FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Semantic integration of Social and Domain Knowledge in a Collaborative Network Platform

Luís Carlos Pacheco Soares Carneiro

Dissertation submitted for partial satisfaction of the requirements for the degree of Master of Informatics and Computer Engineering

Supervisor: António Lucas Soares

20^{th} June, 2010

Semantic integration of Social and Domain Knowledge in a Collaborative Network Platform

Luís Carlos Pacheco Soares Carneiro

Dissertation submitted for partial satisfaction of the requirements for the degree of Master of Informatics and Computer Engineering

Approved in oral examination by the committee:

Chair: Maria Cristina de Carvalho Alves Ribeiro (Professora Auxiliar)

External Examiner: Nuno Alexandre Pinto da Silva (Professor Coordenador)

Internal Examiner: António Manuel Lucas Soares (Professor Associado)

21^{th} July, 2010

Abstract

The Main goal of this dissertation, partially integrated in the works of the European project H-Know, in the the Colnet Group of INESC Porto, is to integrate semantic capabilities of classification and search of contents in a collaborative platform with social networking functionalities.

H-Know is a European research project in the area of old building restoration and maintenance, particularly in the cultural heritage domain. This project envisaged that social networking based forms of interaction can be used to enhance collaboration both intra and inter organizations. The challenge is to develop a innovative platform for collaboration in business networks, going beyond communication oriented social networking, towards the integration of advanced information and knowledge management.

The integration of semantics in the platform aims to enhance the knowledge inference and retrieval of the contents produced in the platform. This semantic module is based on two ontologies: an ontology of social networking and collaboration specifically adapted to collaborative networks of SME's and a domain knowledge ontology in the areas of Civil Construction intervention and cultural heritage. There will be presented the social networking and collaboration ontology developed integrated with the domain ontology. This integration is crucial to relate the socio-collaborative activities managed in the platform with concepts of the domain knowledge. For instance, to express in a automated semantic way, that we have a Blog entry about a Rehabilitation Process published by me.

The research done in the ontology and Semantic Web fields lead to some ontologies for social networking and collaboration, such as the FOAF (Friend of a Friend), the SIOC (Semantically-Interlinked Online Communities Project) and the SKOS (Simple Knowledge Organization System) ontologies. The ontologies created for the H-Know platform are based on these existing ontologies.

For the use of these ontologies in the platform, a technological architecture will be described and a set of functionalities for contents classification and retrieval will be presented and developed in this work.

Keywords: Ontology; Collaborative Network; Social Network; Knowledge Management; Semantic Web; FOAF; SIOC; SKOS; Drupal

Resumo

O principal objectivo desta dissertação, parcialmente integrada nos trabalhos do projecto Europeu H-Know, no grupo Colnet do INESC-Porto, é integrar capacidades semânticas de classificação e pesquisa de conteúdo numa plataforma colaborativa com funcionalidades de rede social.

O H-Know é um projecto de investigação Europeu na área de rehabilitação e manutenção de edíficios antigos, particularmente na área da herança cultural. Este projecto encara as interacções de rede social como uma forma de potenciar a colaboração tanto intra como inter organizações. O desafio é por isso desenvolver uma plataform inovadora para colaboração em redes de negócio, que seja mais que uma rede social apenas focada na comunicação entre pares, tendo em visto a integração de gestão avançada de informação e conhecimento.

A integração de semântica na plataforma pretende melhorar a inferência e pesquisa de conhecimento produzido na plataforma. Este módulo semântico é baseado em duas ontologias: uma ontologia social e de colaboração especificamente adaptada a redes colaborativas de PMEs e uma ontologia de domínio nas áreas de intervenção e herança cultural da Construção Civil. Vai ser apresentada a ontologia social e colaborativa desenvolvida, integrada com a ontologia de domínio. Esta integração é crucial para relacionar actividades socio-colaborativas geridas na plataforma com conceitos do domínio de conhecimento. Por exemplo, para expressar numa forma semântica e automatizada, que temos uma entrada de Blog sobre um processo de rehabilitação publicado por mim.

A investigação levada a cabo no campo das ontologias e da Web Semântica levaram ao apareciamento de ontologias para descrever actividades socio-colaborativas, tais como o FOAF (Friend of a Friend), o SIOC (Semantically-Interlinked Online Communtities Project) ou o SKOS (Simple Knowledge Organization System). As ontologias criadas para a plataforma H-Know são baseadas nestas ontologias já existentes.

Para o uso destas ontologias na plataforma, uma arquitectura tecnológica vai ser descrita e um conjunto de funcionalidades para a classificação e pesquisa de conteúdo vão ser apresentadas e desenvolvidas neste trabalho.

Palavras-chave: Ontologia; Rede Colaborativa; Rede Social; Gestão de conhecimento; Web Semântica; FOAF; SIOC; SKOS; Drupal

Acknowledgements

This section is dedicated to the people that contributed to help me in the realization of this dissertation. To all of them I'm sincerely grateful.

In the first place I would like to thank my supervisor, Prof.Dr. António Lucas Soares. His cordial availability to help and the feedback and orientation guidelines he gave me, were fundamental to the success of my work. I'm thankful for that and also for the freedom he gave me to manage my time and make decisions, which surely contributed for my personal development.

In the second place I would like to thank all the members of the ColNet group, especially Pedro Castanheira for his contribution to help me define the requirements of the dissertation and understand the context of the Project I was involved in, Cristóvão Sousa for his good mood, careful and patient recommendation in implementation and theoretical components of my dissertation and Manuel Silva, a person from a Linguistic area of studies which gave me several times a different vision about my dissertation and with whom I learned things outside my area of studies.

I can't forget to thank Fábio Alves my colleague in INESC-Porto, with whom the collaboration was important for his and my project.

Finally I must thank my family for giving me the support and strength to have success in my dissertation.

Luís Carneiro

Contents

1 Introduction **1**
- 1.1 Context and Motivation . . . 1
- 1.2 Research questions and objectives . . . 2
- 1.3 Technological and methodological approach . . . 2
- 1.4 Contributions and results achieved . . . 3
- 1.5 Structure of the dissertation . . . 4

2 Semantics and Collaborative Social Networking **7**
- 2.1 The importance of semantics in collaborative information management . . . 7
- 2.2 The role of semantics in collaborative processes and social networking . . . 8
- 2.3 Technologies for semantic enabled systems . . . 8
  - 2.3.1 Semantic Web . . . 9
  - 2.3.2 Components of the Semantic Web . . . 10
- 2.4 Semantic Web Storage Management . . . 14
- 2.5 Collaboration and Social Networking Ontologies . . . 15
  - 2.5.1 FOAF . . . 16
  - 2.5.2 SIOC . . . 18
- 2.6 The Semantic Web on Social Media . . . 20

3 Modelling the integration of semantics in the H-Know platform **23**
- 3.1 The H-Know project . . . 23
- 3.2 The H-Know platform . . . 24
  - 3.2.1 Conceptualisation . . . 24
  - 3.2.2 Functionalities . . . 27
  - 3.2.3 The Drupal framework . . . 29
- 3.3 Platform Semantic Conceptualization . . . 33
  - 3.3.1 Conceptualization of the Domain Knowledge . . . 34
  - 3.3.2 Formalization of the Domain Knowledge . . . 38
  - 3.3.3 Competency Questions . . . 42
- 3.4 Mapping platform structure to ontologies . . . 43
  - 3.4.1 Integrating platform and domain ontologies . . . 46
- 3.5 Semantic classification of the platform content types . . . 47
- 3.6 Semantic classification of platform Content information . . . 48
- 3.7 Semantic search of the content . . . 50
  - 3.7.1 Facet-browsing . . . 50

4 Implementing the integration of semantics in the H-Know platform **55**
4.1 Platform Semantic Use Cases 55
4.2 Drupal Semantic Web existing projects 57
4.2.1 Semantic Classification 57
4.2.2 Semantic Querying 58
4.2.3 Semantic Browsing 60
4.3 Platform Semantic Implementation 61
4.3.1 Architectural View 61
4.3.2 Storing the semantic metadata 64
4.3.3 Implementing the Semantic Parser Web Server 66
4.3.4 Implementing the platform structure classification 67
4.3.5 Managing the domain ontology 69
4.3.6 Implementing the platform content classification 70

5 Conclusions and Future Work **75**

References **79**

List of Figures

1.1 Semantic integration methodology . . . 2
1.2 Dissertation structure . . . 4

2.1 Semantic Web Stack diagram [BL05] . . . 10
2.2 Semantic triple graph representation . . . 12
2.3 Description of the FOAF importance . . . 16
2.4 List of FOAF terms [BMa] . . . 17
2.5 Example of FOAF mapping . . . 18
2.6 Overview about SIOC . . . 19
2.7 SIOC classes diagram [BBa] . . . 20
2.8 FOAF, SIOC and Data Portability [BPBD08] . . . 21

3.1 H-know high level architecture . . . 24
3.2 H-know Collaborative Place . . . 25
3.3 H-Know platform Collaborative Place printscreen [Con10] . . . 26
3.4 H-Know platform User Use Cases diagram . . . 27
3.5 H-Know platform Collaborative Place Use Cases diagram . . . 28
3.6 H-Know platform Search Use Cases diagram . . . 29
3.7 Drupal structure layers [ic10] . . . 30
3.8 Drupal "building blocks" . . . 31
3.9 CCK module configuration for the Enterprise content-type [Con10] . . . 31
3.10 The inclusion of semantics in H-Know platform . . . 34
3.11 Collaborative places semantic conceptualization . . . 35
3.12 H-Know knowledge domain, first level concepts . . . 36
3.13 H-Know knowledge domain "Construction Result" concept . . . 37
3.14 H-Know knowledge domain "Construction Process" concept . . . 38
3.15 H-Know knowledge domain "Construction Resource" concept . . . 39
3.16 H-Know knowledge domain "Technical Topic" concept . . . 40
3.17 H-Know knowledge domain described using SKOS . . . 42
3.18 Drupal node structure mapped into ontologies (based on [Cor]) . . . 43
3.19 H-Know platform integration of semantics . . . 44
3.20 Semantically expressing the connection between H-Know users . . . 45
3.21 H-Know content items linking with Domain Concepts . . . 46
3.22 Mockup of the Semantic classification interface of a platform content type 47
3.23 Prototype of the Semantic classification interface . . . 48
3.24 Mockup of the semantic browser filter interface . . . 49
3.25 Mockup of the semantic browser interface . . . 49

3.26 H-Know platform Semantic Search blocks 50
3.27 H-Know content facet-browsing mockup 51
3.28 H-Know users facet-browsing mockup 52
3.29 H-Know collaboration places facet-browsing mockup 53
3.30 H-Know entities facet-browsing mockup 54

4.1 Semantic Use cases of a H-Know "Platform Administrator" 56
4.2 Semantic Use cases of a "Hknow User" 56
4.3 Semantic Use cases of a "Knowledge Manager" 57
4.4 Evoc module configuration 58
4.5 RDF CCK module configuration for the Enterprise content-type 59
4.6 Drupal Sparql Endpoint Module 59
4.7 Example of a exhibit facet-browsing interface [EC] 60
4.8 H-Know platform high-level Semantic Module 61
4.9 H-Know platform Semantic module architecture 62
4.10 OpenRDF Workbench repositories manager 64
4.11 Workbench view of the defined namespaces 65
4.12 Visualization of some part of the knowledge structure loaded into Sesame 65
4.13 Visualization of some part of the knowledge structure loaded into Sesame 66
4.14 Drupal RDF settings page 67
4.15 RDF/XML file exported by RDF CCK 68
4.16 Protégé tool with the SKOSed plugin, editing the domain vocabulary . . . 70
4.17 Ontology Browser interface 72
4.18 Drupal node edit form override 73
4.19 content classification of a node 74

List of Tables

2.1 Triple Stores comparison analysis 15

3.1 Comparison between OWL and SKOS 41

LIST OF TABLES

Abbreviations

API	Application programming interface
ARC	RDF Classes for PHP
CCK	Content Construction Kit
CIK	Construction Industry Knowledge
CMS	Content Management System
Evoc	External Vocabulary Importer Module
FOAF	Friend of a Friend ontology
GWT	Google Web Toolkits
H-Know	Heritage Knowledge
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technologies
INESC	Instituto de Engenharia de Sistemas e Computadores
ISWC	International Semantic Web Conference
JSON	JavaScript Object Notation
OWL	Web Ontology Language
PHP	Hypertext Preprocessor
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
RDFa	Embedding RDF triples in XHTML documents
RDF	Resource Description Framework
RR&M	Rehabilitation, Restoration and Maintenance
RTD	Research & Technical Development
SIOC	Semantically-Interlinked Online Communities
SKOS	Simple Knowledge Organization System
SME	Small and Medium-sized Enterprises
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
WSDL	Web Services Description Language
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Context and Motivation

Semantic technologies are in intense development nowadays, aiming at being used both generically (web) and particularly by teams/organizations/networks for specialised business uses. The semantic web goal is transform the web from a linked document repository into a distributed knowledge base and application platform, thus allowing the vast range of available information and services to be more effectively exploited. In order to have new semantic-enable information systems there is the need to integrate specific semantic vocabularies with semantic artefacts, such as ontologies or taxonomies.

The work described in this dissertation is contributing to the H-Know[1] (Heritage Knowledge), a European research project (2009-2011) in the area of the management of old building rehabilitation, restoration and maintenance.

The complex, multidisciplinary world of building restoration, rehabilitation and maintenance (RR&M) works, is predominantly operated by SMEs. The complexity and dimension of most of the RR&M works places high barriers to SMEs preventing them frequently of taking advantage of business opportunities. The H-Know project aims to develop a socio-collaborative platform where SMEs can share knowledge about restoration and maintenance activities, inducing learning and training of partners and collaboration amongst partners.

From this context we have a goal which is improving the knowledge organization and inference of the content produced in the platform, by semantically expressing the socio-collaborative activities held there connected with the domain knowledge of the H-Know project.

[1] http://www.h-know.eu/

1.2 Research questions and objectives

Considering a platform supporting a collaborative network of SMEs through integrated functionalities of social networking and content management, the specific objectives of this dissertation are:

- To develop a concept and method for "semantic enabled social network" to enable the semantic integration of domain knowledge and collaboration content;
- To develop a set of content classification and retrieval functionalities embedded in collaboration activities (including contents created by collaboration);
- To develop a method for the creation and use of semantic information in social networking/collaboration activities.

1.3 Technological and methodological approach

To build the integration of semantics in a already developed and published socio-collaborative platform, a methodology with several steps is presented in the diagram 1.1.

The approach is organized in 3 different sets of steps, represented with different colours, with the labels of the relations in the diagram representing the order of the steps in the approach.

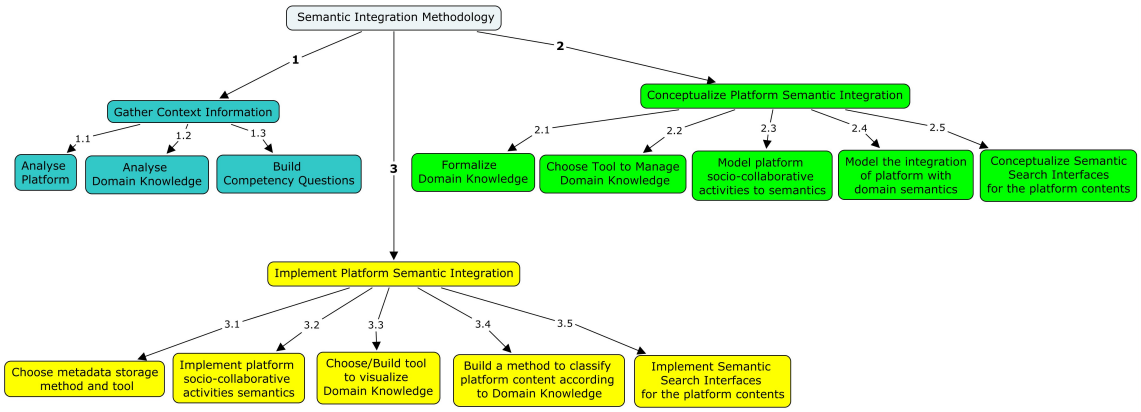


Figure 1.1: Semantic integration methodology

To integrate semantics in a socio-collaborative platform, first of all we need to understand the context of the problem. This means analysing the platform structure, purposes and background (**1.1**) and having a global vision about the domain knowledge managed in the platform (**1.2**).

These two steps allow us to build competency questions (**1.3**), essential to define the importance and the objectives of the semantics for the platform and to understand what

kind of questions the semantic enhancing of the platform can answer that traditional ways of searching cant.

The conceptualization of the semantics in the platform can start after these base tasks are finished. We should start by defining an approach to formalize the Domain Knowledge (**2.1**) and a tool to build and manage that knowledge (**2.2**).

Then, picking up the structure of the platform, its actors and activities, a model to semantically describe the platform socio-collaborative activities should be designed (**2.3**). To relate the domain knowledge of the problem with the socio-collaborative activities, we need to build another model to integrate both semantics (**2.4**). The last step of this conceptualization process should focus on the designing of searching interfaces enhancing the semantic generated metadata of the platform (**2.5**), taking into account the "Competency Questions" previously formulated.

The last group of steps are intended to implement the conceptualizations previously done. First, the method and tool for the semantic metadata storage must be defined (**3.1**). Then, from the semantic model defined to describe the socio-collaborative activities, we should implement it in the platform (**3.2**), so it can generate and store semantic metadata. Since the platform has a specific domain knowledge managed there, a tool to visualize that domain must be chosen or built (**3.3**) together with a method to allow platform users to classify the content produced (**3.4**). The last step of all the process is making use of the semantic metadata generated, implementing the semantic interfaces (**3.5**) that were previously defined.

1.4 Contributions and results achieved

The contribution of this work is the innovative concept of a collaborative social networking platform for SME companies in the area of old building restoration and maintenance, with semantic capabilities as a mean to enhance the search of content and the search of people or entities with specific capabilities or knowledge.

This platform aims to give systematic access to the construction knowledge related to the area of building restoration and maintenance, and to the state-of-the-art processes and materials to be applied.

In more detail, this dissertation project had as expected results:

- An ontology of social networking and collaboration specifically adapted to collaborative networks of SMEs;
- A method for the integration of the social networking ontology with a domain ontology;
- A set of functionalities for collaborative contents classification and retrieval in a content management and social networking platform.

The expected results were almost completely achieved. It was developed a ontology model that aims to integrate both socio-collaborative and knowledge semantics using W3C ontologies (FOAF, SIOC and SKOS), a kind of approach we haven't seen so far in other research projects. In the other hand it was designed and implemented an innovative technological architecture for the problem, where the semantic metadata generated in the platform is stored outside the platform in a native triple store. Part of this architecture is also a "Ontology Browser" built from scratch to enable users to classify the content they produce. Search and retrieval functionalities were thought and designed but not yet implemented.

1.5 Structure of the dissertation

This dissertation is structured in 5 chapters presented in the following diagram:

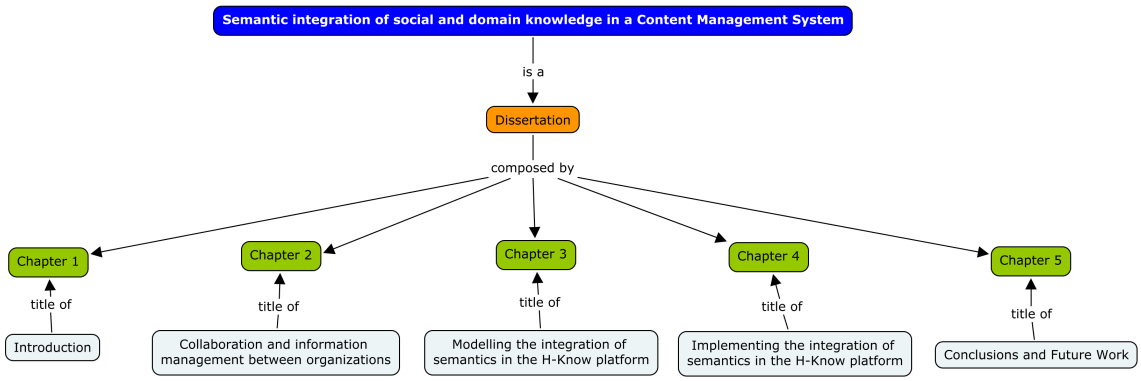


Figure 1.2: Dissertation structure

- In the **chapter 1**, that is now finishing, it is presented the context where this dissertation takes place, so as the objectives it should achieve, the technological and methodological approach to be followed and the expected results from this dissertation;
- **Chapter 2** talks about the advances in the Web Semantic field, presenting the most relevant technologies and tools involved on it, with a focus on the role of semantics in socio-collaborative processes;
- **Chapter 3** presents the conceptualization of the semantics integration in the H-Know platform. It starts by giving an overview about what is the H-Know project and the platform that comes as the deliverable result of it. Then, it's presented an overview about the domain knowledge managed in the H-Know project and the models created to semantically describe that domain. It's also presented in this chapter the model to semantically express the socio-collaborative activities that take

place in the platform. This chapter ends by presenting some non-functional prototypes of user interfaces needed for the semantic integration in the platform.

- **Chapter 4** starts by presenting the use cases that arise from the conceptualization given in the previous chapter. Then the implemented technological architecture is explained, based on the needs discovered. Each one of the modules in the architecture is described.

- **Chapter 5** presents the conclusions of this work, where it's analysed in what extent the objectives defined in the beginning were achieved. After this, there's a very important part which is analysing what should be done in the future to further improve the results of this dissertation.

Chapter 2

Semantics and Collaborative Social Networking

2.1 The importance of semantics in collaborative information management

In a global market environment, where competition is always increasing, collaboration is becoming more and more essential for improving productivity and accelerating innovation in a personal, team, group, enterprise and business coalition levels[SLnn+07]. The exchange of knowledge among people allows them to communicate complex ideas and to collaborate in creating value[Tap06]. By collaboration we mean the increasing richness of means by which objects (things, people and firms) can work together enhanced by the medium of the Internet[Tap06]. The strategy beyond collaboration networks, intends to harness the network to optimize workforce utilization, develop professional staff, retain talent and locate expertise[Bro].

Nowadays, the amount of information existing in the World Wide Web is completely overwhelming. It's becoming difficult, and soon impossible, to process it all. This phenomenon has a big impact on the efficiency how people use information on their collaborations.

Individuals in collaborative networks all across the world are facing information overload. In an attempt to manage information, systems of knowledge management have been developed. These kind of systems allow us to deal with large volumes of documents such as reports or articles. Where many of these systems fail is in the way they provide tools to search, process and manage information in a truly useful way for its users. Most of the Knowledge Management systems don't handle information in different formats and structures in a way that preserves the most important aspect of information: its **meaning**.

So, the problem of information overload persists. We can conclude that the problem of information overloading is not in the amount of information but in the way we handle that information.

The use of semantics, aims to transform knowledge management systems into systems that can understand and analyse the meaning of information. A semantic approach provides a possible solution to deal with the problem of information overload.[1]

2.2 The role of semantics in collaborative processes and social networking

In order to enable efficient collaborations within collaboration partners, structured representations of the organizational processes and knowledge shared are needed[DLHA09]. Assuming that one and same context can be modelled in different ways, allowing different interpretations, it should be specified a conceptualisation, an abstract, simplified view of the world that we wish to represent for some purpose[G+95]. To build this conceptualization, ontologies are used, as logical theories for that conceptualization. This way, all partners collaborating, are communicating in the same dialect and so, it's easier for them to understand each others.

Using semantics along with ontologies, we can specify in a structured way, collaboration activities (blog page editions, forum posts, ect), people and domain knowledge, and arrange them in a meaningful manner. Having all the contents produced in collaborative processes and social networking correctly classified and structured, we are able to do meaningful queries over that contents. This way it is easier to find related information about a specific area of knowledge or to find a collaboration partner that had some contribution for that area.

From this vision of the importance of semantics for collaborative processes, social networking and information management we will give in the following chapter an overview about the semantic web and the technologies related to it.

2.3 Technologies for semantic enabled systems

In the Internet the amount of information is increasing day by day. Frequently, finding what you need is a difficult task since the information is spread all over the Web and many times the contents is not well categorized and so, difficult to find. Searches are imprecise, often returning pointers to many thousands of pages[FHLW03].

Since Web 1.0, computers have been able to understand how a webpage is written - this is the syntax. Every language has its own syntax and for the web, HTML is the syntax.

[1] http://www.intellexer.com/why_semantics.html

Understanding how a webpage is written, and how it should be rendered and displayed is not enough to understand the meaning of its contents. This is what semantic web tries to solve.

2.3.1 Semantic Web

The main idea proposed by the Semantic Web, also known as Web 3.0, [Flo09] is to tell in a web document the meaning of the contents included there, so that not only humans can understand it but also machines can understand it. Extracting the meaning from text can be a very challenging task for computers[FDZJ05]. So, Instead of just saying that in a specific part of a document we have some text in a "h1" style, lets also say that it is a title of a book. We say that that part has a "h1" style and that a title of a book is being specified. The characteristics of the book are detailed in a **ontology**. Ontologies specify metadata schemas, providing a controlled vocabulary of concepts, each with explicitly defined and machine-processable semantics[MS01]. Ontologies specify the classes of objects that exist, the relationships amongst those classes, the possible relationships amongst instances of the classes, and constraints over those instances. [FDZJ05] We don't have only text in internet but also images, videos, etc. Using semantics in web documents, we can make relationships among pieces of data in different formats. Data should be accessed using the general Web architecture, e.g., URI-s. Data should be related to one another just as documents (or portions of documents) are already.[Her]

This is the ultimate goal of the semantic web: making the Web a Web of data[Her], where all the information has exact meaning. It is important to state, that semantic web is not an attempt to substitute the current Web but an extension of the current one in which information is given well-defined meaning, better enabling computers and people to cooperate together [HLB01].

The real use of the Semantic Web is realized when different programs (computer Agents) collect Web contents from diverse sources, process the information and exchange the results with other programs [HLB01]. This process provides huge advantages in information searches, since web contents has this way a meaning.

As we know, the Web itself did not start as commercial service, it began in research facilities with private, personal Web sites. Similarly, nowadays, Semantic Web is currently being used mainly in very specific domains such as the medicine or genetics research areas [AKTV08]. This tendency is gradually changing since more and more semantic web projects for masses appear. One of the biggest projects already in a advanced state of development is DBPedia[2], an initiative to extract structures of information from Wikipedia, allowing users to ask sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data. Other examples can be the RDF indexers Swoogle[3] and

[2] http://dbpedia.org/About
[3] http://swoogle.umbc.edu/

Sindice[4], which allow searching for millions of RDF statements existing on the web.

In the following chapter there will given a brief overview of the components (formats and languages) that make semantic Web possible.

2.3.2 Components of the Semantic Web

Semantic Web as it was conceive by its creator Tim Berners-Lee, as been widely accepted as a hierarchy of languages, with each language exploiting the features and extending the capabilities of the layers below [HPPH05]. The figure 2.1 shows that hierarchy.

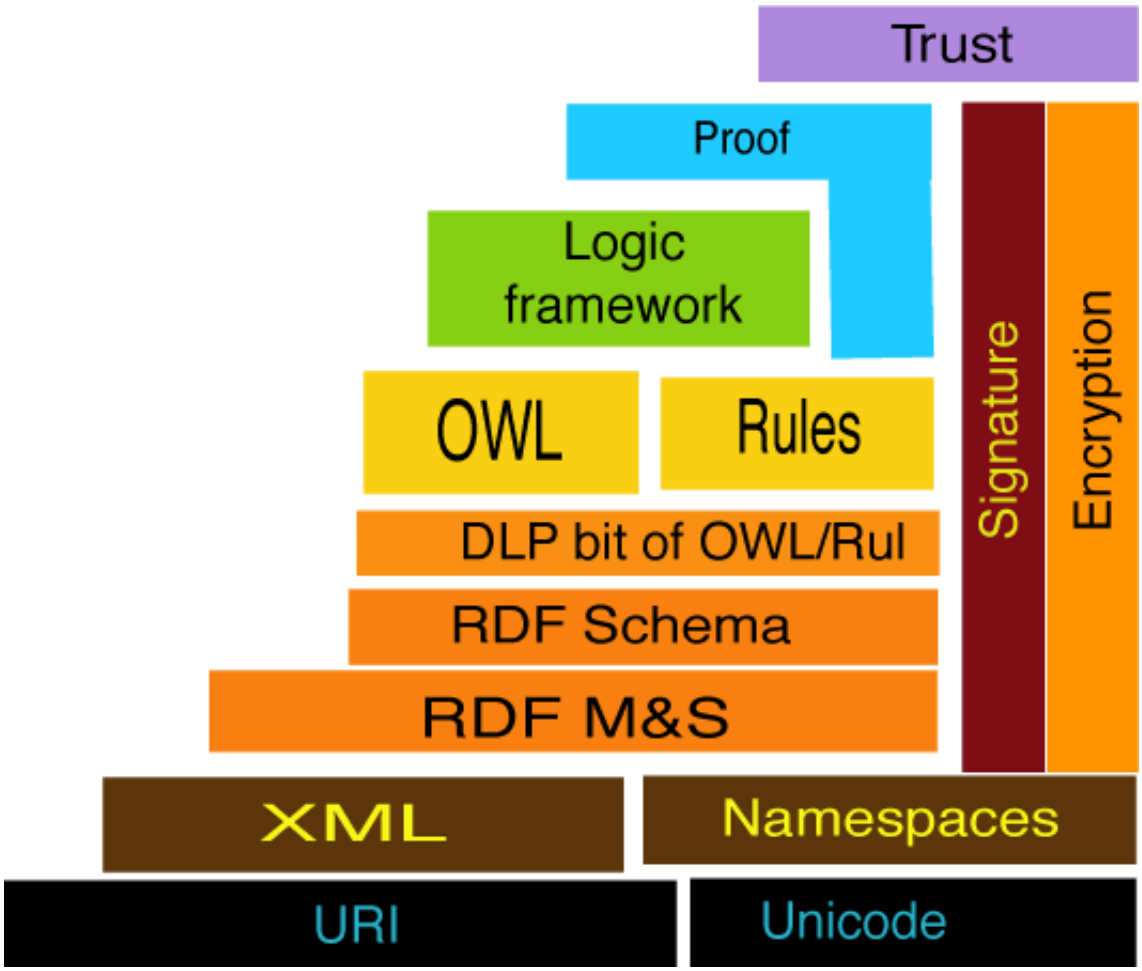


Figure 2.1: Semantic Web Stack diagram [BL05]

The bottom layer, **URI** and **Unicode**, follows the features of the existing WWW. Unicode is a standard of encoding international character sets, allowing humans to write and read on the web using one standardized form. A URI (Uniform Resource Identifier) is a string of a standardized form that allows to uniquely identify resources [Obi02a]. The

[4]http://sindice.com/

most famous subset of URI is the URL (Uniform Resource Locator), which contains access mechanism and a location of a document in a network.

The usage of URI is crucial for the semantic web and for the concept of distributed internet system because it provides understandable identification of all resources.

The **XML** layer with **XML namespace** guarantees that there is a common syntax used in the semantic web. XML namespaces allow us to specify different markup vocabularies in one XML document.

The data representation format for semantic web is **RDF** (Resource Description Framework). RDF stands as a description of a graph formed by triples. Further on in this report we will talk about RDF in more detail.

So users can build their own taxonomies and other ontological constructs, **RDF Schema** (RDFS) was created.

To describe with more detail ontologies, it was created a language: **OWL** (Web Ontology Language), derived from description logics, offering more constructs over RDFS. Like RDFS it is syntactically embedded into RDF.

Since RDFS and OWL have semantics defined, this semantics can be used for reasoning within ontologies and knowledge bases described using these languages [Obi02a].

To provide rules beyond the constructs available from these languages, rule languages are being standardized for the semantic web as well.

In order to query RDF data as well as RDFS and OWL ontologies with knowledge bases, a Query Language was created: **SPARQL**.

The idea is that all the semantics and rules are executed at the layers bellow **Proof** and the results will be used to prove deductions. The formal proof together with trusted inputs for the proof will mean that the results can be trusted.

On top of all these layers applications with user interfaces can be built.

In the rest of this section, it will be described some of the technologies included in the layers presented above.

2.3.2.1 RDF

RDF is the format in the base of the Semantic Web. All data in the semantic web uses RDF as the primary representation language. The normative syntax for serializing RDF is XML in the **RDF/XML** form.

RDF is based on triples **subject-predicate-object** that form a **graph of data** [BM01]. Predicate can also be called a property.

RDF identifies things using Web identifiers (URIs), and describes resources with properties and property values. For example:

- **A Resource (Subject)** is anything that can have a URI, such as "http://luiscarneiro.pt.vu/"
- **A Property (Predicate)** is a Resource that has a name, such as "author".

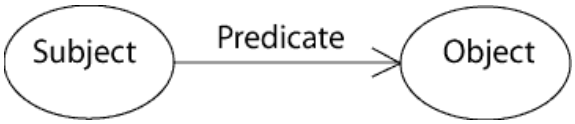


Figure 2.2: Semantic triple graph representation

- **A Property value (Object)** is the value of a Property, such as "Luís Carneiro". A property value can also be another Resource.

Combining these 3 parts we can make a statement, which can be easily read by humans: "The author of http://luiscarneiro.pt.vu/ is Luís Carneiro.".

In RDF the above example would be:

```
<RDF>
<Description about="http://luiscarneiro.pt.vu/">
<author>Luís Carneiro</author>
</Description>
</RDF>
```

2.3.2.2 OWL

The OWL Web Ontology Language is a language for defining and instantiating Web ontologies [SWM01] and is built upon RDF. OWL and RDF are much of the same, but OWL is a stronger language with greater machine interpretation than RDF. OWL has a larger vocabulary and stronger syntax than RDF.

OWL comes in three species:

- **OWL Lite:** for taxonomies and simple constrains;
- **OWL DL:** for full description logic support;
- **OWL Full:** for maximum expressiveness and syntactic freedom of RDF.

To specify the vocabularies being used, we define namespaces. These provide a mean to unambiguously interpret identifiers and make the rest of the ontology presentation much more readable [SWM01]. For example:

```
xmlns ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine"
xmlns:owl ="http://www.w3.org/2002/07/owl"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema"
xmlns:xsd ="http://www.w3.org/2001/XMLSchema">
```

OWL provides a mechanism to describe the classes that individuals belong to and the properties that they inherit by virtue of class membership [SWM01]. An example of a class can be given for the Wine ontology specified by W3C: [Win]

<owl:Class rdf:ID="Region"/>
<owl:ObjectProperty rdf:ID="locatedIn">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl/TransitiveProperty"/>
<rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl/Thing"/>
<rdfs:range rdf:resource="Region"/>
...
</owl:Class>

To describe member of classes, we can declare individuals. Like for example:

<Region rdf:ID="AltoDouro" />

In order to assert general facts about the members of classes, OWL let us define properties. Properties can be:

- **Datatype properties:** relations between instances of classes and RDF literals XML schema datatypes [SWM01]
- **Object properties:** relations between instances of two classes.

To give extra detail to the properties, we can specify their characteristics (transitive, symmetric, functional, etc) and their restrictions (allValuesFrom, someValuesFrom, range, domain etc). [SWM01]

These are the main characteristics of the OWL language. More details are presented in the W3C documentation related with OWL [SWM01].

2.3.2.3 SPARQL

In order to query RDF data, it was developed a language, **SPARQL**, where graph patterns are matched against the direct graph representing the RDF data. SPARQL is a syntactically-SQL-like language [Fei], that includes basic conjunctive patterns, value filters, optional patterns, and pattern disjunction.

Most forms of SPARQL query contain a set of triple patterns called a basic graph pattern [PS]. An example of a SPARQL query can be:

Data: @prefix foaf: <http://xmlns.com/foaf/0.1/> .
foaf:name "Luís Carneiro" .
foaf:mbox <mailto:luiscarneiro@example.com> .
foaf:name "Manuel Oliveira" .

foaf:mbox <mailto:manueloliveira@example.org> .
Query
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
?x foaf:name ?name .
?x foaf:mbox ?mbox
Result
name mbox
"Luís Carneiro" <mailto:luiscarneiro@example.com>
"Manuel Oliveira" <mailto:manueloliveira@example.org>

Just like in SQL language, we can apply filters to the queries and solution sequences and modifiers (order by, offset, limit, etc).[PS]

2.4 Semantic Web Storage Management

In order to enable application developers to use the semantic metadata generated by an application, this semantic metadata needs to be stored. The storage tools that store semantic data are called **Triple Stores**. A triple store is designed to store and retrieve identities that are built from triplex collections of strings [Rus03]. The triples collections represent the **subject-predicate-object** relationship that corresponds to the definition of the RDF standard. Like in a traditional relational database, the triples stored can be retrieved via a query language (SPARQL).

In what concerns the implementation of the triple stores, we can group them in two types: **native** and **non-native** triple stores. Native triple stores are database engines built from scratch, optimized for the storage of triples while non-native triple stores are built on top of existing relational database engines.

The table 2.1 shows a comparison overview of the most relevant existing Triple Stores.

All the triple stores offer more or less the same functionalities. Some of them provide APIs to directly interact with the storage and others just provide connection APIs that allow the usage of generic Semantic Web APIs. Later on this report we will talk about the triple store that was chosen and the reason it was chosen comparing to the other existing triple stores.

Table 2.1: Triple Stores comparison analysis

Name	**Licence**	**Type**	**Technology**	**Main Functionalities**	**API**
Virtuoso [5]	Open Source and Commercial	Native	C	SQL, XML, and RDF data management; SPARQL queries; HTTP upload of data	Provides connection API
Sesame [6]	Open Source	Native	JAVA	Support for RDF Schema inferencing and querying; can be deployed on top of a variety of storage systems	Has its own JAVA API for complete interaction with the storage (query,store,create repository,etc)
Mulgara [7]	Open Source	Native	JAVA	Native RDF support; Simple SQL-like query language; Full text search functionality	Provides connection API
Jena SDB [8]	Open Source	Non Native	JAVA	In-memory and persistent storage; SPARQL query engine; Reading and writing RDF in RDF/XML, N3 and N-Triples;	Provides Java RDF and OWL APIs for complete interaction with the storage
ARC Store [9]	Open Source	Non Native	PHP	RDF Storage (using MySQL); Various parsers; SPARQL Endpoint	Provides a PHP API for complete interaction with storage
Allegro Grapb RDF Store [10]	Free and Commercial Editions	Native	Lisp	persistent RDF graph database; supports SPARQL, RDFS++, and Prolog reasoning;	JAVA Sesame API; JENA API; Phyton API

2.5 Collaboration and Social Networking Ontologies

There have been some efforts on the Semantic Web field, to build ontologies to classify collaboration and social networking activities. In this chapter are presented the two most important achievements so far: FOAF and SIOC.

2.5.1 FOAF

The **Friend of a Friend** (FOAF) project, one of the largest projects in the semantic web [BMb], is a descriptive vocabulary built based on RDF and OWL, for creating a Web of machine-readable pages for describing people, the links between them and the things they create and do [BMb]. It is accepted as standard vocabulary for representing social networks, and many large social networking websites use it to produce Semantic Web profiles for their users [GR08].

FOAF has the potential to become an important tool in managing communities [Dum], and can be very useful to provide assistance to new entrants in a community, to find people with similar interests or to gather in a single place, people's information from several different resources, decentralizing the use of a single social network service for example [GR08].

The things described in the web are connect by people. People attend meetings, create documents, are depicted in photos, have friends, and so on. Consequently, there are a lot of information that might be said about people and the relations between them and objects (documents, photos, meeting, etc) [BMa]. FOAF describes the most common information we usually want to know about a person and because it is built upon RDF, it also uses some vocabulary from other resources, such as the Dublin Core (DC) [BMa].

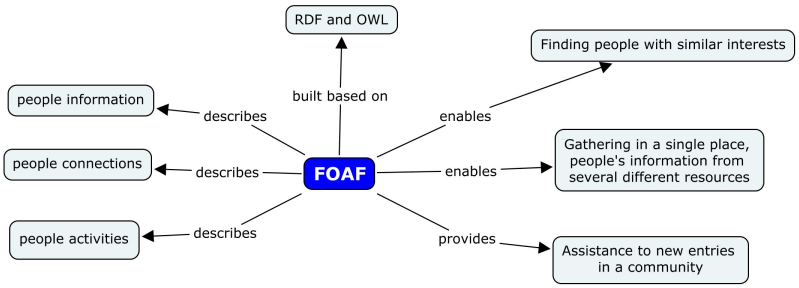


Figure 2.3: Description of the FOAF importance

In the diagram 2.3 we can see a summary of what FOAF stands for.

The base class described in FOAF is the **foaf:Agent** class. The Agent class describes "the things that do stuff" [GR08] and have **foaf:Group**, **foaf:Person** and **foaf:Organization** as sub-classes. FOAF describes resources such as **foaf:Document**, **foaf:Image** or **foaf:OnlineAccount** and people properties like **foaf:name**, **foaf:title** or **foaf:mbox** (email box).

The following figure shows a list of the FOAF ontology classes and properties.

One important property that should be mentioned is the **foaf:knows** property. It can be used to link two people together [Dum]. FOAF identifies other people by stating their properties. An example of a FOAF statement, with the knows property included, can be:

<foaf:Person>
<foaf:name>Luís Carneiro</foaf:name>

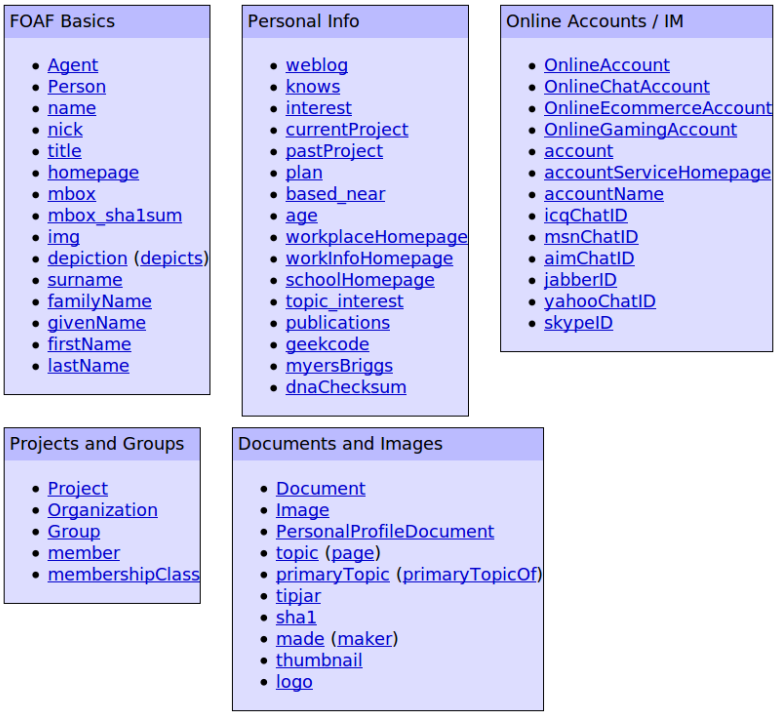


Figure 2.4: List of FOAF terms [BMa]

```
<foaf:mbox rdf:resource="mailto:luis@example.com" />
...
<foaf:knows>
<foaf:Person>
<foaf:mbox rdf:resource="mailto:jose@example.com" />
<foaf:name>Jose Oliveira</foaf:name>
</foaf:Person>
</foaf:knows>
</foaf:Person>
```

The example above describes the Person with name "Luís Carneiro" and email "luis@example.com", that knows the Person with name "José Oliveira", referenced by his email, "jose@example.com".

The diagram 2.5 shows a general use of FOAF to describe users and their interactions in a social community.

There are some concerns related to the use of FOAF [Flo09], such as trust issues, since you can say you know whatever person without any verification and you can create whatever FOAF profile you want.

Nowadays there are already many projects fostering the use of FOAF. Some examples can be:

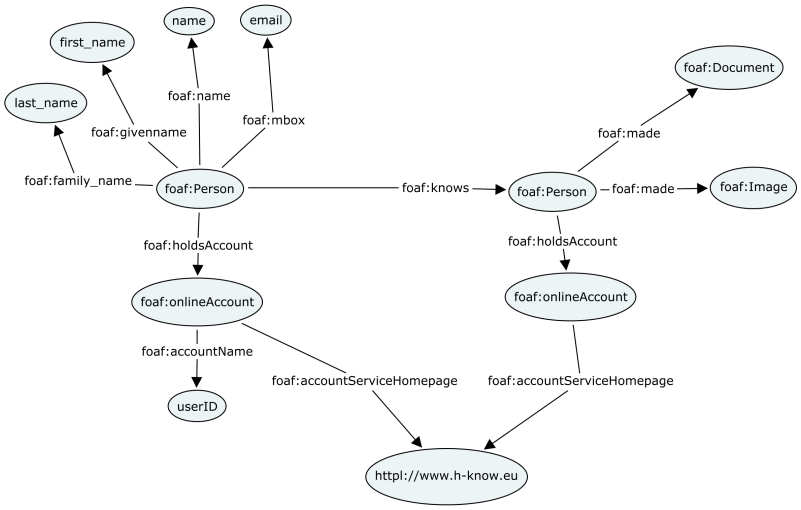


Figure 2.5: Example of FOAF mapping

- **Google Social Graph API:**[11] indexes all the public FOAF data in the Web. Social Graph utilizes public connections their users have already created in other web services.

- **Origo:**[12] a Web-application that enables users to manage their social community profiles utilizing semantic technologies. It allows to unite their different profiles and to browse through their semantic social network across various platforms [VL09], using the FOAF structure and the **RELATIONSHIP ontology**[13] to specify different kinds of relationships between users.

- **Flink:** a system for the aggregation and visualization of online social networks, extracted from a electronic information sources such as web-pages, emails, publication archives and FOAF profiles [Mik05].

2.5.2 SIOC

The **SIOC** project (Semantically-Interlinked Online Communities), is an ontology for representing rich metadata from the Social Web in RDF/OWL, accepted by W3C. It aims to enable the integration of online community information (wikis, message boards, weblogs, etc)[BBb].

SIOC aims to meet the needs of communities and users on the evolving Web, as community-centric content sites become more prevalent and finding relevant items from these communities is now more important than ever [BBFD08].

[11] http://code.google.com/intl/en/apis/socialgraph/
[12] http://code.google.com/p/origo/
[13] http://vocab.org/relationship/.html

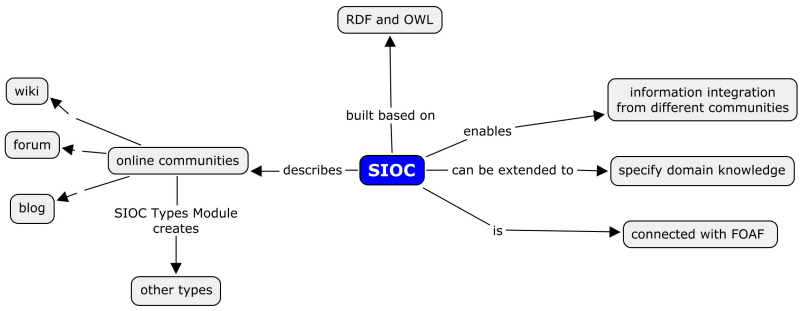


Figure 2.6: Overview about SIOC

The diagram 2.6 sums up the characteristics of SIOC.

As an ontology, SIOC can't incorporate on it everything that might be important to know about communities, about their users and about the contents that users create, otherwise it would be too large[BBa]. Being built over RDF, we can take advantage of other specific description vocabularies, to complement the domain we want to specify. Being built in a modular design, we can create additional ontology modules for specializing and further extending classes and properties contained within the SIOC core ontology[BBFD08]. Currently there are 2 modules defined:

- **SIOC Types Module:** to extend sub-classes of classes such as Forum, Post, Item or Container[BBa].

- **SIOC Services Module:** a sioc:Service allows us to indicate that a web service is associated with (located on) a sioc:Site or a part of it[BBa].

To make the link between SIOC ontology and specific domain ontologies, SIOC Types module uses an **rdfs:seeAlso** property to point SIOC Types objects to the related vocabularies and classes[BBFD08].

In the figure 2.7 we have an overview of the classes that compose the SIOC ontology and the relations between them.

There are SIOC exporter tools that can be used to export RDF information about the contents and structure of Web 2.0 platforms (wikis, forums, blogs, message boards, etc) [BHO07]. This allows information from every page of a site to be represented in RDF, making all the information contained there available in a machine readable form and so, ready for reuse [BBFD08]. Some examples of those exporters are the Wordpress exporter [14] or the vBulletin exporter [15].

[14] http://sioc-project.org/wordpress
[15] http://wiki.sioc-project.org/index.php/VBSIOC

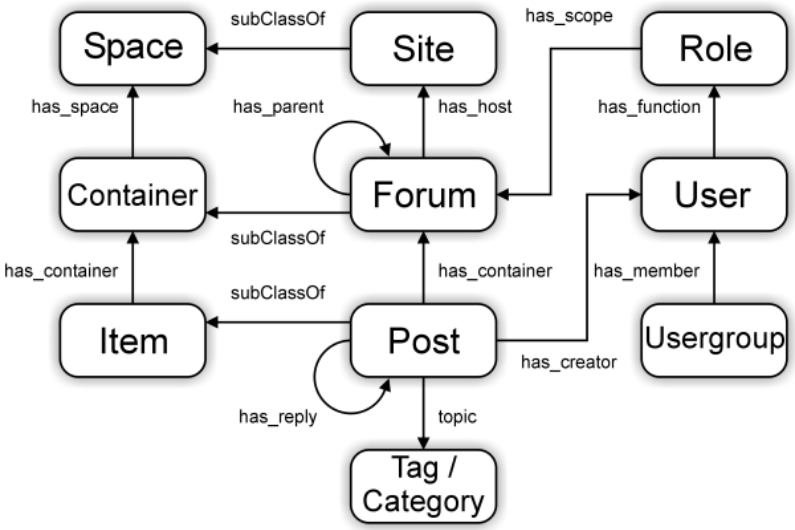


Figure 2.7: SIOC classes diagram [BBa]

2.6 The Semantic Web on Social Media

Nowadays people share their personal and professional information with the world, in several different Social Media platforms like facebook, hi5, orkut, linkdin or twitter. A very important advantage that Semantic Web can provide in this area is related with the possibility of it to standardize the information presented in each one of this sites and allow a cross-platform use of the contents produced there. Having social media contents in a semantic web format, users can feel free to use the platform they like most. Lets say that this way, we can have on a single platform, information coming from facebook and hi5 users for example.

The development of Semantic Social Networks is motivated by the growing need for a formalized representation in the form of meta-data to improve online searches and contents management in a personalized fashion.

Social media sites started open APIs that can be used by other applications to interact in new ways with the site and its data[BBFD08]. The use of this APIs have some limitations of use. We cannot work with clients that have not been designed with the specific API as base [Eng04], their contents cannot be accessed by search engines and other web agents [HJSS06] and each mashup only allows access to data from a limited number of sources chosen by the developer [BBFD08]. Semantic Web can be used to cover this limitations. Semantic Web can be used by generic clients, including RDF browsers, RDF search engines, and web query agents [BBFD08], dismissing the use of APIs.

To build the Semantic Web vision of Social media presented above, two important tools are the FOAF and SIOC ontologies. Imagining a situation where a user Luís as uploaded a video on Youtube for example, we can describe this user using FOAF and

describe the contents he published using SIOC. SIOC and FOAF can be used together to describe the objects in a social network of users[BBFD08]. SIOC can be used to point to external vocabulary to use for a particular specific domain. This way the data about items described in a Blog post or in a forum entry can be structurally classified. SIOC Types module facilitates locating appropriate RDF vocabularies and classes suitable to describe these items[BBFD08].

The figure 2.8 shows an example of how SIOC and FOAF used together can describe social media interactions, providing data portability from different applications.

All the information integrated together, allows us to build a global picture of the objects that a user has interacted with, by creation, discussion or comment, upon across several different network sites, from which the links between the users emerge[BBFD08]. Having all the information described in RDF format, we can ask queries over the data. Semantic Web brings a richer way of describing things than microformat[16]. In microformat, we are limited to the properties of the microformat, hCard or iCal for instance, without being able to make relations between different types of objects. Using RDF, we can make for example a query asking for the persons that have commented about a specific forum topic. If the users share the same identification (for example a URI) across a number of community sites, then information from all of these sites will be returned.

One of the current uses of SIOC on online social communities is "The SIOC explorer [17]" [BHO07], a web application similar to a feed-reader, that allows people to read posts from different web sites in one place.

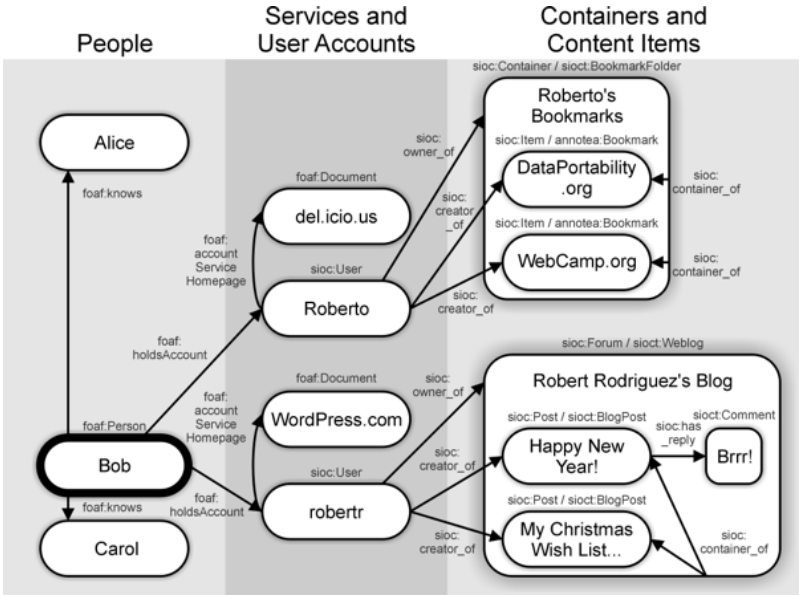


Figure 2.8: FOAF, SIOC and Data Portability [BPBD08]

[16] Webpage about microformats, http://microformats.org/

[17] https://launchpad.net/sioc-ex

Chapter 3

Modelling the integration of semantics in the H-Know platform

3.1 The H-Know project

The work described in this dissertation is contributing to the H-Know (Heritage Knowledge)[1], a European research project (2009-2011) in the area of the management of old building rehabilitation, restoration and maintenance (RR&M), specially in the cultural heritage domain.

The main goal of this project is to develop a methodology, supported by an ICT platform, to manage networks of SMEs and research centres in the RR&M areas, providing facilities for collaboration, information and knowledge and learning management.

The H-Know consortium includes 15 partners, out of which 5 are RTD partners, 8 are SME's, one is an association of the sector and one is a vendor of ICT solutions for virtual marketplaces. The consortium includes all the necessary partners to cover the required multidisciplinary expertise to successfully carry out the required RTD tasks so as to assure the expected results of the project.

The partners of this project are from 5 European countries (Spain, Portugal, Germany, Italy and France), which gives the project a cultural and legal diversity, ensuring that the problem solution will provide a typical real world application scenario for a European-wide reality.

In practical terms, the project solution should offer SMEs possibilities to access specific knowledge by means of a collaborative online community including RTD actors. In such a community, SMEs can share knowledge about restoration and maintenance activities, which induces learning and training of partners and collaboration amongst partners.

[1] H-Know Project Webpage http://www.h-know.eu

3.2 The H-Know platform

3.2.1 Conceptualisation

This section aim is to describe the H-Know platform main concepts, actors and their relationships.

This platform started being developed in November of 2009 by the ColNet [2] (Collaborative Networks) team of INESC Porto and is now being further improved by this team and other partners of the H-Know project. This platform will result in a early prototype of the H-Know project. This dissertation project only contributes to the integration of semantics in the H-Know platform, not for the development of its structure or functionalities.

The H-Know platform is built according to a perspective of collaboration enabled by a social network approach.

Users have a personal profile with their personal and professional information so as the **entities** and **collaborative places** he is connected with as well as their partners ("friends").

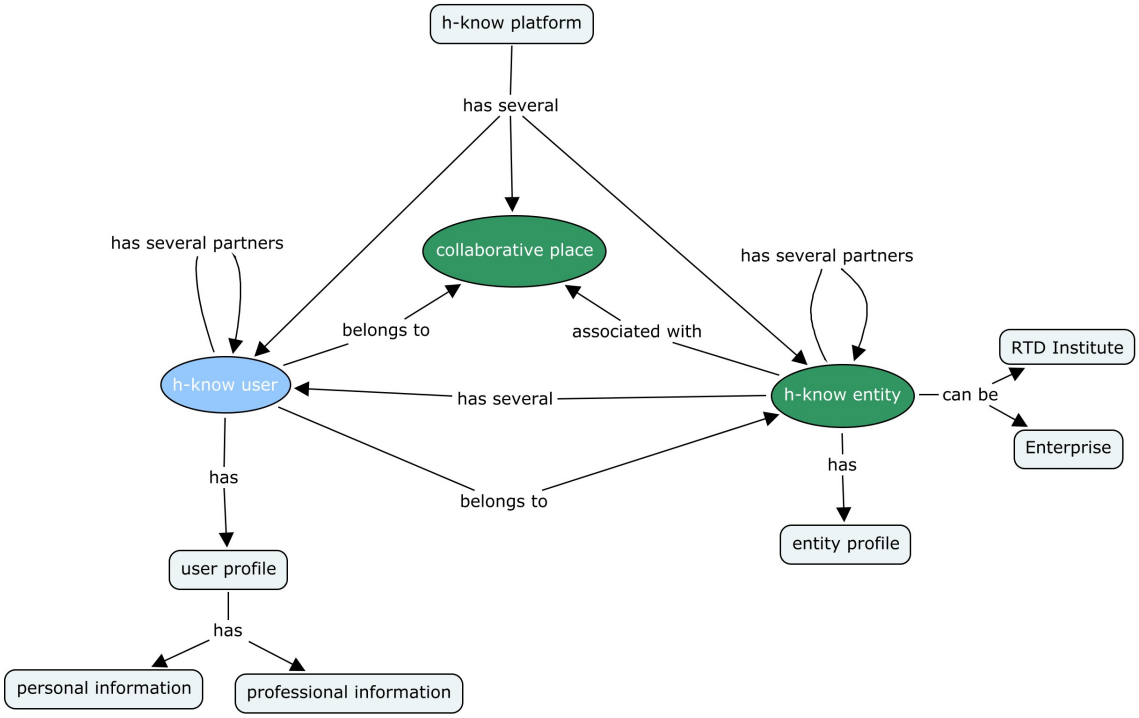


Figure 3.1: H-know high level architecture

In the diagram above we can see the 3 main blocks of the h-know platform and the relations between them. The green concepts represent the two kinds of grouping in the

[2] ColNet Group Webpage, http://dionisio.inescporto.pt/colnet

platform (entities and collaborative places). The blue concept represents the individual users of the platform.

As we can see in the diagram 3.2, a collaborative place (ColPlace) has a specific propose. It is created to solve a problem, discuss a project or a proposal, or collaborate in a research project or business opportunity and have a life cycle and are characterized by creation, operation and dissolution phases.

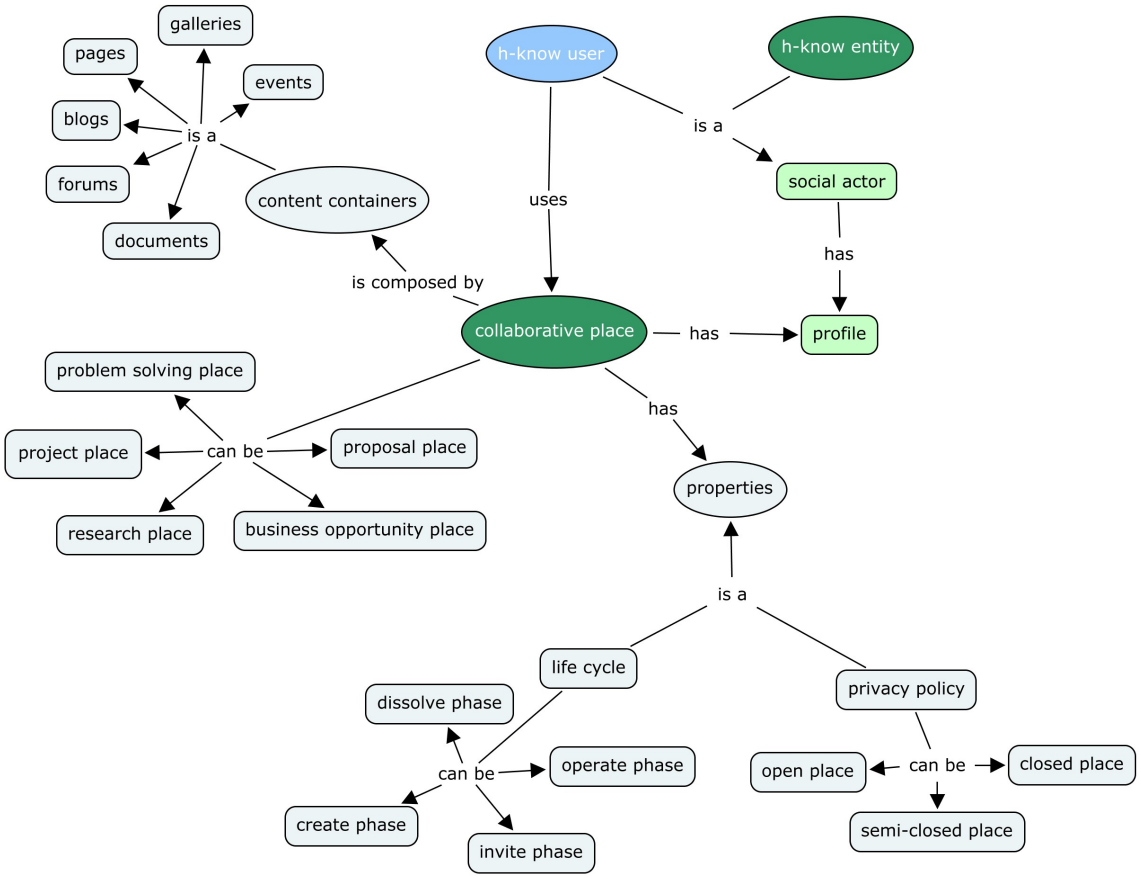


Figure 3.2: H-know Collaborative Place

The main objective of the collaborative place is to provide a mean where different users and entities can share information about the activities they are undertaking together. This is a space to join experts working or interested in a specific Construction Area domain or project.

Both Collaborative Place and Entities have the following set of tools for collaboration:

- **Event manager:** — to create events related with the work group such as meetings or deadlines. The creator of the event can send invitations for another users to attend the event;
- **Gallery manager:** — to share for example images of a construction in progress or an entity's meeting;

- **Pages:** — to publish knowledge about a specific topic like the specification of a project phase;
- **Blog:** — used for example to track the evolution of a project by publishing news of its progress or to announce news about an entity's activities;
- **Forums:** — to enable users to discuss about project details or an entity's issue;
- **File repository:** — a tool where users can publish for example project deliverables. Every group has its own file repository.

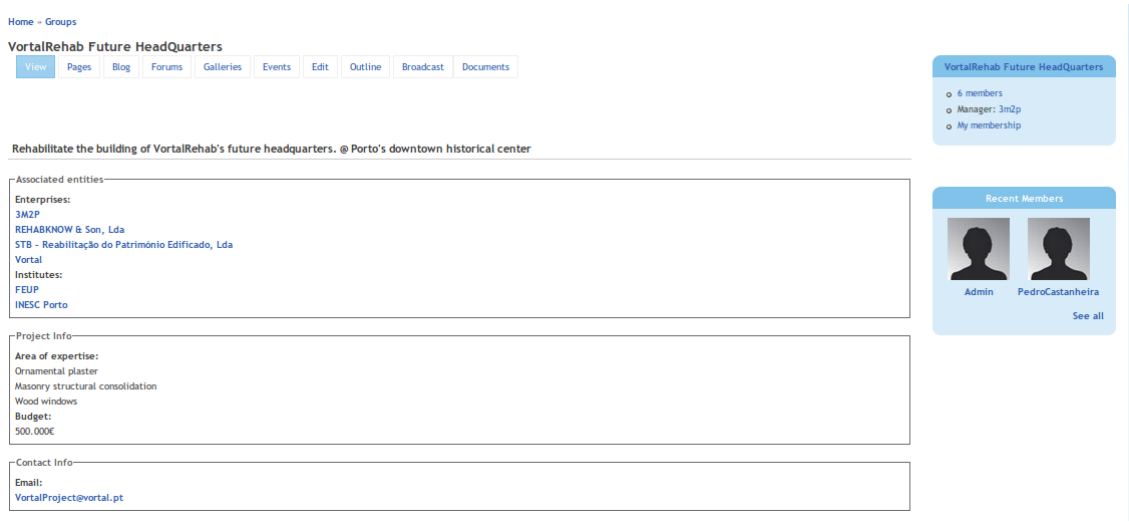


Figure 3.3: H-Know platform Collaborative Place printscreen [Con10]

In the figure 3.3 we can see a printscreen of the platform, showing the layout of the profile page of a Collaborative Place. In the top panel is the navigation bar with the tools presented above, in the center the information about the Collaborative Place characteristics and in the right panel, information about the users associated with it.

3.2.2 Functionalities

In this section there will be presented a brief resume of the main functionalities of the H-Know platform.

Having a good awareness of the platform functionalities is an important step for this study in order to understand what elements and user activities should be semantically expressed and what importance the semantic integration has to the platform usage.

The platform functionalities will be described from the point of view of the platform user, the part that interests for this study.

The following diagram shows the most relevant use cases from the point of view of an User registered in the platform.

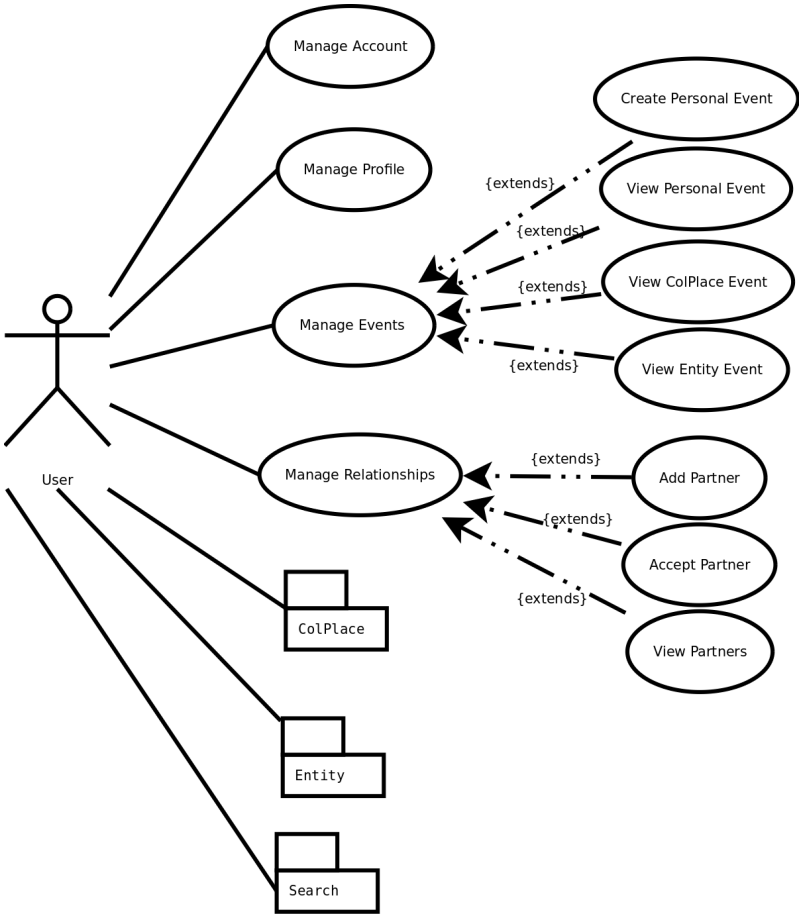


Figure 3.4: H-Know platform User Use Cases diagram

As we can see in the diagram a User in the platform can:

- Manage the account he registered;
- Manage the information he provides in his profile (personal and professional information);

- Manage the events he has scheduled (personal and group events);
- Manage the relationships he establishes with other users in the platform, accepting "partnership" requests or requesting new partnerships.

In addition to these functionalities, in the diagram are specified 3 other use cases packages: use cases for the activities in a Collaborative Place, use cases for the management of Entities and use cases for the searches in the platform.

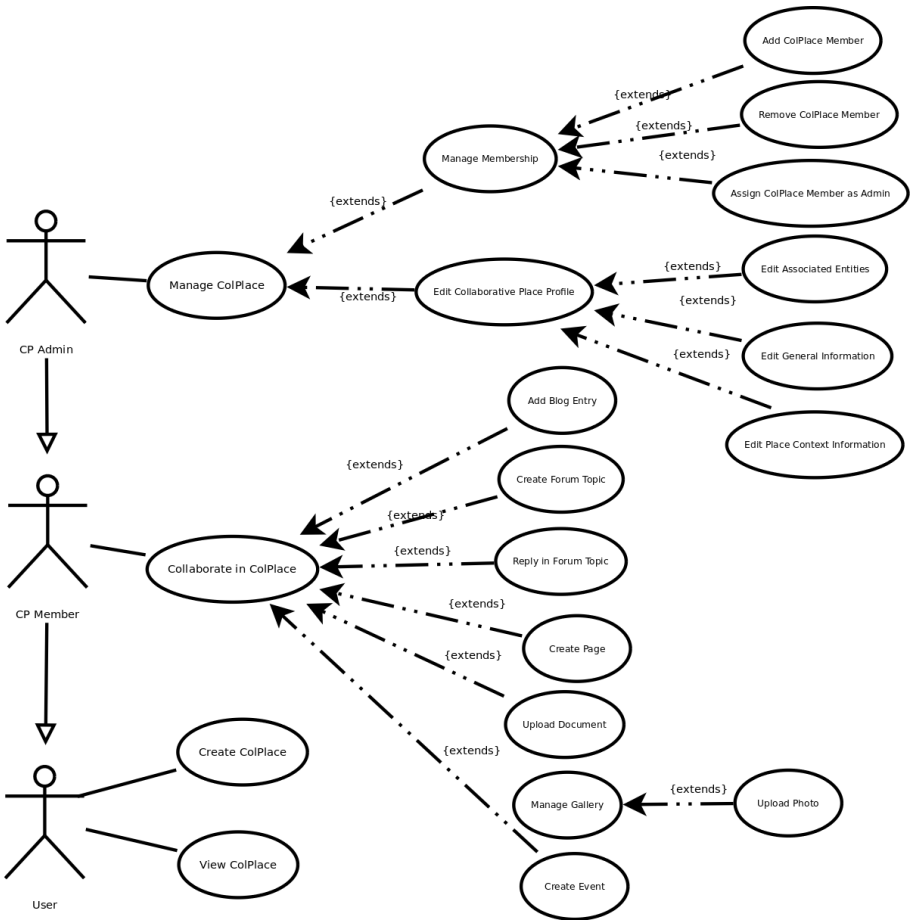


Figure 3.5: H-Know platform Collaborative Place Use Cases diagram

The diagram above shows the possible interactions of users in a collaborative place. Three actors were defined:

- **User:** the common H-Know platform registered user;
- **ColPlace Member:** a registered user that is member of a ColPlace (Collaborative Place);
- **ColPlace Admin:** a registered user that created the ColPlace or that was assigned by another ColPlace Admin as administrator of the ColPlace.

In the context of a collaborative place users can:

- Manage a ColPlace by editing its profile information such as its purpose, areas of expertise, contact information,etc;
- Collaborate to the knowledge produced in the ColPlace by editing and creating new content items.

Both an H-Know Entity and a Collaborative Place are groups just with different purposes: an entity for **intra-organizational** collaboration and a collaborative place for **inter-organizational** collaboration. So the use cases are pretty much the same. The biggest differences between the two are in the information provided in the group profile and the type of relationships established. A Collaborative place may have "associated entities" while an Entity may have "partner entities".

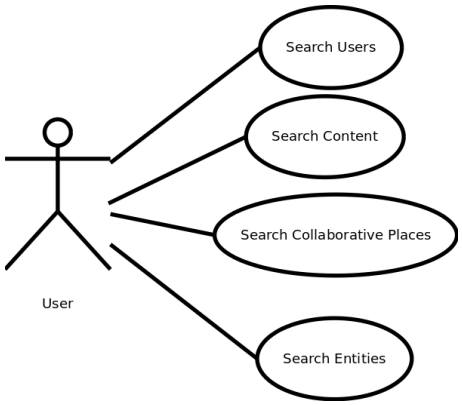


Figure 3.6: H-Know platform Search Use Cases diagram

In the diagram 3.6 we see the type of searches a user is able to do in the platform. He/she is able to search for users, content publicly published in a Collaborative Place or on an Entity, search for an Entity or search for a Collaborative place, specifying some characteristics for them in the searching criteria.

The integration of semantics in the platform intends to enhance the searching and information retrieval capabilities, basically by giving more search options and improving the relations between the searching elements.

These presented functionalities show the main resources the agents of the H-Know platform have. They were developed according to the requirements of the project.

3.2.3 The Drupal framework

The H-Know platform, is built over the Drupal CMS (Content Management System), on its version 6.[3] Drupal is a free software package that allows an individual, a community

[3]Drupal Information Webpage, http://drupal.org/about

of users or enterprises to publish, manage and organize a wide variety of content on a website. This is done in a easy way, without much programming but a lot of configuring and personalizations. Drupal is built for experienced and inexperienced internet developers, allowing them to both design a simple Personal Website or a complex Electronic Commerce platform.

In order to implement the classification of the content produced in the platform, there is a need to first understand how the Drupal CMS is structured, so we can have a perception of the way information flows and is organized.

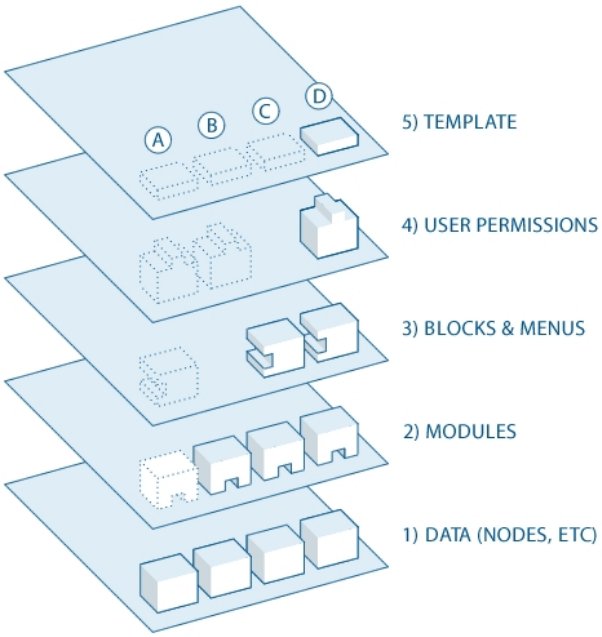


Figure 3.7: Drupal structure layers [ic10]

A Drupal **Template** defines the visual layout of the webpages of our Drupal installation, **user permissions** define the roles of the different kinds of users defined, **blocks** define the positions where content can be positioned and **menus** define the navigation of the website.

Most of the Drupal extensibility is given by **modules**. On its core, Drupal already brings a set of modules. Since the community of Drupal developers is quiet big, there are thousands of modules that can extend the main functionalities available in the core installation.

In the lower level we have the **nodes**. Every page in the Drupal framework is a node. The content produced in the platform is presented there and so this is the main focus of attention.

In the diagram 3.8, we can see the way nodes are related with the other "building blocks" of Drupal.

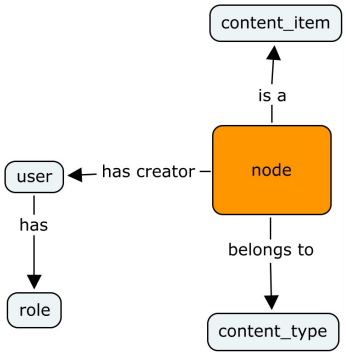


Figure 3.8: Drupal "building blocks"

As we can see in the diagram 3.8, every **node** belongs to a **content type**. content types give us the chance of creating custom-made pages for every kind of content we may have. The content types in Drupal, are managed in a non-core module called **CCK** (content Construction Kit).[4] There we can create new content types adding fields to it which can have different types (text-field, text-area, node reference, etc).

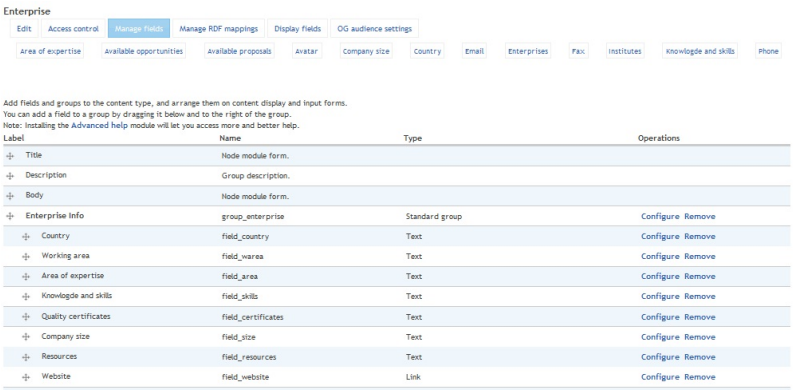


Figure 3.9: CCK module configuration for the Enterprise content-type [Con10]

In our case, the structure presented in the previous chapter can be converted to content types. So, we have all the platform elements mapped to a content type: collaborative places, user profiles, entity profiles, blog entries, forum entries, events and so on.

To create the concept of groups, the Entities and Collaborative Places, the module **"Organic Groups"** [5] was configured, which enabled users to create groups, managing their users and the content created inside the group. In our case we have both Entities and Collaborative Places configured as a group.

[4]CCK Module Webpage, http://drupal.org/project/cck
[5]Organic Groups Module Webpage, http://drupal.org/project/og

Concerning the semantic developments in Drupal, they have been the object of research for a group of Drupal developers for the last 3 years. One of the places where the advances are being published is the "Semantic Web"[6] group of the Drupal official website. As a result of the research made, many Drupal modules were developed for the integration of semantic web in Drupal.

In the article "Produce and Consume Linked Data with Drupal!" [CDC+09], winner of the "Best Semantic Web in Use Paper" in ISWC 2009 (International Semantic Web Conference), the authors present a series of modules developed to integrate semantic web in Drupal.

Later in this report (4.2), there will be presented and described some them.

[6]Drupal group for Semantic Web developments, http://groups.drupal.org/semantic-web

3.3 Platform Semantic Conceptualization

Research on semantically enhanced systems is growing although only few of the projects focus on CMSs. Good examples of that are [RGFR06] or [MLD06] which describe architectures to integrate semantics in CMSs from scratch. In [LAD+10] is presented a generic architecture for the integration of semantic annotation and usage in a CMS. We have a different approach since we use a already available CMS (Drupal).

In the h-know platform it is expected that users generate big amounts of content. Users in a content Repository express the semantics they have in mind while defining the content items and their properties, organizing them in a particular hierarchy [LAD+10]. However, this semantics are not formally expressed and so cannot be used to make meaningful relationships among the content items in an automated way.

Since this is a platform with social characteristics, users and entities information should be classified and connected with the content produced, so we can track in a automated way who produces each content. This semantic classification will be done mostly using existing standard ontologies such as **FOAF** to describe persons and entities and **SIOC** to describe content.

In the other hand, we must take into account that we are dealing with content of a specific domain. So, the content semantics must be connected with the domain semantics. Lets say, besides classifying that a page in the platform is a forum entry, we also want to say what is/are the domain topics described there. We want to semantically express something like: "This is a blog entry written by Luís about the rehabilitation process of the doors of an old church".

In the diagram 3.10, we can understand how the main semantic concepts are related to the H-know platform.

Going deeper in the H-Know platform, we can focus the semantic classification in the Collaborative Places, where the knowledge inter-organizations is created.

Taking this overview into account, we can separate the semantic classification of the platform into 2 areas:

- **Semantic classification of the platform structure:** — The platform is developed under the Drupal framework where every page is called a **node** of a **content type**. So, every content type (collaborative place, user profile, entity profile, etc) and its fields should be classified. This kind of classification is done by administrators, not users. When users create a new content, its structure is automatically semantically classified.

- **Semantic classification of the platform content:** — Besides classifying the structure of the platform contents, we are interested on giving users the ability of classifying the information included in a content. So, in the platform, users should be

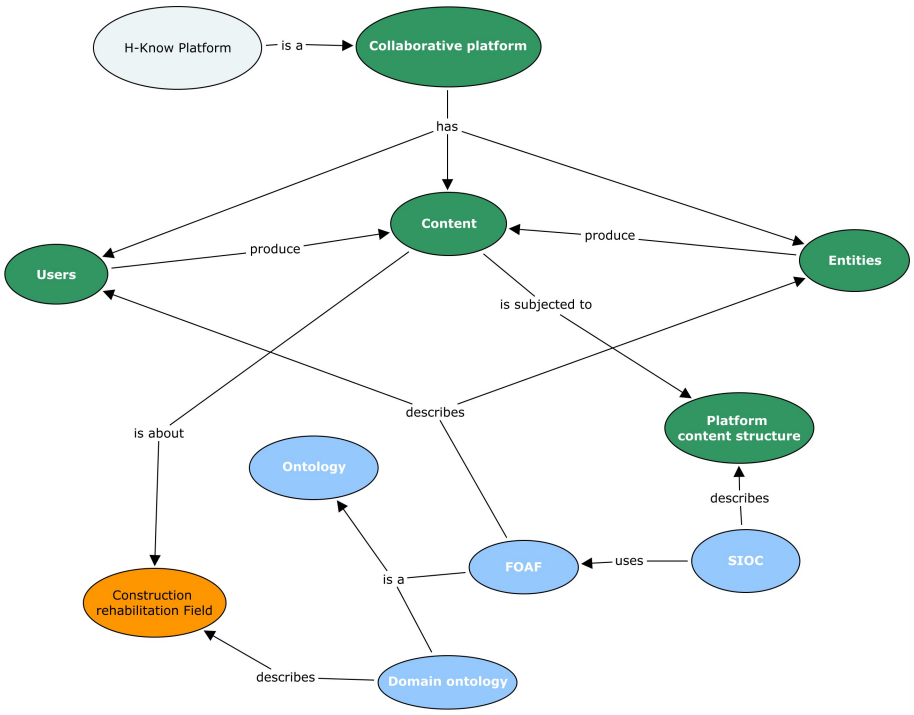


Figure 3.10: The inclusion of semantics in H-Know platform

able to associate content to specific concepts of the domain knowledge. For that, we will need a knowledge browser where the user can choose concepts from. Like we can see in the diagram 3.11, this domain classification is integrated with the **socio-collaborative classification**, so we can relate the produced content to the type of container where it was created and to the person who created it.

3.3.1 Conceptualization of the Domain Knowledge

The goal of this section is to give an overview about the domain knowledge related with the H-Know platform, describing its purpose, structure and main concepts.

The domain knowledge management of the H-Know platform is being conceptualized by members of the ColNet group of INESC Porto.

So we can understand the domain knowledge, related with the H-Know platform, in this chapter there are presented some concept maps, showing how the domain knowledge is structured and designed.

The domain knowledge management of the H-Know platform is being conceptualized by members of the ColNet group of INESC Porto.

Most of the concepts of the knowledge structure are based on the **CIK** (Construction Industry Knowledge) ontology from the project **Know Construct**(2005-2007)[7]. CIK was

[7]Know Construct Project Webpage, http://www.know-construct.com/

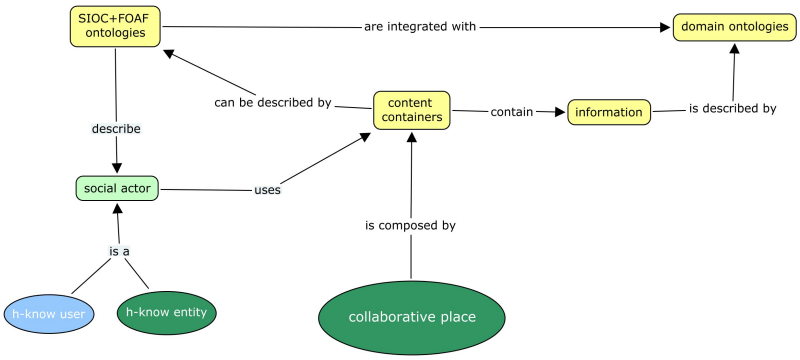


Figure 3.11: Collaborative places semantic conceptualization

built for general knowledge in the sector of construction industry. For H-Know, some modules were eliminated, others were revised and some other were created, specific for the area of old building restoration and cultural heritage. In the diagram 3.12 we see the first level of concepts of our domain.

Basically, we can say that in the **Construction Industry** we apply **Construction Resources** to **Construction Processes** that will lead to **Construction Results**. Construction Resources are constrained by **Technical Topics**. This basic structure is based on ISO 12006-2 [ISO01].

Now lets give some further information about each one of these main concepts shown in the diagram above. Definitions will be given according to the documentation of the ISO 12006-2.

A **Construction Result** 3.13, is a "construction object which is formed or changed in state as the result of one or more construction processes". Examples of construction results can be: Habitation building, a bridge or a ventilation system for example.

As a type of Construction Result we have:

- **Space:** a "material construction result contained within" or associated with a building or other construction entity (a room for example);
- **Construction Complex:Construction Complex:** "two or more adjacent construction entities collectively serving one or more user activity or function" (airport or motorway for example).
- **Construction Entity:** "independent material construction result" serving at least one user activity or function (bridge or tower for example);
- **Construction Entity Part:** "solid material part of a construction entity" (a door or a wall for example);
- **Work Result:** "construction result achieved in the production stage" (installed ventilation duct or applied asphalt surface).

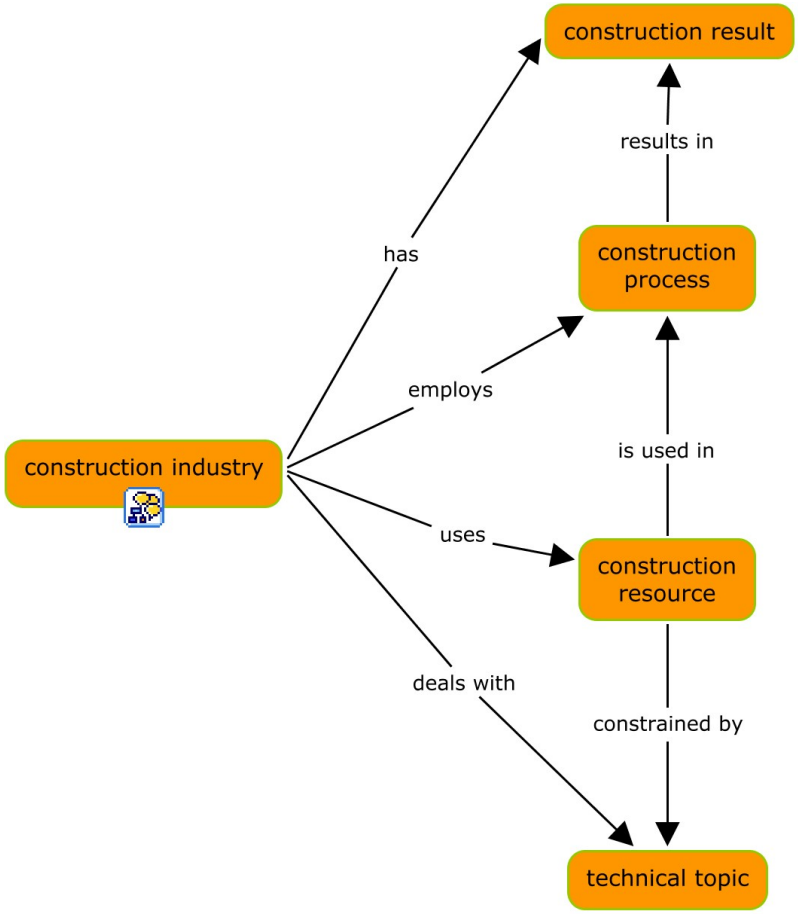


Figure 3.12: H-Know knowledge domain, first level concepts

A **Construction Process** 3.14 is a process "which transforms construction resources into construction results".

As a type of "Construction Process" we have:

- **Management Process:** "construction process with the purpose of planning, administrating or assessing".
- **Work Process:** "predominant construction process which results in a work result" (installing concrete or applying asphalt for example).

A Construction process occurs during:

- **Construction Entity Lifecycle:** "period of time in the lifecycle of a construction entity" (design or production stages for example).
- **Project Stage:** "period of time in the duration of a construction project identified by the overall character of the construction processes which occur within it" (feedback or programme preparation stages for example).

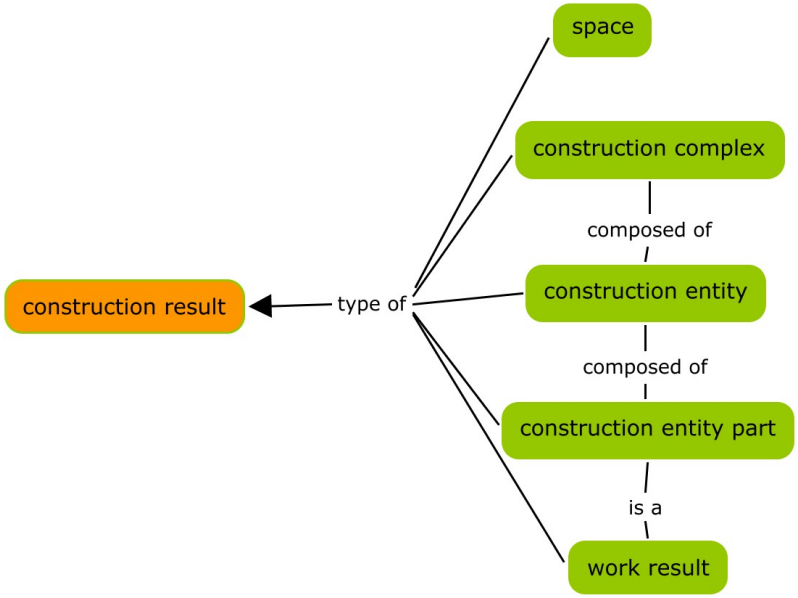


Figure 3.13: H-Know knowledge domain "Construction Result" concept

A **Construction Resource** 3.15 can be seen as a "construction object used in a construction process to achieve a construction result".

As a type of "Construction Resource" we have:

- **Training:** training courses about construction processes based on learning objects (Moodle course for example);
- **Construction Aid:** "material construction resource not intended for incorporation in a permanent manner" (a scaffolding or a machine for example).
- **Construction Agent:** "human participant in a construction process" (an architect or a site manager for example).
- **Construction Information:** "information needed to support one or more construction processes" (a textbook or a drawing for example).
- **Construction Product:** "material construction resource intended for incorporation in a permanent manner" (a door or a window for example).

The **Technical Topic** domain includes concepts related to issues like productivity, safety, constructibility, value engineering, partnering, etc. This domain is not included in the ISO 12006-2. These topics present a softer domain of construction operations and present a kind of "boundary conditions" for the majority of the work in construction. The concepts of this domain are related to a multitude of other concepts in other domains. For example, constructibility could be related to a design, to a construction method, to a construction product (like columns), to a material (like reinforced concrete), etc.

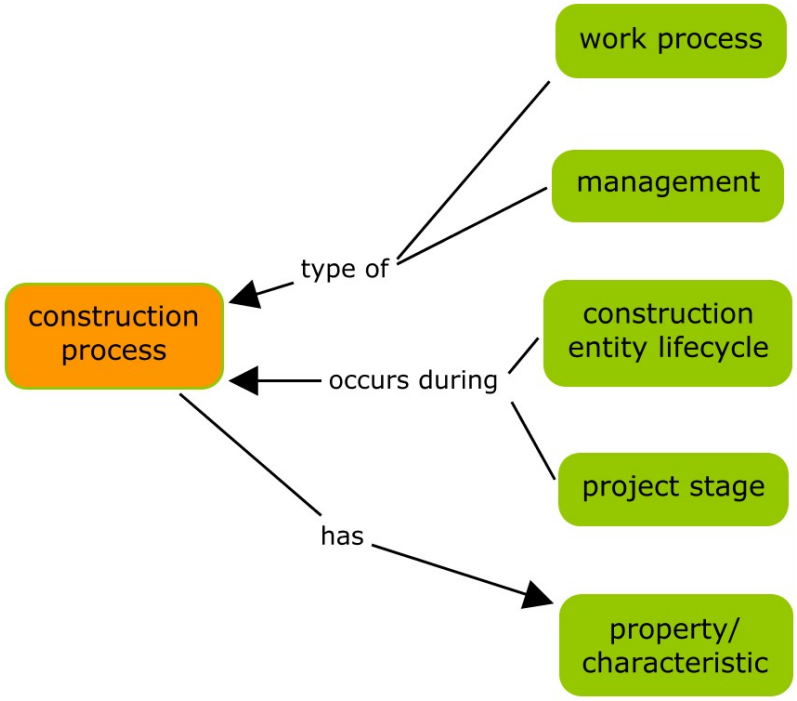


Figure 3.14: H-Know knowledge domain "Construction Process" concept

The knowledge structure presented above intends to be described in **multiple-languages** (for now, the languages of the countries joining the project), in a logic where more languages mean more knowledge.

3.3.2 Formalization of the Domain Knowledge

To describe the domain knowledge presented above in a formal way, ready to be used for machine processing purposes, we have two possible approaches:

- **To use OWL:** In this approach, we define each knowledge concept as a OWL class and we create all the properties needed to establish the relations between the terms;
- **To use SKOS:** In this approach, we define each knowledge concept as a **skos:Concept** and we use the properties defined in this ontology, to make the relations needed in our domain knowledge structure.

Both options have pros and cons. In the first approach, we have the flexibility of building our own **OWL** classes and using the core relational properties such as **rdfs:subClassOf** or **owl:equivalentClass** to build the knowledge structure. In the other hand, since we pretend to use SIOC to describe the types of content we have in the platform, the bottleneck of this approach is that there is no explicit connection between SIOC and a generic OWL

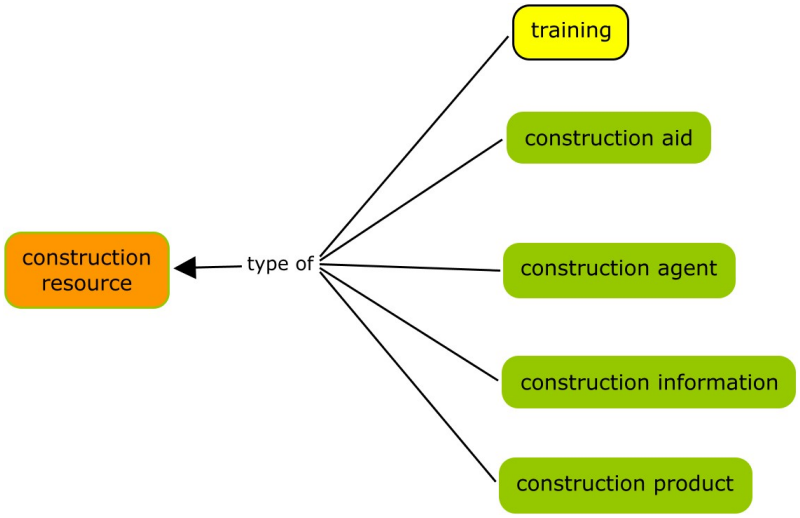


Figure 3.15: H-Know knowledge domain "Construction Resource" concept

class to describe the domain concepts of a content item. OWL is a formal semantics language, which means it can be defined apart from any interpretation of it and that its logical validity can be checked.

In OWL language we can define the domain and range of the classes which we can't do in SKOS. In practical terms, this is specially important because it allows better inference over the ontology concepts. From a OWL vocabulary, we can make instances of its classes.

SKOS is a vocabulary which is intended to represent Knowledge Organisation Systems (KOS) like **thesauri**, **term lists** and **controlled vocabularies**[JBS08].

Using the SKOS data model to translate the domain knowledge, we can define each concept as an individual of the **skos:Concept** class. In this data model, we don't have "pure" classes hierarchy like in a OWL ontology, but we have **skos:narrower** and **skos:broader** properties, to construct the knowledge structure. For non-hierarchic linking we can use the property **skos:related**. OWL language in its basis already defines the transitive closure of the relations between the classes while in the SKOS data model we have to explicitly define it by using the relational properties **skos:narrowerTransitive** and **skos:broaderTransitive**, to say for instance that if <B> is a sub-class of <A> and <C> a sub-class of <B>, <C> is also a sub-class of A. From a SKOS vocabulary, we can't make instances of the concepts because they are already an instance of the class "skos:Concept" and so cannot be further instantiated.

A big advantage of SKOS data model is that it can be connected with SIOC ontology model, by the property **sioc:topic**. Since this is a platform with social and collaborative characteristics this is an important aspect because it simplifies the process of integrating socio-collaborative information with domain knowledge.

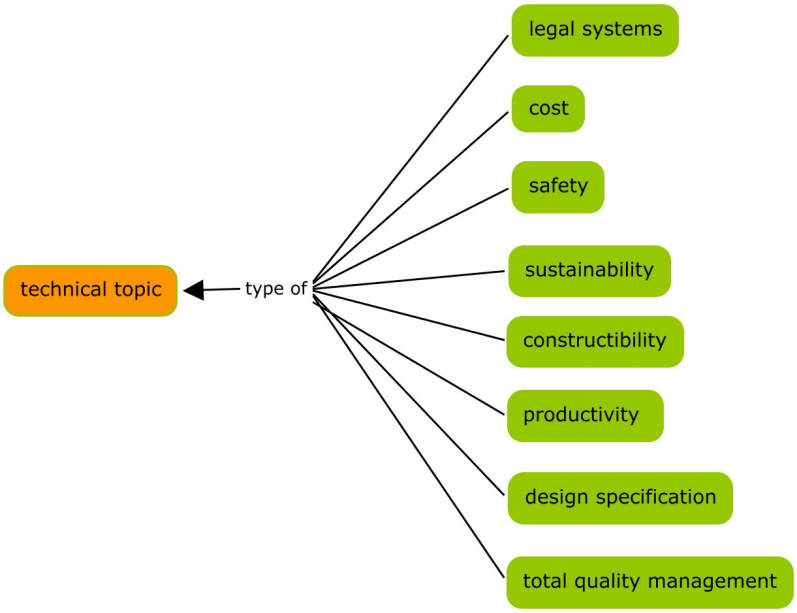


Figure 3.16: H-Know knowledge domain "Technical Topic" concept

Another interesting aspect is the possibility of defining **lexical labels** to the concepts: **skos:prefLabel**, **skos:altLabel** and **skos:hiddenLabel**. The properties skos:prefLabel and skos:altLabel allow us to define a clear meaning for each one of the concepts while the property **skos:hiddenLabel** can be used for example to tag mis-spelled words of the concept, which users usually search for. We also have the possibility of using **language tags** (example: skos:prefLabel "Construction Process"@en) which in our case is a very interesting aspect since the H-Know project should be available in multiple languages. In SKOS we also have definition (**skos:definition**) and notes properties to further describe the concepts. This kind of properties help in the interoperability issues because they provides a way to better understand each concept.

In the table 3.1, a comparison between OWL and SKOS is presented, a summary from the explanation given above.

Analysing these two possibilities, the SKOS approach is the chosen one. It doesn't provide the same formalism as a OWL ontology can have, but provides enough information to describe H-Know domain with the advantage of being easier to manage than a pure OWL ontology. The lexical properties provided by this vocabulary allow us to give the meaning we want to each concept and also specify them in different languages, an important requirement of the H-Know project.

The connection of the domain concepts with the platform contents is solved because SIOC provides a bridge with SKOS vocabularies. This way, we have domain knowledge connected with the content, enabling the socio-collaborative integration we seek with the semantic classification of the platform.

Table 3.1: Comparison between OWL and SKOS

Characteristic	**OWL**	**SKOS**
Building Foundations	RDF	RDF, built over OWL
Standard of W3C	Yes (since February 2004)	Yes (since August 2009)
Application Purposes	explicit modelling/description of a domain	thesauri, term lists and controlled vocabularies
Hierarchy Formality	OWL subclass hierarchies provide a formal interpretation (in terms of sets of instances)	no formal semantics given for the conceptual hierarchies (broader/narrower)
Hierarchy Transitivity	implemented in the basis of OWL with rdfs:subClassOf property	implemented using skos:broaderTransitive and skos:narrowerTransitive
Class Instantiation	owl classes can be instantiated	skos:Concept is a class, particular concepts are instances of that class
Integration with SIOC	no explicit linking	linking using the property sioc:topic connected with skos:Concept
Annotations	owl:AnnotationProperty	skos:prefLabel, skos:altLabel and skos:hiddenLabel,instances of owl:AnnotationProperty

In the diagram 3.17 we can see how the H-Know knowledge is mapped to a SKOS vocabulary.

Every concept in the H-Know knowledge is expressed as an instance of **skos:Concept** class. Using the relational properties of SKOS (broader, broaderTransitive, related, etc), we build the hierarchy of the knowledge.

In the cases we need to define different properties than the ones offered in the core of SKOS, to express extra information, we **specialize the SKOS model**. Following the recommendations of SKOS primer documentation[IS], the SKOS approach allows an application designer to create new properties, declaring them as sub-properties of SKOS properties, using the **rdfs:subPropertyOf** property, from the RDF Schema vocabulary. For example:

hknow:employs rdfs:subPropertyOf skos:narrowerTransitive
hknow:occurs_during rdfs:subPropertyOf skos:broaderTransitive
hknow:is_used_in rdfs:subPropertyOf skos:related

The **hknow:employs** statement between two concepts, for instance, can be formally interpreted by a Semantic Web reasoning engine. This interpretation will take into account

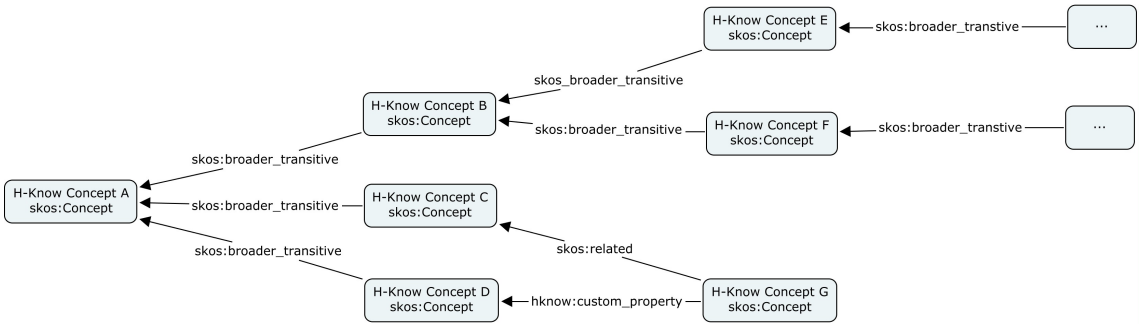


Figure 3.17: H-Know knowledge domain described using SKOS

the inference of a **skos:narrowerTransitive** property between these concepts, a piece of information which may then be explored by SKOS tools.

To fully express the meaning of every H-Know concept, the lexical properties **skos:prefLabel**, **skos:altLabel** and **skos:definition** are assigned, using in addition the language tag of a lexical label, to restrict its scope to a particular language. For example:

hknow:Space rdf:type skos:Concept;
skos:prefLabel "Space"@en;
skos:prefLabel "Espaço"@pt;
skos:altLabel "Space"@en;
skos:altLabel "Espaço"@pt;
skos:definition "A material construction result contained within or associated with a building or other construction entity"@en;
skos:definition "Um resultado de contrução material, contido ou associado a um edifício ou outra entidade de construção"@pt.

After having the domain knowledge expressed as a SKOS vocabulary, it is ready to be incorporated in the context of the H-Know platform.

3.3.3 Competency Questions

Following the description of the integration of ontologies in the structure of the platform and after giving an overview about the domain knowledge managed there, this chapter aims to give some information about the competency questions that arise from the semantic classification of the platform.

Competency questions are a form of specifying the requirements of an ontology based information system in what concerns to the information to be obtained by querying the system.

Since this H-Know platform has social networking characteristics, we want to be able to answer questions like:

- Who is an expert in the area of expertise xpto?
- Which projects exist about the legal systems of a construction?
- Who is publishing more about the rehabilitation process of a bridge?
- With whom have I discussed lately about the application of the construction process xpto?

In a sum, we want to get domain information related with the users and entities connected to it. This is why we have social collaboration semantic classification combined with domain ontologies. On this report it will be further given some competency questions when describing the semantic search interfaces for the platform (3.7).

3.4 Mapping platform structure to ontologies

This chapter explains the model developed to map the platform structure to ontologies. This is a very important chapter since it explains how the socio-collaborative activities are semantically expressed.

The first important thing to do to model the platform structure to ontologies is to focus on the very essential of every Drupal page: a **node**. Like stated before, everything from a user profile to a content item is a node.

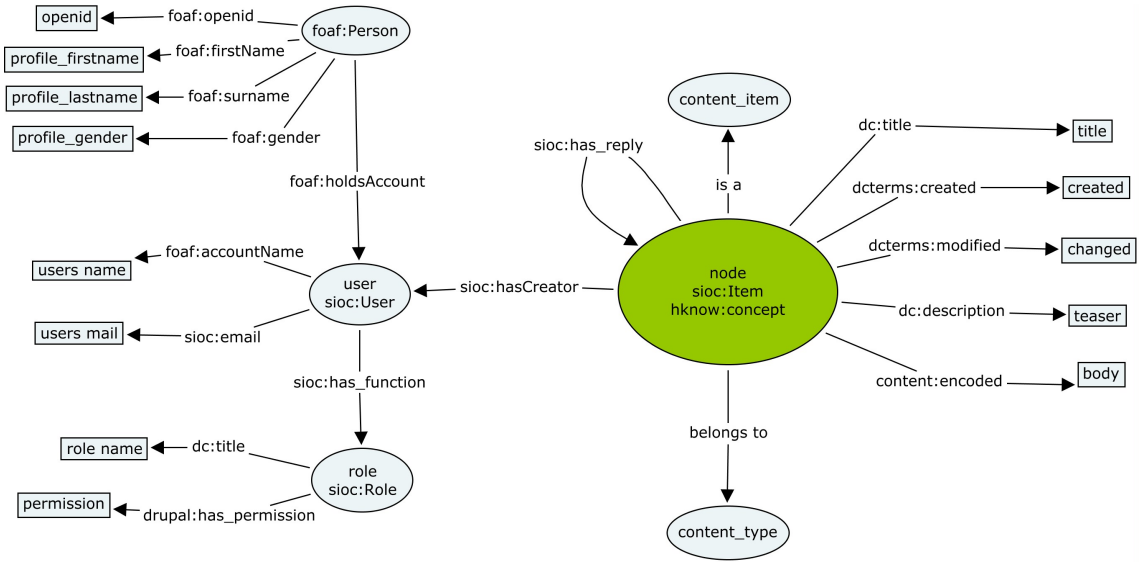


Figure 3.18: Drupal node structure mapped into ontologies (based on [Cor])

In this presented schema 3.18, every Drupal node is considered a **sioc:Item**. A sioc:Item is a class that describes something that can be in a container [BBa]. It has subclasses that can specify different types of Items such as **sioc:Post** to describe a Forum post. With

the SIOC Types Module, we can create new types of **sioc:Item** to describe other types of content.

Every Drupal node can have comments associated with it (which are also a node). To combine a node with its comments we can use the property **sioc:has_reply**.

Platform users are represented with the class **sioc:UserAccount**, "an online account of a member of an online community". A user, apart from the platform, is classified as a foaf:Person, with his own set of characteristics and interests, independent from the platform. A User is connected to the Person he represents by the property **foaf:holdsAccount**.

To connect the node with the user that created it, we use the property **sioc:has_creator**.

Associated with every node we will have concepts of the H-Know domain ontologies, represented in this schema by the **hknow:concept** tag. We will talk about the integration of domain concepts in the platform in the next section 3.4.1.

Since we have different types of content pages in our platform, we should classify them according to their specific genre. Picking up all the elements that defined the platform,the diagram 3.19 represents the mapping of the different elements to a ontology class and the relations held between them.

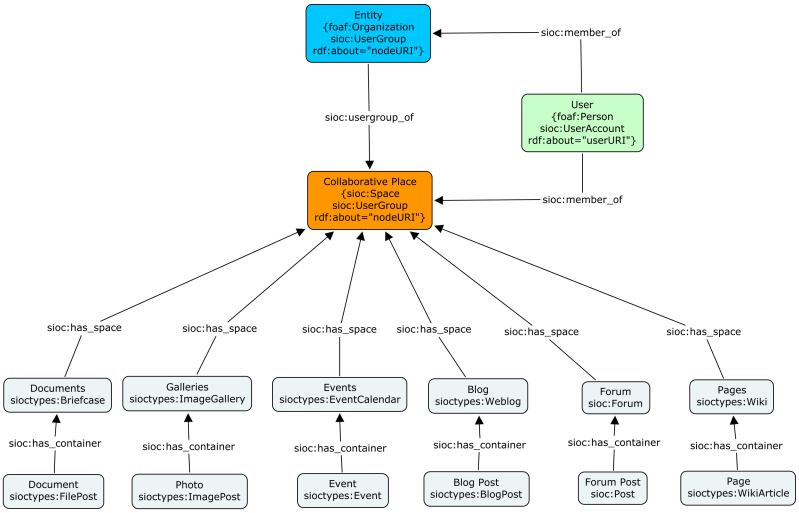


Figure 3.19: H-Know platform integration of semantics

From the diagram 3.19, we can discuss some of the decisions took to semantically express the platform.

In this diagram we specify for the main elements, their **rdf:about property**. This property is used to define the subject in a triple statement, the resource being described.

First of all, it was decided to define a Collaborative Place with two different SIOC classes: **sioc:Space** and **sioc:UserGroup**, using as resource identifier the URI of the node that is the index page of that Collaborative Place. This approach was followed because of the multiple behaviour of a Collaborative Place. It is both a place to aggregate

information and platform agents (users and entities). So, since a **sioc:Space** "is defined as being a place where data resides" [BBa], we use it as the location for a set of **Containers** of content items. Any data that resides in a **sioc:Space**, can be linked to it using the property **sioc:has_space**.

In this diagram, we can also see the different content containers that a Collaborative Place has (Documents, Galleries, etc) with their content items associated. The content containers of a Collaborative Place are classified using **SIOC** and **SIOC Types**, an extension to the core SIOC ontology[8]. Each content Item is linked to its container using the property **sioc:has_container**.

In the other hand, we have the classification of the platform entities: Enterprises and RTD Institutes. An **entity**, in its individual identity, is defined as a **foaf:Organization**, "a kind of Agent corresponding to social institutions such as companies, societies, etc". In the context of the H-Know platform, an Entity can also be seen as a **sioc:Usergroup**,"a set of UserAccounts whose owners have a common purpose or interest" [BBa], since it has users associated.

To classify the group behaviour of a Collaborative Place the **sioc:UserGroup** structure is used. To link the H-Know users to the Collaborative Places they are part of, we use the **sioc:member_of** property. The association entities have with a ColPlace is expressed by the property **sioc:usergroup_of**, "a **Space** that the **Usergroup** has access to" [BBa].

H-Know Users have a set of connected partners. These connections form the social network vision of the platform. To semantically express these connections, we use the **foaf:knows** property to link different **foaf:Person** elements. According to the FOAF reference, **foaf:knows** property represents "a person known by this person (indicating some level of reciprocated interaction between the parties".

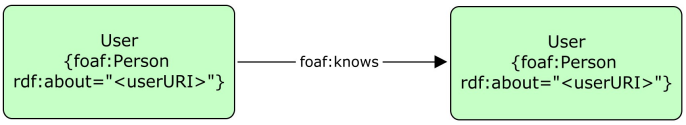


Figure 3.20: Semantically expressing the connection between H-Know users

The diagram 3.20 represents the connection between two people, linked by the **foaf:knows** property. This property is not reciprocated, this means, I can know someone but that someone may not "know" me. In the case of the H-know platform, we force this reciprocity because a connection has to be accepted by the requested user and so, if Person A knows Person B, Person B certifies that he also knows Person A. In this diagram an extra information was added: rdf:about. This means that a foaf:Person in our platform is identified by the **userURI** which is the page of the user profile, a unique identification of him.

[8]SIOC Types Ontology Module Namespace, http://rdfs.org/sioc/types$#$

All these described elements (Users, Collaborative Places and Entities), may have domain concepts associated with, enriching the content classification. This will be the focus of the next section.

3.4.1 Integrating platform and domain ontologies

Following the discussion presented on chapter 4.3.2, it was decided to adopt the approach of representing the domain knowledge as a vocabulary, using the **SKOS** data model.

To connect platform content items with the domain knowledge classification done by the users, it's used as a bridge the property **sioc:topic**. This property can be applied to most of the classes defined in the SIOC ontology. So, we can for example assign a set of concepts to a container and then propagate those topics to its items. Since we have in our case, a very specific, specialized knowledge domain, we map all the knowledge concepts with SKOS using the **skos:Concept** class.

In the diagram 3.21, we can understand the linking described above, for content items.

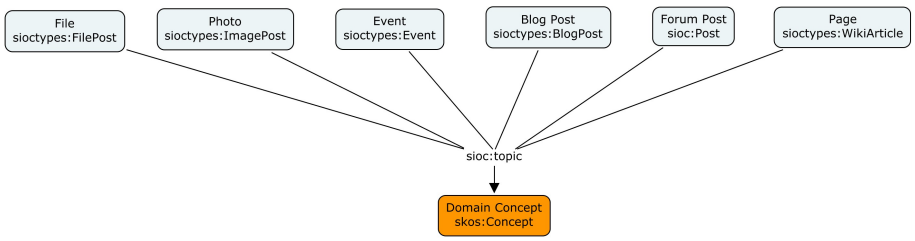


Figure 3.21: H-Know content items linking with Domain Concepts

Each content item may have a set of Domain Concepts related to it. So, we will have triples in the form:

- **Subject:** The URI of a content item, classified with the SIOC vocabulary;
- **Predicate:** The sioc:topic property, making the bridge between a content item and a concept;
- **Object:** A skos:Concept class representing a concept in the domain knowledge.

This way we can specify the concepts which relate to each content item of the platform.

Another usage of the domain SKOS vocabulary, is to map platform content template field's. The triples in this case have the following form:

- **Subject:** The URI of a content item;
- **Predicate:** An instance of a rdf:Property representing a domain knowledge property;

- **Object:** A skos:Concept class representing a concept in the domain knowledge.

The predicate may be any content type field, semantically defined as a rdf:Property, for example the field "area of expertise" to represent the area of expertise of a user, defined in a vocabulary of the platform structure.

Applying this classification to content items, we have the complete classification graph, allowing us to make socio-collaborative queries with domain concepts involved.

3.5 Semantic classification of the platform content types

This chapter describes how the edition of the platform node's template (content type) semantic classification should look like, presenting a simple non-functional prototype.

After having defined the mappings of every content type and its fields in the platform to ontologies, we need a way to easily allow platform administrators to apply and edit those mappings.

Since every page in H-Know platform belongs to a content type, we can automatically classify the structure of all the pages created by users if in the back-end we assign the mappings between the content types and its fields to the ontologies describing them.

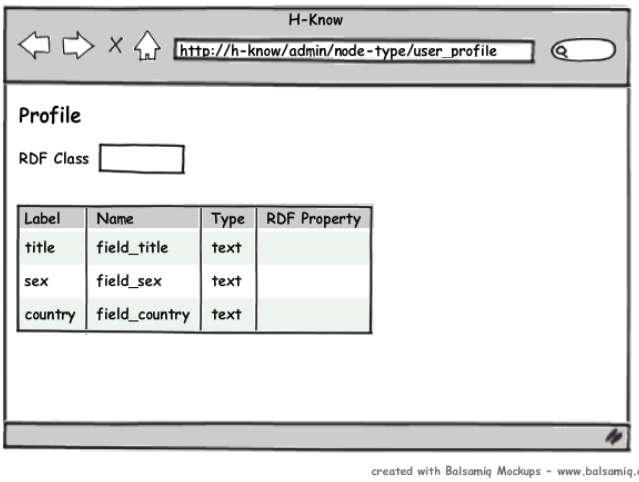


Figure 3.22: Mockup of the Semantic classification interface of a platform content type

In the prototype 3.22 we can see an example of an user interface to map a content type ("User Profile" in this example) to ontologies. The idea is that, after this process, all the content items structure is automatically semantically classified when they are created or edited.

3.6 Semantic classification of platform Content information

In this chapter there will be explained how the platform content information can be semantically classified, presenting some non-functional prototypes of user interfaces for those classifications.

Apart from the automatic classification of the content items structure, defined by administrators, users need to classify the pages according to the existing domain knowledge. To do that, they should be able to choose terms from the domain ontology.

So, a visual representation of the ontology is needed to provide users with an interface where they can choose concepts to classify the content they produce.

When creating or editing content, users should have an interface that allows them to associate domain concepts with the content they are producing.

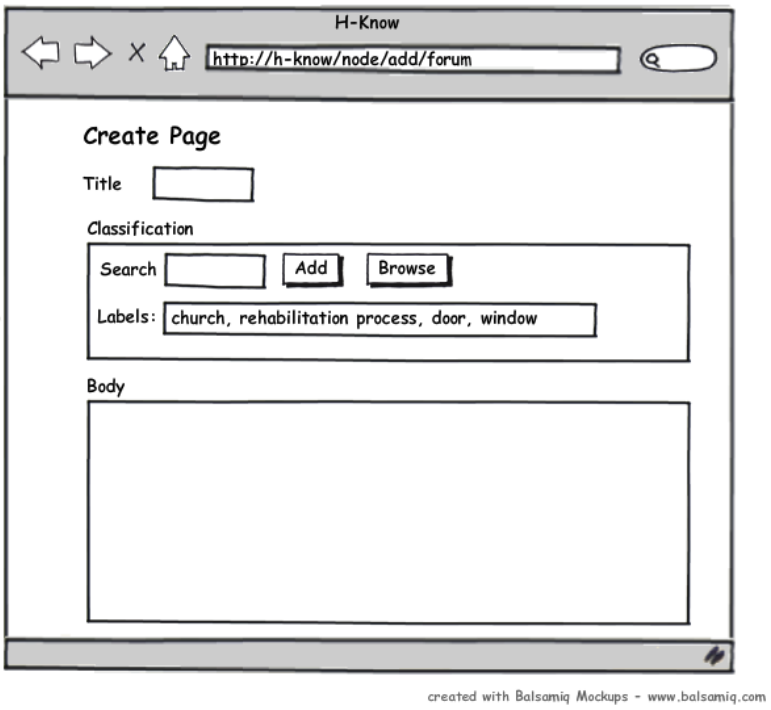


Figure 3.23: Prototype of the Semantic classification interface

In the prototype 3.23, we can see the approach to classify a content item. Users can search for a concept of the domain ontology using a input text with auto-complete functionality and associate that concept with the content. Once they add that concept, it will appear in a list of concepts associated with that content. Another possibility is searching for a concept in a ontology hierarchical browser.

Using the Ontology Browser, to make it easier to find the concept the user needs to classify the content, it should be provided an initial dialog where the user can introduce a broad concept of the ontology that will be the initial branch of the ontology browser. For

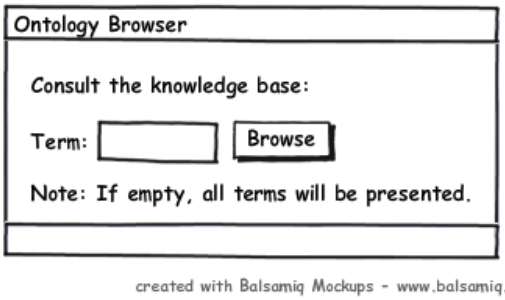


Figure 3.24: Mockup of the semantic browser filter interface

instance, if the user wants to classify a content page with a type of "Residental Building", he searches for the term "Residental Building" and the browser shows a tree showing the concepts which are narrower of "Residental Building".

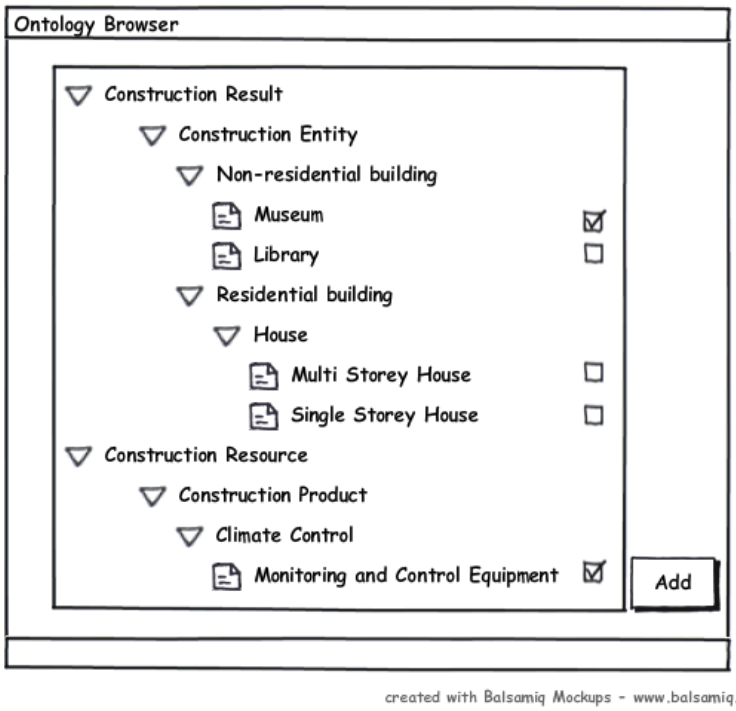


Figure 3.25: Mockup of the semantic browser interface

Once the user is in the "Ontology Browser", he checks the instances related to the content item he is classifying and adds them to the list of domain concepts related with it.

3.7 Semantic search of the content

After having the information produced in the H-Know platform semantically classified, users should be able to make searches taking advantage of that classification.

The searching interfaces should be as "user-friendly" and simple as they can be, so users can easily find the content they need.

The H-Know platform has a social perspective (users and entity descriptions), the content produced by them and the collaboration places where that content can be created. So, taking this overview into account, we may define four different searching profiles:

- **Users search:** when a user needs to find users with specific characteristics such as area of expertise, country of residence or availability to work in the present.
- **Entities search:** when users want to find an entity with specific characteristics, to start a collaboration or to get some useful information about topics which that entity is an expert of.
- **Collaboration Places search:** when a user wants to find a "starting point" to get further information about a specific area of knowledge. Another example can be an user, that wants to contribute to a Collaboration Place about a topic he is an expert.
- **Domain Content search:** the most crucial kind of searches users may need to do. When a user wants to find information about knowledge areas he is interested in. Those searches should allow users to filter results both for socio-collaborative information and concepts of the domain knowledge.

In the diagram 3.26, it's summarized the searching profiles of the platform.

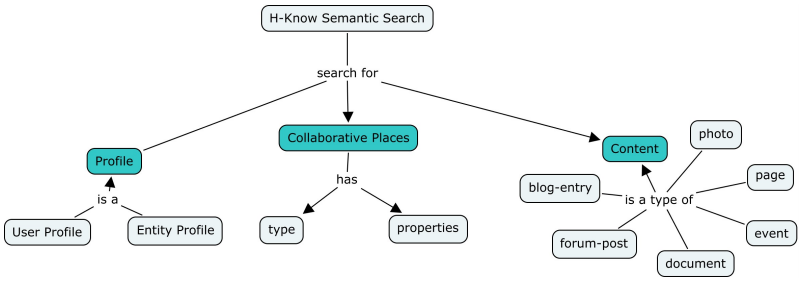


Figure 3.26: H-Know platform Semantic Search blocks

3.7.1 Facet-browsing

One possible approach is to build **facet-browsing** search interfaces, where users can, by the means of filters, continuously redefine their searching criteria. When users are trying

to find information, most of the times they don't want only one exact result for their search, but instead of it, a set of results they can evaluate the interest.

A faceted classification system allows the assignment of multiple classifications to an object, enabling the classifications to be ordered in multiple ways, rather than in a single, pre-determined, taxonomic order [**?**]. In a facet-browsing interface, each **facet** typically corresponds to the possible values of a property or set of properties common to a set of digital objects, web references in the H-Know case. So, starting from a non-filtered set of web references, users can use pre-defined filters to decrease the set of results, getting just what they were trying to find.

The idea is, using all the metadata generated in the platorm, to get results relating domain knowledge concepts with the socio-collaborative activities.

Taking into account the four searching profiles presented above, four different facet-browsing interfaces, can be conceptualized: one for **Contents search**, one for users **profile search**, one for **entities profile search** and one for **collaborative places search**.

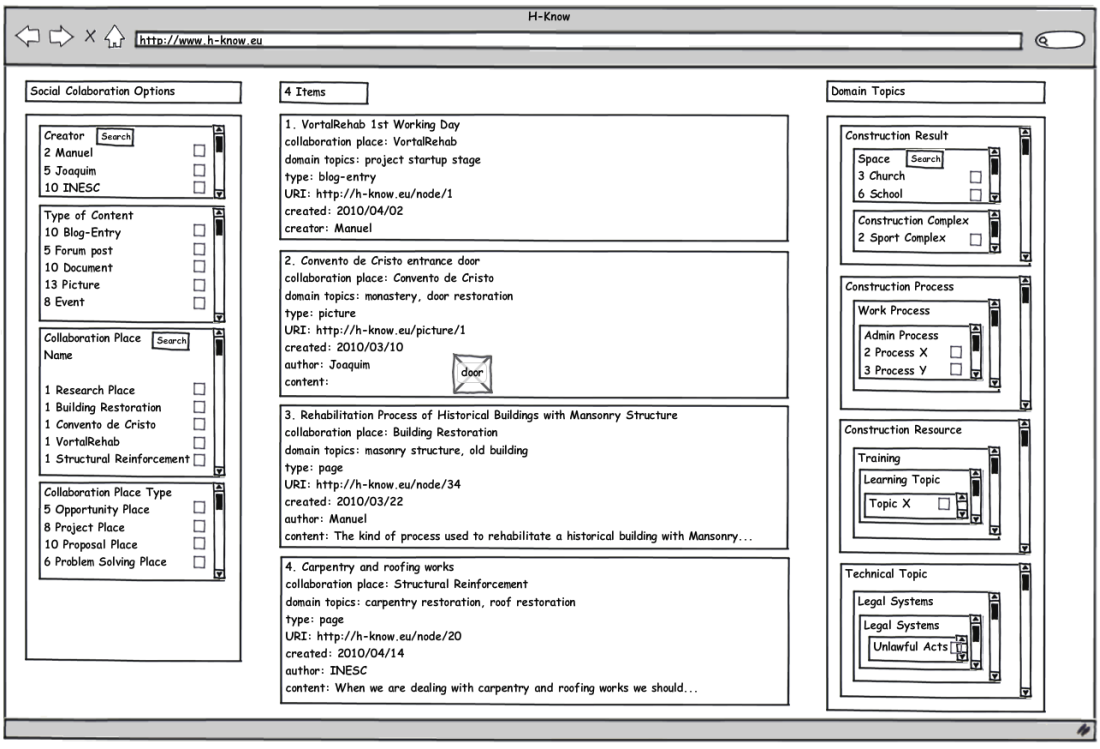


Figure 3.27: H-Know content facet-browsing mockup

In the figure 3.27 we can see a prototype of what could be a content search interface, using the facet-browsing technique. Every content item has socio-collaborative characteristics and domain concepts related to it. So, we define 2 sets of filters: one for the **socio-collaborative characteristics** of the content item, such as the author of a content,

the type of content or the type of collaboration place where it is produced and another set of filters related to the topics defined in the **domain ontology**.

This way we can get an answer for a question like: "Which blog-entries Manuel created about churches?", filtering creator for "Manuel", type for "blog-entry" and Construction Result Space for "Church". Every field of the filters have a number associated with. That represents the number of resources classified with that instance. For example, there are 10 content resources classified with the "Proposal Place", Collaboration Place type.

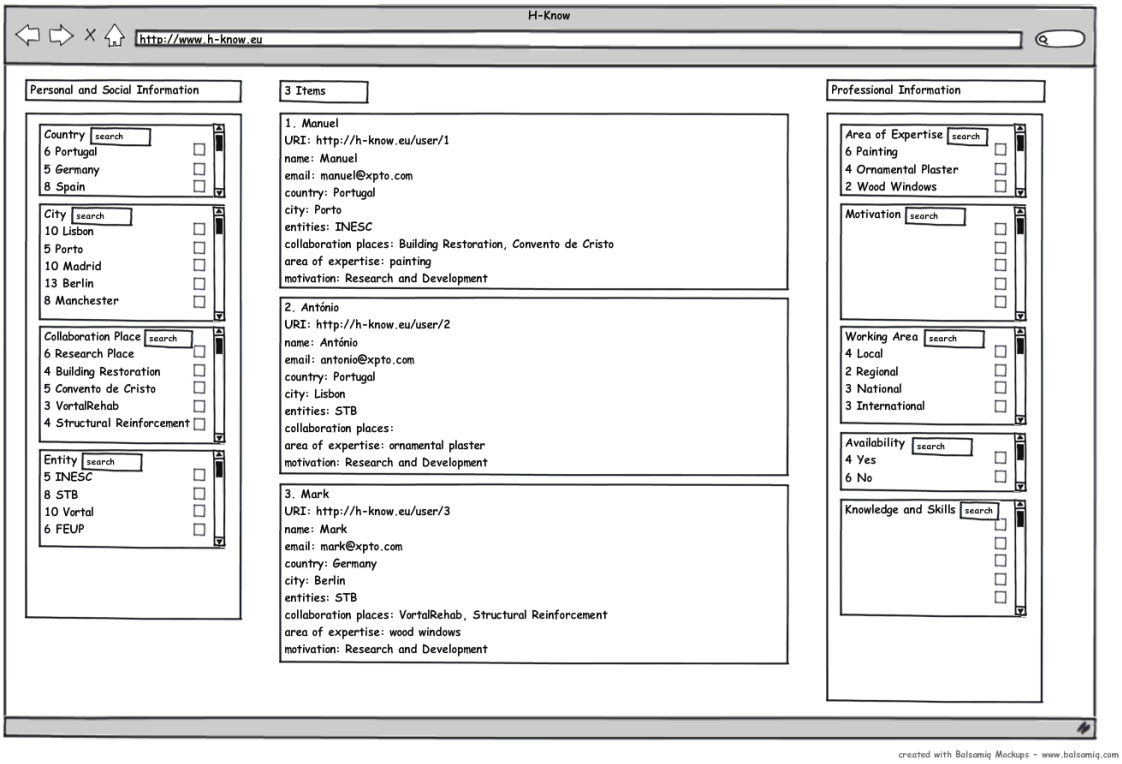


Figure 3.28: H-Know users facet-browsing mockup

The figure 3.28 exemplifies how a users searching interface can be. There are two types of information we can have about a user: his personal and social information in the platform and his professional domain information. In a searching interface like this, users can answer questions like: "Who knows about painting in Portugal and is available to start a new project?", selecting "Portugal" as the country, "Painting" as the area of expertise and availability as "yes".

In the search for a collaboration place (3.29) we are interested on finding places either about a specific area of expertise or related to some entities or users.

In a search for an entity (3.30), users might be interested on answering questions like: "Which entities in Portugal are expert in Landscape maintenance?", combining social and domain information.

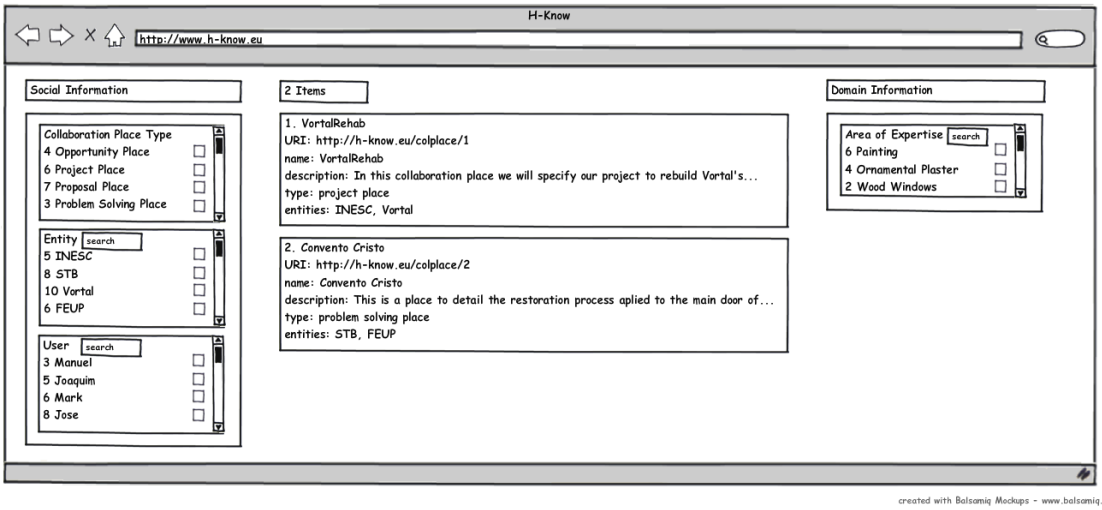


Figure 3.29: H-Know collaboration places facet-browsing mockup

These presented facet-browsing interfaces are a possible usage of the semantic metadata generated in the platform.

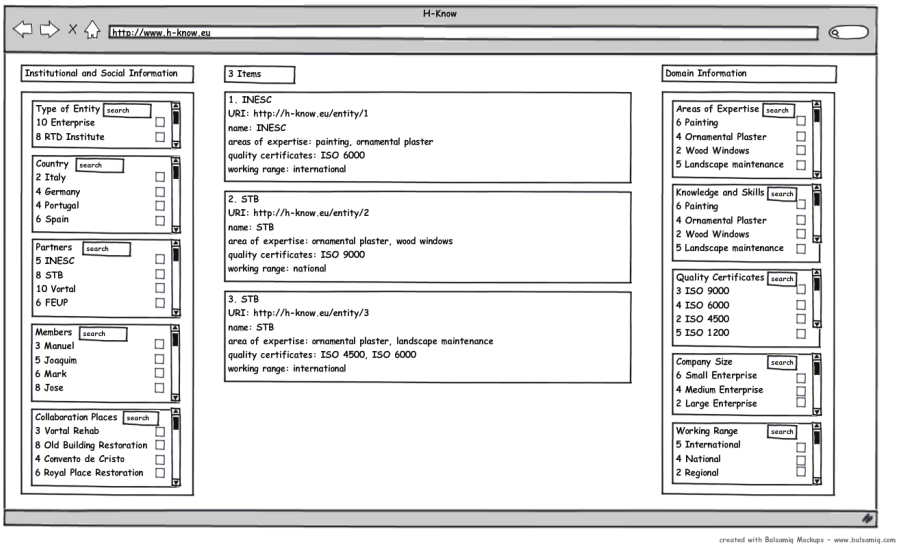


Figure 3.30: H-Know entities facet-browsing mockup

Chapter 4

Implementing the integration of semantics in the H-Know platform

This chapter will present all the implementations of the conceptualizations done in the previous chapter, picking up the designed models and prototypes that were thought. The technologies and decisions taken are explained.

4.1 Platform Semantic Use Cases

In this section there will be presented the use cases related with the Semantic integration in the platform, that arise from the conceptualization presented in the previous chapter.

Thinking about the semantic activities that involve the platform, we can identify three actors that play a role in the semantic module of the platform:

- **The platform Administrator:** the person that manages the platform structure and functionalities.
- **Platform User:** the person that uses the platform.
- **Knowledge Manager:** the person that is an expert in the domain knowledge area of this platform. He is the responsible for managing the domain ontology used for the information classification.

In the following diagrams the use cases of each one of these actors will be presented and described.

Beginning with the **Platform Administrator**, this actor should manage in the back-end the RDF Data produced in the platform. This means he should be able to make changes on stored triples or perform some queries over them. He is also responsible for applying the semantic mapping designed by the **Knowledge Manager**.

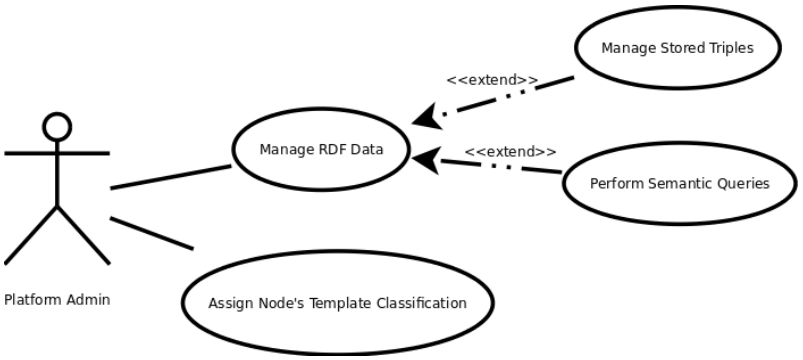


Figure 4.1: Semantic Use cases of a H-Know "Platform Administrator"

Following the designed classifications for the content items templates by the "Knowledge Manager", the platform administrator is responsible for applying and editing those classifications in the platform.

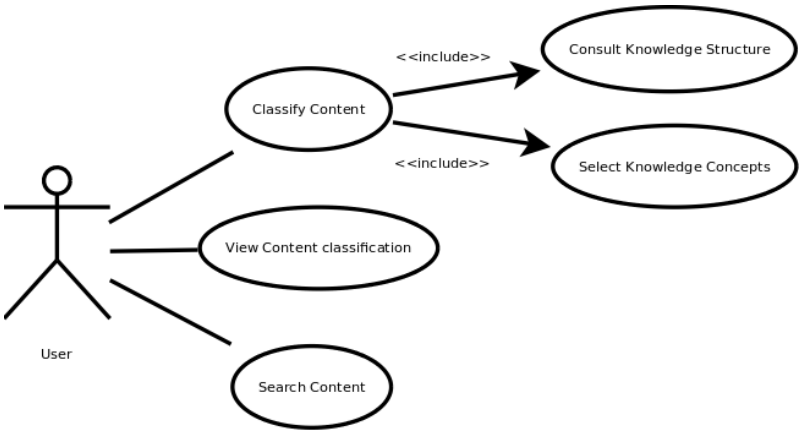


Figure 4.2: Semantic Use cases of a "Hknow User"

A user collaborating in the platform, should classify the content he produces. To do that, he needs to consult the knowledge structure and select suitable concepts from there that fit the meaning he wants to give to a specific content item. When navigating in the platform, he should be able to visualize the classifications done for each content he is viewing.

Taking advantage of the semantic classification of the platform, a user should be able to perform searches. In the previous chapter (3.7) it was presented the conceptualization of the semantic searching module.

The "Knowledge Manager" needs to manage the domain ontology used in the platform to classify the content items. To do that he needs to visualize the list of concepts included in the domain ontology and be able to add/delete concepts and add or edit its properties.

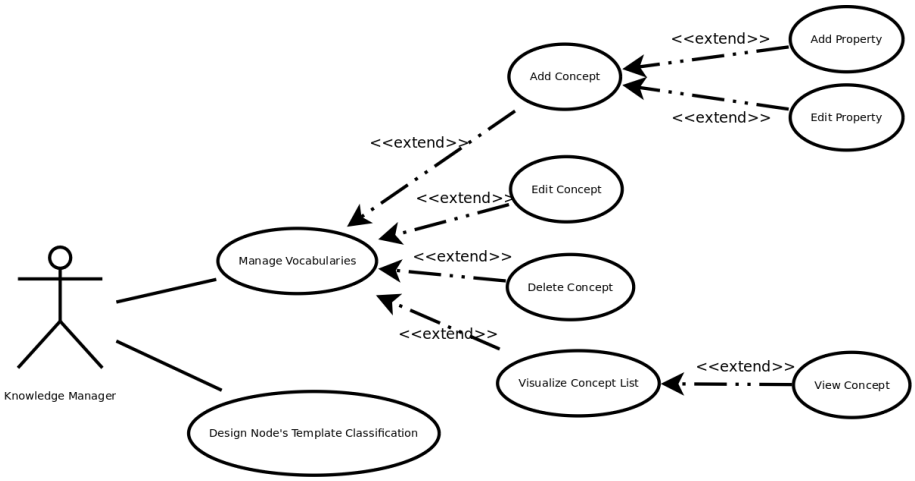


Figure 4.3: Semantic Use cases of a "Knowledge Manager"

In the other hand, this actor is responsible for designing the classification of each content item template of the platform.

4.2 Drupal Semantic Web existing projects

In this chapter we will talk about some of the developments made so far in the area of the semantics in the Drupal framework. This presentation is divided in 3 semantic areas: **semantic classification**, **semantic querying** and **semantic browsing**.

The presented modules were configured in a H-Know platform testing installation.

4.2.1 Semantic Classification

The base module for the use of semantic web in Drupal is **RDF API**[1], a module which the main purpose is to provide a uniform API and storage abstraction layer for querying and storing RDF statements in repositories, giving the chance to other Drupal modules hook into this to provide the actual implementation of storage backends and any other higher-level functionality. This module uses **ARC2**[2], a system of RDF classes for PHP.

In Drupal, like stated before, administrators use CCK to define the types of nodes they need, which are then used by content authors to populate the site. Taking advantage of this principle, it was developed a module, **RDF CCK**[3], which auto-generates RDF classes and properties for all content types and fields. This module creates a so called "site vocabulary", which describes the content types and fields used in the data model as classes and properties, based on the CCK field and type IDs. RDF CCK then **exports** all

[1]RDF API Module Webpage, http://groups.drupal.org/node/8930
[2]http://arc.semsol.org/
[3]RDF CCK Module Webpage, http://drupal.org/project/rdfcck

the semantic data to "node/*/rdf". With a patch applied to the original module, RDF CCK can also expose CCK fields in **RDFa**. This is a different way of exposing semantic web metadata, where the semantic information is embedded in HTML documents.

There was also developed another module called **Evoc** (External Vocabulary importer module)[4] that enables the reuse of RDF vocabularies across Drupal sites. This module will cache any external RDF vocabulary in Drupal and expose its classes and properties to other modules.

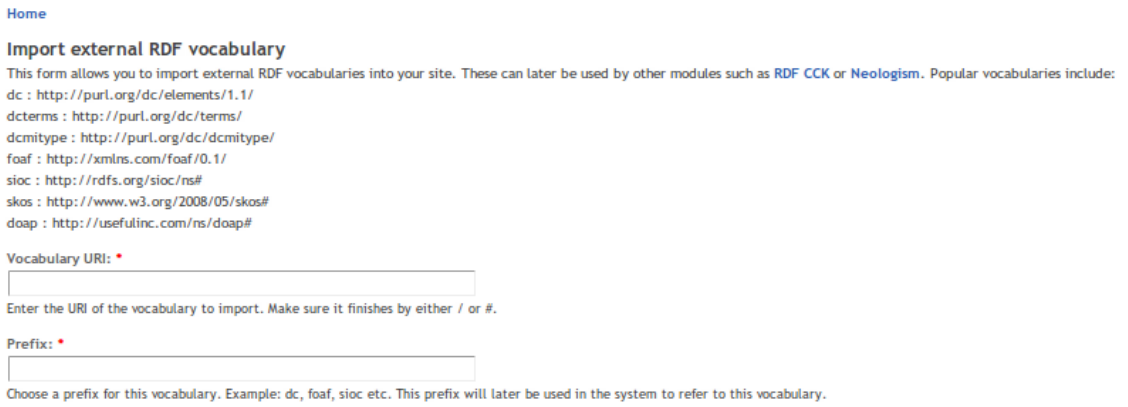


Figure 4.4: Evoc module configuration

Like we can see in the figure 4.4, in the evoc module we specify the URI of the vocabulary and the prefix that identifies it and the evoc module loads to the Drupal database the classes and properties of the given ontology.

Using these capabilities, RDF CCK module gives administrators the chance of mapping CCK types and fields to external vocabularies, using the classes and properties loaded by the evoc module.

In the figure 4.5, we can see the possibility of mapping both the type of content and its fields to external vocabularies.

These are the RDF modules that enable the semantic classification of the structure of a Drupal site. Now we will see some examples of modules developed to take advantage of that classification.

4.2.2 Semantic Querying

The RDF SPARQL Endpoint[5] module indexes the RDF data publicly available on a Drupal site into an ARC2 RDF store, providing a SPARQL endpoint to query RDF local data. The module automatically creates a local RDF repository which hosts all the RDF data generated by the RDF CCK module, in the Drupal installation database.

[4]Evoc Module Webpage http://drupal.org/project/evoc
[5]Sparql Endpoint Module Webpage, http://drupal.org/project/sparql$_$ep

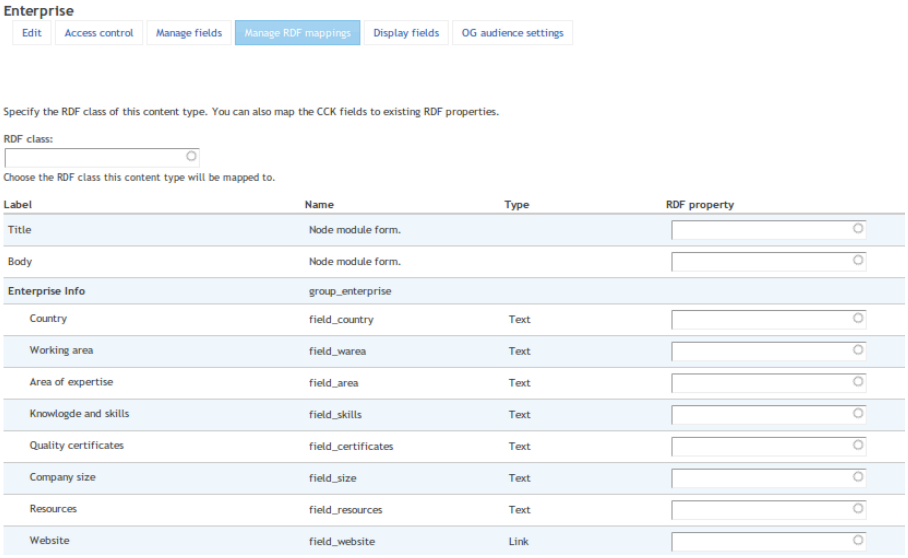


Figure 4.5: RDF CCK module configuration for the Enterprise content-type

In the figure 4.6 we can see the SPARQL endpoint created by this module, which enables users to perform queries over the triples generated by RDF CCK and export the results to several different formats (HTML, XML, RDF, Tuples, JSON, etc).

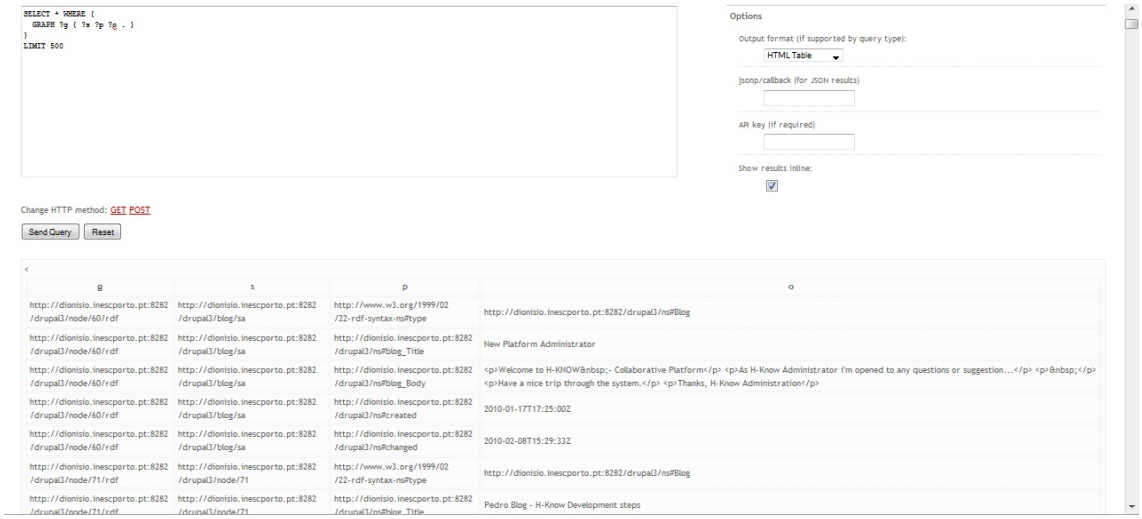


Figure 4.6: Drupal Sparql Endpoint Module

Since data is spread all over the Web, sometimes there is a need to use data retrieved from other resources. For that, it was created the module **RDF SPARQL Proxy**[6] which allows developers to instantiate RDF resources on demand (lazy loading) using SPARQL

[6]RDF Proxy Module Webpage http://drupal.org/project/rdfproxy

Construct queries. Site administrators can define profiles which specify the mapping rules of the remote data schema to the local Drupal RDF schema.

4.2.3 Semantic Browsing

A way to search for content based on semantic classification of the Drupal platform content can be the module **Exhibit Drupal**. This module embeds in Drupal, Exhibit, a facet browsing publishing framework, that allows developers to create web pages with support for sorting, filtering and rich visualizations.[7]

Exhibit is built over a collection of Javascript files. When pages are loaded on Exhibit by a browser, the Javascript reads the source data and builds a local database in the memory of the machine running the browser. Data can then be filtered and sorted directly in the browser without having to re-query by the server.

The first step to build an exhibit view is to configure the feeds of data it will use. Exhibit accepts the following formats: JSON, RDF/XML, N3, Excel, Tab-separated values, Bibtex, Google Spreadsheet and RDFa. All of these non-JSON formats are converted by Exhibit to Exhibit JSON format.

After defining the feeds of data, we should build our exhibit. For that we have to configure the view of the **content** and its **facets**. In the figure you can see an example of a facet browsing built with Exhibit. We can see in the left the content and at the right the facets.

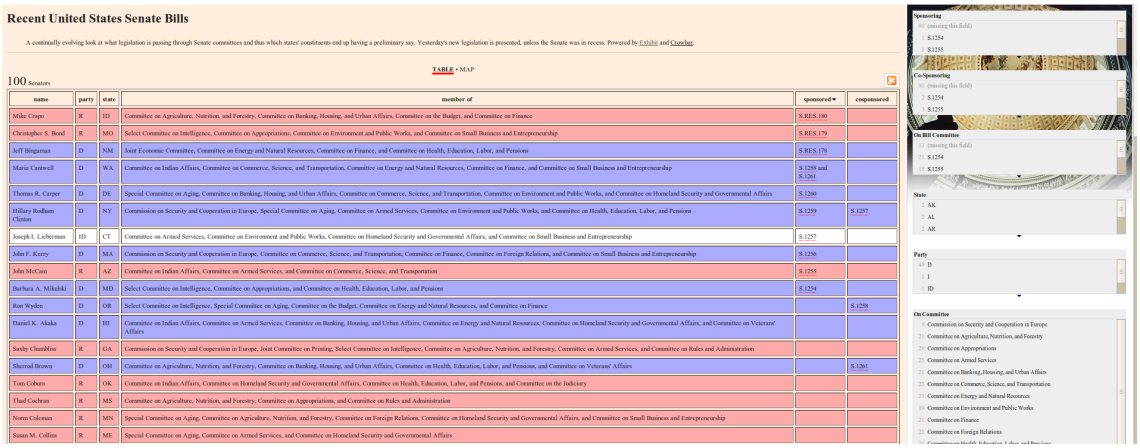


Figure 4.7: Example of a exhibit facet-browsing interface [EC]

[7] Exhibit Framework Website, http://www.simile-widgets.org/exhibit/

4.3 Platform Semantic Implementation

4.3.1 Architectural View

This chapter aims to give an overview about the technological architecture of the semantic solution implemented in the H-Know platform.

As a starting point to develop the semantic components of the H-Know platform, lets talk about the building blocks we need.

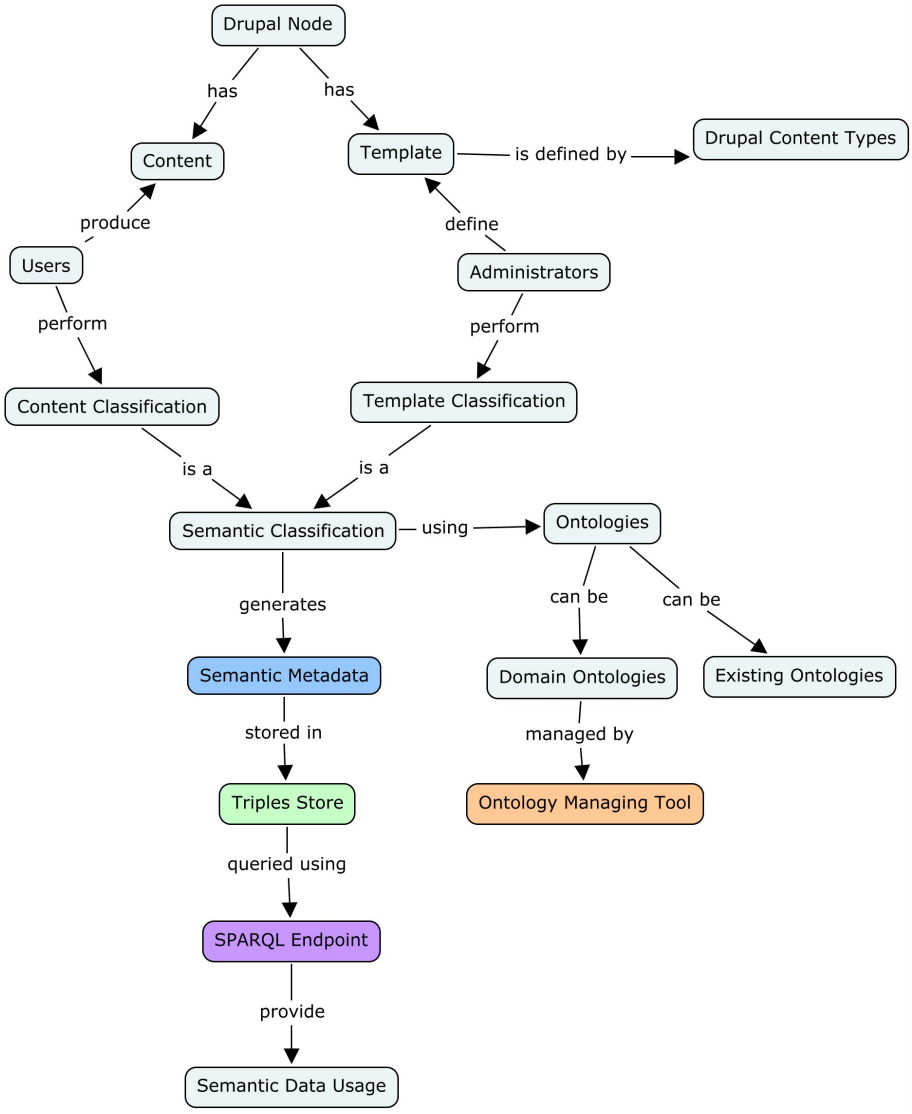


Figure 4.8: H-Know platform high-level Semantic Module

Like stated before, in the Drupal platform we have nodes, with a specific structure and with content produced there. The **structure** of the nodes (templates) should be classified by administrators (invisible for the users), while the **content** must be classified by the users which produce content in the platform. So, we need a back-end editor to classify

the structure of the nodes (the content types) and a front-end tool that gives users the chance of classifying the content they produce according to the domain knowledge. The domain ontologies should be managed by a **Ontology Managing Tool** that enables users to modified existing ontology classes and properties and to create new ones.

Both of these two classifications, generate **Semantic Metadata** in a standard semantic web format such as the RDF or Turtles. This Semantic Metadata must be stored somewhere. It can be **stored** in a Triplestore (like Virtuoso, Sesame or RDF Broker) or it can also be stored in a conventional relational database such as MySQL or PostgreSQL.

To allow the **usage** of the metadata produced in the platform, we need a **SPARQL endpoint** that can query it. By metadata usage we mean the searches users can do on top of the semantic metadata or the presentation of related content when navigating in the platform.

Now, regarding the needs explained above, lets present the designed solution.

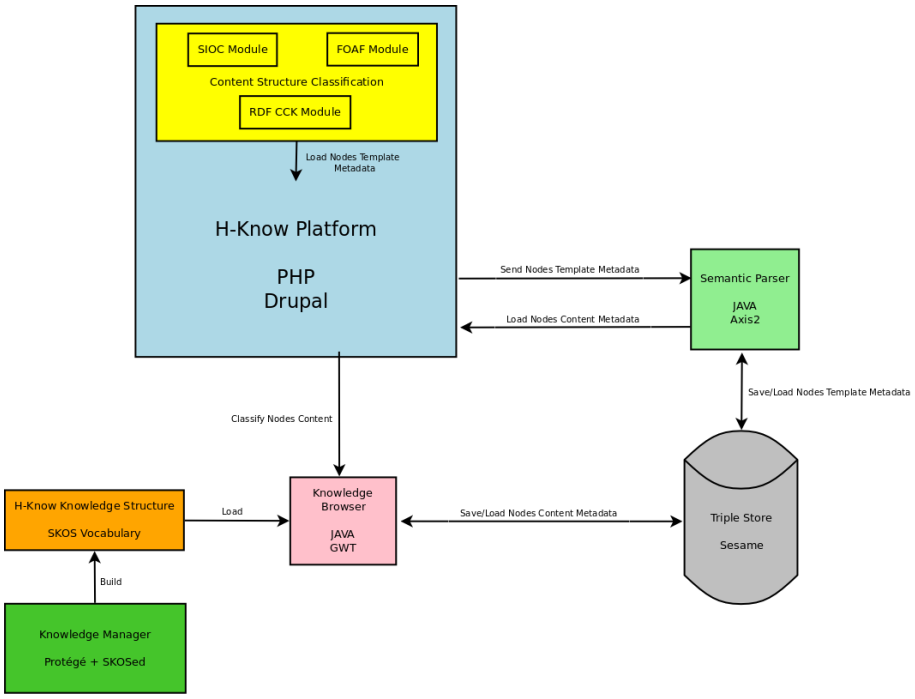


Figure 4.9: H-Know platform Semantic module architecture

So, starting from the bottom layer of the architecture, to **store** the metadata triples, **Sesame** framework was chosen in its 2.3.1 version. Sesame is an open source JAVA framework for storage, inferencing and querying of RDF data[8].

Sesame was chosen as the triple store because it provides a fast and reliable native triples store, with a comprehensive back-end managing interface (called Workbench) and a very complete and easy to use JAVA API with functions to interact with it. When

[8]Sesame Webpage, http://www.openrdf.org/

comparing this triple store with others, like presented in NCBO (National Centers for Biomedical Computing) report evaluation on February of 2009[9] , Sesame proves to be the fastest one loading data, comparing with Mulgara, JENA SDB and Virtuoso, three of the most important existing triples store. In this group, Sesame is the only one that provides an API for fine level access. We have a ready to use tool for storing Drupal's semantic metadata (RDF Sparql Endpoint module, presented before), but that solution, provides a triples store inside the Drupal's database, a non-native type of triples store. This kind of storage is not as efficient as a native triples store and, since one of the requirements of the H-Know project was to store the semantic metadata outside of Drupal, this was not an eligible solution.

Changing the focus to the Drupal platform, for the **template semantic classification**, a conjunction of three different customized Drupal modules (**RDF CCK Module**, **FOAF Module** and **SIOC Module**) was the solution. RDF CCK is an "out-of the-box" module, that provides an extension to the content types manager (CCK), to map each content type (node template) and its fields to ontology vocabularies. This module like explained before, exports the semantic metadata of each node template in different ways, such as RDFa or to a file with the semantic information of a node. In addition to this module were used the FOAF and SIOC modules, with some modifications, to describe the platform socio-collaborative characteristics. These modules act the same way as RDF CCK, exporting FOAF and SIOC information of the nodes to RDF/XML files.

To establish a connection between Drupal and Sesame, since Drupal is built over PHP language, the first idea was to use a PHP API providing direct communication between Drupal and Sesame. There is such an API, Phesame[10], but the only version available only works in Sesame 1.x versions, which is quiet different from Sesame 2.x versions, making it unusable for our solution. So, a new solution had to be designed. Since Sesame provides a JAVA API, this is the most suitable option. But then we have Drupal which is built on PHP. To solve this issue, a "**Semantic Parser**" was built to work as an intermediate between Drupal and Sesame, using a Web Service built using the Axis2 engine[11]. This application is used both to save Drupal metadata in Sesame and to load metadata from Sesame into Drupal. This application then uses the API of Sesame to perform the required actions (saving or loading of metadata).

To implement the semantic classification of the nodes content, done by the users, there is no existing ready to use, Drupal solution. Users need a tool to choose concepts from the knowledge domain of the platform and associate them with the content they produce. There are some existing projects which can be used to visualize ontologies and vocabularies, but they are rather commercial or not good enough for our purposes or difficult

[9]NCBO report, http://www.bioontology.org/wiki/images/6/6a/Triple$_$Stores.pdf
[10]Phesame API Webpage, http://www.hjournal.org/phesame/
[11]Apache Axis2 Webpage, http://ws.apache.org/axis2/

to customize. So, it was developed from scratch a JAVA GWT application,"**Knowledge Browser**", to load the domain knowledge structure of the platform and give users the chance of choosing concepts which relate with the content they are producing. This browser loads the knowledge structure using a JAVA API called SKOS API[12] and interacts with the Sesame triple store using its API.

The "Knowledge Structure" is managed using Protégé in its 4.0.2 version with a special plugin for the construction of SKOS vocabularies (SKOSed). The choice over the "Ontology Managing Tool" was Protégé because this is the worldwide most used Ontology editor, and the only one that provides a specific plugin for the edition of SKOS vocabularies, the chosen technology to formalize the knowledge domain. The fact that inside the H-Know project, members are already experienced with this tool, is another plus.

Completed this generic overview, in the next sub-chapters there will be given further details about each one of the parts described above.

4.3.2 Storing the semantic metadata

The semantic information produced in the platform is stored in the Sesame triple store. Sesame runs as a Web Application using Tomcat as the servlet container. In this case, we have Sesame 2.3.1, running on Apache Tomcat/6.0.26[13]. Sesame store is managed using OpenRDF Workbench web application. This tool enables users to create and edit **RDF repositories** (storage containers) and to manage its **contexts** (also known as named graphs, providing a mechanism for grouping RDF statements) , namespaces (prefix used to identify vocabularies) and types. It also provides some extra tools such **Explore** to navigate over the triples, **Query** to make Sparql queries over the triples or **Export** to export the stored metadata.

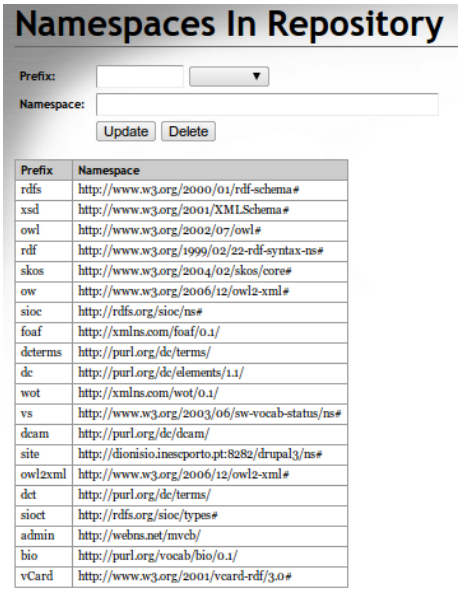
Figure 4.10: OpenRDF Workbench repositories manager

The figure 4.10 shows the aspect of the Workbench repositories manager. Sesame allows us to create 9 different types of repositories [B.V10]. In our case, it was created a

[12] SKOS API Webpage, http://skosapi.sourceforge.net/
[13] http://tomcat.apache.org/

repository for the platform called "hknow" of the type "Native Java store RDF schema", so we are able to do inferencing queries over our triples.

In the **namespaces manager**, we add the prefixes of the vocabularies, used in the platform: foaf, sioc, sioc-types, dc, dcterms, dctypes and skos. When rdf data is loaded into Sesame, it automatically creates the namespaces defined on that data. In this case it was done manually so we can have a better control of the existing namespaces in the store. To store the triples generated in the platform it was created a new context with the



Prefix	Namespace
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
skos	http://www.w3.org/2004/02/skos/core#
ow	http://www.w3.org/2006/12/owl2-xml#
sioc	http://rdfs.org/sioc/ns#
foaf	http://xmlns.com/foaf/0.1/
dcterms	http://purl.org/dc/terms/
dc	http://purl.org/dc/elements/1.1/
wot	http://xmlns.com/wot/0.1/
vs	http://www.w3.org/2003/06/sw-vocab-status/ns#
dcam	http://purl.org/dc/dcam/
site	http://dionisio.inescporto.pt:8282/drupal3/ns#
owl2xml	http://www.w3.org/2006/12/owl2-xml#
dct	http://purl.org/dc/terms/
sioct	http://rdfs.org/sioc/types#
admin	http://webns.net/mvcb/
bio	http://purl.org/vocab/bio/0.1/
vCard	http://www.w3.org/2001/vcard-rdf/3.0#

Figure 4.11: Workbench view of the defined namespaces

URL of the platform. The knowledge structure was also loaded into the repository, so as the standard vocabularies foaf, sioc, sioc-types, dc, dcterms, dctypes. This was done in order to have stored in our database the complete structure of these online published vocabularies used in our classifications.

skos:Collection	rdf:type	owl:Thing	<file://hknow.rdf>
skos:Collection	skos:definition	"A meaningful collection of concepts."@en	<file://hknow.rdf>
skos:Collection	skos:scopeNote	"Labelled collections can be used where you would like a set of concepts to be displayed under a 'node label' in the hierarchy."@en	<file://hknow.rdf>
skos:Concept	rdf:type	owl:Thing	<file://hknow.rdf>
skos:Concept	skos:definition	"An idea or notion; a unit of thought."@en	<file://hknow.rdf>
skos:ConceptScheme	rdf:type	owl:Thing	<file://hknow.rdf>
skos:ConceptScheme	skos:scopeNote	"A concept scheme may be defined to include concepts from different sources."@en	<file://hknow.rdf>
skos:ConceptScheme	skos:definition	"A set of concepts, optionally including statements about semantic relationships between those concepts."@en	<file://hknow.rdf>
skos:ConstructionComplex	rdf:type	owl:Thing	<file://hknow.rdf>
skos:ConstructionComplex	rdf:type	skos:Concept	<file://hknow.rdf>
skos:ConstructionComplex	skos:prefLabel	"Construction Complex"@en	<file://hknow.rdf>
skos:ConstructionComplex	skos:altLabel	"Construction Complex"@en	<file://hknow.rdf>
skos:ConstructionComplex	skos:definition	"This is a construction complex."@en	<file://hknow.rdf>
skos:ConstructionComplex	skos:altLabel	"Complexo de Construção"@pt	<file://hknow.rdf>
skos:ConstructionComplex	skos:prefLabel	"Complexo de construção"@pt	<file://hknow.rdf>
skos:ConstructionComplex	skos:definition	"Este é um complexo de construção."@pt	<file://hknow.rdf>
skos:ConstructionComplex	skos:broaderTransitive	skos:ConstructionResult	<file://hknow.rdf>
skos:ConstructionEntity	rdf:type	skos:Concept	<file://hknow.rdf>
skos:ConstructionEntity	rdf:type	owl:Thing	<file://hknow.rdf>
skos:ConstructionEntity	skos:prefLabel	"Construction Entity"@en	<file://hknow.rdf>

Figure 4.12: Visualization of some part of the knowledge structure loaded into Sesame

The picture above shows part of the knowledge domain, loaded into the Sesame triple store. Using the Sesame Explorer we can navigate through the stored triples.

4.3.3 Implementing the Semantic Parser Web Server

To make the connection between Drupal and our Sesame triple store, it was built a Java application to work as an intermediate between the two. The "Semantic Parser" is a Java application implemented using the Axis2 framework running as a Web Application on Tomcat.

This application receives requests from Drupal and performs actions on the Sesame triple store using its Java API. In the diagram 4.13 we can better understand this server/client architecture.

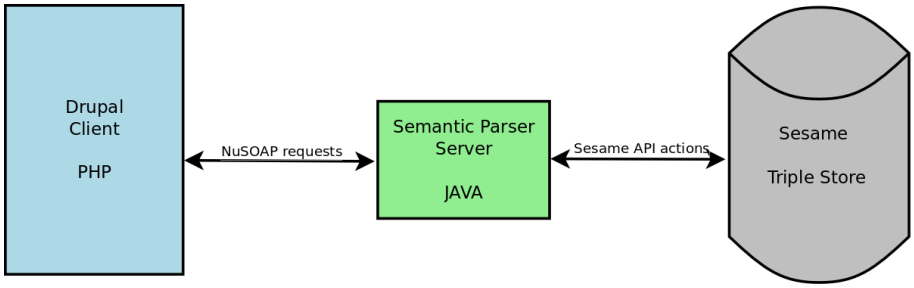


Figure 4.13: Visualization of some part of the knowledge structure loaded into Sesame

As we can see in the diagram 4.13, Drupal plays as the **Client** application connecting with the "Semantic Parser", published as a web service. The communication with the Server is done using the NuSOAP API[14], which allows developers to create and consume web services based on SOAP[15], WSDL and HTTP. In our case we connect by WSDL.

After creating the connection with the service, inside Drupal, we call methods implemented by the "Semantic Parser", to perform functions of loading and storing of RDF data to Sesame.

The request is received and processed by the server and the result is then analysed in the client (Drupal).

In the side of the **Server**, he receives requests to load or store RDF data in Sesame using the Sesame API. The server then processes those requests and sends the results to the client (Drupal). There is always a connection with the Sesame repository hosting the platform triples and then the call of functions to add and query/load metadata from the repository where the platform triples are stored.

In the **Semantic Parser** we implemented so far the following functions:

[14]NuSOAP API Webpage, http://sourceforge.net/projects/nusoap/
[15]SOAP W3C Webpage, http://www.w3.org/TR/soap/

- **String storeNodeRDF(String file_url):** receives the url of a RDF/XML file and stores it in the Sesame triple store.
- **String[] sendNodeConcepts(String node):** receives the url of a Drupal node and retrieves the concepts which are related with that node.

This "Semantic Parser" is a crucial piece to enable the storing of metadata outside the Drupal environment.

4.3.4 Implementing the platform structure classification

This chapter aims to explain how the semantics of the platform structure is implemented, following the conceptualization defined (3.4).

To classify the template of every content type, we use the RDF CCK. This module fulfils the need we have to allow the administrators to map each one of the content types managed in CCK to a existing ontology vocabulary so as its fields. So, after configuring this module in our Drupal installation, for each content type we have, we implement the mappings designed.

For example, an Entity of the H-Know platform is described by its profile page. That profile is a node of the content type "entity". In the management of that content type and using the RDF CCK module we can do the semantic mapping for that type.

RDF CCK module, uses as default a site vocabulary defined by the RDF API Drupal module. So, even if administrators don't define the vocabularies that match with some specific content type or field of it, there will always be a auto-generated site vocabulary concept for it. This module, as a feature that enables the export of the metadata defined for every node to a URL of the type **"node/node_id/rdf"**, using the RDF API.

In the settings of the RDF API module, we can select the format of the export, like we see in the next picture.

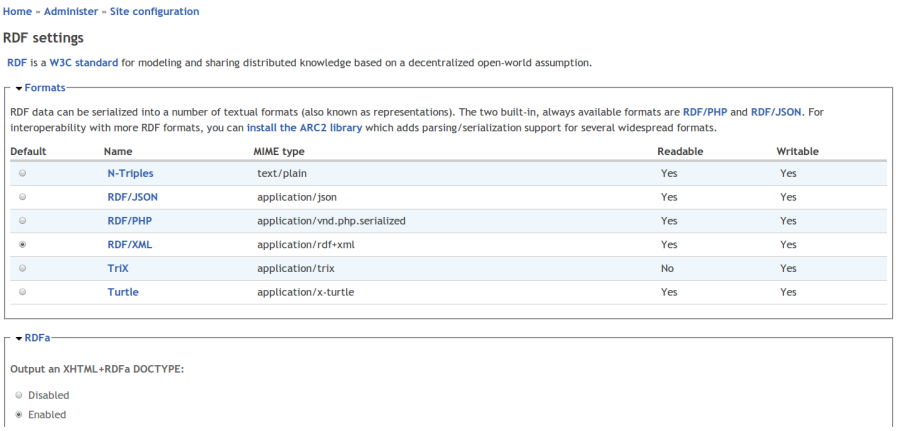


Figure 4.14: Drupal RDF settings page

In this page, it was also enabled the option "Output an XHTML+RDFa DOCTYPE". This means that the classifications defined by RDF CCK are also available embedded in the HTML pages. The used namespaces for the vocabularies are declared in the html and its type and fields are identified with "property" tags.

xpto@xpto.com

This way we also have semantic information stored directly in our pages, available for anyone that wants to use a Web Crawler and re-use the classified information.

In the following picture, we can see an example of a exported node rdf file metadata, in the RDF/XML format.

```
−<rdf:RDF>
 −<site:Place rdf:about="http://dionisio.inescporto.pt:8282/drupal3/node/78">
    <site:place_Title>VortalRehab Future HeadQuarters</site:place_Title>
   −<site:place_Body>
      <p>Rehabilitate the building of VortalRehab's future headquarters. @ Porto's downtown historical center</p>
    </site:place_Body>
    <site:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2010-02-08T15:54:23Z</site:created>
    <site:changed rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2010-03-02T13:57:28Z</site:changed>
    <site:place_Image rdf:resource="http://dionisio.inescporto.pt:8282/drupal3/sites/default/files/oldbuilding.jpg"/>
    <site:place_Enterprise rdf:resource="http://dionisio.inescporto.pt:8282/drupal3/enterprise/3m2p#this"/>
    <site:place_Enterprise rdf:resource="http://dionisio.inescporto.pt:8282/drupal3/enterprise/rehabknow-son-lda#this"/>
    <site:place_Enterprise rdf:resource="http://dionisio.inescporto.pt:8282/drupal3/enterprise/stb-%E2%80%93-reabilita%C3%A7%C3%A3o-do-patrim%C3%B3nio-edificado-lda#this"/>
    <site:place_Enterprise rdf:resource="http://dionisio.inescporto.pt:8282/drupal3/enterprise/vortal#this"/>
    <site:place_Institute rdf:resource="http://dionisio.inescporto.pt:8282/drupal3/group/feup#this"/>
    <site:place_Institute rdf:resource="http://dionisio.inescporto.pt:8282/drupal3/group/inesc-porto#this"/>
    <site:place_Area>Ornamental plaster</site:place_Area>
    <site:place_Area>Masonry structural consolidation</site:place_Area>
    <site:place_Area>Wood windows</site:place_Area>
    <site:place_Budget>500.000€</site:place_Budget>
  </site:Place>
</rdf:RDF>
```

Figure 4.15: RDF/XML file exported by RDF CCK

Originally, this module uses as **Subject** for the triples, URIs with the title of the pages (for example: ".../project/XPTO Project"). That was changed to use instead of it, URIs in the "node/node_id" format, so we have a standard URI format for all the kind of nodes, that is not affected when the title of the page is changed.

RDF CCK just configures the semantics of the fields which are part of a content type template. It doesn't configure important "under covered" semantic information such as the creator of a content or the creation data of it.

To configure this kind of information we configure a different module: SIOC module[16]. This module as a similar behaviour to RDF CCK module. It exports to the url's **".../sioc/node/node_id"**, the generated metadata of the **nodes**, using the SIOC vocabulary, in RDF/XML format and to **".../sioc/user/user_id"** the metadata of a **User**. In addition, this module is configured to describe in SIOC, native Drupal content types such as Blog or Forum. This is important to express the relations and the typical structure of these contents.

This module was changed to also express the semantics of the membership of Users to Collaborative Places and Entities and the association of Entities to Collaborative Places, using the **sioc:has_member** and **sioc:usergroup_of** properties as the triples predicate, like described in the previous diagram **??**. The relationships of members to groups are

[16]SIOC Module Webpage, http://drupal.org/project/sioc

kept in a table created by the Organic Groups module, the module responsible for creating the groups in the platform (ColPlaces and Entities). For each group in the platform, that table is queried to get its partners.

To express the metadata representing the **Person** and **Organization** elements of H-Know, which in the platform activities, are expressed as **User** and **Entity**, we use the module FOAF[17].

In the case of users, in addition to the generation of **foaf:Person** class for each user, it was patched this module to also semantically classify the partners each Person has. This module exports to **".../foaf/user_id"** the generated metadata for users and to **".../foaforg/entity_id"** the generated metadata for entities.

Picking up the same idea of the **RDF Sparql Endpoint** module that indexes all the nodes metadata generated by RDF CCK inside Drupal's database, it was built a new module that aims to do the same but **store the metadata outside Drupal**, in the Sesame triple store. We have done an addition to this module perspective: plus getting the metadata exported by RDF CCK, we also index in the Sesame store the metadata generated by the FOAF and SIOC modules.

So, for the platform nodes, it was built a script that iterates through all the published nodes in the platform, builds their RDF CCK export url and the SIOC export url and then sends them to the "Semantic Parser", using the Web Service function "storeNodeRDF". The URLs are received by the "Semantic Parser" that then uses them to store the metadata of each node in the Sesame triple store, using its API.

This script also iterates through the users and entities registered in the platform, building their FOAF export url and also sending them to the "Semantic Parser", that stores the metadata in Sesame.

With the conjunction of these three exports, we have all the metadata of our platform structure, stored in Sesame.

4.3.5 Managing the domain ontology

This chapter aims to describe how the domain ontologies of the platform are managed.

To manage the domain knowledge of the platform, many tools were investigated. Protégé[18] was the selected one because all the team members of H-Know project are used to it, because it has a big number of active developers improving it and because the existence of a plugin like SKOSed specific for the construction of SKOS vocabularies fits the project needs.

[17] FOAF Module Webpage, http://drupal.org/project/foaf
[18] http://protege.stanford.edu/

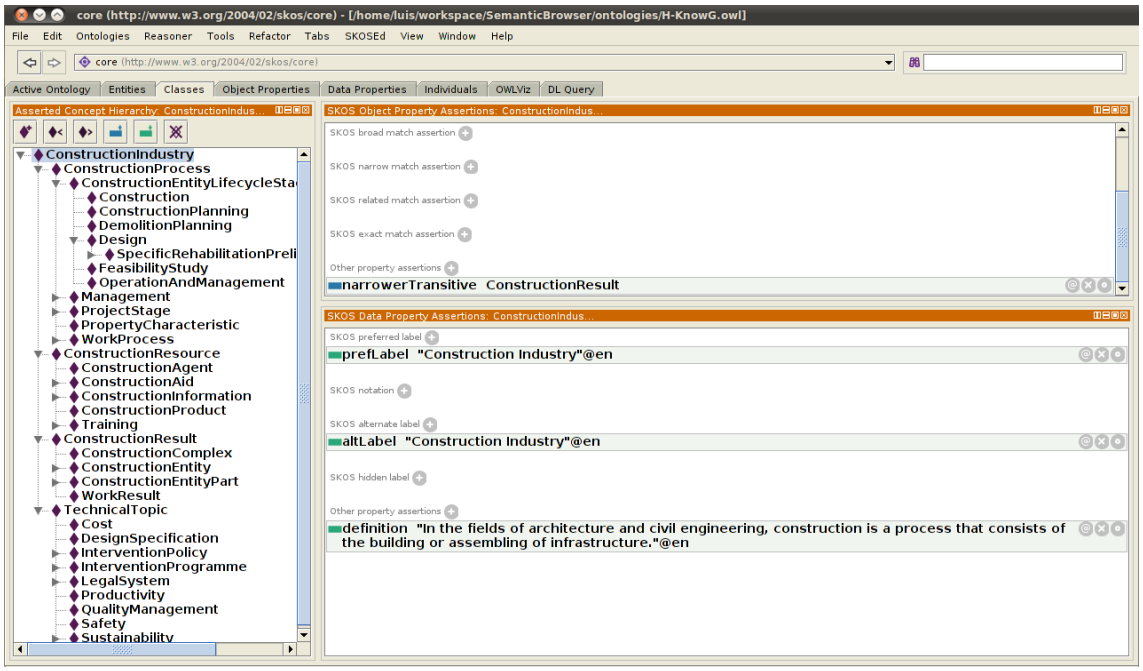


Figure 4.16: Protégé tool with the SKOSed plugin, editing the domain vocabulary

SKOSed[19], the plugin that you can see in the figure 4.16, enables the edition of SKOS vocabularies, by the creation of SKOS concepts, their properties and their hierarchical arrangement.

Picking up the domain knowledge structure presented before, we defined all the domain concepts as a SKOS concept and created them with the plugin so as their properties. For each concept we define object properties (the relations broader, narrower, topConcept, etc) and the data properties (prefLabel, altLabel and definition). The data properties were defined in different languages, responding to the H-Know requirements. The resulting vocabulary was then exported to a OWL file and loaded from the "Ontology Browser".

An interesting project using SKOS is "Pool Party" [SB10], a web application to create and maintain SKOS vocabularies with a easy to use user-interface. This solution was not used because it's a commercial product and in the H-Know project, one of the requirements is to use Open-source solutions.

4.3.6 Implementing the platform content classification

Here in this chapter it is explained how users can browse the domain knowledge ontology and associate the content they produce with concepts from it.

[19] http://code.google.com/p/skoseditor/

In the H-Know platform users need to classify the content they produce. In order to do that, it was created a Web Application to allow users to browse the knowledge structure and select the concepts which relate with the content they are producing.

This was built on Java using GWT [20] (Google Web Toolkits) and SmartGWT[21] (an extension of GWT) to build the user interface. The GWT application is published as a Web Application in the Tomcat server. GWT is a development toolkit for building browser-based applications, allowing developers to build complex applications on Java with a pretty interface without losing time with the tedious frequent browsers visualization problems. It implements in the back-end Javascript and Ajax functions, without the user having a big knowledge about these technologies. GWT is used by many Google applications such as Google Wave and Google AdWords.

GWT works in a client-server architecture. So, in the client side we implement all the interface details and in the server we process all the needed information. This way, there is a complete separation between the content and the layout. Client and server communicate via services implemented in the client. Further on, there will be given further information about this architecture.

The main idea of this tool is to provide an easy to use interface, where users can select the most suitable concepts of the domain knowledge to classify the content they produce.

Basically, this user interface 4.17 has two information blocks:

- **H-Know Domain:** a tree list of the domain knowledge concepts;
- **Concept information:** relevant information about a concept.

The tree list presents the knowledge structure loaded from a RDF/OWL file with the domain knowledge of the H-Know project. In the server side, it was implemented a parser for RDF/OWL files using the SKOS API [22]. Like it was explained before, in a SKOS vocabulary, the Concept class is the foundation piece. So, this parser reads a file with the description of a SKOS vocabulary and builds "Concept" objects. It was created a **Concept** class that implements a JAVA "Serializable", which has the following structure:

- String uriPath: the URI describing the concept;
- String conceptId: internal id created to identify a loaded concept.
- String name: the name of the concept;
- **Lexical properties:**

[20] GWT Project Webpage, http://code.google.com/intl/en/webtoolkit/
[21] SmartGWT Webpage, http://code.google.com/p/smartgwt/
[22] http://skosapi.sourceforge.net/

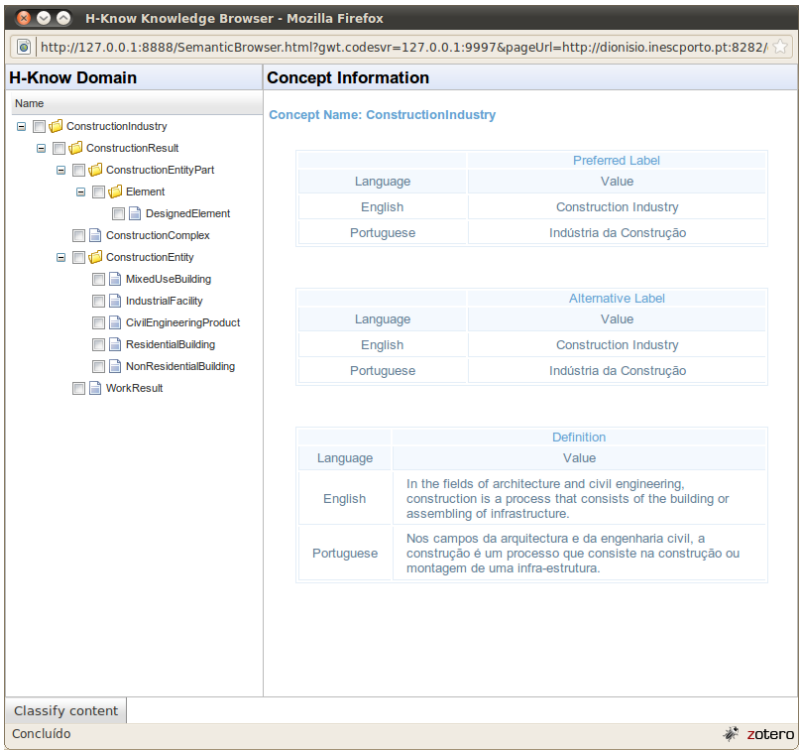


Figure 4.17: Ontology Browser interface

- Map<String,String> prefLabel: skos property that defines the preferred label to describe the concept. It is a map that has as key the language and as value the label.
- Map<String,String> altLabel: skos property that defines the alternative label to describe the concept. It as the same structure as prefLabel (Language, Label).
- Map<String,String> defintion: skos property that gives the definition of a concept. Like the other two lexical properties as a Language, Label structure.
- **Relational properties:**
- String[] broaderName: the name of the broader concepts.
- String[] narrowerName: the name of the narrower concepts.

The "Concept" objects created from the parser are sent to the client on a JAVA List. Then, the client creates the nodes of the tree with the information gathered from the Concepts. It was created a SmartGWT class called **ConceptNode**, which has the same structure as a "Concept" with the difference that is a graphical object of the user interface.

It was created a **SmartGWT Tree**, with an **Array of ConceptNodes** as the data elements. In the properties of the Tree, it was defined that the node's name is the Concept name, the node's id is the Concept id and that the parent's node is the Broader Id of a Concept. This way we have a hierarchical built tree.

This Tree was then putted inside a **SmarGWT TreeGrid**, that has as a selection method check-boxes so users can select concepts from the tree for the classification they are doing.

When a user selects a concept of the tree, the information about that concept is presented in the **Concept Information** block. We show the users the most relevant information about a concept in all the defined languages so he can do a proper selection of the concept he needs to classify a specific content. This is done by implementing the **onCellClick handler** of the Tree which changes the content information of a node when a cell is clicked.

After selecting all the concepts, the user clicks in the **Classify content** button and a report of the selected concepts is presented and the classification is done.

In the handler of this button, there is a call to a server function which receives an array with the URL's of the selected concepts. In the side of the server, the triples are built and stored in Sesame using its API.

The triples have the following format:

- **Subject:** The URL of the node being classified.
- **Predicate:** The sioc property "topic", linking the concept to the node.
- **Object:** The URL of a skos concept class.

All the triples generated are stored in the same context as the other platform triples.

This classification browser is integrated inside Drupal. In every node edition, a user can use this browser to classify the content he is publishing. To do this, the edition of a node was changed by "overriding" the **node edit form**.

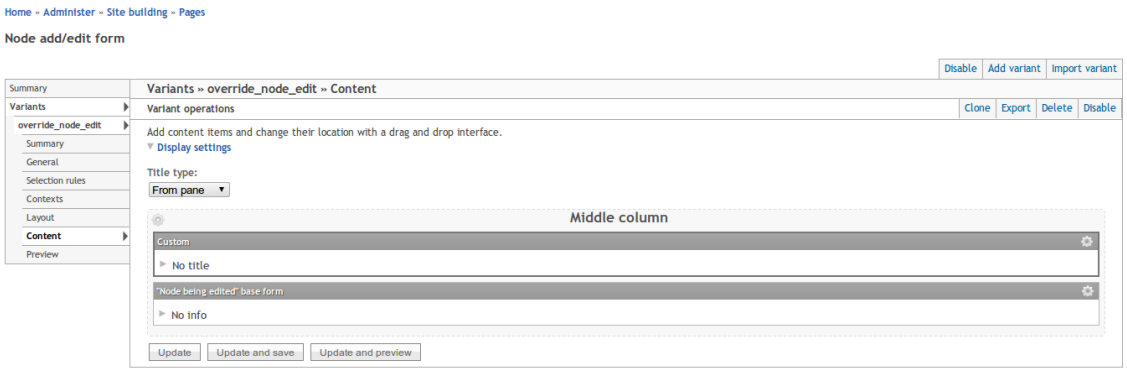


Figure 4.18: Drupal node edit form override

In the layout of this form, it was added a custom pane, for a **Semantic Classification box**. In this pane, users can open the **Knowledge Browser** and select concepts to classify the node. The browser is opened with an extra variable in the path which is the url of the node, to be used in the construction of the triples.

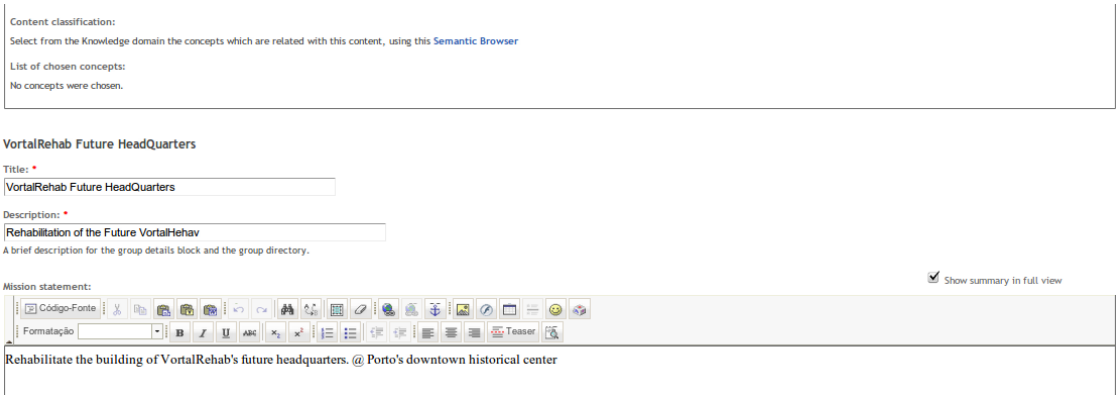


Figure 4.19: content classification of a node

In the content classification pane is also presented a **list of the concepts** that the user associated with the node. This is achieved using a request from Drupal to the "Semantic Parser" (the "sendNodeConcepts" function) that retrieves a list of the concepts associated with the node.

Chapter 5

Conclusions and Future Work

The objectives defined for this dissertation work were fulfilled. The foundations of the semantic integration in the platform are done. It was developed and implemented a model to express the socio-collaborative activities managed in the platform and defined a methodology to formalize the domain knowledge. It was also defined a model to integrate those activities with the domain concepts. In the other hand, it was designed a technological architecture to implement the integration of semantics in the H-Know platform.

All the generated metadata is stored outside of Drupal in a native triple store application, Sesame. In a sum, we have the content created in the H-Know platform semantically described, ready to be consumed by any application that wants to take advantage of the information available in the H-Know platform.

The semantic usage of the platform generated metadata is an area that lacked of implementations during this dissertation. It was studied and conceptualized some facet-browsing interfaces to perform searches based on the semantic classifications, but those interfaces were not developed because of time limitations. The current solutions to implement facet-browsing interfaces based on RDF metadata stored in triple stores are not sufficiently developed for quick usage and testing. It's an area that needs further investigation and developments.

This dissertation work is obviously not perfect and complete. So, there are a list of **improvements** and **new developments** that can follow this dissertation project.

Thinking about the developments done in this dissertation we can point out the following list of **improvements**:

- Improve the **"Semantic Browser"**: the usability of this browser can be improved by adding search functionalities to it so users can easier find the suitable concepts

they need to classify the contents they are producing. A search dialog with auto-complete functionality can be added to the browser, giving users a quicker way of finding concepts.

- Synchronize platform changes with the stored semantic metadata: at the moment, we have a module that exports at a time, all the semantic metadata of the platform structure to the Sesame triple store. The metadata of the platform content should be updated when creating, editing or removing contents.

- Improve the usability of the **"contents classification"**: users of the platform should have the minimum possible efforts to classify the content they produce. The "auto-complete" search box conceptualized in the non-functional prototype may be a good way of enabling a quicker form of finding a concept for classification.

- Embed RDFa of all the semantic metadata generated in the platform pages: right now, only the metadata generated by RDF CCK is embedded in the pages as RDFa. Embedding all the metadata generated in the pages can improve the reuse of the metada by other Semantic Web applications.

Picking up the results that came out of this dissertation and the ideas that appeared during this period, here are some **new developments** that can be done from the point we are now:

- Implement the conceptualized **facet-browsing interfaces**: Exhibit presented earlier in this report, may be a good option for this but, since this is a recent paradigm of searching, new solutions may arise and be better than Exhibit.

- Extend the **"Semantic Browser"** implementing the display of platform content: present in the browser resources which have been classified. User selects concepts from the three of concepts and visualizes the resources in the platform classified with those concepts. Filters for the socio-collaborative metadata can also be added. This may be another approach for a kind of "facet-browsing" interface.

- Use the semantic metadata generated in the platform in applications using different forms of knowledge representation (Topic Maps for example[1])

- Use the FOAF and SIOC metadata generated in the platform in applications developed for the use of these vocabularies: the profile information of the users and entities can be used in other places or the information generated in forums or blogs of the platform, can be reused in other systems managing the same type of knowledge in the Civil Construction field.

[1] http://www.topicmaps.org/

- Generalization of the Technological semantic architecture: the technological architecture for the integration of semantics in a socio-collaborative platform connected with domain knowledge presented is specific for the Drupal framework. It might be interesting to advance to higher level of abstraction and create an architecture that can be implemented in any CMS just with some integration and configurations.

References

[AKTV08] A. Ankolekar, M. Kr\ötzsch, T. Tran, and D. Vrandečić. The two cultures: mashing up web 2.0 and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):70–75, 2008.

[BBa] Uldis Bojars and John Breslin. SIOC core ontology specification. http://rdfs.org/sioc/spec/.

[BBb] Uldis Bojars and John Breslin. sioc-project.org | Semantically-Interlinked online communities. http://sioc-project.org/.

[BBFD08] U. Bojars, J. G. Breslin, A. Finn, and S. Decker. Using the semantic web for linking and reusing data across web 2.0 communities. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):21–28, 2008.

[BHO07] U. Bojars, B. Heitmann, and E. Oren. A prototype to explore content and context on social community sites. In *SABRE Conference on Social Semantic Web (CSSW 2007)*, 2007.

[BL05] Tim Berners-Lee. Web for real people. http://www.w3.org/2005/Talks/0511-keynote-tbl/, 2005.

[BMa] Dan Brickley and Libby Miller. FOAF vocabulary specification. http://xmlns.com/foaf/spec/.

[BMb] Dan Brickley and Libby Miller. The friend of a friend (FOAF) project | FOAF project. http://www.foaf-project.org/.

[BM01] Dave Beckett and Brian McBride. RDF/XML syntax specification (Revised). http://www.w3.org/TR/REC-rdf-syntax/, Maio 2001.

[BPBD08] U. Bojars, A. Passant, J. G Breslin, and S. Decker. Social network and data portability using semantic web technologies. In *2nd Workshop on Social Aspects of the Web (SAW 2008) at BIS2008*, page 5–19, 2008.

[Bro] M. Brown. "Facebook for the enterprise": Catchy phrase or a strategy for collaboration?

[B.V10] Aduna B.V. User guide for sesame 2.3. http://www.openrdf.org/doc/sesame2/users/, 2010.

[CDC+09] S. Corlosquet, R. Delbru, T. Clark, A. Polleres, S. Decker, A. Haller, M. Marmolowski, W. Gaaloul, E. Oren, B. Sapkota, et al. Produce and consume linked data with drupal! November 2009.

[Con10] H-Know Consortium. H-know platform early prototype testing instalation. http://dionisio.inescporto.pt:8282/drupal3/, 2010.

[Cor] Stéphane Corlosquet. Drupal RDF schema proposal | groups.drupal.org. http://groups.drupal.org/node/9311.

[DLHA09] F. Dengler, S. Lamparter, M. Hefke, and A. Abecker. Collaborative process development using semantic mediawiki. In *Proceedings of the 5th Conference of Professional Knowledge Management. Solothurn, Switzerland*, 2009.

[Dum] Edd Dumbill. XML watch: Finding friends with XML and RDF. http://www.ibm.com/developerworks/xml/library/x-foaf.html.

[EC] Exhibit and Crowbar. Recent united states senate bills. http://www.simile-widgets.org/exhibit/examples/senate/senate.html.

[Eng04] Y. Engestr\öm. Collaborative intentionality capital: Object-Oriented interagency in multiogranizational fields. *University of California, San Diego*, 2004.

[FDZJ05] T. Finin, L. Ding, L. Zhou, and A. Joshi. Social networking on the semantic web. *The Learning Organization*, 12(5):418–435, 2005.

[Fei] Lee Feigenbaum. SPARQL protocol and query language: SPARQL frequently asked questions. http://thefigtrees.net/lee/sw/sparql-faq.

[FHLW03] Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster. Spinning the semantic web. *IEEE INTELLIGENT SYSTEMS*, 2003.

[Flo09] L. Floridi. Web 2.0 vs. the semantic web: A philosophical assessment. *Episteme*, 6(1):25–37, 2009.

[G+95] T. R Gruber et al. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, 43(5):907–928, 1995.

[GR08] J. Golbeck and M. Rothstein. Linking social networks on the web with foaf: A semantic web case study. In *AAAI*, pages 1138–1143, 2008.

[Her] Ivan Herman. W3C semantic web FAQ. http://www.w3.org/2001/sw/SW-FAQ.

[HJSS06] A. Hotho, R. J\äschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. *The Semantic Web: Research and Applications*, pages 411–426, 2006.

[HLB01] J. Hendler, O. Lassila, and T. Berners-Lee. The semantic web. *Scientific American*, 284(5):35, 2001.

[HPPH05] I. Horrocks, B. Parsia, P. Patel-Schneider, and J. Hendler. Semantic web architecture: Stack or two towers? *Principles and Practice of Semantic Web Reasoning*, page 37–41, 2005.

[ic10] Drupal individual contributors. The drupal overview | drupal.org. http://drupal.org/getting-started/before/overview, 2010.

[IS] Antoine Isaac and Ed Summers. SKOS simple knowledge organization system primer. http://www.w3.org/TR/skos-primer/.

[ISO01] Iso 12006-2 building construction — organization of information about construction works - framework for classification of information, 2001.

[JBS08] S. Jupp, S. Bechhofer, and R. Stevens. SKOS with OWL: don't be full-ish! In *Fifth International workshop on OWL Experiences and Directions*, 2008.

[LAD+10] G. B. Laleci, G. Aluc, A. Dogac, A. Sinaci, O. Kilic, and F. Tuncer. A semantic backend for content management systems. *Knowledge-Based Systems*, 2010.

[Mik05] Peter Mika. Flink: Semantic web technology for the extraction and analysis of social networks. *Journal of Web Semantics*, 2005.

[MLD06] Lau L. Minh Le D. An open architecture for ontology-enabled content management systems: A case study in managing learning objects. *On the Move to Meaningful Internet Systems 2006, France*, 2006.

[MS01] A. Maedche and S. Staab. Ontology learning. *IEEE INTELLIGENT SYSTEMS*, 2001.

[Obi02a] M. Obitko. Translating between ontologies for agent communication. *Gerstner Laboratory for Intelligent Decision Making and Control. Series of Research Reports*, 2002.

[Obi02b] M. Obitko. Translating between ontologies for agent communication. *Gerstner Laboratory for Intelligent Decision Making and Control. Series of Research Reports*, 2002.

[PS] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[RGFR06] Garcia R., Gimeno, Perdrix F., and Gil R. The rhizomer semantic content management system, 2006.

[Rus03] Jack Rusher. Triple store. *Semantic Web Advanced Development for Europe (SWAD-Europe)*, 2003.

[SB10] Thomas Schandl and Andreas Blumauer. Poolparty: Skos thesaurus management utilizing linked data. *The Semantic Web: Research and Applications*, 2010.

[SLnn+07] J. Soriano, D. Lizcano, M. A Ca\ nas nas, M. Reyes, and J. J Hierro. Fostering innovation in a mashup-oriented enterprise 2.0 collaboration environment. *UK, sai: sisn*, 24:62–68, 2007.

[SWM01] Michael Smith, Chris Welty, and Deborah McGuinness. OWL web ontology language guide. http://www.w3.org/TR/owl-guide/, Maio 2001.

[Tap06] D. Tapscott. Winning with the enterprise 2.0. *New Paradigm Learning Corporation*, 2006.

[VL09] Mario Volke and Thorsten Liebig. Origo - a client for a distributed semantic social network. 2009.

[Win] Wine ontology. http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf.