pagina Directory in Fig. 17 viene usata per eseguire ricerche pro-attive. Il sistema mostra all'utente una lista di categorie che corrispondo alle classi radice del modello ontologico gestito dal portale. L'utente può selezionare una categoria per ottenere una pagina contenente la lista di istanze della categoria e la lista delle sue sottoclassi. L'utente può navigare verso classi più specifiche o ispezionare una delle istanze trovate.

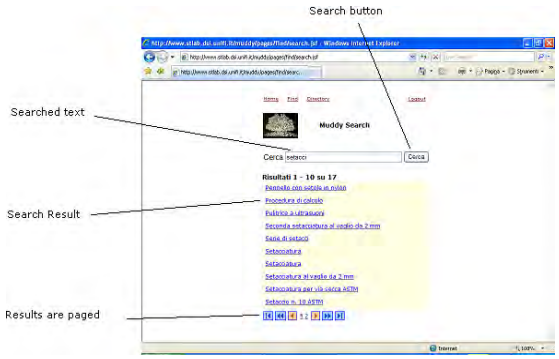
La pagina Search vedere Fig. 18 implementa un'estensione dei metodi di ricerca full-text. Gli utenti possono specificare una o più parole che devono essere contenute nelle risorse desiderate. Essi possono anche specificare il tipo di relazione che lega le risorse desiderate alle parole specificate. Per esempio, l'espressione "neededTools = sieve" permette ad un utente di richiedere tutte le risorse che hanno "sieve" tra "neededTools".

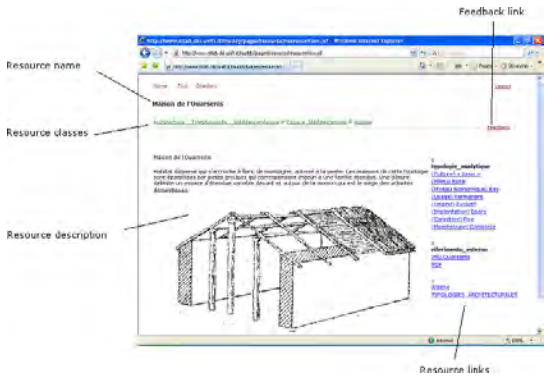
Questa pagina prova a semplificare l'uso di SPARQL agli utenti.

Resource View Page

Questa pagina mostra le informazioni riguardanti una risorsa (Fig. 19), e permette agli utentidi velocizzare la navigazione verso risorse semanticamente relate.







Resource View Page

This page shows information about a resource (Fig. 14), and allows users to speed up navigation towards semantic related resources. The portal also allows users to give feedback about accessed resources which is used to calculate appreciation indexes about resources.

Conclusions and ongoing activity

We are further developing the portal and its underlying architecture, facing various interrelated issues, with less scientific relevance but 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; in this perspective, the orientation towards change in the overall architecture, and in particular the concept of presentation ontology implemented in the mapper, provide major assets to face iterative refinement in design choices;
- 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.

Il portale abilita gli utenti a rilasciare feedback che permettono di assegnare una valutazione alle risorse consultate; da qui si rende possibile calcolare gli indici di apprezzamento per le risorse.

Conclusioni e futuri sviluppi

Il portale è costantemente sotto sviluppo così come la sua architettura sottostante, in conseguenza sia degli sviluppi teorici relativi alle modalità di utilizzo di sistemi di conoscenza basati su ontologie, sia in relazione allo sviluppo nel tempo dei componenti software che popolano e si integrano nel portale. Da questo punto di vista se le problematiche interrelate sono di minor rilevanza scientifica risultano cruciali per estendere la fase prototipale ad uno sviluppo robusto del contesto d'uso. A questo scopo gli sviluppi attesi sono rilevabili in:

- una pianificazione di uno studio di usabilità per valutare la capacità del portale di supportare l'utente a mantenere il contesto nella navigazione attraverso la struttura web-like dei contenuti del portale; in questa prospettiva, l'orientamento verso il cambiamento dell'intera architettura ed in particolare del concetto di ontologia della presentazione implementato nel mapper fornisce il maggior contributo per applicare raffinamenti iteratici alle scelte progettuali;
- la profilatura preliminare delle prestazioni indica le che prestazioni possono essere largamente migliorate con l'integrazione di un repository RDF più elaborato;
- le estensioni funzionali sono possono essere sviluppate per implementare la derivazione automatica di un editor di individui basato sulla struttura dei concetti dell'ontologia ed uno strumento per l'annotazione di risorse web con riferimento ai concetti di un modello ontologico.

Index

Finito di stampare nel mese di ottobre 2008
in Pisa dalle Edizioni ETS
Piazza Carrara, 16-19, I-56126 Pisa
info@edizioniets.com
www.edizioniets.com