Paulo Alves Figueiras

A framework for supporting knowledge representation – An ontological based approach

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: Professor Ricardo Jardim-Gonçalves, FCT-UNL
Co-orientador: Professor Celson Lima, UFOPA

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

2012

COPYRIGHT

A framework for supporting knowledge representation - An ontological based approach

Dedico esta dissertação à minha Família

ACKNOWLEDGEMENTS

I want to express my thanks firstly to my thesis advisor Professor Ricardo Gonçalves and co-advisor Professor Celson Lima for the idea for this work, and for all the support they always provided me during the preparation of this dissertation, and secondly to Ruben Costa without whom I could not have done this work.

I also extend my thanks to those who supported me and encouraged me during this process, particularly to my parents, Carlos Figueiras and Clara Figueiras, to my girlfriend Catarina and to all my friends for their endless support and inspiration.

RESUMO

Em apenas alguns anos, a World Wide Web teve um tremendo impacto tanto a nível social como empresarial, ao fazer com que a informação se tornasse instantâneamente disponível. Durante esse tempo, no qual os meios de transporte de informação passaram de físicos a electrónicos, o conteúdo e a codificação de tal informação permaneceu guardado como linguagem natural e apenas identificado pelo seu URL. Este é talvez o maior obstáculo para processos comerciais e empresariais pela Web. Para que tais processos possam ser executados sem intervenção humana, as fontes de conhecimento, tais como documentos, deverão ser facilmente entendidos por máquinas e deverão conter outra informação além do seu conteúdo principal e respectivo URL. A Web Semântica é uma visão de uma futura Web de informação perceptível por máquinas. Nesta Web, será possível que programas examinem facilmente do que tratam estas fontes de conhecimento.

O trabalho apresentado introduz uma estrutura conceptual e a sua implementação para auxiliar a classificação e descoberta de fontes de conhecimento, suportada pela visão introduzida em cima, na qual a informação contida em tais fontes é estruturada e representada através de um vector analítico que calcula a relevância semântica destas fontes de conhecimento em relação ao domínio de interesse de cada utilizador. Este trabalho também realiza o enriquecimento de tais representações de conhecimento, através do uso da relevância estatística de palavras-chave, baseada no conceito clássico do Vector Space Model, e extendendo-a com suporte ontológico, usando conceitos e relações semanticas contidos numa ontologia de domínio específico. Estes vectores semânticos são comparados uns com os outros para obter o grau de similaridade entre diferentes fontes de conhecimento, disponibilizando capacidades de procura e descoberta destas aos utilizadores finais.

Termos Chave: Representação do conhecimento, Web Semântica, *Information Retrieval*, engenharia de ontologias, *Vector-Space Model*.

ABSTRACT

The World Wide Web has had a tremendous impact on society and business in just a few years by making information instantly available. During this transition from physical to electronic means for information transport, the content and encoding of information has remained natural language and is only identified by its URL. Today, this is perhaps the most significant obstacle to streamlining business processes via the web. In order that processes may execute without human intervention, knowledge sources, such as documents, must become more machine understandable and must contain other information besides their main contents and URLs. The Semantic Web is a vision of a future web of machine-understandable data. On a machine understandable web, it will be possible for programs to easily determine what knowledge sources are about.

This work introduces a conceptual framework and its implementation to support the classification and discovery of knowledge sources, supported by the above vision, where such sources' information is structured and represented through a mathematical vector that semantically pinpoints the relevance of those knowledge sources within the domain of interest of each user. The presented work also addresses the enrichment of such knowledge representations, using the statistical relevance of keywords based on the classical vector space model concept, and extending it with ontological support, by using concepts and semantic relations, contained in a domain-specific ontology, to enrich knowledge sources' semantic vectors. Semantic vectors are compared against each other, in order to obtain the similarity between them, and better support end users with knowledge source retrieval capabilities.

Keywords: Knowledge representation, Semantic Web, Information Retrieval, ontology engineering, Vector-Space Model.

TABLE OF CONTENTS

FIGURES INDEX

TABLE INDEX

SYMBOLS AND NOTATIONS

AJAX	**A**synchronous **J**avaScript **A**nd **X**ML
API	**A**pplication **P**rogramming **I**nterface
CoSKS	**CoS**paces **K**nowledge **S**upport
DAML	**D**ARPA **A**gent **M**arkup **L**anguage
DL	**D**escription **L**ogic
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
ERD	**E**ntity-**R**elation **D**iagram
HTML	**H**yper**T**ext **M**arkup **L**anguage
IDE	**I**ntegrated **D**evelopment **E**nvironment
IDF	**I**nverse **D**ocument **F**requency
IP	**I**ntegrated **P**roject
IR	**I**nformation **R**etrieval
JAX-WS	**J**ava **A**PI for **X**ML **W**eb **S**ervices
JSON	**J**ava**S**cript **O**bject **N**otation
KM	**K**nowledge **M**anagement
MVC	**M**odel **V**iew **C**ontroller
OIL	**O**ntology **I**nference **L**anguage
OWL	**W**eb **O**ntology **L**anguage
RDF	**R**esource **D**escription **F**ramework
RDFS	**R**esource **D**escription **F**ramework **S**chema
RPC	**R**emote **P**rocedure **C**all
SECI	**S**ocialization, **E**xternalization, **C**ombination, **I**nternalization
SEKS	**S**emantically **E**nhanced **K**nowledge **S**ources
SGML	**S**tandard **G**eneralized **M**arkup **L**anguage
SPARQL	**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
TF	**T**erm **F**requency
UCD	**U**ML **C**lass **D**iagram
UML	**U**nified **M**odeling **L**anguage
URI	**U**niversal **R**esource **I**dentifier
URL	**U**niversal **R**esource **L**ocator
USD	**U**ML **S**equence **D**iagram
VSM	**V**ector **S**pace **M**odel
WSDL	**W**eb **S**ervice **D**efinition **L**anguage
W3C	**W**orld **W**ide **W**eb **C**onsortium
XML	e**X**tensible **M**arkup **L**anguage

1 INTRODUCTION

1.1 Rationale & Motivation

Knowledge is a key factor in all aspects of today´s industries, and in their effort to keep up on such a competitive environment (Ichijo & Nonaka, 2007). This fact is established by accounting for how much knowledge is gathered and shared between companies in their projects' life spans. Usually, a project achievement strongly relies on a truly exchange of knowledge and competencies. Project information, budgets, human resources lists, time schedules, manuals and guides, video tutorials, material lists, documents and all other forms of knowledge sources are, basically, created or recycled knowledge. From the organisation point of view, knowledge goes through a spiral progression denominated as knowledge lifecycle (Nonaka & Takeuchi, 1995). In this configuration, knowledge is created and nurtured in a continuous flow of conversion, sharing, combination, and dissemination, where all the aspects and contexts of a given organisation, are considered, such as individuals, communities, and projects. Knowledge assets are the basic foundations for sustainable work, collaboration and organization within a company and between different companies, due to knowledge's ability to be recycled, shared and reused between different users and teams and in different formats, saving time and money spent on investigation and information gathering, on behalf of employers and employees.

Today's World Wide Web has had a tremendous impact on such collaboration and knowledge sharing processes by making knowledge instantly and universally available within and across companies. With the rapid increase of knowledge assets on the Web, and during the transition from physical to electronic means for their transmission, the content and encoding of such assets has remained natural human language. The current World Wide Web is, basically, a huge library, or database, of knowledge assets (web pages, documents, video and audio files, etc.) interconnected by a hypermedia of unique links, or identifiers, for each asset, and it acts as an applications and multimedia platform (Goble, et al., 2001). Thus, the World Wide Web can be defined as the *Syntactic Web*, because computers present information (usually in HTML) to human users, without any attempt to evaluate, classify or interpret the outputted information. It is up to the user to decide which of the presented knowledge is relevant (Breitman, et al., 2007). So, the question is: Why leave the process of creation of concepts' mental representations and contextualization through semantic relatedness to humans?

Furthermore, most knowledge assets are only comprised by their main contents and identified by Uniform Resource Locators (URLs), without any knowledge structure or external metadata associated to them, which means that knowledge assets are merely unstructured pieces of data or information. This implies that some kind of knowledge indexation mechanism is needed for

achieving an optimized maintenance and capitalization throughout the entire knowledge lifecycle (Alavi & Leidner, 1999). Specifically, in a project oriented context, teams of professionals can improve their work performance if supported by knowledge indexation and transfer mechanisms as part of a collaborative working environment. Hence, the industrial medium has the need to develop new ways of collaborative work, specifically, electronic collaboration (e-collaboration), supported by effective knowledge management (KM) and indexation mechanisms within their Information Retrieval (IR) tools, thus improving efficiency and enabling added-value ways of operation between collaborative projects' partners (Costa, et al., 2010).

Over the last years, a vision of structured, contextualized and semantically relevant knowledge over the World Wide Web has been developing: the term "*Semantic Web*" (Berners-Lee, et al., 2001) refers to W3C's vision of the Web of linked data. The Semantic Web fosters a common framework that allows contextualized, structured data to be shared and reused across application, enterprise, and community boundaries (World Wide Web Consortium (W3C), 2004). Due to the growing interest in semantic technologies and the advent of the Semantic Web, the above issues and limitations of today's World Wide Web are being addressed, by making computers able to measure semantic relatedness or similarity[1] between concepts or expressions on large-scale information sources, through semantic-based knowledge indexation mechanisms.

The presented work, along with the Semantically Enhanced Knowledge Sources (SEKS) system, approaches one particular issue of the IR area of research: semantic knowledge indexation of knowledge sources in order to retrieve them by its semantic relevance and context within a given *domain* or *environment* (i.e., the organisation itself or a collaborative workspace), improving collaboration between different parties at different stages of a given project life cycle and ensuring that relevant knowledge is properly capitalised in similar situations (Lima, et al., 2010).

1.2 Goals

As referred before, the aim of this work is to formalize knowledge representation techniques for knowledge sources, supported by Semantic Web technologies, developing a contribution for KM and IR areas. In order to achieve this, SEKS system enriches the representation of knowledge sources with knowledge extracted from metadata, annotations and a domain-specific ontology. In this work, the knowledge domain of the used ontology is the building and construction domain and the test-bed for the presented work was collaborative engineering projects.

SEKS is not implemented to be a semantic search engine, but rather a semantic knowledge source indexation and extraction framework. Nevertheless, a search engine-like interface will allow users, that play some role on such collaborative engineering projects, to express their information needs in terms of keywords, but at the same time will use the semantic knowledge regarding the

[1] Semantic relatedness measures are automatic techniques that attempt to imitate human judgments of relatedness

domain of the application, explicit on the domain-specific ontology, to obtain semantically relevant results that often are not achieved in traditional syntax-based search engines, like Google (Google Inc., 2012) or Yahoo! (Yahoo! Inc., 2012). Then the system performs knowledge indexation through the creation of semantic vectors[2] for each knowledge source, and posterior semantic similarity analysis between user queries and sources' vectors (queries are considered pseudo-knowledge sources), under the scope of the Building and Construction domain.

Therefore, the research question for this work can be formulated as: *Will the inclusion of a semantic dimension on knowledge representation techniques allow the creation of better representations of knowledge sources?* The knowledge representation topic is approached in Chapter 2. More specifically, the goals for the presented work are:

- Extract underlying knowledge from main contents of knowledge sources and associated metadata, through the utilization data- and text-mining tools, in order to build a statistic vector of term occurrences and positioning in such sources;
- Given a set of keywords or terms that occur frequently in a knowledge source, or that have a position of relevance in that source (e.g.: term in a document's title), build keyword-based semantic vectors by matching these keywords with ontological concepts;
- Given a set of ontological concepts and respective statistical weights, build a keyword-based semantic vector based on such concepts' semantic relevance, not only within a particular knowledge source, but also their relevance across all sources within the SEKS knowledge source universe;
- Given a keyword-based semantic vector, analyze the relevance of concepts present in such vector in terms of their taxonomic relationship, and build a semantic vector according to concepts related by a hierarchical relation (taxonomy-based semantic vector). This taxonomy-based vector takes into account the families, or bags, of terms. For instance, "*Design Actor*" and "*Architect*" may be seen as taxonomically related, being "*Architect*" a child concept of "*Design Actor*", much in the same way as concepts "*Dog*" and "*Animal*".
- Given a keyword-based or taxonomy-based semantic vector, analyze the relevance of concepts present in such vector in terms of their ontological relations, and build an ontology-based semantic vector. This ontology-based vector is based on the underlying semantic relationships between ontology concepts. For instance, concepts "*Architect*" and "*Building*" may be considered linked by a semantic relation, such as "*designs*" or "*is designed by*".

[2] Semantic vectors are represented as two columned matrixes or vectors: The first column contains the terms that build up the knowledge representation of the source, i.e. the most relevant terms for contextualizing the information within the source; the second column keeps the degree of relevance, or weight, that each term has on the description of the knowledge source.

- Compare semantic vectors using a similarity measure that allows analyzing whether or not those vectors are semantically similar or not. If two vectors are semantically similar, then the knowledge sources associated to them are also similar, in terms of their meaning and context.

1.3 Context of Development

The work presented here was firstly developed for UNINOVA, under the CoSpaces Integrated Project (IP) (CoSpaces Project Consortium, 2010), funded by the European Comission. The aim was to implement a knowledge management system to use on the CoSpaces collaborative workspace, comprised by three main modules: First, the knowledge extraction module, or Miner, would extract data from a project's knowledge source, like databases and knowledge source repositories, mining over text and data to build a statistic representation of available knowledge sources. Secondly a semantic knowledge indexation mechanism, named CoSpaces Knowledge Support (CoSKS) (Lima, et al., 2010) and considered the first development iteration for the presented work, would build semantic representations and contextualize knowledge assets according to actors, projects, issues, meetings, etc. Finally, knowledge assets were presented to users depending on their necessities by the Companion module, an autonomous proactive and reactive software infrastructure (Antunes, 2010). The work developed under the CoSpaces IP led to the creation of several conference approved articles (Lima, et al., 2010; Costa, et al., 2010). CoSKS is the main contributor for this work: CoSKS ontology was modified to create SEKS ontology; most of the ontology interaction methods used on SEKS were implemented for CoSKS; and both architectures are similar.

1.4 Thesis Structure

Besides this introductory chapter, the thesis structure comprises six more chapters as follows:

Chapter 2 – Overview of the Main Areas Involved

This chapter provides an overview over the main areas involved in the presented work, and points the way to this work's goals and requirements.

Chapter 3 – Theoretical and Technical Foundations

A focus on the theory behind concepts, models and mathematical functions behind the SEKS system is presented in this chapter.

Chapter 4 – SEKS Requirements and Conceptual Model

This chapter presents the main requirements established for the presented work, and introduces the conceptual model, taking a close look at functional and architectural visions that support the implementation of this research work.

Chapter 5 - SEKS Implementation

A detailed view of all implementation aspects regarding SEKS system, adopted technologies, ontology, database, processes and architecture is the subject for this chapter.

Chapter 6 – Assessment

This chapter covers examples and assessments made for the presented proof of demonstration, presenting functionalities and results.

Chapter 7 - Conclusions and Future Work

The last chapter synthesizes the presented work, comparing achieved results with goals previously presented, mentions contributions given by the developed research and implemented system, and introduces future work possibilities for further enhancement of the presented work.

2 OVERVIEW OF THE MAIN AREAS INVOLVED

This chapter summarizes the current state of the art of the main areas involved in the presented work, divided into three subchapters, namely: *Knowledge*, *Collaboration & Knowledge Management* and *Information Retrieval*.

2.1 Knowledge

Looking back at human History it is easily realizable that knowledge is the key element behind the evolution of mankind and it has always been captured and passed among people from different parts of the globe, using different techniques and tools, and its growth has always depended on human interaction (Costa, et al., 2010). For example, one can recall that Egyptians used papyrus to store information that could be transformed into knowledge for coming generations; Sumerians used clay tablets for the same purpose; and Alexandria's library was, in a given period of history, probably the largest source of knowledge available on earth, gathered from and shared by many different sources, from Greek to Ancient Muslim authors.

Knowledge is one (if not the) major factor that makes personal, organisational, and societal intelligent behaviour possible (Wiig, 1993). There are many definitions for knowledge and it is a very ambiguous philosophical concept, varying its definition depending on the context it is being approached. Research on the matter defines knowledge either as a state of mind, an object, a process, a condition of having access to information or a capability (Alavi & Leidner, 1999). In the organizations' point of view, knowledge is embedded in and carried through multiple entities including organization culture and identity, routines, policies, systems and documents, as well as individual employees. Specifically, knowledge is a justified belief that increases an entity's capacity for taking effective action (Nonaka, 1994). For the sake of clarity, it is important to distinguish three basic concepts commonly referred throughout the presentation, namely: data, information and knowledge. Data[3], in its most simple context, represents stored information to be examined and used. Information is comprised by facts or details about a specific topic, presented within a context. Knowledge is the awareness and cognitive skills acquired through experience or education.

The knowledge structure model used on this work is based on the above knowledge categorization and considers that knowledge goes through an evolving spiral when it is transformed from tacit knowledge (the inner knowledge, intangible) to explicit knowledge (visible, the tangible one) (Nonaka & Takeuchi, 1995). Explicit knowledge is described in formal, systematic details, and can be expressed in words and numbers, whereas tacit knowledge is described as a knowledge

[3] Data has also a particular meaning in the Semantic Web's context, which will be explained in Chapter 3.

built on experiences, and it is difficult to capture and share, because it is highly personal and not easily visible or expressed.

Such knowledge evolution process is represented by the SECI (Socialization, Externalization, Combination and Internalization) Model, shown in Figure 2.1, which comprises the four transformation processes involving the two knowledge types, namely: Socialization (from tacit to tacit), Externalization (from tacit to explicit), Combination (from explicit to explicit), and Internalization (from explicit to tacit).

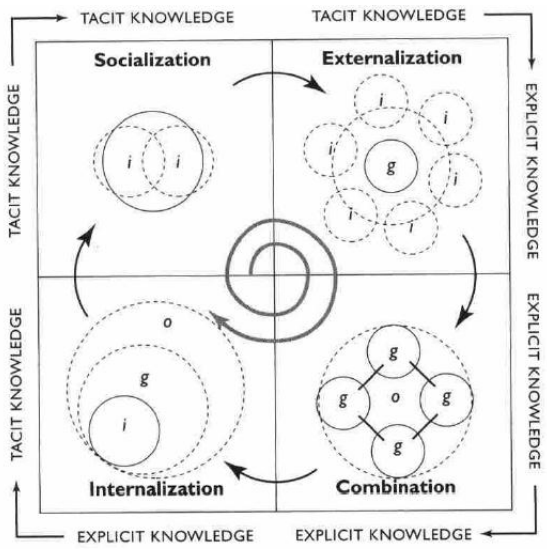

Figure 2.1: SECI Model (Nonaka & Takeuchi, 1995)

The presented work takes into account this dynamic dimension of knowledge represented in the SECI Model, by constantly assessing knowledge update and creation processes, and building knowledge representations according to such assessment.

2.2 Collaboration and Knowledge Management

Collaboration may be defined as *the partnering of activities, knowledge and assets by multiple workers in a dynamic environment, aiming at gaining business and expertise advantage* (Mathew, 2002). Collaboration occurs when people within an organization or across different organizations implement something together through joint effort and decision making, and share ownership of the final product or service (Linden, 2002).

In the context of product development, it is possible to define three levels of joint endeavour intensity: coordination, cooperation and collaboration (Kern & Kersten, 2007; Camarinha-Matos & Afsarmanesh, 2006), as shown in Figure 2.2. Which type should be chosen depends on the required interaction intensity caused by the need of a specific development situation. For example, the required interaction intensity increases with increasing process-related interdependencies between

the defined subtasks, and with increasing necessity of integrating the knowledge and experience of the development partners (Costa, et al., 2010).

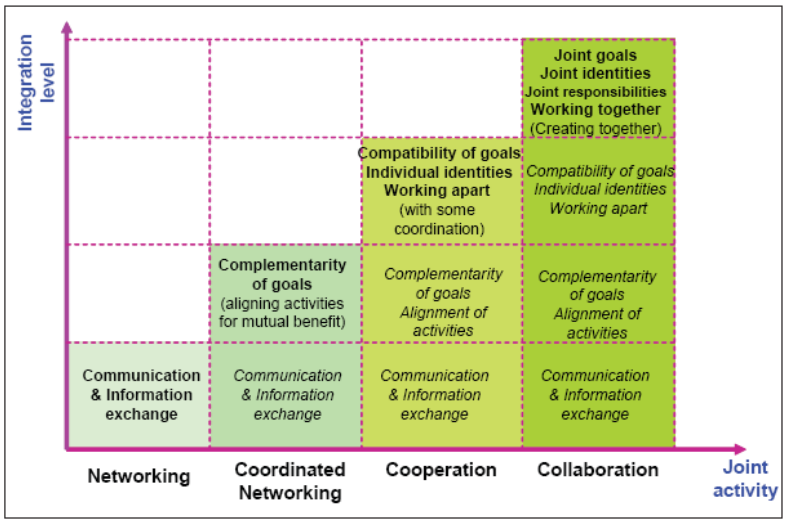

Figure 2.2: Levels of joint endeavor (Camarinha-Matos & Afsarmanesh, 2006)

Knowledge sharing during collaboration processes is essential to its proper capitalization. On one hand knowledge sharing is heavily dependent on technical capabilities and, on the other hand, since the social dimension is very strong during collaboration, there is also an increased need to take into account how to support the culture and practice of knowledge sharing. For instance, issues of trust are critical in collaborative enginnering projects, since the distribution of knowledge and expertise means that it becomes increasingly difficult to understand the context in which the knowledge was created as well as to identify who knows something about the issue at hand, and so forth.

The sharing and capitalisation of knowledge and expertise within a project is highly amplified through the use of the Internet, since it can reach a bigger audience in a more effective way. With the exponential development of the Internet, most organizations and companies have moved their approach to e-business or e-collaboration where the internet is the main platform to interact with each other and with their customers (Antunes, 2010). Two new and intertwined concepts have emerged in order to face the new challenge posed by collaboration over Internet: *Collaborative environment* and *collaborative workspace*, with the difference between them being that the latter is a particular case of the first, directed only to work- or project-based collaboration. Collaborative environments are virtual environments in which individual participants or teams, from geographically dispersed locations, can work in a common project and collaborate with each other through a single combined online environment as if they were virtually in the same room, with the aim of solving the challenge of geographically spread collaboration among organizations.

Over the process described by the SECI model within collaborative environments, knowledge is: (i) transformed in a evolving way over time; (ii) managed around specific problems

and needs of the collabration partners, in order to be proper capitalised in terms of the collaboration context and (iii) enriched with the appropriate support of knowledge organization tools and models (Costa, et al., 2010). Thus, knowledge management (KM) is of primary importance due to the myriad of created and combined knowledge which has grown to unconceivable levels, materialized with the introduction of collaborative environments over the Web. Such growth brings other issues, such as the lack of structured and organized knowledge on the Web, the ambiguity of contexts attributed to a concept on the panoply of Web pages and Web documents existent and the scattered distribution of knowledge over databases, documents, emails, etc.

KM is a set of techniques, tools or technologies that identify, organize, collect, present, and create knowledge from a collection of data or individuals' expertise, and facilitate the communication between knowledge creators and receivers, enabling them to use the achieved knowledge to make supported decisions or take well-thought actions (Rezende & Souza, 2007). There are several KM models or cycles (Wiig, 1993; von Krogh, 1998; Nonaka, et al., 2001). However, research over an integrated and unified KM model, built according to previous works, has realized the Integrated KM Model (Dalkir, 2005), shown in Figure 2.3. The presented work is based on this knowledge management model, in order to maintain knowledge representations of knowledge sources updated and contextualized in terms of their consistency with the source itself and with all other sources present in the system's knowledge source repository.

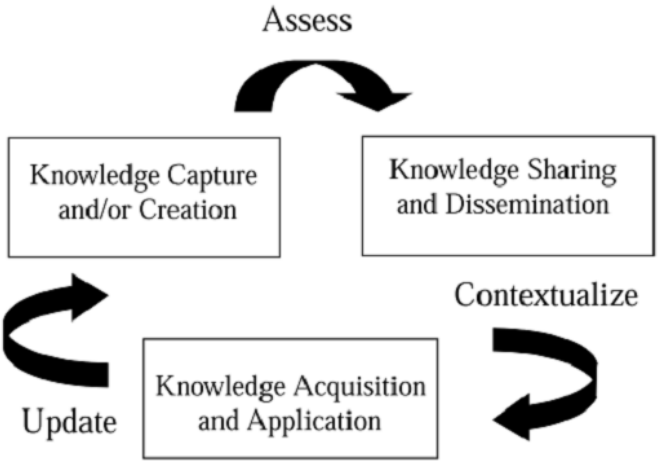

Figure 2.3: Integrated Knowledge Management Model (Dalkir, 2005)

After knowledge creation and capture, knowledge has to be *assessed* to verify its integrity, relevancy and validity according to the context of use. Only then is knowledge ready to be shared or disseminated, according to the users' needs. The next process is responsible for *contextualizing* knowledge. Only contextualized knowledge can be properly acquired and applied. For instance, if one gives a book of Chemistry to an architect, it won't be of much use. Knowledge must be delivered according to users' contexts. The final process is knowledge *update*: The knowledge management cycle is then reiterated as users understand and decide to make use of content. Users

will validate usefulness, will signal when it becomes out of date or when this knowledge is not applicable and will help validate the scope of the content or how generalizable the best practices and lessons learned can be. They will also, quite often, come up with new content, which they can then contribute to the next cycle iteration (Dalkir, 2005). One important aspect to keep in mind is that an organization cannot share all of the knowledge that it produces: there is a great amount of knowledge assets within a company that is considered to be confidential or private. Such confidential information must be preserved on the act of collaboration, and specially in knowledge sharing. This means that, for companies willing to collaborate and as a consequence, to share knowledge, privacy issues must be taken into account, and also to enable a secure collaboration in order to build trust between organizations, must be implemented by security and trust policies. There are mechanisms that manage security and privacy on collaborative environments but, as referred before, they are not part of the developed work.

The main purpose for the use of KM technologies in the presented work is to provide a way for knowledge structuration and indexation, which are processes linked to the next subject to be presented: the Information Retrieval area. In some sense, KM technologies aim to provide better Information Retrieval tools, by organizing knowledge in ways in which they can be easily found and retrieved to the user. The indexation process output is a knowledge representation of a knowledge source. There are many knowledge representation techniques, divided into hierarchical-based, rule-based and logic-based representations but they will not be subject to further explanation on the presented work. Sufficed it is to say that this work uses a knowledge representation approach based on an ontology, semantic vectors and the Vector Space Model[4], all subjects of Chapter 3.1.1.

2.3 Information Retrieval

IR can be defined in a broad set of senses and contexts. However, as an academic field of study, IR might be defined as (Manning, et al., 2008):

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies a need for information from within large collections (usually stored on computers).

In the past, IR was an activity that only a few people engaged in, such as librarians, paralegals, and similar professional searchers. With the advent of the Internet and the World Wide Web, hundreds of millions of people started engaging in IR in a day-to-day basis by using a web search engines or searching their email. IR is the dominant form of information access, overtaking traditional database style searching, which requires specific identification. IR can also solve other

[4] The Vector Space Model will be further explained in Chapter 3.1.1.3. For now, it is sufficed to say that the Vector Space Model is a mathematical model that represents documents as vectors in a vector space.

knowledge problems. The term "*unstructured data*" refers to data which does not have clear, semantically overt, easy-for-a-computer structure. It is the opposite of *structured data*, the canonical example of which is a relational database, of the sort companies usually use to maintain product inventories and personnel records (Manning, et al., 2008).

The field of IR also covers supporting users in browsing or filtering collections of knowledge sources, or further processing a set of retrieved sources. Given a set of knowledge sources, *clustering* is the task of coming up with a good class grouping such sources based on their contents. Given a set of topics, textual information needs, or other kind of collections, *indexation* is the task of deciding which class or classes, if any, each set of sources belongs to.

Three research works in the IR area, or that presented a contribution for the IR area, were used both as inspiration and comparison objects. (Castells, et al., 2007) propose a system focused in the IR area and based on an ontology, just as in the presented work's case. It also uses the TF*IDF algorithm, matches knowledge sources' keywords with ontology concepts, creates semantic vectors and uses the Euclidian distance to compare created vectors. A major difference between this IR system and the presented work is that ontology relations are not considered, nor the hierarchical relations between concepts (taxonomic relations). (Li, 2009) present a way of mathematically quantifying such hierarchical or taxonomic relations between ontology concepts, based on relations' importance and on the co-occurrence of hierarchically related concepts, and reflect this quantification in semantic vectors of knowledge sources, as presented in Subchapter 5.4.1. This work's aim is to create an IR model based on semantic vectors to apply over personal desktop knowledge sources, and has no relation to Web IR applications, as is the case of SEKS system. Nevertheless, this work has some manual input parameters that the presented approach tries to insert automatically.

Finally, (Nagarajan, et al., 2007) propose a knowledge source indexation system supported by Semantic Web technologies, just as in the presented work. They also propose a way of quantifying ontological relations between concepts, and represent that quantification in semantic vectors of knowledge sources, as explained in Subchapter 5.4.1.4. There are some differences between this work and the presented approach, though. Firstly, (Nagarajan, et al., 2007) do not focus their work on a contribution to the IR area, not providing any means of knowledge source extraction or retrieval. Secondly, this work does not distinguish between taxonomic and ontological relations, as SEKS system does. SEKS is based on the main goal of (Castells, et al., 2007), but tries to improve it, by applying an hybrid approach for the taxonomic and ontological relations' quantification processes proposed by (Li, 2009) and (Nagarajan, et al., 2007).

3 THEORETICAL AND TECHNICAL FOUNDATIONS

This chapter provides a summarized theoretical background over all relevant subjects used to design and develop the SEKS system. First, a brief description of the technical foundations supporting the presented work is given. Afterwards the focus will be directed to the mathematical bases behind the semantic knowledge source indexation and search mechanisms developed for SEKS system.

3.1 Technical Foundations

This subchapter presents the technical foundations under which this work was developed. The first subject to be approached is the Semantic Web. Afterwards, a brief introduction over Knowledge Extraction mechanisms and an explanation about homologous and non-homologous concepts are given.

3.1.1 The Semantic Web

The term "*semantics*" is defined as the meaning or relationship of meanings of a sign or between a set of signs (Kashyap, et al., 2008). Other references define semantics as the study of the meaning of words in terms of their context (Larousse, 1994). Nowadays there is still no formal definition for the Semantic Web, and it often leads to different definitions of it, according to different groups of individuals. Nevertheless, the term "Semantic Web" was originally created by the World Wide Web Consortium (W3C) and introduced on the article "The Semantic Web" (Berners-Lee, et al., 2001). Two different definitions for the Semantic Web from relevant sources are presented below:

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries (World Wide Web Consortium (W3C), 2011).

The Semantic Web is the extension of the World Wide Web that enables people to share content beyond the boundaries of applications and websites (semanticweb.org, 2011).

Two concepts are intimately connected to the Semantic Web subject: *Linked Data* and *Web of Data*. Linked Data refers to the use of the Web to create typed connections between data from different sources, by publishing structured data on the Web and adding connections between such data (Bizer, et al., 2009). Specifically, Linked Data defines a set of best practices for publishing and linking information online. Web of Data is an interchangeable definition for the Semantic Web. It is the objective of the Semantic Web, because the Semantic Web can be seen as a set of technologies and standards to achieve or realize the Web of Data. In other words, if Linked Data is published by using Semantic Web technologies and standards, then the result would be a Web of Data (Yu, 2011). These concepts are the basis for the layers of knowledge enrichment used on the

Semantic Web. The layered structure for the Semantic Web is shown in Figure 3.1 (Berners-Lee, et al., 2001) (Antoniou & Harmelen, 2004).

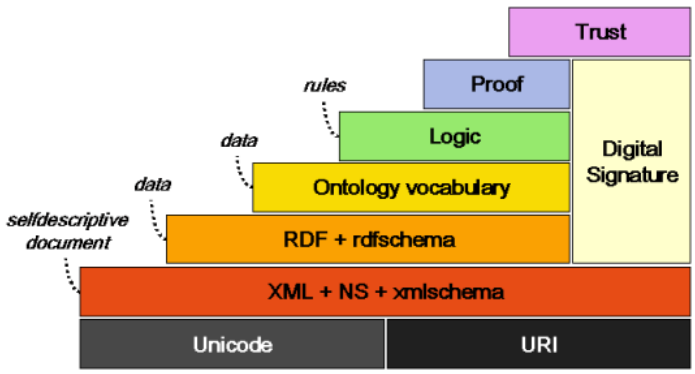

Figure 3.1: A layered approach to the Semantic Web (Berners-Lee, et al., 2001)

Supporting documents or, in this work's case, knowledge sources, are *Unicode document encoding* and *document URIs*. Besides their main contents and X Markup Language structures, *selfdescriptive documents* comprise document *annotations* and *metadata*. Metadata and annotations refer to domain- or application-specific embedded descriptions of knowledge sources, which will be used over the Semantic Web to annotate such sources, and are represented by XML-based languages. Above documents is the *RDF and RDFS layer*. RDFS provides modeling primitives for organizing RDF resources into hierarchies, or taxonomies. Schemas, like RDFS, are a particular case of structured metadata which may contain semantic information (Kashyap, et al., 2008). The *Ontologies layer* lies down on top of RDF and RDFS. Ontologies refer to the underlying vocabulary and semantics of the metadata annotations, allowing the representation of more complex relationships between Web knowledge sources. Collections of domain-specific ontological concepts may extract underlying domain-specific contexts and meanings from main contents of such Web sources.

The *Logic layer* is used to enhance the ontology language further and to allow the gathering of application- and domain-specific declarative knowledge, by providing semantic rules that will help in the semantic reasoning process. The Logic layer is the *Proof layer*, comprises the deductive process and the representations of proofs and compatibilities in lower-level Web languages, and validation of such proofs. On top of the structure is the *Trust layer*, which is developing through the recurrent use of digital signatures and other types of knowledge, supported by trusted agents, certification agencies and user groups (Antoniou & Harmelen, 2004). As referred on Chapter 2, the Trust layer is built upon the confidence on knowledge shared between collaborating companies. On one hand, some knowledge is confidential and private within a company, so it must not be shared; on the other hand, sharing must be reciprocal, which means that both sides involved in sharing processes must give and take back equal amounts of relevant knowledge.

In order to achieve this layered perspective on the Semantic Web, two conditions must be satisfied: Downward compatibility and upward partial understanding. Downward compatibility implies that applications oriented to the use of a specific layer, for instance, the ontology layer, must be able to take full advantage of knowledge from lower layers, for example the RDF and RDFS layer. Upward partial understanding entails that although an application is using a lower layer of information, like RDF and RDFS, it must be capable of extract some knowledge from upper layers, as the ontology layer, by using only the primitives of RDF and RDFS present within ontologies, and disregarding other upper layer primitives.

Semantic protocols used to infer over the desired logic and proof and, at the same time, provide query support to semantic markup languages, like RDF and OWL, are already available: SPARQL for RDF (World Wide Web Consortium (W3C), 2008) and SPARQL-DL for OWL-DL (World Wide Web Consortium (W3C), 2011). The presented work uses SPARQL-DL indirectly, through a semantic framework that enables query execution over taxonomies and ontologies, Jena Semantic Framework (The Apache Software Foundation, 2011), so it does not deserve much more attention in this presentation. More insights on this semantic framework will be given on Chapter 5.

3.1.1.1 Annotations, Taxonomies and Ontologies

Both annotations and ontologies are, as previously referred, based in structured languages, like XML. In fact, structured languages are intrinsically related to the advent of the Semantic Web, as can be seen in because such languages are designed to be both human- and machine-readable, and can contain semantic content and relations associated to data.

Annotations are comprehended by structured information embedded in knowledge assets that define contexts and meanings to knowledge assets main contents' unstructured data. Often they are based on XML or RDF. Taxonomies are hierarchical structures that organize concepts in *bags*, under a specific domain, and are normally represented in RDF. Concepts are related by kin, or familiarity, meaning that existing relations define concepts' parenthoods by stating that some concepts are particular forms of other concepts ("Architect" is a particular form of a construction environment "Actor"). Ontologies are taxonomies that enable machine assertions or claims about relations between concepts that are not taxonomically related, and are represented using the Web Ontology Language, or OWL.

The first step towards semantic structured knowledge representation languages was the international standard SGML (Standard Generalized Markup Language), which aimed to define different information presentation methods, platform-independent, and enabling human- and machine-readability. HTML, which is based on SGML, is the standardized language for Web pages nowadays, and it is completely data presentation-driven, which means that it is not designed to

comprise structural information, like annotations. It is a purely a document format and display language.

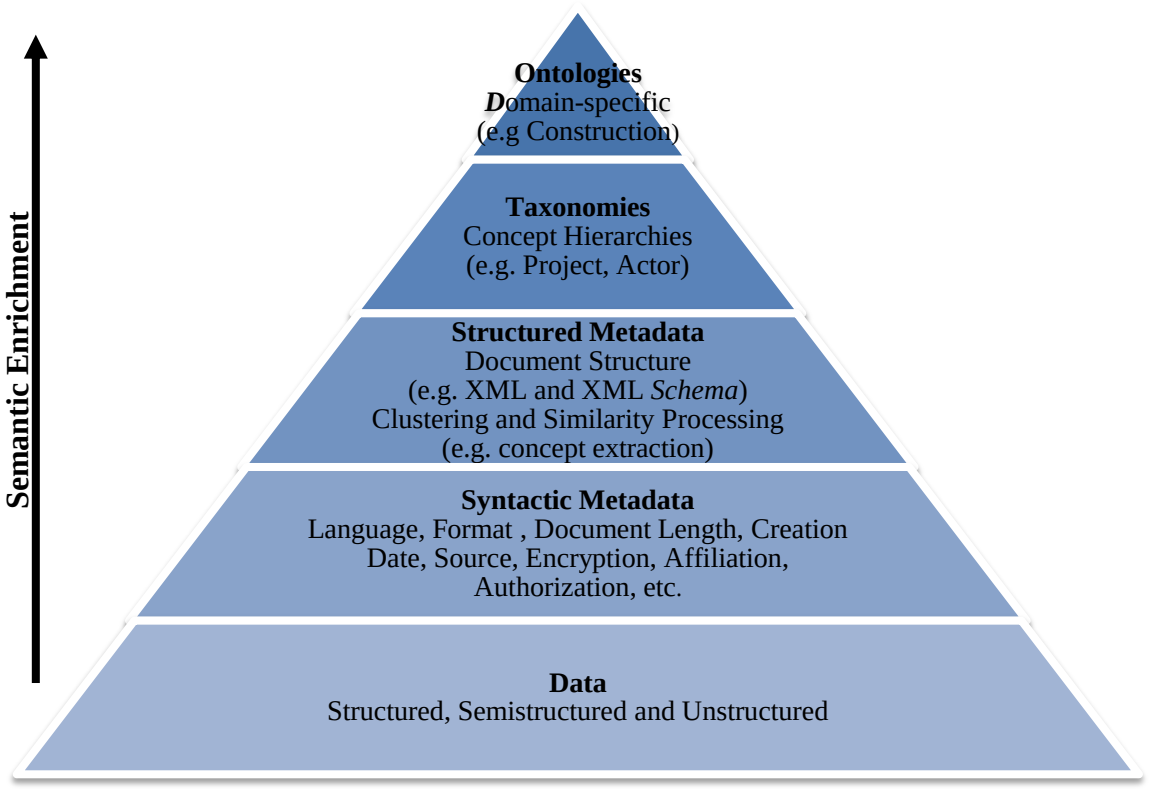

Figure 3.2: The pyramid of Semantic Web Content (adapted from (Idehen, 2010))

XML development was driven by shortcomings of HTML. Both HTML and XML use a data representation model supported by *tags*. Both permit to structure information, through the introduction of tags nested within other tags. Both are markup languages, meaning that they allow the distinction between content and underlying structured data, describing the role that content plays, and both are human- and machine-readable. In contrast to HTML, XML documents are far more easily readable by machines, because every piece of information within it is described, and underlying relations are also defined through the nesting structure.

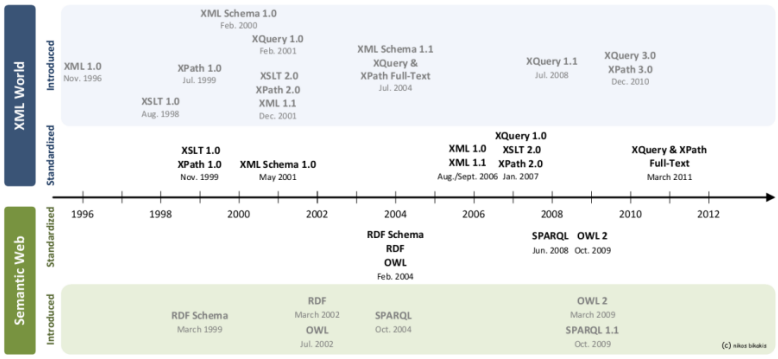

Figure 3.3: XML and Semantic Web Chronology (Bikakis, et al., 2012)

It is this evolution in markup languages that serves as foundation for the extraction of meaning and context from unstructured information behind the Semantic Web, and not the mentioned languages themselves. Over time, other languages and information structures were implemented to bring the Semantic Web's concept to what it is today: RDF and OWL languages and taxonomic and ontological structures, described below.

3.1.1.1.1 Taxonomies and RDF

The Semantic Web introduces new usage possibilities to recently developed XML-based languages. The next iteration of XML-based languages towards the Semantic Web vision is RDF (Resource Description Framework). RDF is a data-modeling language based on XML syntax which defines *statements* about *resources* and *relations* among them.

Resources define domain concepts, meaning "things" belonging to a certain domain. For instance, "Actor", "Project phase" or "Architect" are concepts, or resources, which have particular meanings in the construction environment domain. Every resource has a Universal Resource Identifier (URI) that identifies it. In RDF, resources can be arranged in taxonomic arrays of classes, and respective sub-classes, of concepts using the nesting ability of XML. XML does not offer any explicit way to represent the hierarchy of information in a taxonomic fashion, and it is up to each application to interpret the nesting of tags.

RDF permits machines to reason the hierarchical taxonomy of concepts regarding a certain domain. A class A is said to be a subclass of another class B if every resource in A is also included in B. For instance, one can easily see that "Architect" is a sub-class of "Actor", meaning that architects are one type of actor in the construction environment. In such cases one can state that the concept "Architect" is an *instance* of "Actor". Properties, or relations, are a particular type of resources, they describe relations between resources, for instance, the relation "is a". Using again the example given above, one can comprehend that an "Architect" "is an" "Actor" in the construction domain's context. RDF properties are also identified by URIs. The idea of using URIs to identify concepts and relations because it introduces a global unique naming scheme, which reduces homonym issues in data representation (Antoniou & Harmelen, 2004).

Statements assert the properties of resources. Each RDF statement is built according to a triplet form, comprising a subject (resource), a predicate (property or relationship) and an object (property value). Statement objects can either be resources or *literals*, which are atomic values that can be integers, strings, real numbers, etc. (Baker & Cheung, 2007).

Nevertheless, RDF is domain-independent, meaning that no assumptions about a particular domain are made. It is still up to the human user to define the assertions about the underlying domain. This terminological definition is made using a schema language denominated RDF Schema (RDFS). RDFS defines domain-specific vocabulary, specifies which properties apply to

which type of resources and what values are permitted describes the domain-specific connections between resources and implements the concepts' class hierarchy. This means that RDFS makes semantic information, represented in RDF, machine-accessible.

Even so, RDF, and consequently RDFS, present problems to the Semantic Web vision of linked data. RDF only allows binary properties, meaning that a statement is always comprised by one subject, one property and one object. Much of the times humans use predicates with more than two resources as arguments. These can be simulated by binary statements, but such simulation is not simple and does not look natural. Another issue is property handling. On RDF, properties are regarded as a special type of resource, which means that they can be used also as statements' objects or subjects. This approach can be flexible, but it entails modeling complexity and can sometimes be confusing to human modelers or users. This issue is also verified in statements. RDF allows statements about statements, which introduces complexity levels not suited for a foundational layer for the Semantic Web.

Recapitulating, RDF and RDFS are widely- and commonly-used standards in the Semantic Web context, but neither is expressive enough to provide the formal knowledge representation support that is intended for computer processing. While RDF and RDFS allow representations of *some* ontological knowledge, like vocabulary organization in typed hierarchies: subclass and subproperty relationships, domain and range restrictions and instances of classes, their expressivity is deliberately limited: RDF is limited to binary ground predicates and RDFS is limited to subclass and property hierarchies. Other Semantic Web's important features are missing (Antoniou & Harmelen, 2004):

- Properties' local scope. RDFS defines the range of a property as global, meaning that one cannot declare range restrictions that apply only to some resources.
- Disjoint classes. RDFS only permits statements about subclass relationships ("male" and "female" are subclasses of "person"), not allowing the definition of disjoint or contrary classes ("male" and "female" are disjoint concepts).
- Boolean combinations of classes. RDFS does not comprise Boolean interactions between classes, which entails that classes cannot be united, intersected or complemented with each other, to form new classes.
- Cardinality restrictions. RDFS does not allow the placement of restrictions on how many distinct values a property can or must have.
- Properties' special characteristics. RDFS does not comprise *transitive*, *unique* or *inverse* characteristics over properties.

Hence, a richer ontology language that provides the above features is needed. More profoundly, the number of particular use-cases for the Semantic Web that would require much more expressiveness than provided by RDF and RDFS led to an initiative between research groups both in the United States of America and Europe to define a richer ontology modeling language. DAML+OIL (DARPA Agent Markup Language + Ontology Inference Language) (Antoniou & Harmelen, 2004). This language was the starting point for the definition of the W3C Web Ontology Group's (World Wide Web Consortium (W3C), 2001) Web Ontology Language, or OWL, the next subject to be discussed.

3.1.1.2 Ontologies and OWL

The term "*ontology*" has its origins defining a subfield of philosophy, the study of the nature of existence, meaning the identification of types of things that exist, and how to describe them. However, in the Semantic Web's point a view, "*an ontology*" can be described as a formal description or conceptualization of a domain of knowledge, consisting of a collection of classes and the relations between these classes. Classes denote domain concepts and are arranged hierarchically, just as in taxonomies. Ontologies provide a shared understanding of a specific domain. Apart from subclass relationships, ontologies contain other types of knowledge, such as properties (e.g. *A* "builds" *B*), value restrictions (e.g. "only *A* can build *B*"), disjointness statements (not available in RDF and RDFS), and specification of logical relationships between objects (e.g. "*B* is built by at least ten *A*'s").

OWL is the W3C current standard for representing ontologies on the Web, and aims to solve RDF and RDFS's lack of expressiveness issues mentioned above. Furthermore, OWL's main requirements are a well-defined syntax, a formal semantics, convenience of expression, efficient reasoning support and sufficient expressive power. A well-defined syntax is important for obvious reasons: enable better machine-processing over knowledge. RDF, RDFS and OWL all comply with this condition. On the other side, the syntax should easily readable by human users, condition poorly satisfied by any of the mentioned languages. Nevertheless, condition is not absolutely necessary, for ontologies will be developed using ontology development tools, for instance Protégé ontology development environment (Stanford Center for Biomedical Informatics Research, 2011).

A formal semantics defines the precise meanings of knowledge, which means that such meanings do not refer to subjective intuitions or individual interpretations. Knowledge under a domain-specific ontology must not be subject to different descriptions. Semantics is a prerequisite for reasoning support, because it allows one to reason about specific knowledge under a specific domain. Regarding particularly ontological knowledge reasoning, it is possible to infer about (Antoniou & Harmelen, 2004):

- Class Membership. If x is an instance of a class A, and A is a subclass of B then x is also an instance of B.
- Class Equivalence. If a class A is equivalent to class B, and class B is equivalent to class C, then A is also equivalent to C.
- Knowledge Consistency. If x is an instance of a class A and A is a subclass of the union $B \cap C$ and also subclass of D, then, considering B and D as disjoint classes, the ontology has an inconsistency regarding the knowledge rules already imposed, because A has the instance x in it, but due the disjointness of B and D, A should be empty.
- Knowledge Classification. If certain property-value pairs are a sufficient condition for membership in class A and if an individual x satisfies such condition, then x must be an instance of class A.

The panoply of requirements for an ontology language, specially reasoning support and the combination between an expressive language, as RDFS, and the power of description logic, led W3C's Web Ontology Group to define OWL as comprised by a full logic-driven language with a powerful expressiveness, OWL Full, which fulfills the complete set of requirements stated above, and two sublanguages, geared towards a better computational efficiency and to be user-friendly, not provided by OWL Full: OWL DL and OWL Lite (Yu, 2011) (Antoniou & Harmelen, 2004). To regain the lost computational efficiency of OWL Full, OWL DL (Description Logic) was created. OWL DL restricts how OWL and RDF tags, or constructors, may be used. Although OWL DL is not fully RDF and RDFS compatible, it permits efficient reasoning support. Other iteration to OWL regarding efficiency and towards being more user-friendly is OWL Lite. It excludes enumerated classes, disjointness statements and arbitrary cardinality. Although it has a limited expressivity, it is easier to implement and to understand than OWL DL. All OWL languages use RDF and RDFS for their syntax and their instances are declared as in RDF. OWL classes, datatype properties and object properties are specializations of their RDF counterparts.

The SEKS project's ontology is an OWL DL ontology, because both good expressiveness and reasoning efficiency are needed for semantic knowledge source indexation. The SEKS project's ontology is an OWL DL ontology, because both good expressiveness and reasoning efficiency are needed for effective semantic knowledge source indexation. The ontology is used by SEKS system to infer about underlying semantic relations between ontological concepts to achieve semantically enhanced knowledge representations of knowledge sources and user queries.

3.1.1.3 Semantic Vectors and the Vector Space Model

The Vector-Space Model (VSM) (Salton, et al., 1975) is a mathematical model. A mathematical model can be defined as a consistent mathematical structure designed to correspond to some

physical, biological, social, psychological, or conceptual entity (Dubin, 2004). VSM was developed for the SMART IR system by Gerard Salton and his colleagues (Salton, Wong, & Yang, 1975). SMART system was pioneer in the development of many of the concepts that are still in use in modern search engines (Manning, et al., 2008).

VSM formulates and approaches the *statistical semantics hypothesis*: if statistical patterns are applied over human syntactic word formation and natural language term usage, then it is possible to understand the underlying meaning behind that usage (Turney & Pantel, 2010). The statistical semantics hypothesis is supported by several more specific hypotheses that are relevant to the context of the presented work*: bag of word hypothesis*, *distributional hypothesis*, *extended distributional hypothesis* and *latent relation hypothesis*.

The bag of words hypothesis states that one can estimate the underlying relevance of knowledge sources in relation to a query by representing both sources and queries as bags of words. Hence, the frequencies of occurrences of a query's words in knowledge sources tend to represent the relevance of each source in relation to the query. This hypothesis is the building block that supports the application of VSM in IR (Salton, et al., 1975). The distributional hypothesis affirms that words or expressions that occur in similar contexts tend to have similar meanings (Harris, 1954) (Sahlgren, 2008). The distributional hypothesis enables the application of the VSM in word similarity measures. The extended distributional hypothesis is based on the concept that patterns that co-occur with similar pairs of expressions or words also tend to have similar meanings. As the name implies, this hypothesis is an extension to the distributional hypothesis. Pattern similarity can be used to infer that one sentence is a paraphrase of another (Lin & Pantel, 2001). Finally, the latent relation hypothesis states that pairs of words or expressions that co-occur in similar patterns tend to have similar semantic relations (Turney, 2008).

Through the application of the above hypotheses, VSM has several attractive properties for Semantic Web applications (Turney & Pantel, 2010). VSMs use automatic knowledge extraction over a given corpus, which requires lesser computation power than other approaches to semantics. VSMs are considered a very powerful tool when measuring the similarity of meaning between words, phrases, and documents. Most search engines use VSMs to measure the similarity between a query and a knowledge source (Manning, et al., 2008).

According to the statistical semantics hypothesis, the idea behind VSM is to represent each knowledge source in a collection as a point in a space (a vector in a vector space). Queries are represented as points in the same space as knowledge sources (Queries are considered pseudo-knowledge sources). When measuring distances between vectors in the semantic relevance vector space, vectors that are close together in this space are semantically similar and vectors that are far apart are semantically distant. In this way, knowledge sourcess are sorted in order of increasing

distance (decreasing semantic similarity) from the query and then presented to the user (Turney & Pantel, 2010). Formally, VSM-based IR methods mainly use three types of matrixes as output: term-document, word-context and pair-pattern matrixes (Turney & Pantel, 2010).

SEKS system uses VSM to realize knowledge representations of both knowledge sources and queries, and to define a common format between these representations, allowing comparisons between them. Specifically, SEKS uses approximations to term-document vectors in its methods to build statistic and semantic vectors. Term-document matrixes (Salton, et al., 1975) are used to measure document similarity. The matrix's row vectors correspond to terms or expressions and the column vectors correspond to documents, with the document vector representing the corresponding document as a bag of words.

In the presented work, statistic vectors are an approximation to term-document matrixes, built with extracted words and expressions from knowledge sources and with a quantification of the relevance of those expressions in the source, also called weight. The difference between the traditional term-document matrixes and SEKS system's statistic vectors is that instead of having a column for each knowledge source and a row for each expression, the matrix has two rows: one with expressions, or keywords, and another with the respective relevance quantification for each expression. The whole matrix is a statistic representation of the knowledge source.

Semantic vector models include a family of related models for representing concepts with vectors in a high dimensional vector space (Widdows & Ferraro, 2008). Semantic vector creation is supported by the VSM and it is the basis for the presented approach because it implements the extraction of knowledge and meaning from knowledge sources and the agglomeration of this knowledge in a matrix form, better suited for mathematical applications than the raw text form of documents. Furthermore, semantic vectors present the following advantages (Widdows & Ferraro, 2008):

- They can be built using entirely unsupervised distributional analysis of free text.
- While they involve some nontrivial mathematical machinery, they make very few language-specific assumptions (e.g., it is possible to build a semantic vector model provided only that one has reliably tokenized text).
- Similar techniques have been used in other areas, e.g., for image processing.
- The ease with which very simple distributed memory units can collaboratively learn and represent semantic vectors has been noted for its potential cognitive significance.
- Being strongly distributional and associative in character, they have complementary strengths to some of the more traditional formalist and symbolic semantic techniques such as those based on propositional logic and lambda calculus.

The presented work again uses an approximation to the term-document matrix concept but, as in the case of statistic vectors, semantic vectors are matrixes with two rows that represent knowledge within a knowledge source. The difference is, instead of expressions that occur in a particular source, semantic vectors have a row for semantic concepts that comprise a large number of expressions or words. For instance, expressions like *architecture*, architectonics, building design or urban planning may refer to a semantic concept, in other words, a concept that extends its meaning to these expressions. In this case, the concept of *architect* may be considered to embrace all of the previous expressions in its context.

3.1.2 Knowledge Extraction

Knowledge extraction is usually a process embracing three stages: word extraction, regular expressions filtering, and statistic vector creation. Word extraction is the process in which words and expressions are extracted from a knowledge source and divided by data- and text-mining techniques. Text mining corresponds to the extension of more traditional data mining approach to unstructured textual data and is concerned with various tasks such as extraction of information implicitly contained in collections of knowledge sources, similarity-based structuring and visualization of large sets of texts (Rajman & Besançon, 1998). Another responsibility of text mining tools is regular expression filtering, which is the process of removing frequently occurring terms that have no relevance for the context of a particular source, but normally appear several times in all knowledge sources, such as grammatical articles (*the*, *a*/*an*, *some*), pronouns (*mine*, *yours*, *me*, *she anybody*, etc.), prepositions (*for*, *in*, *under*, *toward*, etc.) or adverbs (*much*, *few*, *quite*, *slowly*, etc.).

This process is performed by noise reduction and filtering procedures, such as stemming filters, which cut words to their stem forms (e.g. the stem form of *painting* is *paint*). The last stage, statistic vector creation, is the process that builds the statistical representation of a knowledge source, through the analysis of each term extracted from the source, in terms of its frequency, emphasis and position within the corpus of such source. The statistic vector is organized in matrix form and is composed by the extracted terms, or keywords[5], and by the statistical weight of each keyword within the knowledge source, based on the frequency analysis introduced above. This vector of keywords and respective weights is represented by an approximation to the term-document matrix vector, which is denominated statistic vector. This vector is the main input parameter for the SEKS system.

There are many state-of-the-art data-mining and text-mining frameworks and techniques available. Knowledge extraction techniques are the basis for IR, supporting syntactical comparison

[5] Throughout the text, the author uses the words “keyword” and “term” in a way that may appear ambiguous. The word “keyword” is applied whenever the author means the terms extracted from the document. The word “term” is applied whenever a reference to the term-frequency vector is used; both mean the same thing in different contexts.

between users' queries and words or expressions contained in knowledge sources. Most search of today's Web search engines function according to this principle. The software used to create SEKS statistic vectors was RapidMiner (Rapid-I GmBH, 2011), which is a knowledge extraction tool in the form of an easy-to-use IDE, and that builds statistic vectors of knowledge sources over sources' corpora. RapidMiner has several filter tools, such as stemming and lemmatization filters, and provides access to MySQL (Oracle Corporation, 2011) databases, storing itself the statistic vectors created.

A particular type of filter used in knowledge extraction mechanisms is stemming filters. A stem is the part of a word that contains no inflectional morphology (Kroeger, 2005): for instance, the stem word for "*decentralization*" is "*centralize*", being "*de-*" the prefix and "*-ation*" the suffix. Stem filtering is used in knowledge extraction because it reduces time in word comparison processes, by getting the essential part of a set of words with just one stem word.

3.1.3 Homologous and Non-homologous Concepts

A kin relation is represented as a "is-a" taxonomic relation. For instance, one can say that the concept "design phase", defining a construction project's designing phase, is related to the more general concept "phase", defining any phase on the project's development. One can also say that "design phase" "is-a" "phase". This relation traduces the fact that "phase" is an ancestor node for "design phase" in the ontological tree structure. Specifically, the kin relations can be expressed through the following definitions (Li, 2009):

Definition 3.1: In the hierarchical tree structure of the ontology, concept A and concept B are homologous concepts if the node of concept A is an ancestor node of concept B (Figure 3.4). Hence, A is considered the nearest root concept of B, $R(A,B)$. The taxonomical distance between A and B is given by:

$$d(A,B) = |depth(B) - depth(A)| = |depth(A) - depth(B)| \quad (1)$$

In Equation 1, $depth(X)$ is the depth of node X in the hierarchical tree structure, with the ontological root concept's depth being zero (0).

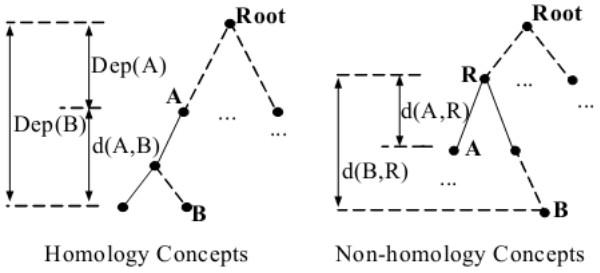

Figure 3.4: Homologous and non-homologous concepts (Li, 2009)

Definition 3.2: In the hierarchical tree structure of the ontology, concept A and concept B are non-homologous concepts if concept A is neither the ancestor node nor the descendant node of concept B, even though both concepts are related by kin; If R is the nearest ancestor of both A and B, then R is considered the nearest ancestor concept for both A and B concepts, $R(A,B)$; The taxonomical distance between A and B is expressed as:

$$d(A,B) = d(R,A) + d(R,B) \tag{2}$$

The taxonomical similarity, $Sim(A,B)$, is calculated differently to both homologous and non-homologous taxonomical relations defined previously:

$$Sim(A,B) = \left(1 - \frac{\alpha}{depth(A)+1}\right) \cdot \frac{\beta}{d(A,B)} \cdot \frac{son(B)}{son(A)}, \tag{3}$$

$$\text{if } d(A,B) \neq 0 \text{ and } A \text{ and } B \text{ are homologous}$$

$$Sim(A,B) = \left(1 - \frac{\alpha}{depth\big(R(A,B)\big)+1}\right) \cdot \frac{\beta}{d(A,B)} \cdot \frac{son(A)+son(B)}{son(R)}, \tag{4}$$

$$\text{if } d(A,B) \neq 0 \text{ and } A \text{ and } B \text{ are non} - \text{homologous}$$

$$Sim(A,B) = 1, \text{if } d(A,B) = 0 \tag{5}$$

In the above functions, $son(X)$ represents the total number of nodes in the sub-tree with root X. The cited author (Li, 2009) states that parameters α and β are managed by filed experts. Equation 5, states that if the distance between two concepts is null, or equal to zero (0), then those two concepts are the same concept.

3.2 Mathematical Foundations

This chapter introduces the basic mathematical foundations needed for the implementation of the presented work.

3.2.1 TF-IDF Function Family

One of the first measures of term relevance on documents used in IR was the *inverse document frequency* (IDF) (Jones, 2004). The IDF measure is defined by the number of documents in a document set which are indexed by, or contain, a specific term, word or expression. It is basically a heuristic implementation of the idea that a term that occurs in a great number of documents should have less relevance, or weight, than one that occurs in few documents.

This idea has given its proofs in the field of IR, especially when coupled with the *term frequency* (TF) measure. IDF does not measure term relevance within a specific document, but rather on a universe of documents. TF completes the task, by measuring the frequency of

occurrences of a term on a document. The approximation used by SEKS system for IDF is considered its basic formula:

$$idf(t_i) = \log \frac{N}{n_i} \quad (6)$$

The above equation states that the weight of a term t_i within a document search space with a total of N documents is given by the logarithm of the quotient of N by n_i, the total number of documents in the search space that are indexed by t_i. The realization of this equation on IR entails an issue that is worthy of consideration: The documents universe, or search space, must be well defined because the total number of documents in it has to be known by the system. The TF approximation used by SEKS system calculates a probability of relevance of a term within, by dividing the frequency or the weight of the specific term, $w_i = freq(t_i)$ by the frequency or weight of the most frequent term in the document $max_t w_t = freq(t_{most\ frequent})$, as defined by:

$$tf(t_i) = \frac{freq(t_i)}{freq(t_{most\ frequent})} = \frac{w_i}{max_t w_t} \quad (7)$$

By coupling TF and IDF, one can not only weigh the relevance of a term in one document, but also the relevance of that term in all the documents in a document search space. Several approximations for the two frequency measures are referenced (Robertson, 2004). The TF*IDF approximation used by the SEKS approach can, thus, be defined by multiplying Equations 6 and 7:

$$tf * idf(t_i) = \frac{w_i}{max_t w_t} \log \frac{N}{n_i} \quad (8)$$

Equation 8 represents the relevance of a particular term on a document, regarding also the relevance of that term for the domain associated to the documents' corpus universe.

3.2.2 Euclidean Distance

The most popular way to measure the similarity of two frequency vectors (raw or weighted) is to take their cosine, or their Euclidian distance (Turney & Pantel, 2010). The Euclidian distance is defined as the distance between two vectors on a Euclidian vector space. A Euclidian vector space is defined by Euclid's five postulates, which are the basis for Classic Geometry:

- A straight line, or vector, may be drawn from any one point to any other point (any 2 points determine a unique vector).
- A finite vector may be produced to any length in a straight line.
- A circle may be described with any center at any distance from that center.
- All right angles are equal.

- If a vector meets two other vectors, so as to make the two interior angles on one side of it together less than two right angles, the other vectors will meet if produced on that side on which the angles are less than two right angles.

Considering $\mathbb{R}^n$ as the domain of a Euclidian vector space with n dimensions, and defining vectors $x, y \in \mathbb{R}^n$ as:

$$x = (x_1, x_2, \dots, x_n) \qquad y = (y_1, y_2, \dots, y_n)$$

Then, the Euclidian distance between x and y, shown in Figure 3.5, is defined as the *inner product* between x and y, and given by (Deza & Deza, 2009):

$$\cos\theta = \frac{\text{"adjacent side"}}{\text{"hypotenuse"}} = \frac{x \cdot y}{\|x\|\|y\|} \tag{9}$$

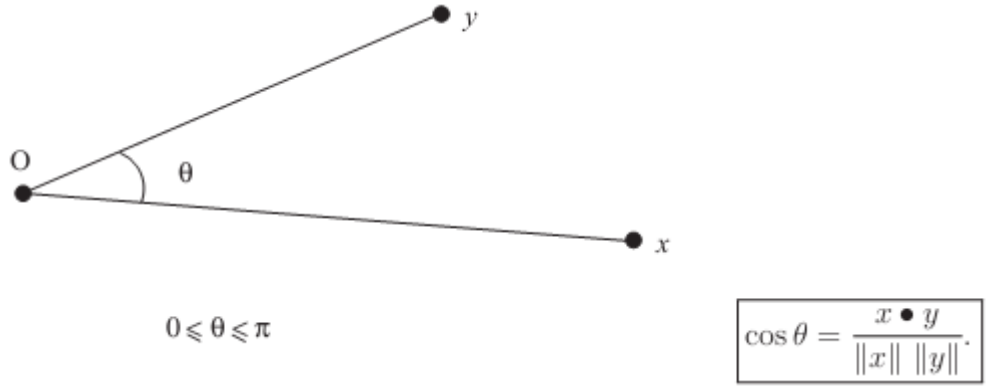

Figure 3.5: Euclidian distance between two vectors

Definition 3.3: *For any $x, y \in \mathbb{R}^n$, the* inner product *of x and y, also known as the* dot product*, is the number*

$$x \cdot y = \sum_{i=1}^{n} x_i y_i \tag{10}$$

Definition 3.4: *The* norm *of a vector $x \in \mathbb{R}^n$ is the number*

$$\|x\| = \sqrt{\sum_{i=1}^{n} x_i} \tag{11}$$

One can easily conclude from Equation 11 that the Euclidian distance dictates that vectors' lengths must match. This entails a problem with the comparison between vectors that do not have the same length. However, there are some cases in which the Euclidian distance can be used to calculate the distance between vectors with different lengths, as it is shown in the next subchapter.

3.2.3 Sparse-Matrix Multiplication

As one can obviously comprehend, semantic vectors do not have necessarily the same size. This means that the Euclidian distance cannot be directly applied on semantic vectors. In order to compute the cosine function between two vectors with different sizes, one must use a sparse-matrix multiplication approximation.

A sparse-matrix is a matrix with only a small percentage of nonzero values. When multiplying two sparse matrixes, these can have different sizes, because zero values can be added to the smallest vector, making it of the same size of the biggest vector. Specifically, considering two vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, with $m > n$, as:

$$x = (x_1, x_2, \dots, x_n) \qquad y = (y_1, y_2, \dots, y_m)$$

In order to perform the multiplication of x by y, the size for vector x has to be augmented by adding $m - n$ zero values to x. SEKS system uses this sparse-matrix multiplication approach due to the fact that the inner product distance measure for vectors x and y, presented above, can be decomposed into three values: one depending on the nonzero values of x, $f_2(x_i)$, another depending on the nonzero values of y, $f_3(y_i)$, and the third depending on the nonzero coordinates shared both by x and y, $f_1(x_i, y_i)$. Formally:

$$\cos\theta = f_0\left(\sum_{k=1}^{m} f_1(x_i, y_i), f_2(x_i), f_3(y_i)\right) \tag{12}$$

In Equation 12, f_0, $f_1(x_i, y_i)$, $f_2(x_i)$ and $f_3(y_i)$ are given, respectively, by:

$$f_0(a, b, c) = \frac{a}{bc} \tag{13}$$

$$f_1(a, b) = ab \tag{14}$$

$$f_2(a) = f_3(a) = \sqrt{\sum_{k=1}^{m} a_k^2} \tag{15}$$

The four expressions defined above can be combined to realize Equation 11. In this case, even though the inner product method initially requires that both vectors have the same size, when applying the sparse-matrix multiplication approach presented above the vectors' sizes don't necessarily have to coincide.

If one vector is smaller than the other, then it means, in practice, that the smaller vector has zero values for all the concepts that are missing to reach the size of the bigger vector. On the other hand, calculating $f_1(x_i, y_i)$ is only required when both vectors have at least one shared nonzero coordinate. If the vectors do not possess any shared concept, i.e. a nonzero coordinate, the value for the function above is zero, and the vectors do not present any similarity. This also means that $f_2$and

$f_3$do not need to be calculated, significantly reducing the computation needed (Turney & Pantel, 2010).

4 REQUIREMENTS AND CONCEPTUAL MODEL

This chapter explains the conceptual model that supported this thesis. It presents a possible utilization scenario, the main requirements and the basic functional and architectural visions adopted in this work.

4.1 Scenario

Consider the following scenario: A meeting associated to an engineering project is being held on a collaborative project environment, represented in Figure 4.1. Several project actors are present on the online environment, working on decision-making and problem-solving processes around the project. Collaboration participants may need to search the knowledge source repository of the collaborative environment for information that has some relevancy on such processes. The SEKS search process begins when a participant inputs a query in the form of keywords in the SEKS User Interface. The knowledge source repository is comprised by knowledge assets or sources, such as documents, that were previously uploaded to the system, and that were statistically and semantically indexed.

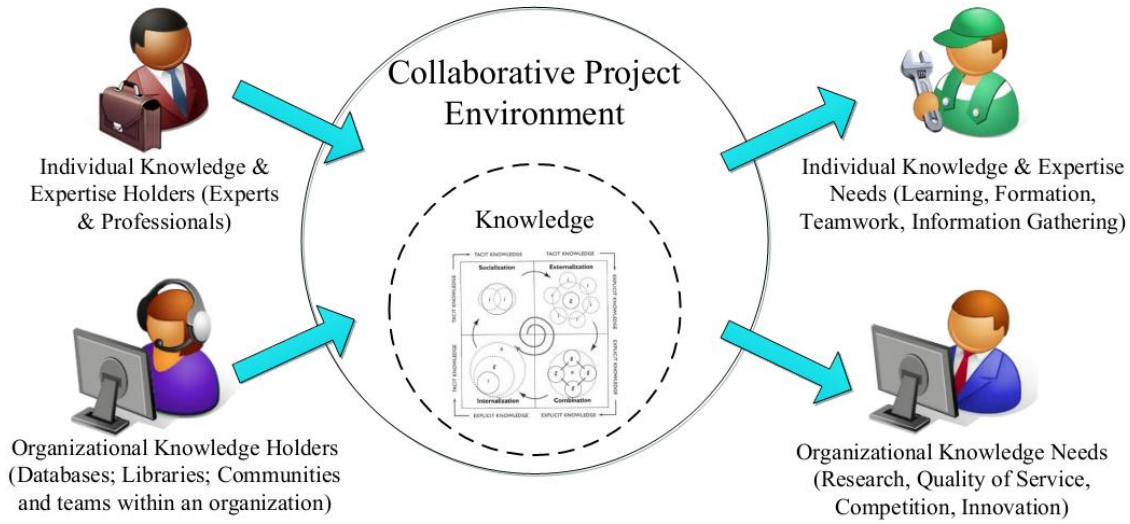

Figure 4.1: Collaborative Project Environment

4.1.1 Actors

Explicitly, there are two types of actor considered in the conceptualization and development of this work. Knowledge holders are actors which possess knowledge in various forms to share with other collaborative environment partners. If such knowledge is in the form of knowledge sources, SEKS enables knowledge holders to upload their knowledge sources to SEKS database. The other type of actors is composed by collaboration environment users that have some sort of knowledge need. If such knowledge is in the form of documents or other kind of knowledge source, then SEKS enables

such users to search the SEKS knowledge source repository, by typing a query in the SEKS User Interface.

4.1.2 Preconditions

SEKS system assumes a set of preconditions that it must attend, namely:

- Actors must be able to upload files to the system. This precondition enables the use of an external knowledge source repository, so that SEKS may have upload capabilities.
- Actor should receive only relevant results, according to their environment (in this case, the Building and Construction environment). SEKS has to create statistical and semantic representations of knowledge sources, in the form of vectors, to extract relevant knowledge from them.
- Actors must be able to search SEKS knowledge source universe by simply typing keywords on the search bar in SEKS User Interface, just as in regular search engines. Although SEKS is not a search engine, as previously referred, it has to possess knowledge source search capabilities over the SEKS search space. Also, actors' queries should be treated as pseudo-knowledge sources, meaning that they also have to possess some type of knowledge representation, similar to knowledge sources. This will enable the ranking of knowledge sources, through the comparison of the respective semantic vectors.
- The knowledge source indexation process is not made each time a knowledge source is uploaded to the system; rather, SEKS performs a server-side periodical task once a day that indexes all files uploaded on that particular day.

4.1.3 Assumptions

SEKS also takes into account some assumptions. First, SEKS assumes that the used domain ontology covers the entire domain in question. This assumption may have performance consequences, in terms of the specificity and relevance of extracted knowledge. Specifically, when indexing a knowledge source, the representation of such source will be built upon knowledge extracted from that source, in the form of relevant terms and expressions, and then matched against ontological concepts, in an attempt to extrapolate the semantic relevance of the extracted knowledge from a statistical account of frequency, relevance and positioning in the text of such relevant terms and expressions. SEKS uses semantic vectors to, introduced in Chapter 3, to build such knowledge representations. The definition of "semantic vector" used in this work is inherently linked to this extrapolation of knowledge, through the use of domain ontologies. Second, SEKS assumes that external Knowledge Extraction mechanisms will extract the relevant terms or expressions, named keywords, from knowledge sources, and build a statistic representation, in the form of a statistic vector, of such sources. Statistic vectors are the main input for SEKS system.

4.1.4 Post Conditions

After the input of a query by a random user, the results may or not be satisfactory. In the case of being satisfactory, users should have a sorted list of knowledge sources, by similarity to the user's query. If, on the other hand, results are unsatisfactory, meaning that the user cannot fulfill his or hers information needs, the query must be refined, by adding new keywords to it, or to generalize keywords to much broader terms.

4.2 Requirements

The above scenario is the basis of work for the presented project, and the set of established requirements are supported by it. The system must be able to handle semantic indexation of knowledge sources and to provide fast and contextualized results to users within the Building and Construction domain. In such scenario there are several system requirements, divided into three main groups: Functional, Architectural and Technical Requirements, as shown in Figure 4.2.

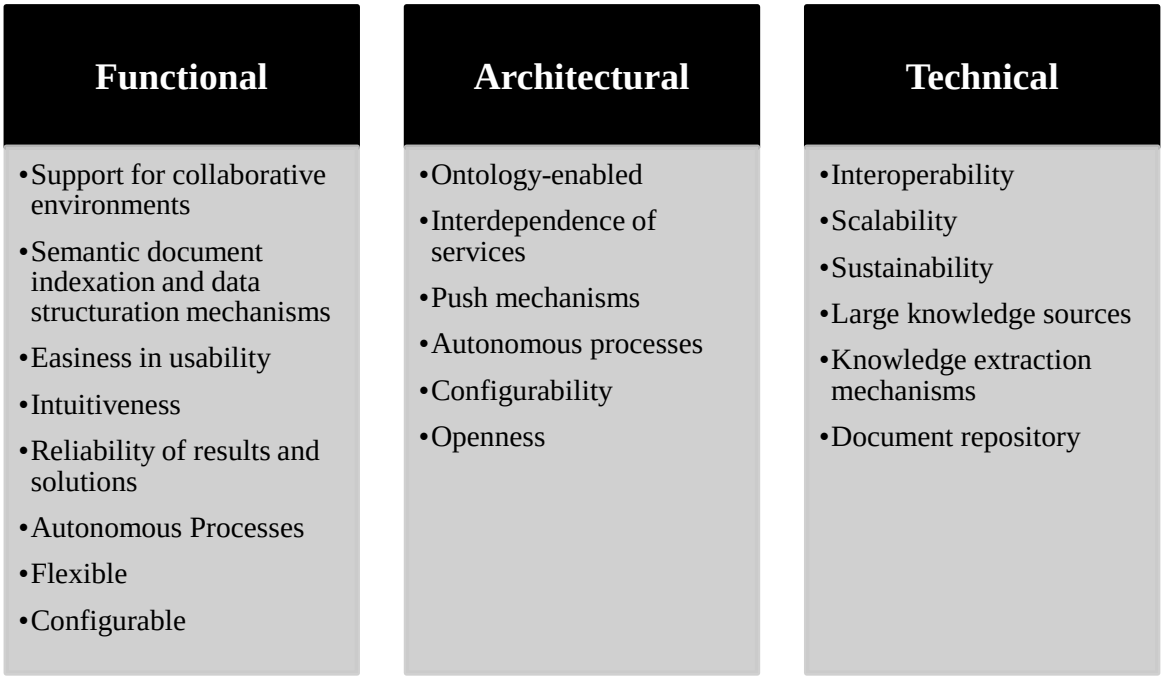

Figure 4.2: SEKS system main requirements

As previously referred, the scope of this dissertation is aiming at supporting collaborative engineering projects, and at the Building and Construction domain. Nevertheless, applications in other environments are possible, due to SEKS flexibility and configurability, as referred later in Chapter 7. The system also must present user interaction mechanisms that are already recognized by the general public as intuitive and easy to use, without putting at risk the reliability of the pretended results. Furthermore, the main requirement of the presented work is to provide tools and mechanisms for semantic knowledge source indexation and to give some contribution to the unstructured data problematic, previously stated in Chapter 3.2.

In terms of technical and architectural requirements, the system is adopting a n-tier architecture, in which low-level and high-level services interact with each other in order to achieve the required objective. Such services must be scalable and sustainable, not only in terms of its software infrastructure, but also considering the use, for example, of other ontology regarding a different domain or environment. Push mechanisms are a way of pushing the information to the user that would be interested in that, depending on his or her context of interest, and autonomous processes are processes able to execute a number of basic steps without human intervention in order to achieve a given goal and make proper decisions. The presented work will not provide knowledge extraction from knowledge sources; instead, it will use an external technology to handle statistic vector creation. Furthermore, the knowledge source repository needed for storing collaborative environments' knowledge sources will also be from an external source.

4.3 Conceptual Model

SEKS design and modeling was formalized by adopting the Unified Modeling Language (UML), a standard general-purpose visual modeling language that is used to understand, design, maintain and control information about a software system, helping on the visualization and documentation of systems' or processes' models, including their structure and design, in such way that complies with specifications and requirements. UML may be viewed as set of graphical modeling notations (diagram objects), and textual modeling notations (syntax explaining how objects are linked).

The conceptual model paradigm used on SEKS project is the Model View Controller (MVC) paradigm, used to guide the development and structuration of software systems. It is divided in three distinct layers: Model, View and Controller. The Model layer corresponds to databases, repositories and ontologies, and it represents the data model in which the system lays down. The View layer is comprised by the system's interfaces, and it is the link between system and user. Finally, the Controller layer responsible for all interaction between View and Model layers, processing users' requests from the View layer, fetching the necessary information from the Model layer, and presenting it on the View layer again.

The conceptual model for SEKS will be presented in two different scopes: functional scope and architectural scope. The functional vision will approach the possible utilization scenarios for SEKS system and what are the tasks running behind the project's curtains in each scenario. The architectural vision will present the tasks implemented by the system, divided into conceptual modules.

4.3.1 Functional Vision

The functional vision of the implemented software infra-structure is given in terms of UML Use Case Diagrams representing functionalities that users should expect to be provided by SEKS

system. The SEKS system's approach can be seen as three distinct processes: knowledge extraction, knowledge source and query indexation, and semantic vector comparison, as shown in Figure 4.3.

Although the knowledge extraction process is needed for the system's correct functioning, it is not part of the work developed nor will it be thoroughly explained on this document. However, it is necessary to make a conceptual introduction to the subject in this Chapter, and then mention the tools used to satisfy the requirements for this module, regarding that knowledge extraction is, as referred above, one of the requisites of the SEKS system.

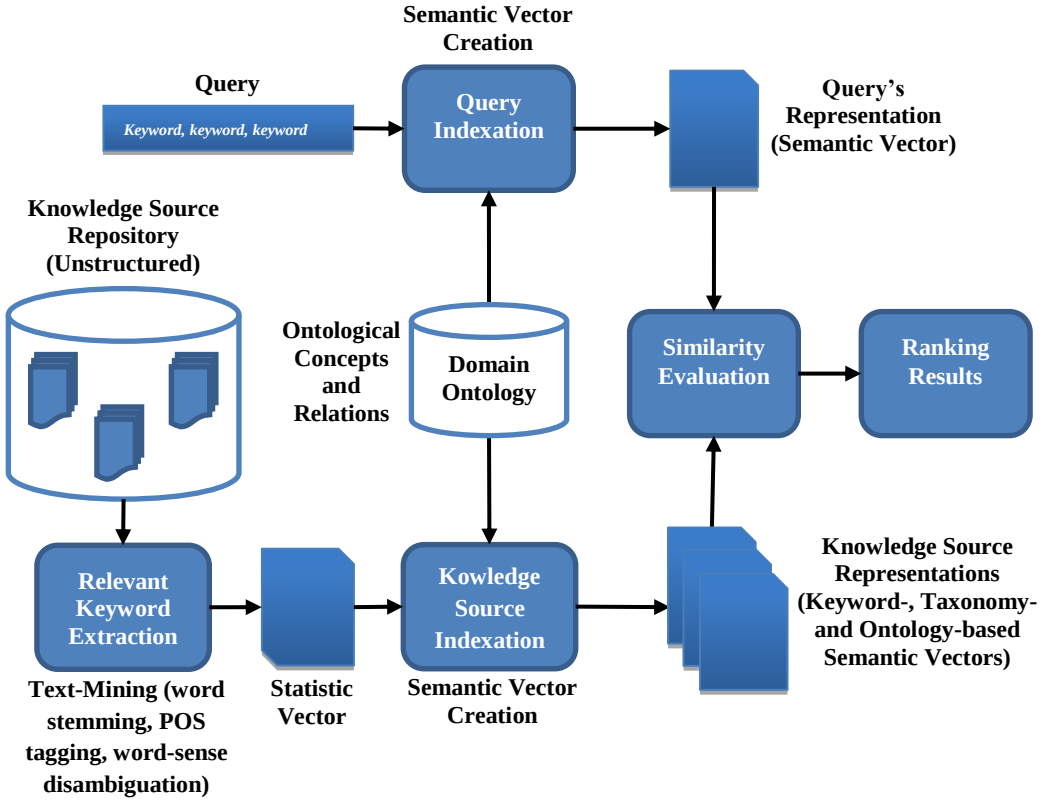

Figure 4.3: Knowledge source indexation and comparison

Knowledge extraction mechanisms process raw information from knowledge sources (e.g. text, metadata) and extract the most relevant terms from that information, in terms of their occurrence frequency and position in such sources, filtering unwanted or irrelevant terms which occur many times on a source, but do not introduce any relevance to its knowledge representation. Knowledge source indexation manages the creation of knowledge sources' semantic vectors, constructing a knowledge representation not only of the source itself, but also of the underlying meaning inherent to its most relevant terms. Knowledge source comparison processes queries, by treating them as pseudo-knowledge sources and attributing special semantic vectors to them, and

compares the resultant query vector with semantic vectors belonging to sources present in the system's search space[6].

4.3.1.1 Semantic Vector Types

The presented approach takes into account tree different, but complementary iterative procedures for building up semantic vectors: Keyword-based, taxonomy-based and ontology-based semantic vectors. Each of these types of semantic vectors is a more realistic iteration of the knowledge representation of a knowledge source, in terms of semantic enrichment. In the context of this work, semantic enrichment may be defined as the process of analyzing underlying relations between ontological concepts within the knowledge representation and, depending on the relevance or force of such relations, boost the relevance of certain concepts within the representation of the knowledge source or add new concepts that are relevant to the knowledge source representation through their relation with concepts already present in that specific representation. For instance, if a document has several references to concepts that may be considered as belonging to a certain taxonomy, as "Architect", "Engineer" and "Designer" may belong to the family of concepts under the concept "Design Actor", it is reasonable that:

- The relevance for concepts "Architect", "Engineer" and "Designer", in the representation of the knowledge source, should rise in accordance to the fact that all of these concepts can be considered of the same kin, as will be explained later in Chapter 5.
- The concept "Design Actor" should be added to the representation of that knowledge source, because it is taxonomically related to several concepts within such representation and, thus, can be considered to be semantically relevant to represent that particular knowledge source.

Keyword-based semantic vectors are built upon the statistic representation of a knowledge source in the form of expressions that occur in the source, according to their emphasis and frequency of occurrence both locally (in the knowledge source itself) and globally (in the knowledge source corpus' universe). Taxonomy-based vectors push one notch further in the representation of a knowledge source by adjusting the weights between expressions according to their taxonomic kin with each other, i.e., expressions that are related with each other with the "is a" type taxonomic relation. If two or more concepts that are taxonomically related appear in a keyword-based vector, the existing relation can boost the relevance of the expressions within the source's representation. Ontology-based vectors are the last iteration of the semantic vector creation process. The creation process for this type of vector uses keyword-based and taxonomy-based vectors as inputs to analyze the inherent ontological relation patterns between the input vectors' concepts. Such ontological relations define semantic patterns between concepts which can

[6] The author also refers to the universe of knowledge sources comprised within the SEKS system as the system's "search space".

be used to enhance the representation of the knowledge source. For instance, if a vector has two concepts that are related to each other by an ontological relation, and if this ontological relation occurs frequently across the system's knowledge sources universe, then the relevance of both concepts being together within the knowledge source increases the weight of these concepts in the vector. Other hypothesis covered by the presented system is the addition of new concepts that have relevant, or strong, taxonomic and ontological relations with concepts already present in the input vectors.

The importance of these relations is given by a threshold that is manually included in the relation by an ontology manager. Although the automatic calculus of a relation's importance threshold is not part of the presented work, research over the subject has been made by other authors. For instance, SemRank uses a blend of semantic and information theoretic techniques along with heuristics to determine the rank, or relevance, of semantic and taxonomic relationships in an ontology. In the SEKS system's case, the importance of taxonomical relations is automatically computed, as will be stated in the following chapters, but the importance of ontological relations is, as referred before, manually inputted. The reasons behind the creation of these three iterations are, firstly, to compare the results between iterations, and secondly, to provide users various kinds of search specificity, depending on their needs. Also each of the semantic vector types used is an attempt to better reflect the underlying information within the knowledge source, in order to better respond to the system users' needs.

4.3.1.2 Knowledge Source Indexation and Semantic Vector Creation

When a knowledge source is uploaded into the system, two main tasks are triggered, as shown in Figure 4.4, but not at the same time. First the source is stored in the system's knowledge source repository, which is not part of the implemented work. There are many online free-to-use document repositories, so there was no necessity to implement one. All dashed use cases in Figure 4.4 were not implemented from scratch because there were already good, open-source choices available, as is the case of the Knowledge Extraction module.

SEKS system performs knowledge source indexation tasks every day, at a scheduled time, for all sources uploaded during that day. The knowledge source indexation process's function is to create a semantically enriched knowledge representation of the knowledge source from the syntax-based statistic vector outputted by the knowledge extraction process. This representation comes in the form of semantic vectors, which is a particular type of vector extracted from a document-term matrix, used in the knowledge extraction process. The difference between the general term-document form and the more specific semantic form is that the semantic vector is composed by concepts which may not be directly present on the source, but have an inherent relation with the terms present within the statistic vector.

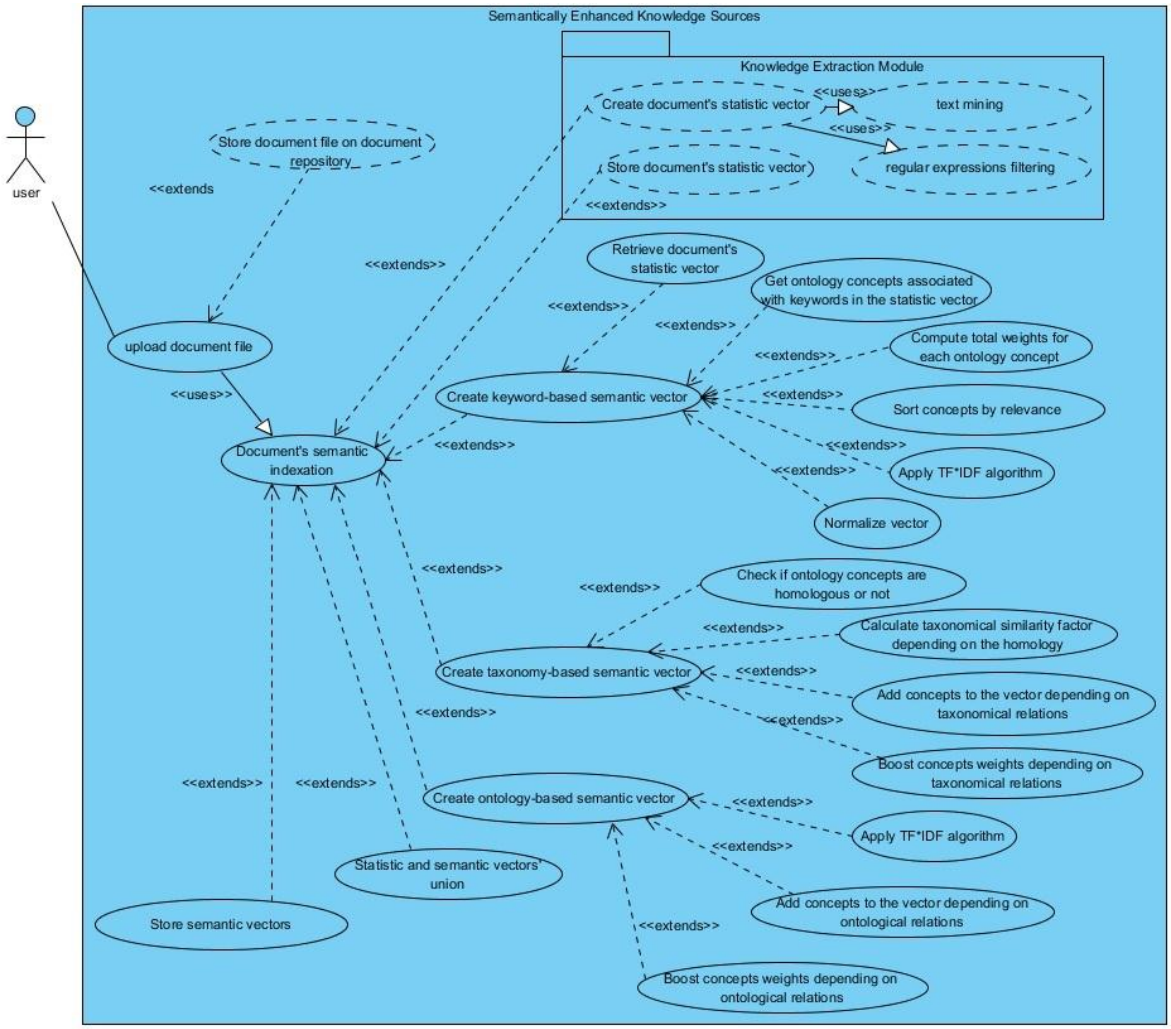

Figure 4.4: UML Use Case Diagram – Knowledge Source Indexation

The UML Activity Diagram representing knowledge source indexation process behavior is shown in Figure 4.5. The process of fetching concepts from a knowledge source's statistic vector subject will be approached in Chapter 5. However, it is worth to say that the concept fetch process is made by matching the terms in the statistic vector with a *domain-specific* ontology, comprehending concepts from that domain and semantically related keywords to each of the concepts. In the case of SEKS system, and as referred before, the domain is the Construction Environment domain. For instance, a possible concept on a construction domain ontology can be "architect" and the related keywords may be, besides the concept itself, "architecture", "building design", "architectonics", and all other terms and expressions which are related with the concept. It is important to mention that a keyword may not be only composed by a word; instead it can be an expression, a phrase or any form of text which can be associated to a certain concept.

Furthermore, instead of having the weights depending on the frequency of appearance of a term in a particular knowledge source, the weights in the semantic vector reflect the semantic relevance of a concept within that source.

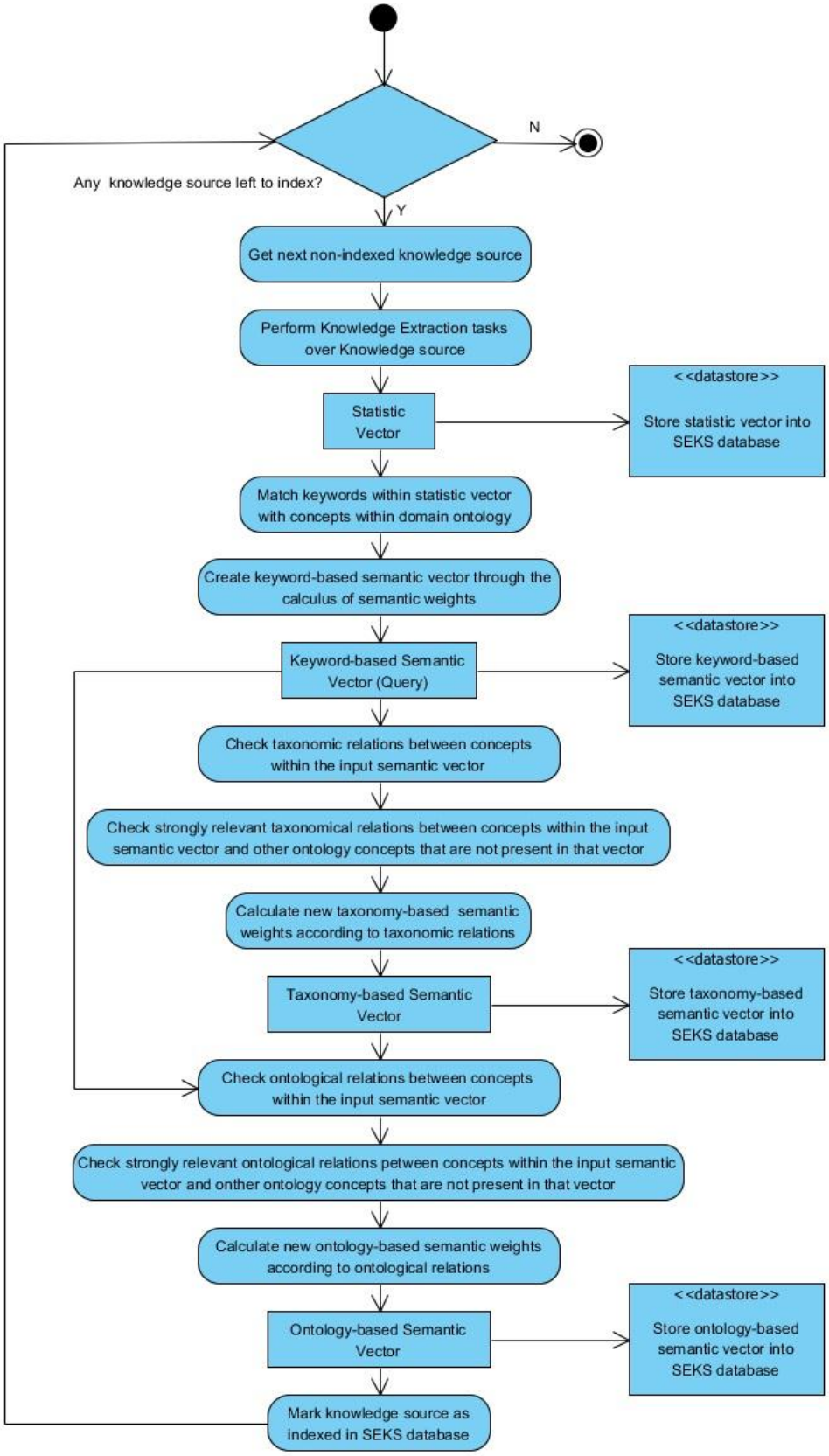

Figure 4.5: UML Activity Diagram - Knowledge Source Indexation

Such semantic relevance is based not only on the frequency of appearance and emphasis given in a particular source, but also on its significance within the knowledge source itself,

regarding the construction domain, and throughout the knowledge sources' universe comprised by SEKS system. More will be said about this subject in Chapter 5.

The knowledge source indexation process starts by building a statistic vector based on the frequency of occurrences for the most frequent or better positioned (e.g. a word in the title is more relevant than one in the text content of a document) terms or expressions. The statistic vector is organized in matrix form and is composed by the extracted terms, or keywords[7], and by the statistical weight of each keyword within the knowledge source, based on the frequency analysis introduced above. This task is made by the Knowledge Extraction module, which uses text-mining techniques and regular expression and stemming filtering processes, presented in Chapter 3.1.2, to gather such terms and expressions.

These terms and expressions are called *keywords*. Both semantic and statistic vectors of each knowledge source are united, in the sense that, keywords that belong to the statistic vector but are not associated to any ontology concepts present in the semantic vector will be added to the semantic vector. This overrides the issue of having a poorly developed domain-specific ontology. The statistic vector is then stored in the system's vector database. The real semantic indexation process begins here, with the creation of the three semantic vectors' iterations.

4.3.1.3 Queries, Knowledge Source Comparison and Result Ranking

The second use case is knowledge source search (Figure 4.6) which is comprised by query treatment and knowledge source comparison. When a user inserts a search query on the system's interface, the first step is to treat the user's query. The system creates statistic and semantic vectors for queries, with some differences when comparing with those of knowledge sources, as will be explained in the next chapters.

In syntax-based IR systems, the comparison between a query and a knowledge source is made, in a simplistic view, by syntactically matching query keywords with documents' expressions, usually through knowledge extraction techniques. This process may have regular expressions filtering and take into measure the position and emphasis of expressions in texts, much like the SEKS statistic vector's creation process. The SEKS system handles knowledge source comparison through semantic vector comparison, using vector intersection techniques, as discussed later on this Chapter.

[7] Throughout the text, the author uses the words "keyword" and "term" in a way that may appear ambiguous. The word "keyword" is applied whenever the author means the terms extracted from the document. The word "term" is applied whenever a reference to the term-frequency vector is used; both mean the same thing in different contexts.

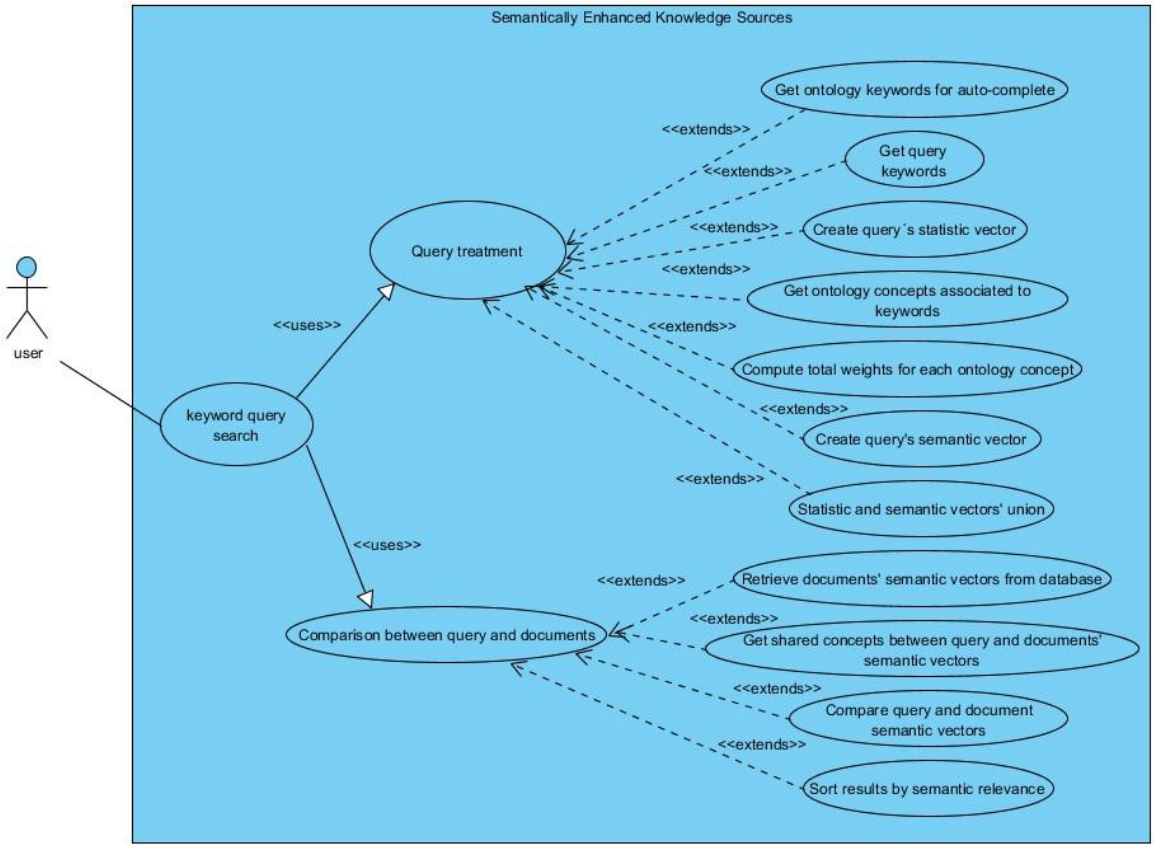

Figure 4.6: UML Use Case Diagram – Knowledge source search

This enables the system to respond to users' queries in a "semantic" way, matching knowledge representations of sources and queries and outputting contextually relevant results. The system also provides one other form of query, built directly with ontology concepts, which will be explained in Chapter 5. The UML Activity Diagram representing knowledge source search process behavior is shown in Figure 4.7.

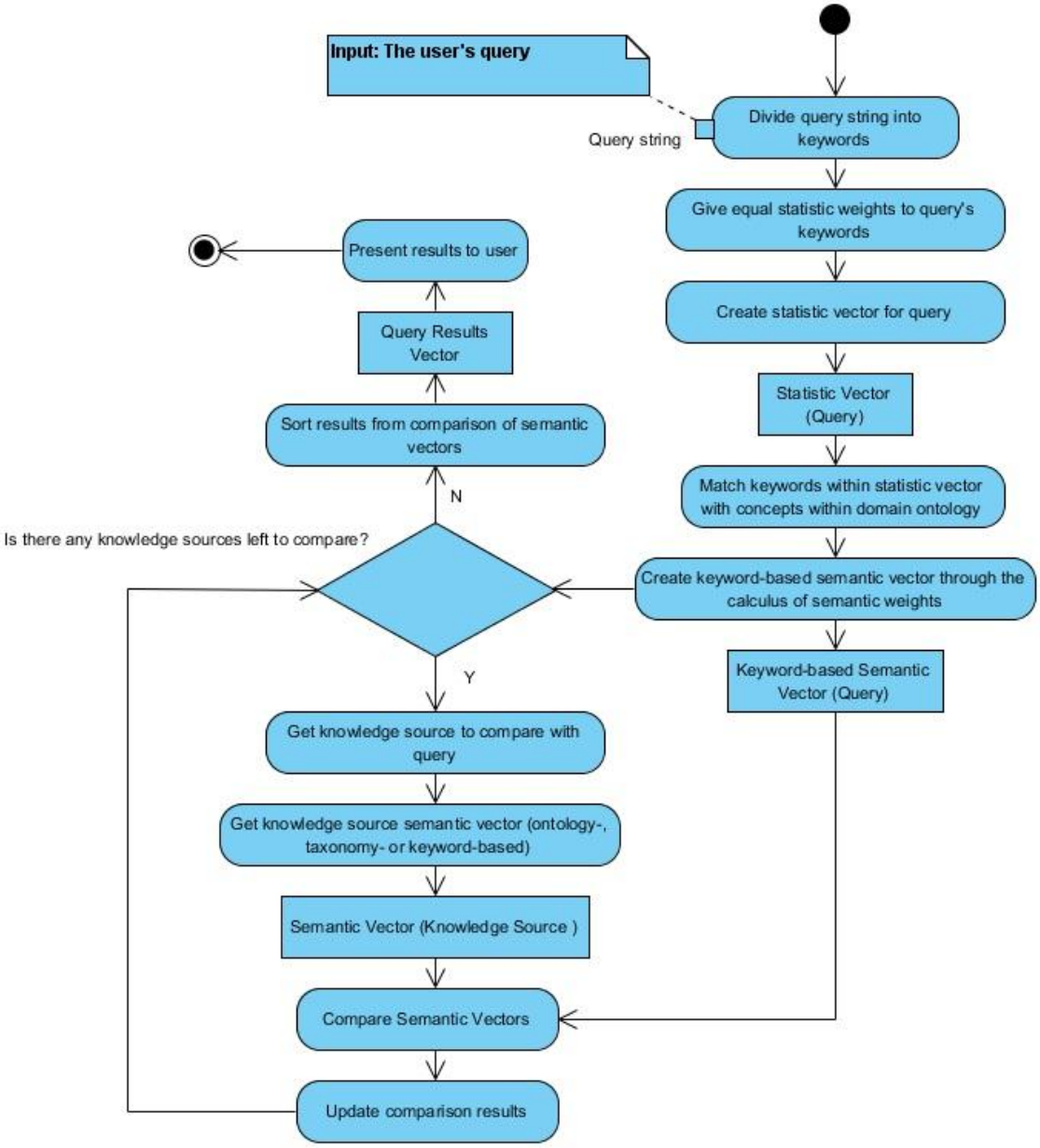

Figure 4.7: UML Activity Diagram - Knowledge source search

4.3.2 Architectural Vision

As previously mentioned, SEKS system is conceptually based in the MVC paradigm, presenting a n-tier architecture, as shown in Figure 4.8. View layer comprises the system's interfaces and client interaction modules. Controller layer includes both SEKS Basic and Advanced Services and the external Knowledge Extraction Module. Model layer contains databases, knowledge source repositories and the system's ontology.

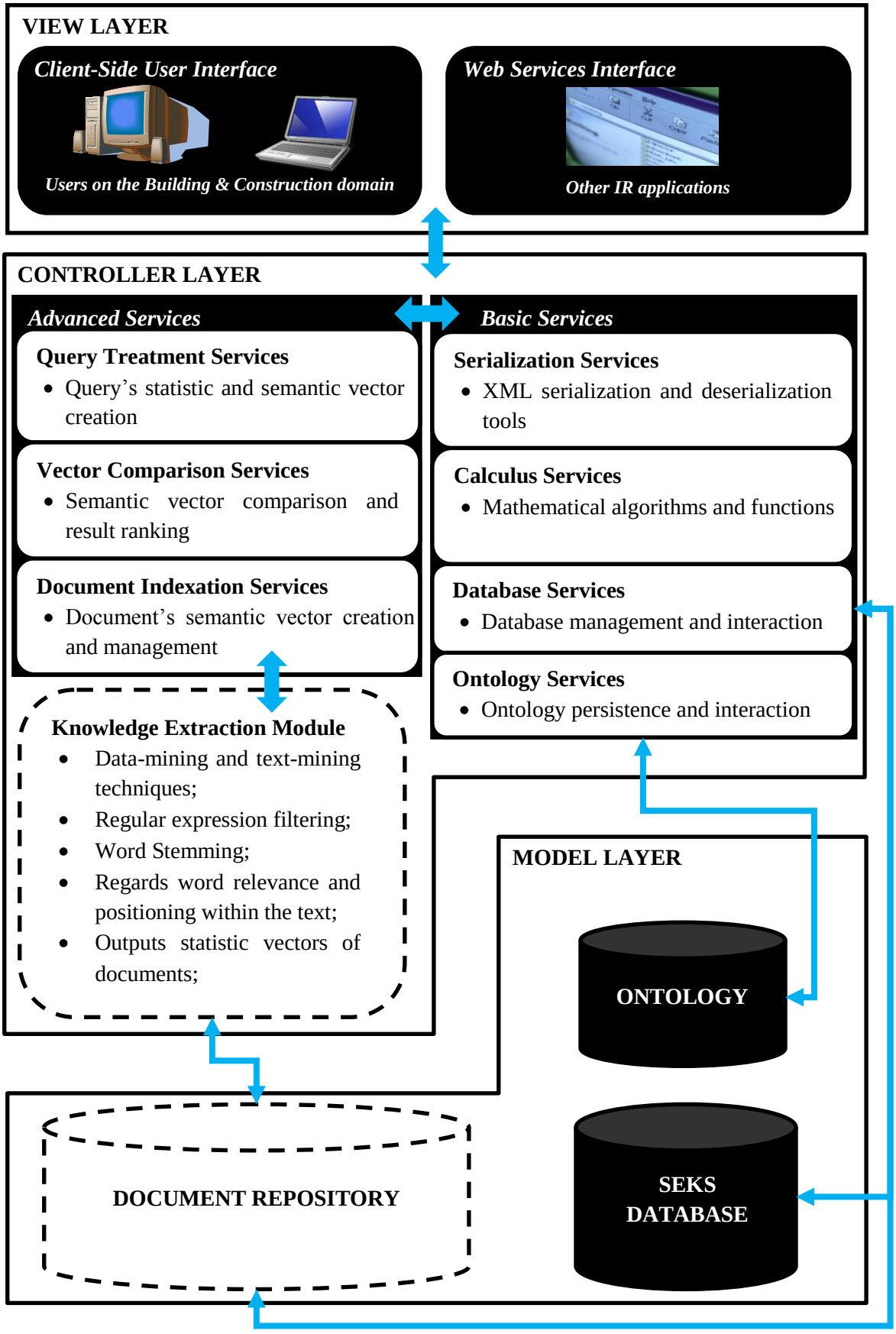

Figure 4.8: SEKS Architecture

4.3.2.1 View Layer

This layer comprises, besides a client-side application, all classes that manage interactions with SEKS system's users, linking the system's server-side services with the client-side application. Two interfaces are comprised in this layer: Web Services Interface and User Interface. The Web Services Interface will allow other software systems to use SEKS knowledge source indexation capabilities, and will provide remote access to SEKS ontology. The User Interface is a simple search engine-like Web interface, with a search field where users can insert their queries.

4.3.2.2 Controller Layer

The Controller layer is responsible for all Basic and Advanced Services' methods, and it is considered the core behind SEKS system's provided functionalities. Basic Services are services or methods that perform low-level functions like direct interaction with databases and ontology, mathematical computation or data serialization for the Web Services Interface. Advanced Services are high-level procedures and functions that are supported by Basic Services functionalities. Some of examples of Advanced Services are semantic vector creation, query treatment and knowledge source comparison.

4.3.2.2.1 Basic Services

Basic Services are comprised by four service modules: *Serialization Services*, *Calculus Services, Ontology Services* and *Database Services*. Serialization Services are used by the Web Services Interface to marshal and unmarshal information to and from XML format. Calculus Services are responsible for the needed mathematical computations, as the TF*IDF algorithm and the cosine algorithm, both introduced in Chapter 3 and further explained in Chapter 5. Database Services manage connections and interactions with the system's database and knowledge source repository, as shown in Figure 4.8. Finally, Ontology Services comprise all methods to persist the system's ontology on the Web, as if it were a database itself, and manages all calls to it.

4.3.2.2.2 Advanced Services

The Advanced Services layer interacts with all other Controller sub-layers: Knowledge Extraction and Basic Services. It is responsible for realizing SEKS system's main functionalities, as the system's core, and it is comprised by three high-level service modules that implement the use cases previously presented on Figures 4.4 and 4.6. *Document Indexation Services* comprises all functions associated to the creation of all three iterations of semantic vectors, and it is the only module that interacts with the *Knowledge Extraction Module*. *Query Treatment Services* are responsible for user queries' treatment and statistic and semantic vector creation. Last but not least, *Document Comparison Services* contain all methods that support the comparison between semantic vectors and ranking the results of this comparison.

4.3.2.3 Model Layer

SEKS uses three repositories: a knowledge source repository, where knowledge sources are stored, an ontology-persisted database map, to provide the system access to the ontology through the Web, and SEKS database, to store statistic and semantic vectors and link them to knowledge sources. The knowledge source repository will be provided by an exterior tool or service and the ontology map is automatically created by the semantic framework in charge of ontology interaction processes, as is explained later on Chapter 5. The Entity-Relation Diagram for the system database is shown in Figure 4.9.

The *Document* table serves as link between knowledge sources in the knowledge source repository and the vectors for those sources. This link is provided by having an identification number for the knowledge source repository, with which knowledge sources can be retrieved according to their semantic vectors. *StatisticWeight* stores knowledge sources' statistic vectors and *KeywordBasedSemanticWeight*, *TaxonomyBasedSemanticWeight* and *OntologyBasedSemanticWeight* tables store the three iterations of semantic vectors created by SEKS. *OntologyRelation* and *TaxonomyRelation* tables are used to keep track of ontological and taxonomical relation occurrences within the knowledge source repository, respectively. *RelationImportance* table stores the ontological relation importance used on ontology-based semantic vectors creation. The SEKS database also has some routines implemented, that will be presented in Chapter 5.

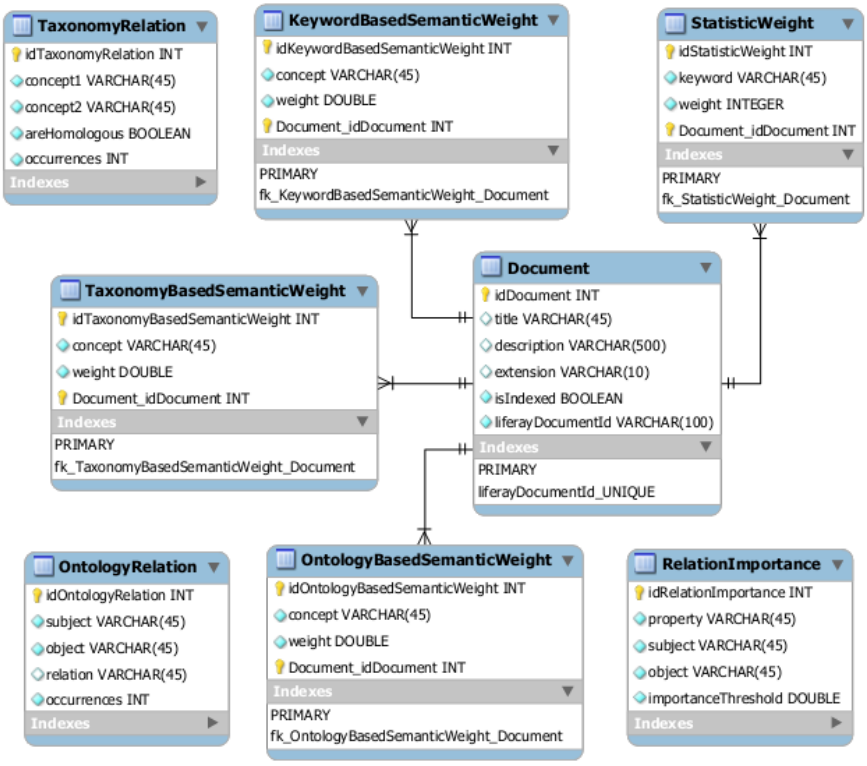

Figure 4.9: Entity-Relation Diagram for SEKS system database

5 SEKS IMPLEMENTATION

In this chapter, the adopted tools and technologies and the whole implementation process are described. Then, both static and dynamic visions of SEKS architecture are given.

5.1 Adopted Tools and Technologies

The conceptual work previously presented can be implemented using many technologies. Through the work's design phase, the system was modeled with Visual Paradigm for UML, a UML modeling tool. The system's database was implemented in MySQL, and designed with MySQL Workbench, which is a visual tool for SQL database development, and the ontology was coded in OWL-DL with the Protégé Ontology Editor, a visual ontology-editing tool supporting OWL and RDF.

The Controller Layer, shown in Figure 4.8, is coded in Java programming language and was developed using Eclipse IDE 3.7, which provides a visual integrated environment for several programming languages and paradigms. The whole system runs on Apache Tomcat 7 server, which is web application container that supports Java applications. The interaction with the ontology is managed by Jena Semantic Framework.

Communication between View and Controller layer is made with Java Servlets 3.0 technology. In the View Layer, the user interface is implemented with HTML 5 and CSS 3, and uses jQuery JavaScript Library to perform AJAX requests to the server, event handling and animations, and the Web Services interface is implemented using JAX-WS RI framework, which provides tools and infrastructure to implement Web Services. Appendix B shows the main technologies used and a brief introduction to each one.

5.2 Ontology

The domain-specific ontology used in this work was entirely developed using Protégé ontology editor (Stanford Center for Biomedical Informatics Research, 2011), and it is written in OWL-DL language. Main inputs to build SEKS ontology are the OmniClass Standards for the Construction Environment (OCCS Development Committee Secretariat, 2011), the BuildingSmart IFD Library (BuildingSmart, 2011), the CoSKS ontology (Lima, et al., 2010) and the Construction Information and Knowledge Portal ontology (Zhang, 2010).

The SEKS ontology has been developed for test and validation purposes and does not try to enclose all aspects of the Building and Construction domain; rather, it includes a set of ontological concept families, and respective properties, that enable semantic knowledge extraction from knowledge sources related to collaboration entities on such domain. Specifically, the set of concept families is shown in Figure 5.1 and it is comprised by the following:

- **Actor:** The set of actors that act on a construction project. All individuals or groups of individuals that perform an action with a particular objective under the Construction Environment context are considered actors. Some examples of concepts encompassed by 'Actor' are architect, contractor, team, laborer, engineer, etc.
- **Agenda:** Different projects have different agendas. An agenda is a task list for a specific project and it comprises project issues, actor tasks, and timelines. Three types of agendas are contemplated: technical financial and administrative agendas.
- **Issue:** Issues and problems that occur in a project. If issues are catalogued and indexed in the form of knowledge sources, then it is possible to find old solutions from other projects to solve apparently new problems on a recent project. Three types of issues are contemplated: technical financial and administrative issues.
- **Knowledge Item:** Knowledge sources' types such as, for instance, documents (such as books, articles, manuals, contracts) or images (such as maps, graphs, drawings etc.).
- **Meeting:** Each project can be viewed as a set of meetings, comprising project kick-off, intermediate and final meetings. Intermediate meetings may be problem solving, planning or feedback meetings.
- **Product:** Products used in a construction project, such as electric appliances, sanitary products, equipment and furnishings, and manufactured structures.
- **Project:** The project itself, or its final function as, for example, museums, libraries, commercial facilities, production and industrial facilities or stadiums.
- **Project Form:** The form of a specific construction project as well as size, number floors or type of terrain that the project is built in. Some examples are high-rise buildings, movable structures and land forms.
- **Project Phase:** The different phases of a construction project, such as design, execution and demolition phases.
- **Skill:** Skills gathered by construction actors, such as social, cognitive or physical skills.
- **Task:** Types of task that must be performed by actors to achieve objectives proposed on meetings, annotated on agendas, and belonging to a specific project. Three types of tasks are contemplated: technical financial and administrative tasks.

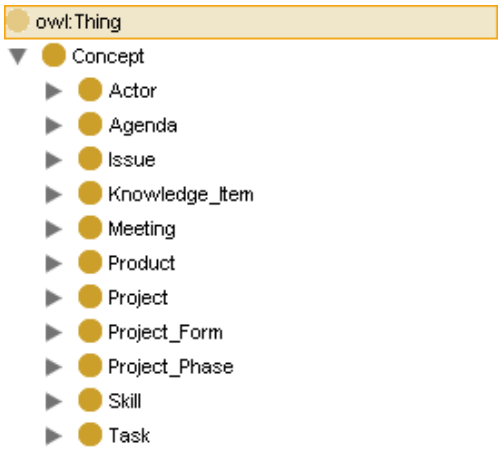

Figure 5.1: Main concepts of SEKS ontology

Several levels of specificity are given for all concept families, as shown for concept "*Actor*". These specificity levels represent concepts hierarchies and, ultimately, taxonomic relations such as '*Architect*' '*is a*' '*Design Actor*' and '*Design Actor*' '*is an*' '*Actor'*, as shown in Figure 5.2. All classes, or concepts, have an instance, which corresponds to the class, and comprises the keywords or expressions gathered and related to each concept, through an ontological datatype property denominated 'has Keyword'.

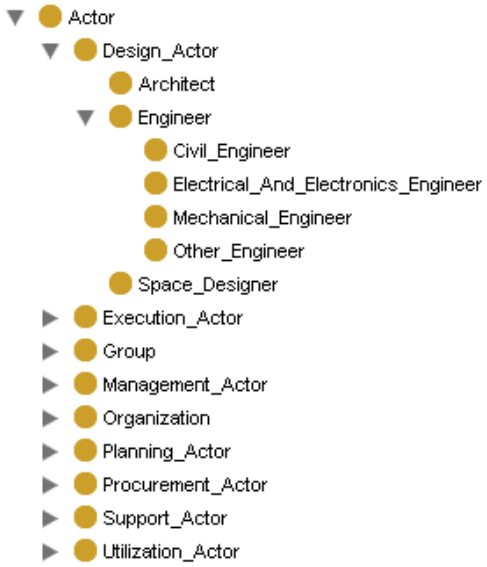

Figure 5.2: Subclasses for concept 'Actor', 'Design Actor' and 'Engineer'

All concepts are themselves keywords, because all concepts are expressions or terms that may occur in a knowledge source. Apart from themselves, concepts possess keywords that are terms or expressions relevant for capturing different semantic aspects of such concepts. For instance, the '*Architect'* class has an '*Architect Individual*', and this instance has several keywords such as '*architect*', '*architecture*', '*building design*' '*urban planner*', etc., as shown in Figure 5.3. Also, concepts are connected by ontological object properties, used to enrich semantic vectors

according such relations' relevance in the context of the Building and Construction domain. As examples, five ontological object properties of SEKS ontology are shown in Table 5.2.

Table 5.1: Examples of ontological properties or relations

Property	Subject	Object	Description
operates in	Actor	Project Phase	Actors operate in one or several particular project phases
is involved in	Actor	Project	Actors are involved in projects
has skills	Actor	Skill	Actors have some skills and expertise
has skill needs	Project	Skill	Projects need actors' skills and expertise
is decomposed in	Project	Task	Projects may be considered sets of tasks

It is important to notice that SEKS system in not implemented to work only with SEKS ontology. Any ontology that is built according to the SEKS ontology's structure will function on the system, contributing in some sense for the fulfillment of scalability, flexibility and sustainability requirements.

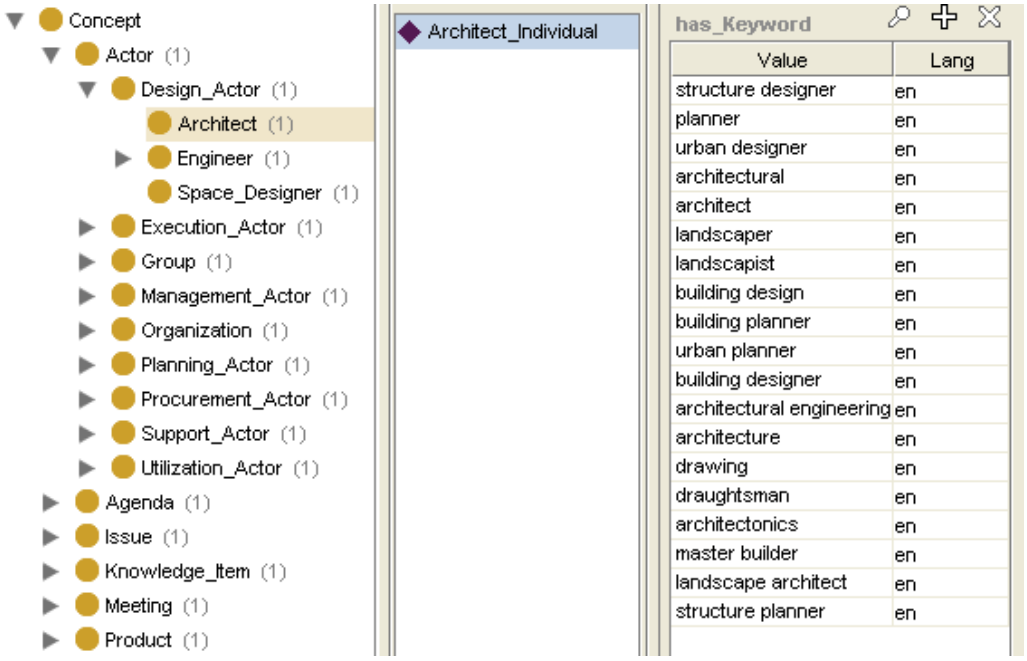

Figure 5.3: Individual and keywords for concept 'Architect'

5.3 Databases

The presented work uses three different MySQL[8] databases in order to store knowledge sources and semantic vectors separately, and to map an online persistent model of the ontology to a MySQL database. The knowledge source repository database is fully provided by the Liferay Content Management System[9] document repository, and its sole purpose is to store knowledge sources. The persistent ontological model is created by Jena Semantic Framework, and it is used to enable access to the ontology through the Web, and also protecting its contents, since the system never interacts directly with the ontology, but rather with the persistent model. Thus, more explanations about these matters are unneeded. The SEKS database has some built-in routines, which are described in Table 5.3.

Table 5.2: SEKS Database MySQL routines

Routine	Input : type	Description
getAllDocumentIDs		Fetches all primary keys from table *Document*
getDocumentNumWithConcept	*concept* : varchar	Fetches the number of primary keys from instances of table *KeywordBasedSemanticWeight* that have their *concept* fields equal to the input parameter
getKeywordBasedWeightsWith DocID	*documentID* : int	Selects all instances of table *KeywordBasedSemanticWeight* that have their *Document_idDocument* fields equal to the input parameter
getMaxTaxonomyRelation Occurrences		Selects the instance of table *TaxonomyRelation* that has the higher value for field *occurrences*
getNotIndexedDocumentIDs		Selects all instances of table *Document* that have their fields *isIndexed* equal to *false* (0)
getOntologyBasedWeightsWith DocID	*documentID* : int	Selects all instances of table *OntologyBasedSemanticWeight* that have their *Document_idDocument* fields equal to the input parameter
getOntologyRelationOccurrences	*concept1* :varchar	Selects all instances of table

[8] (Oracle Corporation, 2011)
[9] (Liferay Inc., 2011)

	concept2 : varchar	*OntologyRelation* where the fields *subject* and *object* are equal to the input parameters
getRelationImportanceWith Concepts	*concept1* :varchar *concept2* : varchar	Selects all instances of table *RelationImportance* where the fields *subject* and *object* are equal to the input parameters
getRelationImportanceWith MinimumThreshold	*threshold* : int	Selects all instances of table *RelationImportance* where the field *importanceThreshold* is equal to the input parameter
getStatisticWeightsWithDocID	*documentID* : int	Selects all instances of table *StatisticWeight* that have their *Document_idDocument* fields equal to the input parameter
getTaxonomyBasedWeightsWith DocID	*documentID* : int	Selects all instances of table *TaxonomyBasedSemanticVector* that have their *Document_idDocument* fields equal to the input parameter
getTaxonomyRelation Occurrences	*concept1* :varchar *concept2* : varchar	Selects all instances of table *TaxonomyRelation* where the fields *subject* and *object* are equal to the input parameters
getTotalDocumentNum		Fetches the total number of instances of table *Document*
insertKeywordBasedWeight	*concept* : varchar *weight* : double *documentID* : int	Inserts a new instance on table *KeywordBasedSemanticWeight*, setting its values with the input parameters
insertOntologyBasedWeight	*concept* : varchar *weight* : double *documentID* : int	Inserts a new instance on table *OntologyBasedSemanticWeight*, setting its values with the input parameters
insertOntologyRelation	*subject* : varchar *object* : varchar *relation* : varchar	Inserts a new instance on table *OntologyRelation*, setting its values with the input parameters
insertRelationImportance	*property* : varchar	Inserts a new instance on table

	subject : varchar *object* : varchar *threshold* : double	*RelationImportance*, setting its values with the input parameters
insertTaxonomyBasedWeight	*concept* : varchar *weight* : double *documentID* : int	Inserts a new instance on table *TaxonomyBasedSemanticWeight*, setting its values with the input parameters
insertTaxonomyRelation	*subject* : varchar *object* : varchar *relation* : varchar	Inserts a new instance on table *TaxonomyRelation*, setting its values with the input parameters
updateOntologyRelation Occurrences	*concept1* :varchar *concept2* : varchar *relation* : varchar	Increments the field occurrences of all instances of table *OntologyRelation* that have their fields *subject*, *object* and *relation* equal to the input parameters
updateTaxonomyRelation Occurrences	*concept1* :varchar *concept2* : varchar *relation* : varchar	Increments the field occurrences of all instances of table *TaxonomyRelation* that have their fields *subject*, *object* and *relation* equal to the input parameters

5.4 SEKS Process Detail

This chapter thoroughly explains the document indexation and search processes introduced previously, in Chapter 4, and which comprise the main functionalities provided by SEKS system.

5.4.1 Knowledge Source Indexation and Semantic Vector Creation

The implemented knowledge source indexation process, represented in Figure 5.4, is consists of two main processes: knowledge extraction and semantic vector creation.

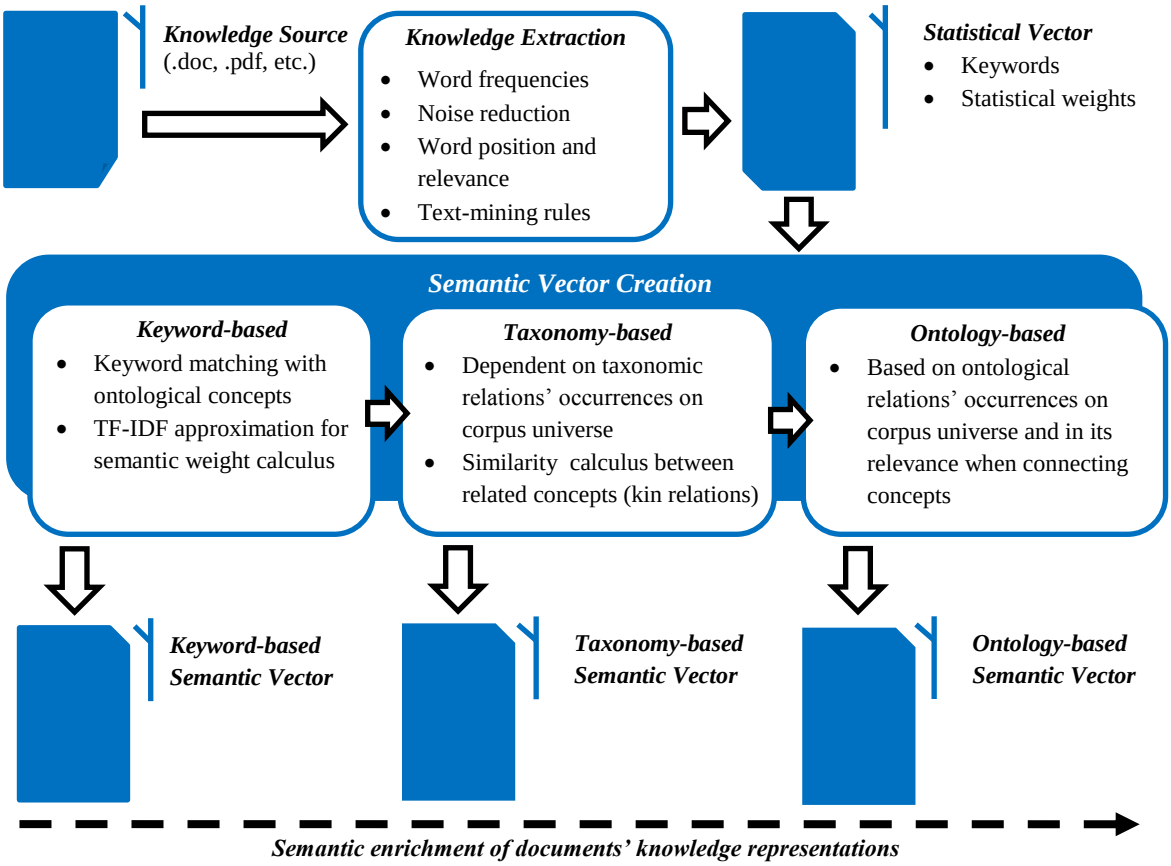

Figure 5.4: Knowledge source indexation process

This chapter briefly presents the Knowledge Extraction Module, and then makes a thorough focus on the process of semantic vector creation.

5.4.1.1 Keyword-based Semantic Vector Creation

The keyword-based semantic vector creation comprehends the matching of keywords from a statistic vector with keywords present in the Building and Construction domain-specific ontology, the subsequent extraction of the concepts related semantically with matched keywords, and the attribution of semantic weights to each concept, depending on its relevance within the knowledge sources' universe.

The first two functions, namely keyword matching and concept extraction, are based on extracting all the keywords and equivalent terms associated to concepts in the ontology, comparing them with the statistic vector's keywords, and subsequently creating a list of all the concepts on the ontology tree which have one or more matched keywords associated. Such process depends on Knowledge Extraction Module results and can face some issues. The external tool used as the Knowledge Extraction Module, RapidMiner, stems keywords, meaning that one keyword from RapidMiner's statistic vector can match several ontology keywords, because SEKS ontology's keywords are not stemmed. For instance, if the statistic vector has the word *land* as keyword,

matches range from the actual word *land*, to *landscape*, *landlord*, etc. On the other hand, all of these results are legitimate, and can be considered right, since the system has no way to tell which the original word in the knowledge source was. Hence, this process contemplates all ontology keywords that start with the statistical vector's stemmed version.

Other problem is that keywords are not formed only by one word. Both statistic vector and ontology have keywords with two or more words, but statistic vector's keywords contain only stemmed words. The presented work approaches this problem by only comparing statistic vector and ontology keywords that have the same word count. The statistical weight for each concept is given by:

$$w_c = \sum_{i=1}^{n} w_{k_i} \tag{16}$$

In Equation 16, n is the total number of keywords from the statistic vector that are associated with the concept, w_c is the resultant statistic weight for the concept and w_{k_i} is the statistic weight for keyword i, associated to the concept. This means that the total statistic weight of a concept within a knowledge source is defined by the sum of the statistic weights assigned to the keywords associated with the concept, if the source contains such keywords, or zero otherwise. For instance, if an ontological concept has two keywords associated to it, with statistic weights 0.05 and 0.02, respectively, then the total statistic weight for that ontological concept would be 0.07.

Equation 16 introduces an important question: what happens if a keyword present in the statistic vector has no match in the domain-specific ontology? If a statistic vector's keyword is not matched with any ontology concept, and since SEKS system uses a domain-specific ontology, this could imply that the keyword is not part of the construction domain or that it has no underlying meaning in the context of this domain. Nevertheless, SEKS system has a functionality that unites statistic and semantic vectors, as explained in Chapter 5.4.1.4, so that unmatched keywords can also be used when comparing vectors.

Again, the aim of SEKS System is to improve the specificity of IR results according to user's context and needs, in this case someone who plays a role on the construction domain. As example, if an architect queries a general search engine for the word "door", the results can range from relevant (e.g. a door catalog) to completely irrelevant (e.g. album covers from the band The Doors), even if the query comprises other keywords which enrich its semantic meaning, regarding the construction domain. The result of the application of Equation 16 on all ontology concepts extracted from a knowledge source is a vector comprising those concepts and the corresponding statistical weights. In this stage, the resultant vector is not yet considered a semantic vector.

The next step further into the knowledge source indexation process is the attribution of semantic weights to each of the concepts. A semantic weight is a weight attributed to a concept

according to its semantic meaning within the construction domain, and regarding not only the current source's context, but also the different contexts of all knowledge sources comprised within the SEKS system. The SEKS approach uses an approximation to the TF*IDF family of weighting functions, introduced in Chapter 3, to calculate the semantic weight for each concept resultant from the concept extraction process. The TF*IDF[10] approximation algorithm used is given by the expression:

$$w_x = \frac{w_{x,d}}{max_y w_{y,d}} \cdot \log \frac{\mathcal{D}}{n_x} \tag{17}$$

In Equation 17, $w_{x,d}$ is the statistical weight for concept x in knowledge source d's statistical vector, $max_y w_{y,d}$ is the statistical weight of the most relevant concept, y, within the statistical vector of knowledge source d, $\mathcal{D}$ is the total number of knowledge sources present in the SEKS search space, n_x is the number of sources present in the search space which have concept x in their semantic vectors, and w_x is the resultant semantic weight of concept x for knowledge source d.

The keyword-based semantic vector is then stored in the database in the form $\left[\sum_{i=1}^{n} x_i \; ; \; \sum_{i=1}^{n} w_{x_i}\right]$, where n is the number of concepts in the vector, x_i is the syntactical representation of the concept and w_{x_i} is the semantic weight corresponding to concept i. Statistical normalization is performed over the keyword-based semantic vector's weights, in order to obtain values between zero (0) and one (1). This normalization is computed as:

$$w_x^* = \frac{w_x}{\sum_{i=1}^{n} w_{x_i}} \tag{18}$$

In Equation 18, w_x is the semantic weight for concept x for knowledge source d, n is the total number of concepts in source d's semantic vector, w_{x_i} is the semantic weight for concept i, and w_x^* is the normalized semantic weight of concept x for source d. Hence, normalization is simply dividing each semantic weight by the total sum of semantic weights present in knowledge source d's semantic vector. This will be crucial for the upcoming vector comparison result ranking processes, because it will ease the computation processes needed and the attribution of relevance percentage to the results. The last step is to store both concepts and semantic weights from the keyword-based semantic vector into table *KeywordBasedSemanticWeight* on SEKS database, through a call to the storage procedure *insertKeywordBasedWeight*. At the end of any semantic vector creation process, semantic vectors are stored into their corresponding tables, depending on if such vectors are keyword-, taxonomy or ontology-based. SEKS database has built-in storage procedures for each of the three semantic vector iterations.

[10] (Castells, et al., 2007)

5.4.1.2 Taxonomy-Based Semantic Vector Creation

The next iteration on semantic vector creation is to define a taxonomy-based vector, based on the relations of kin between concepts within the ontological tree. As previously referred, if two or more concepts are taxonomically related, this underlying relation may trigger two different processes[11]:

- **Process 1: When C_x (an ontology concept that belongs to the semantic vector) is taxonomically related to C_y (another ontology concept), and C_y is also present on the semantic vector.**

In this case, the weights corresponding to C_x and C_y are boosted within the semantic vector, by applying the following equations:

$$tw_{C_x} = w_{C_x} + \sum (all\ related\ C_y s) \left[w_{C_y} * \left(TI_{C_x C_y} \right) + \text{Co-Occurrence}_{C_x C_y} \right] \quad (19)$$

$$tw_{C_y} = w_{C_y} + \sum (all\ related\ C_x s) \left[w_{C_x} * \left(TI_{C_x C_y} \right) + \text{Co-Occurrence}_{C_x C_y} \right] \quad (20)$$

Such weight boost is only performed if the taxonomic importance $TI_{C_x C_y}$ is greater or equal than a certain threshold. This constraint only accepts the weight boost if the two related concepts are linked by a relation that is strong (i.e. both concepts are taxonomically near in the ontology tree). $TI_{C_x C_y}$ is given by homologous and non-homologous similarity equations, Equations 3 and 4, presented in Chapter 3. In the case of Equation 5, the weight boost is not applied for obvious reasons. There are two different thresholds for, respectively, homologous and non-homologous concepts.

In Equations 19 and 20, tw_{C_x} and tw_{C_y} are the new taxonomy-based semantic weights for concepts C_x and C_y, respectively; w_{C_x} and w_{C_y} are the weights present on the keyword-based semantic vector, for concepts C_x and C_y, respectively. $\text{Co-Occurrence}_{C_x C_y}$ represents the frequency of occurrence of both concepts over the knowledge sources' search space, and it is computed by an approximation to the IDF formula, presented in Equation 6:

$$\text{Co-Occurrence}_{C_x C_y} = idf\left(C_x C_y\right) = \log \frac{\mathcal{D}}{n_{C_x C_y}} \quad (21)$$

In Equation 21, $\mathcal{D}$ is the total number of knowledge sources in the system's repository, and $n_{C_x C_y}$ is the number of knowledge sources that have concepts C_x and C_y in their keyword-based semantic vectors. Afterwards, the semantic vector's weights have to be normalized again, using Equation 18, so that each weight represents a percentage of relevance on the knowledge representation of a knowledge source again.

[11] (Nagarajan, et al., 2007)

In the presented work, parameters α and β, used in Equations 3 and 4, are adjusted in a different fashion, in order to reduce human intervention in the semantic vector creation process. Both parameters are calculated according to the usage of each kin relation within the knowledge source corpus' universe. More specifically in Equation 3, parameter β adjusts the importance of the distance between concepts, and parameter α adjusts the relevance given to the depth of root concept A. The existent homologous relation dictates that parameter β has a bigger relevance, because the depth of the root concept does not define the proximity relation between the homologous concepts. Hence, for Equations 3 and 4, parameters α and β are expressed as:

$$\beta = \frac{\text{number of vector occurrences for taxonomical pair } A, B}{\text{number of occurrences for the most frequent taxonomical related pair}} \tag{22}$$

$$\alpha = 1 - \beta \tag{23}$$

Equation 22 states that parameter β is computed by dividing the total number of occurrences, in the system's search space, of the taxonomical pair (A, B), by the total number of occurrences of the most frequent taxonomical related pair of concepts in the search space.

Parameter α is considered to be the inverse of parameter β, as stated by Equation 23, which implies that if concepts A and B appear often on the system's semantic vector universe, then the distance between them will be much more relevant than the depth of ancestor concept A. If, in contrast, the pair (A, B) is not a frequently occurring pair, then the relevance given to the distance between concepts is downgraded in favor of root concept A's depth.

- **Process2: When C_x (an ontology concept that belongs to the semantic vector) is taxonomically related to C_y (another ontology concept), and C_y is not present on the semantic vector.**

In this case, C_x is not modified and C_y is added to the semantic vector, and its weight is computed as:

$$tw_{C_y} = w_{C_y} + \sum (all\ related\ C_x s) \left[w_{C_x} * \left(TI_{C_x C_y} \right) \right] \tag{24}$$

$$w'_{C_x} = w_{C_x} \tag{25}$$

In this case, the system has to calculate the TF*IDF weight for concept C_y, w_{C_y}, which brings a conceptual problem: C_y does not possess any statistic weight resulting from the Knowledge Extraction process. The chosen approach in this case is to apply only the IDF term of Equation 17. This is made by attributing the value one (1) to the divisor and the dividend of the TF term of Equation 17, which means IDF is applied exactly according to Equation 6.

As in the previous process, the new concept is only added to the taxonomy-based semantic vector if the taxonomic relevance, $TI_{C_xC_y}$, is greater or equal than a threshold. $TI_{C_xC_y}$ is computed as in the previous process. When both processes finish, the taxonomy-based semantic vector is stored in the database.

5.4.1.3 Ontology-Based Semantic Vector Creation

The third iteration of the semantic vector creation process is the definition of the semantic vector based on the ontological relations' patterns present in the knowledge source corpus. The first step is to analyze the ontological relations between concepts present on the input semantic vector. In this case, both keyword- and taxonomy-based semantic vectors can serve as inputs for this analysis. As in taxonomy-based semantic vector creation, there are two processes involved on the ontological relationship analysis: the first boosts weights belonging to concepts within the input semantic vector, depending on the ontology relations between them; the second ads concepts that are not present in the input vector, according to ontological relations they might have with concepts belonging to the vector:

- **Process 1: When C_x (an ontology concept that belongs to the semantic vector) is ontologically related to C_y (another ontology concept), and C_y is also present on the semantic vector.**

Consider two concepts that are linked by an ontological relation, and occur on the input vector. Then, an ontological boost is applied to the concepts' weights, increasing the relevance of these concepts within the source's representation. Such boost is calculated recurring to the following equations:

$$ow_{C_x} = w_{C_x} + \sum (all\ related\ C_ys) \left[w_{C_y} * \left(OI_{C_xC_y} \right) + \text{Co-Occurrence}_{C_xC_y} \right] \quad (26)$$

$$ow_{C_y} = w_{C_y} + \sum (all\ related\ C_xs) \left[w_{C_x} * \left(OI_{C_xC_y} \right) + \text{Co-Occurrence}_{C_xC_y} \right] \quad (27)$$

In Equations 26 and 27, ow_{C_x} and ow_{C_y} are the new ontology-based semantic weights and w_{C_x} and w_{C_y} are the input vector's weights for concepts C_x and C_y, respectively. $\text{Co-Occurrence}_{C_xC_y}$ is computed with Equation 6, but this time it will be used to account for the frequency of occurrence of the ontologically related concepts throughout the knowledge source corpus. It does not make any sense to apply a TF*IDF algorithm in this case, because the TF term is always equal to one (1), due to the fact that the relation importance between two concepts is unique to that pair of concepts. $OI_{C_xC_y}$ is the ontological relation's importance, or relevance, and it is not automatically computed; rather, it is retrieved from the system's database, where it is stored on a table filled by the ontology manager. This is the only non-automatic step on SEKS knowledge

source indexation process but there are already systems that perform this kind of automatic importance analysis over semantic and ontological relations, as is the case of SemRank[12].

- **Process 2: When C_x (an ontology concept that belongs to the semantic vector) is taxonomically related to C_y (another ontology concept), and C_y is not present on the semantic vector.**

In this case, and again as in the taxonomy-based semantic vector creation process, C_x is not modified and C_y is added to the semantic vector, and its weight is computed as:

$$ow_{C_y} = w_{C_y} + \sum (all\ related\ C_x s) \left[w_{C_x} * \left(OI_{C_x C_y} \right) \right] \quad (28)$$

$$w'_{C_x} = w_{C_x} \quad (29)$$

Once more, the system has to calculate the TF*IDF weight for concept C_y, which presents the conceptual problem introduced previously, and again the chosen approach is to apply only the IDF term of Equation 6. As in Taxonomy-based semantic vector creation, the new concept is added to the ontology-based semantic vector only if the ontological relation importance, $OI_{C_x C_y}$, is greater than or equal to a threshold, for the same constraint purposes. Finally the ontology-based semantic vector is stored in the database.

5.4.1.4 Vector Union

After all three iterations of the semantic vectors creation process are concluded, there is still one more step: create a unified vector for each knowledge source, which results from the union between the statistic vector and the three types of semantic vectors, depending again on the specificity of the search. This union process will further enrich the semantic vector with the statistic vector's keywords that do not match any ontology concept. Using only the semantic vector for comparison and retrieval would mean performing such comparison having only as basis the keywords in the knowledge source that overlap with concepts in the SEKS ontology. In cases of a limited domain model, or absence of ontology concepts associated to the source, one can imagine that the semantic vector would be very sparse. Considering that the goal of the presented work is to augment the knowledge representation of the knowledge source, the SEKS system uses a combination of both statistic and semantic vectors.

5.4.2 Queries, Knowledge Source Comparison and Result Ranking

This chapter describes the processes of query treatment and vector comparison, introduced in Chapter 4.

[12] (Kemafor, et al., 2005; Nagarajan, et al., 2007)

5.4.2.1 Treating Queries

Queries are treated like pseudo-knowledge sources. This means that all queries suffer an indexation process similar to the one applied to such sources (Figure 5.5). Initially, the query is divided into keywords. These keywords are then used to create a statistic vector for the query, equal to the statistic term-frequency vector used for knowledge source indexation. But, instead of passing the query through the knowledge extraction process the statistic vector is created by giving the same statistic weight to all keywords contained in the query. Such rule implies that the system assumes the same importance to all of the query's keywords, because it does not know which keywords are more relevant to the system's users.

Furthermore, the attribution of equal weights increases SEKS system's performance: while knowledge source indexation may be made server-side, at a scheduled time of the night (SEKS can have all its computational power directed to the indexation of several knowledge sources gathered in the course of a day) query indexation has to be made on-the-fly, because users do not want to wait for a clinical and time wasting indexation, but rather swift and clean results retrieval. Considering the above condition, the query indexation process is much lighter than the knowledge source indexation process, because it encompasses only keyword-based semantic vector creation and it does not need the knowledge extraction mechanisms for gathering keywords and statistic weights. The statistic weights for each keyword are computed as follows:

$$w_{x,q} = \frac{1}{n_q} \tag{30}$$

In Equation 30, n_q is the total number of keywords contained in the query q and $w_{x,q}$ is the resultant statistic vector for keyword x contained in q. This calculus entails that the sum of all of the keywords' statistic weights is equal to one (1), and that all keywords bounded to a query possess the same value.

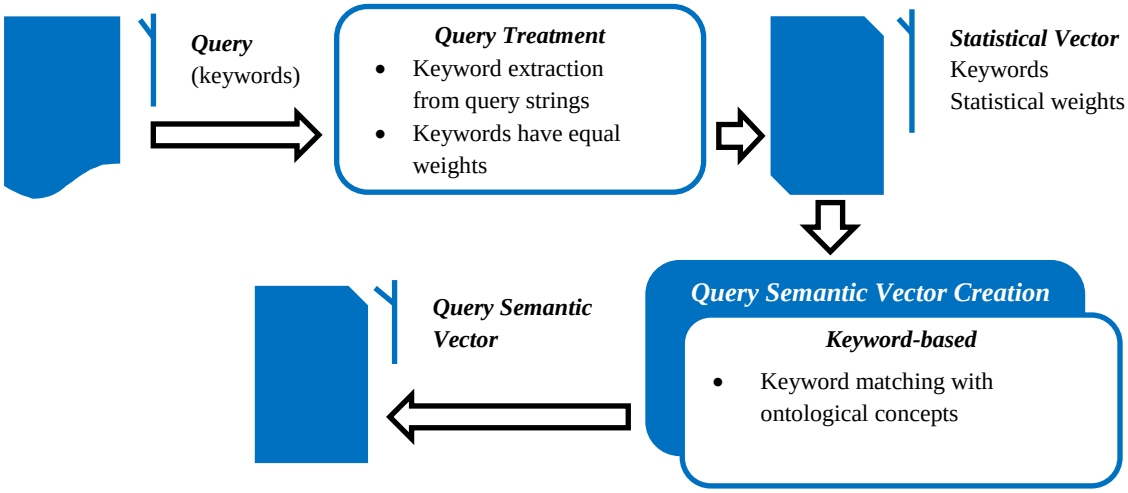

Figure 5.5: Query Indexation Process

:The subsequent phase in query indexation is exactly the same has in knowledge source indexation. First, keywords from the statistic vector are matched against ontology keywords, and the correspondent ontology concepts are extracted. Then, the statistic weights corresponding to the extracted concepts are calculated using Equation 16. Following the knowledge source indexation process, the next stage would be the application of the TF*IDF approximation algorithm over the concepts and weights' vector. In the case of queries, the TF*IDF algorithm is not applied, nor is the query's vector inputted into any of the taxonomy-based and ontology-based vectors' creation processes. This occurs for the same reason that makes all the weights of a query's keywords equal: the system cannot guess what keywords may be more meaningful in the user's context.

Subsequently, the query's semantic vector goes through the weights normalization process, using Equation 18. The output vector is in all ways equal to a keyword-based semantic vector. This vector form is again crucial to the knowledge source comparison process due to the fact that the queries' vectors are now ready to be compared with knowledge sources' semantic vectors, in terms of their semantic similarity, as described next. Finally statistic and semantic vector union is made, just as in the process of knowledge source indexation. As an example, the system's interface also allows users to query the system in yet another way that does not need query treatment or statistic and semantic vector union: search by ontology concepts. In this case, instead of the normal search field, the interface provides a visual ontology tree, from which users can drag-and-drop concepts to a search box, and subsequently perform a search. Thus, as previously mentioned, the query treatment process is not needed, although a statistic vector is created, just like in the keyword-based search. An example will be given later on in this chapter.

5.4.2.2 Comparing and Ranking Knowledge Sources

SEKS system performs knowledge source comparison by calculating the similarity between their semantic vectors. In this case, comparison refers to the similarity computation between the query's semantic vector in one side, and all the semantic vectors which define knowledge representations of knowledge sources in SEKS search space. The SEKS approach for vector similarity calculus takes into account the Euclidian distance between two vectors, denominated cosine, and the sparse-matrix multiplication method, which is based on the observation that a scalar product of two vectors depends only on the coordinates for which both vectors have nonzero values, as introduced on Chapter 3. This chapter also states that the cosine of two vectors is defined as the inner product of those vectors, after they have been normalized to unit length. Let d be the semantic vector representing a knowledge source and q the semantic vector representing a query. The cosine of the angle θ between d and q is given by[13]:

[13] (Castells, et al., 2007; Li, 2009)

$$\cos\theta = \frac{d}{\|d\|} \cdot \frac{q}{\|q\|} = \frac{\sum_{k=1}^{m} w_{dk} w_{qk}}{\sqrt{\left(\sum_{k=1}^{n} w_{dk}^2\right)\left(\sum_{k=1}^{m} w_{qk}^2\right)}}, \text{ with } n > m \tag{31}$$

In equation 31, m is the smallest vector's size, n is the biggest vector's size, w_{dk} is the weight for each concept that represents d and w_{qk} is the weight for each concept present on the query vector q. Equation 31 entails an important limitation: the vectors' sizes have to be equal. To surpass this issue, and because almost all semantic vectors have different sizes, a sparse-matrix multiplication approach is introduced, as stated in Chapter 3.2.3. Formally, applying Equation 12 to Equation 31, $f_1(w_{dk}, w_{qk})$, $f_2(w_{dk})$, $f_3(w_{qk})$ and f_0 are given by, repectively:

$$f_1(w_{dk}, w_{qk}) = \sum_{k=1}^{m} w_{dk} w_{qk} \tag{32}$$

$$f_2(w_{dk}) = \sqrt{\sum_{k=1}^{n} w_{dk}^2} \tag{33}$$

$$f_3(w_{qk}) = \sqrt{\sum_{k=1}^{m} w_{qk}^2} \tag{34}$$

$$f_0\left(f_1(w_{dk}, w_{qk}), f_2(w_{dk}), f_3(w_{qk})\right) = \frac{f_1(w_{dk}, w_{qk})}{f_2(w_{dk}) f_3(w_{qk})} \tag{35}$$

As also mentioned in Chapter 3.2.3, both vectors are considered to have the same size, because the smallest vector is complemented with $m - n$ zero-valued entries. The result of the vector similarity algorithm is a list with relevance percentages for each knowledge source present in SEKS search space. These relevance percentages enable an easy result ranking process, just by applying a simple sorting and ranking technique, like quick sort or bubble sort algorithms, for instance. In this work, a bubble sort algorithm is used.

5.5 SEKS Architecture

SEKS system was implemented as a Java Web application, compliant with Java 6, Java Servlets 3 and JAX-WS 2.2.6 and built to run on Apache Tomcat 7. The system was developed using Eclipse[14] Integrated Development Environment (IDE) and its configuration files, Java packages and class structure are shown in Figure 5.6.

The *.owl* file stores the SEKS ontology. It is available to be accessed by the persistent model, whenever there is a need for it, such as the creation of a new version of the ontology, or a swap of domain-specific ontologies. The three *.xml* files are database access configuration files that

[14] (Eclipse Foundation, 2004)

configure databases' hosts, ports, databases names and MySQL usernames and passwords: *jenaConfig.xml* configures the access to the persistent model of the ontology; *lportalConfig.xml* manages the database connection with Liferay's document repository; and *svdbConfig.xml* is responsible for the access to statistic and semantic vectors from SEKS database.

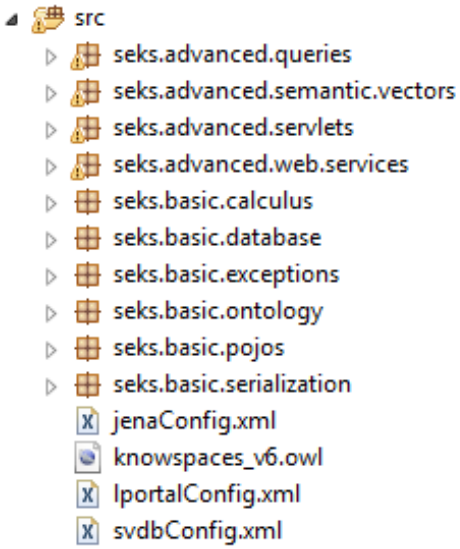

Figure 5.6: SEKS system class structure

SEKS system includes basic and advanced service packages. The implementation classes presented in Figure 5.6 were designed and modeled using UML Class Diagrams (UCD's) and UML Sequence Diagrams (USD's), and are the subject of the following subchapters.

5.5.1 Static Vision

SEKS static vision is represented using UCD. Such diagrams visually introduce class packages, their respective classes and interfaces, their components and how classes are related to each other. UCD's will be presented following the high-level architecture presented in Subchapter 4.2.2.

5.5.1.1 Basic Services

Basic services contain five class packages, beginning with the package name *seks.basic*, four of which implement its services, as shown in Figure 4.8: *calculus*, *database*, *serialization* and *ontology*. The last class package, *seks.basic.pojos*, comprises Java object classes needed in the process for system performance and database data retrieval purposes, and is represented in Figure 5.7.

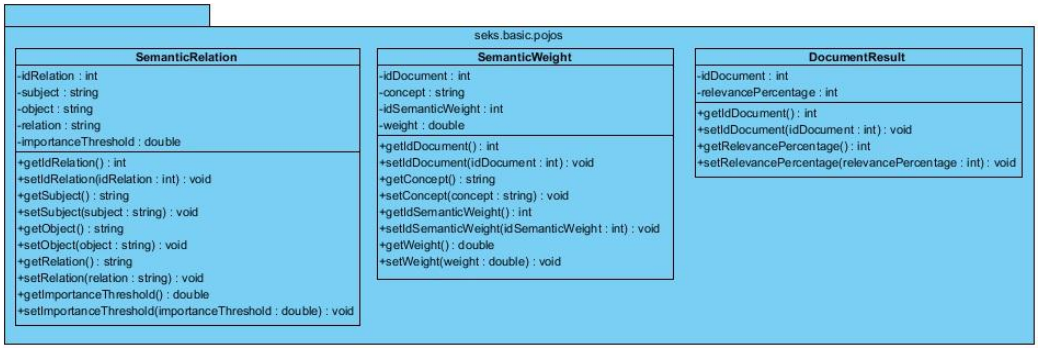

Figure 5.7: UCD - Plain Old Java Objects (POJOs) classes

Class package *seks.basic.calculus*, represented in Figure 5.8, contains the TF*IDF algorithm, a vector normalization function, the homologous and non-homologous factor computation algorithm and the Euclidian distance algorithm.

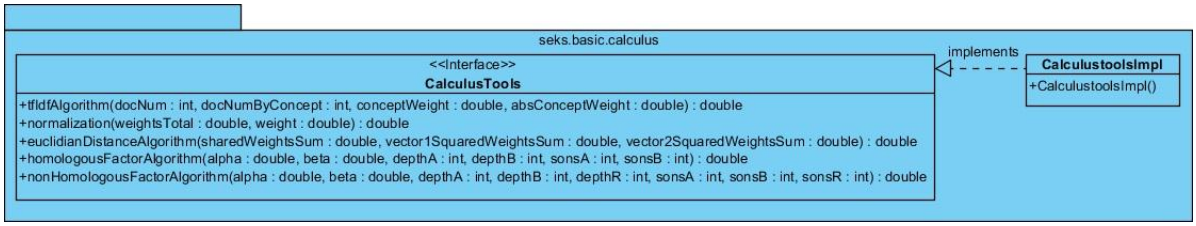

Figure 5.8: UCD - Calculus Services classes and interfaces

Class package *seks.basic.database*, shown in the UCD of Figure 5.9, is responsible for opening and closing MySQL connections to interact with the system's databases and repositories, and for calling database routines presented in Table 5.3.

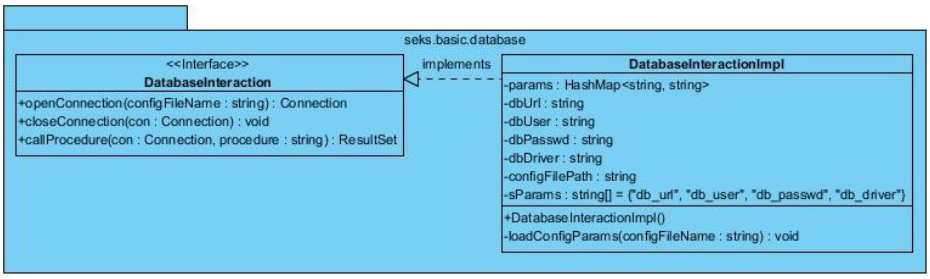

Figure 5.9: UCD - Database Services classes and interfaces

Class package *seks.advanced.ontology* (Figure 5.10) has two interfaces with corresponding classes: *OntologyPersistence.java* and *OntologyInteraction.java*. *OntologyPersistence.java* class creates a database map of the ontology for online interaction with SEKS or other systems that use the SEKS Web Services Interface. *OntologyInteraction.java* contains all methods that interact with SEKS ontology. These methods are supported by the Apache Jena Semantic Framework.

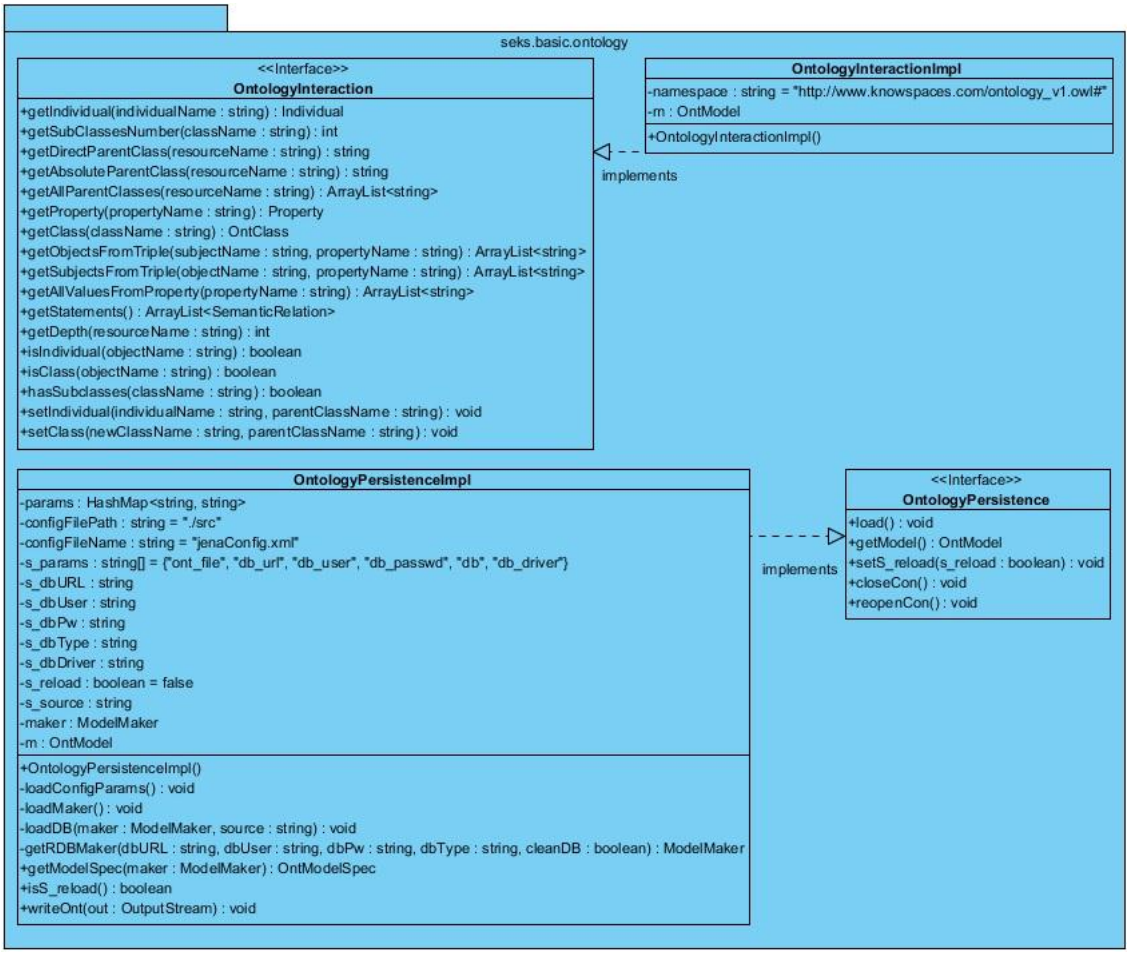

Figure 5.10: UCD - Ontology Services classes and interfaces

Finally, class package *seks.basic.serialization* (Figure 5.11) is responsible for the serialization and deserialization methods used by the Web Services Interface to transmit responses to other systems that use SEKS functionalities.

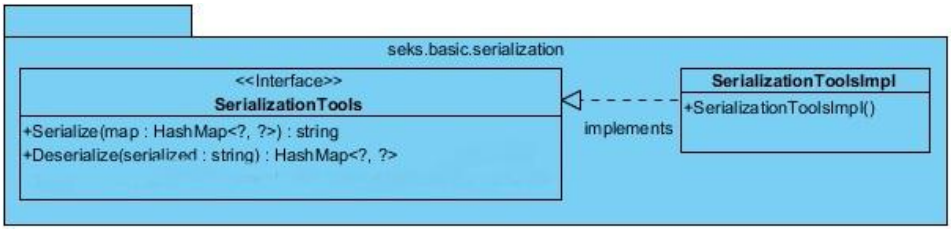

Figure 5.11: UCD - Serialization Services classes and interfaces

Such transmission is often made with XML strings, as is the case for the presented work. Serialization mechanisms were needed specifically because semantic vectors are managed by the system as *java.util.HashMap* objects, which are not automatically serialized by JAX-WS framework.

5.5.1.2 Advanced Services

Advanced Services cover two packages: one for query handling and the other for knowledge source semantic vector creation and vector comparison. Package *seks.advanced.sematic.vectors* (Figure 5.12) manages the creation of all three semantic vector iterations and also handles vector comparison. It comprises one class and one interface for each of the processes mentioned. *KeywordBasedSVCreation.java* class interacts with all Basic Services (except *seks.basic.serialization* class package which only interacts exclusively with the Web Services Interface) to create keyword-based semantic vectors.

The same interaction applies to *TaxonomyBasedSVCreation.java* and *OntologyBasedSVCreation.java*, which responsibility is to create the respective taxonomy- and ontology-based semantic vectors. Finally, SemanticVectorComparison.java handles all methods needed for vector comparison, including statistic and semantic vector union and interacting also with all Basic Services class packages.

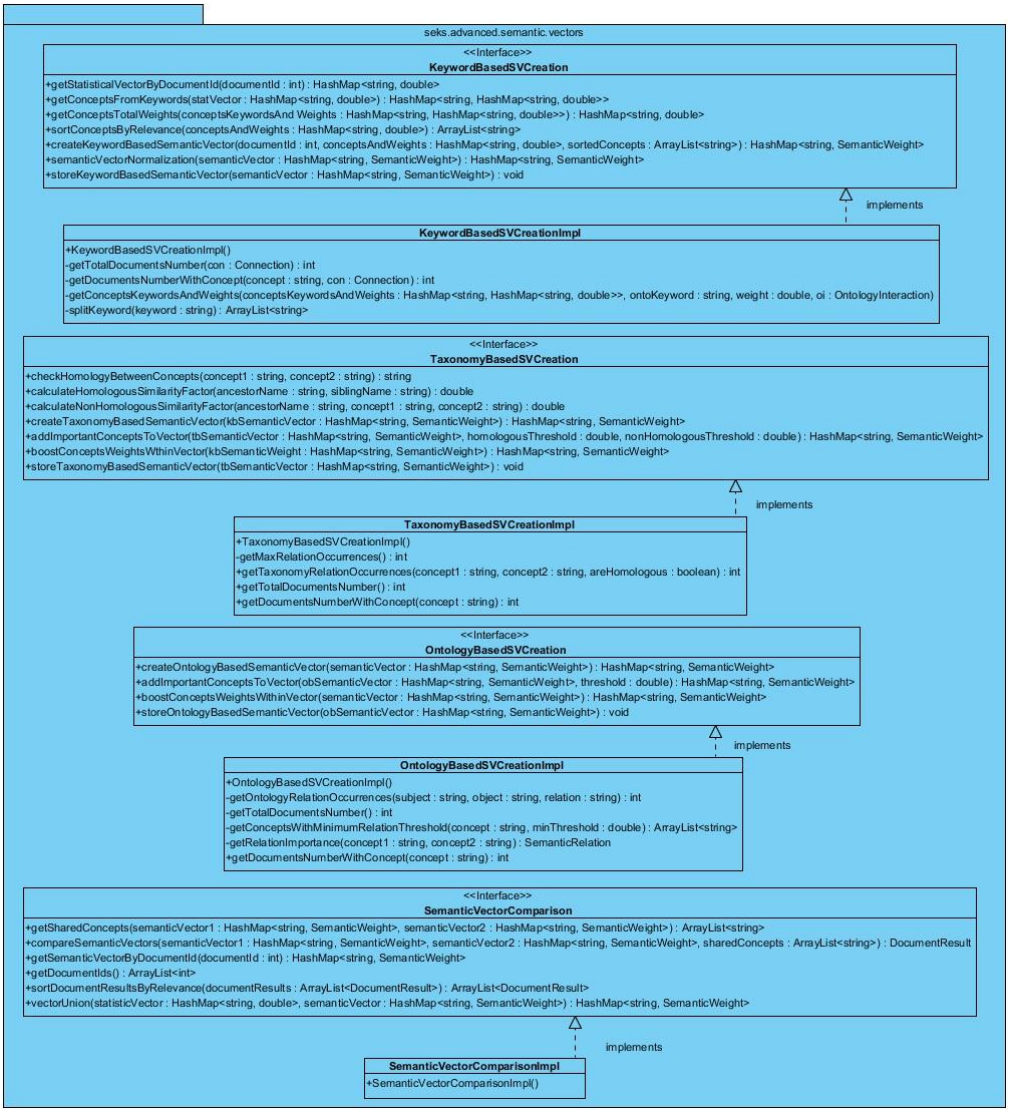

Figure 5.12: UCD - Semantic Vector Services and Document Comparison Services classes and interfaces

Package *seks.advanced.queries* (depicted in Figure 5.13) is responsible for splitting query strings into keywords, creating statistic and semantic vectors for queries, and to get all ontology keywords, used by User Interface for autocomplete purposes over the keyword search field. When a user starts typing its query in the User's Interface search field, an autocomplete mechanism is triggered so that ontology keywords that start with the letter or letters inserted by the user are shown below the search field. Users can then select the desired keyword if it exists in the ontology, saving time in query typing. If an user's keyword does not exist in the ontology, the autocomplete is disabled until the next keyword insertion.

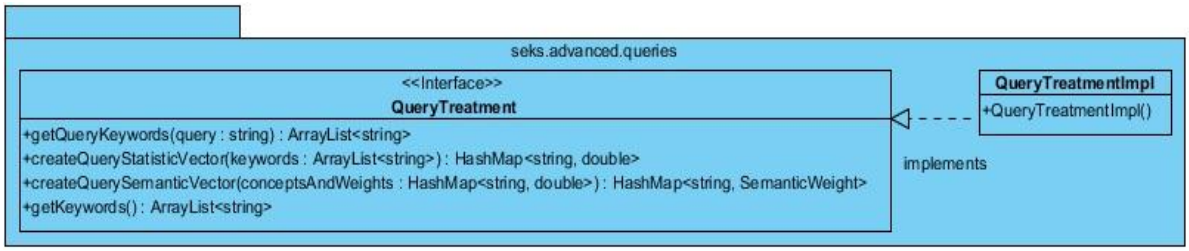

Figure 5.13: UCD - Query Treatment Services classes and interfaces

5.5.2 Dynamic Vision

Contrasting with static vision, the presented software infrastructure's dynamic vision presents interactions between and within Java class instances, or objects, showing system's responses for users' actions. Such vision of Java objects is expressed in terms of USD's. Each of these diagrams represents a particular functionality from previously presented UML Use Case Diagrams. USD's necessary for all use cases and the inherent interactions between SEKS Advanced and Basic Services classes and interfaces are presented in Appendix A.

Requests are represented as if they were sent by other system using SEKS Web Services Interface, but processes are the same for SEKS User Interface, with the exception that, instead of being the Web Service class handling requests, in this case, is the Servlet object that processes requests. For this reason, USD's only represent calls to Web Services, and not to Servlets. If it were otherwise, it would be irrelevant, with the only difference being that SEKS User Interface does not use SEKS Serialization Services functionalities. The first use case is that of knowledge source indexation, shown in Figure 4.4, and then the focus will turn to the knowledge source search use case, represented in Figure 4.6.

Keyword-based semantic vector creation is illustrated in Figure A.1, Appendix A. The two processes previous to this one, matching keywords with ontology concepts and compute each concept's total weight is not shown because they are simple, but very vast methods, and their USD's would be enormous. Taxonomy-based semantic vector creation is similar to ontology-based semantic vector creation, being comprised of the concepts' weight boost process and new concepts addition processes. As in the case of the process of matching keywords with ontology concepts,

both boost and addition processes are too vast to be represented in UML Sequence Diagrams. Taxonomy-based semantic vector creation is represented in Figure A.2. All vectors are stored by the same process, shown in Figure A.3. The difference between all three store procedures, corresponding to each of the semantic vector iterations is the MySQL database routine called by each algorithm. Query statistic and semantic vector creation are represented, respectively, in Figures A.4 and A.5. Finally, the process of analyzing the sharing of concepts between two semantic vectors and subsequent vector comparison are shown in Figures A.6 and A.7, respectively.

5.6 Interfaces

The system presents two different interfaces to the exterior. The User Interface acts like a normal search-engine web site, in order to provide a common and intuitive visual interaction with its users. The Web Services Interface is viewed more as a framework, providing functions that can be used by other systems, if their developers wish to use SEKS functionalities. Interfaces' server-side Java classes and Servlets are also presented using UCD's.

5.6.1 Web Services Interface

Classes provided through the Web Services Interface, represented in Figure 5.14, mirror SEKS Advanced Services' classes. *DocumentSemanticVectorsService.java* and *QuerySemanticVectorsService.java* classes provide all mechanisms for semantic knowledge source and query indexation, respectively. *VectorComparisonService.java* offers access to all knowledge source comparison and result ranking capabilities provided by SEKS. Finally, the *ClientSupportService.java* class provides access to the JavaScript Object Notation (JSON) visual ontology tree, used for supporting drag-and-drop ontology concept search, and to the ontology keywords, for instance, for autocomplete purposes. SEKS Web Services are developed with JAX-WS Framework, which automatically generates the Web Service Definition Language (WSDL) files needed for Web Services operation.

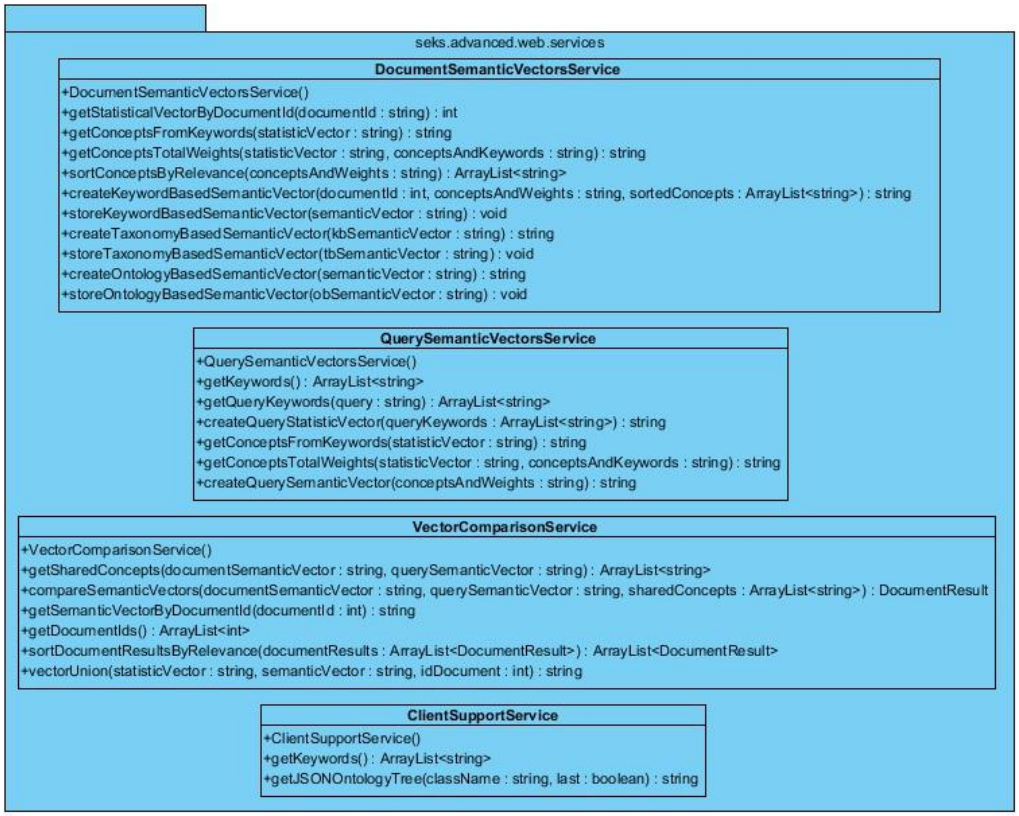

Figure 5.14: UCD - Web Services Interface classes

5.6.2 User Interface

The User Interface classes are Java Servlet classes, as shown in Figure 5.15. Such classes only respond to client-side requests, and react to those requests, sending responses. The exception in this case is *InitDocumentIndexationServlet.java* class that is called directly from server-side, to initiate scheduled knowledge source indexation processes. *KeywordSearchServlet.java* manages users' queries and starts the knowledge source search process. The *UploadFileServlet.java* class interacts with the knowledge source repository in order to upload users' documents. *GetKeywordsServlet.java* is used to fetch all ontology keywords, for autocomplete purposes, as previously mentioned. *ConceptsTreeServlet.java* and *ConceptSearchServlet.java* classes provide support for ontology concept-based search by creating the JSON visual ontology tree, and for performing subsequent search based on ontology concepts chosen from the JSON tree, respectively.

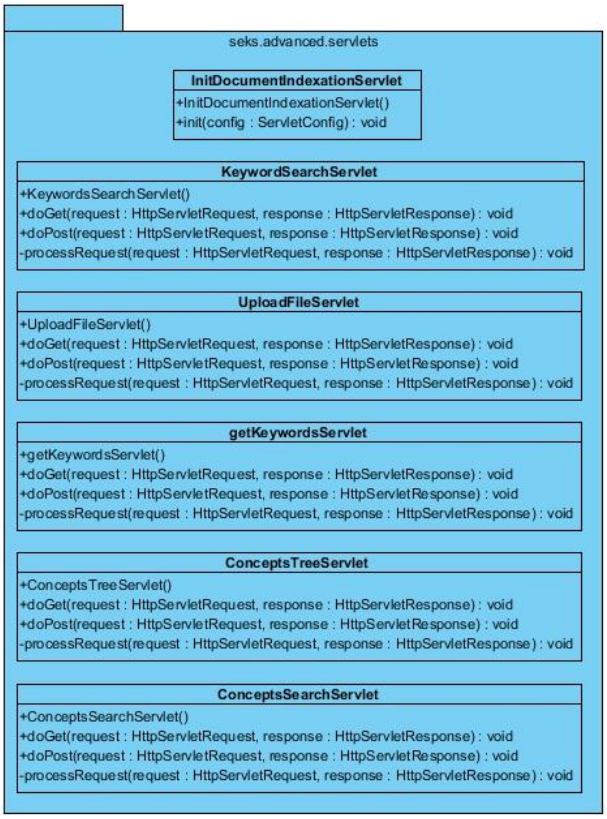

Figure 5.15: UCD – User Interface classes

The User Interface is shown in Figure 5.16. The client-side application allows users to upload knowledge sources and to search SEKS' knowledge source search space through keyword- and ontology concept-based queries, as will be presented in the following Chapter.

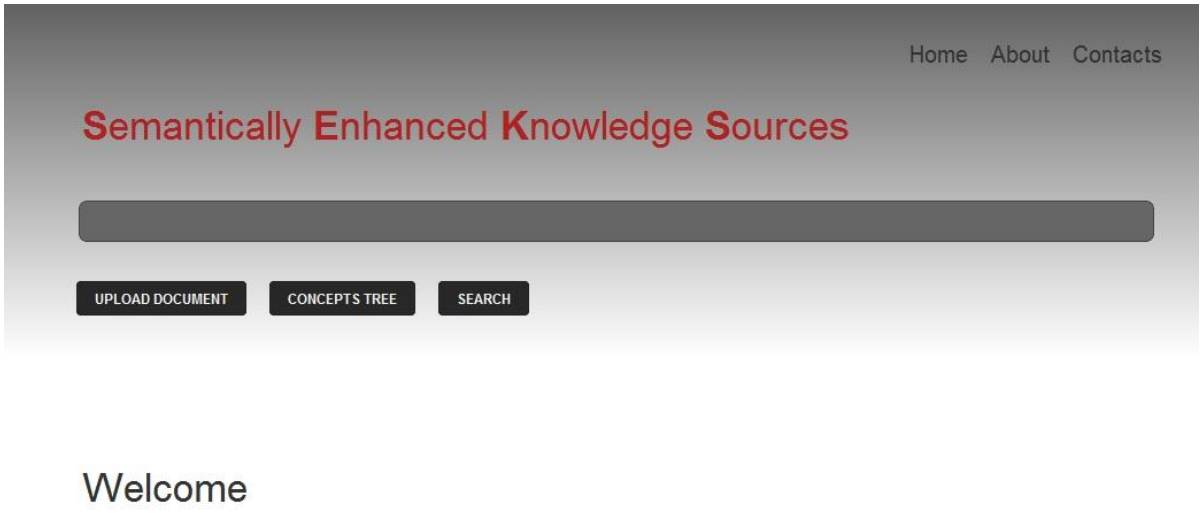

Figure 5.16:SEKS User Interface

6 ASSESSMENT

This chapter illustrates the assessment process of the SEKS system. First, the document upload process will be explained in relation to the client-side application. Finally, both query types are exemplified. The User Interface is encompassed by the visual search environment shown in Figure 5.16. In addition to the regular *About* and *Contacts* links, users can choose from three different functionalities: *Upload Document*, *Concepts Tree* or *Search*. The *Upload Document* link opens the knowledge source upload popup window, shown in Figure 6.1, which is supported by jQuery. Such support provides AJAX-RPC calls to the server, uploading selected files without having to reload the whole application. This enables the system to process initial information only once, improving performance.

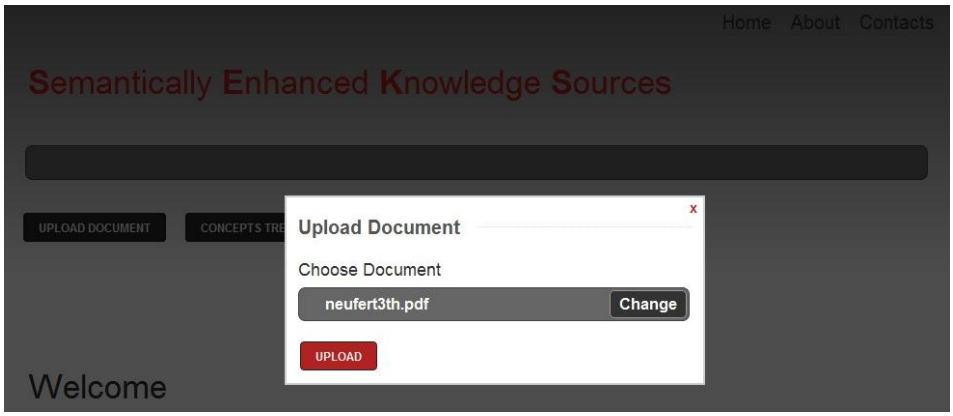

Figure 6.1: User Interface - Upload knowledge source function

When a knowledge source is uploaded, it is directly stored in the knowledge source repository, and the system creates an instance of table *Document* from the system's database to link statistic and semantic vectors to the knowledge source in the repository. This *Document* instance is flagged as not indexed, meaning that the source has yet to go through the indexation process. Although the Knowledge Extraction module should be automatic it is not in the SEKS scenario. RapidMiner is a desktop application so, to create statistic vectors, the system administrator has run the knowledge extraction process manually, directly on a computer. This manual process has to be done before the scheduled hour for the indexation process to begin; otherwise the system does not have the necessary statistic vectors for non-indexed knowledge sources. If everything runs normally, RapidMiner inserts statistic vectors of non-indexed files into table *StatisticWeight* on SEKS database. An example of all statistic weights for a test document is shown in Table 6.1.

Table 6.1: Example of a stored statistic vector

Keyword	Statistic weight (rounded values)
access	0.037

addit	0.007
advanc	0.311
agreement	0.550
applic	0.067
author	0.090
compli	0.114
concern	0.083
content	0.006
district	0.027
engin	0.011
ensur	0.067
feder	0.196
found	0.212
fund	0.376
govern	0.153
includ	0.004
inspect	0.150
local	0.166
make	0.040
manag	0.045
modul	0.144
offic	0.019
parti	0.114
project	0.011
provis	0.317
purpos	0.055
record	0.250
report	0.033
repres	0.094
request	0.083
requir	0.022
section	0.047
singl	0.116
state	0.150
summari	0.070

As one can easily see from Table 6.1, words are stemmed. This means that there is a certain error margin for each keyword in terms of its matching with ontology concepts. For instance, stem word *engin* may have different matches for keywords in ontology, such as *engineer* or *engine*, and both concepts are not related under the Building and Construction domain. On the other hand, stem words like *manag* do not present many related words out of the context of the referred domain, because *manager* and *management* are two applicable keywords in such domain. At the schedule hour, the system looks for all non-indexed files of that day, and starts the semantic indexation process. An example for the first iteration of semantic vectors created by the presented work, keyword-based semantic vectors, corresponding to the same test document of Table 6.1 is shown in Table 6.2. The keyword-based semantic vector creation process was tested on sixty five test knowledge sources.

Table 6.2: Example of a keyword-based semantic vector, matched: Matched ontology concepts and keywords, and respective weights

Concept	Keyword	Matched ontology keywords	Keyword-based semantic weight (rounded values)
Engineer	engin	engineer	0.009
Inspector	inspect	inspector, inspection	0.114
Management_Actor	manag	manager, management actor	0.028
Association	feder	federation	0.124
Trainer	manag	manager	0.028
Report	report	report	0.025
Territory	state	state	0.095
Request	request	request	0.063
Issue	compli	complication	0.087
Manufacturer	make	maker	0.025
Presence_Detection _And_Registration	record	recording	0.189
District	district	district	0.015
Foundation	found	foundation	0.134
Project	project	project	0.007
Consultant	author	authority	0.0572

Next, SEKS system performs taxonomy- and ontology-based semantic vector creation processes. The objectives are, on one side, to boost weights corresponding to concepts within semantic vectors that are taxonomically or ontologically related to each other, and, on the other side, add new concepts to these semantic vectors, according to their taxonomic or ontological

relations with concepts within such vectors. The assessment of these processes was made over six random knowledge sources present in the SEKS knowledge source repository. An example of these objectives for taxonomic relations of ontology concepts from one of these six test knowledge sources is shown in Table 6.3.

Table 6.3: Old keyword-based semantic weights and new taxonomy-based semantic weights

Concept	Keyword-based weight (rounded values)	Taxonomy-based weight (rounded values)
Engineer	0.009	0.004
Inspector	0.114	0.057
Management_Actor	0.028	0.014
Association	0.124	0.062
Trainer	0.028	0.014
Report	0.025	0.012
Territory	0.095	0.048
Request	0.063	0.032
Issue	0.087	0.043
Manufacturer	0.025	0.013
Presence_Detection_ And_Registration	0.189	0.095
District	0.015	0.008
Foundation	0.134	0.067
Project	0.007	0.004
Counsultant	0.057	0.029
Design_Actor	n.a.	0.271
Distributor	n.a.	0.105
Contractor	n.a.	0.063
Coordinator	n.a.	0.062

Concepts presented in bold represent new concepts added to the taxonomy-based semantic vector due to their taxonomic relation with one or more concepts that were already within the keyword-based semantic vector. In this case, the values corresponding to the homologous and non-homologous threshold are, respectively, 0.05 and 0.07. For more specificity in the addition and boost of weights, the value of such thresholds should be raised. It is obvious that "*Design_Actor*" gained more relevance than all the other concepts. This happens because the concept

"*Design_Actor*" has a strong taxonomic relation with (i.e. is taxonomically near to) several concepts:

- Homologous relation with "*Engineer*";
- Non-Homologous relation with "*Management_Actor*", "*Inspector*", "*Trainer*" (other actors);

The ontology-based semantic vector creation process assessment is similar to the one presented previously (Table 6.3). The assessment over these processes was only made over six knowledge sources due to performance issues in the test server machine (the processes of taxonomy- and ontology-based semantic vector creation for one knowledge source took almost one hour).

This leads to knowledge source search by semantic vector comparison. Keyword-based search interface and the provided autocomplete functionality are presented in Figure 6.2. Autocomplete is performed by loading all ontology keywords to the client-side application on its initialization. When users insert queries, they may use the autocomplete function or not. Then, the search process begins.

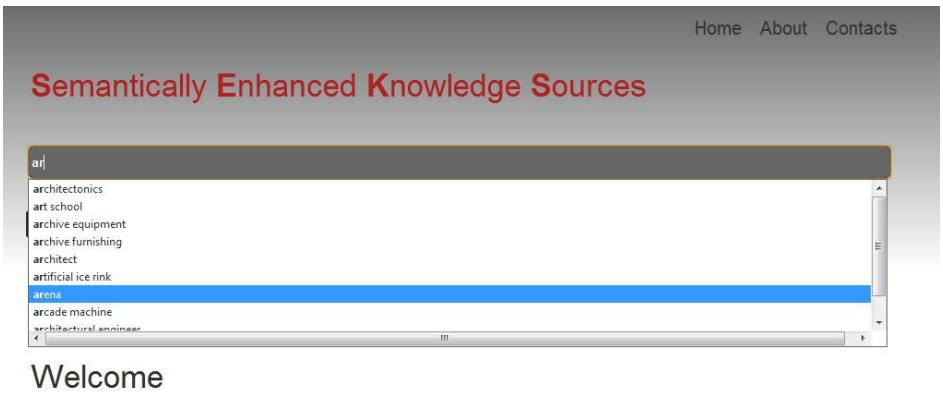

Figure 6.2: User Interface - Keyword-based search and autocomplete function

As previously referred, this work used sixty five knowledge sources associated to the Building and Construction domain as test subjects. Just as an example, a test query search for "*architect, door frame*" is inserted in the interface's keyword-based search field, meaning that the user is an architect and is interested in find knowledge sources about door frames, such as a catalog, for instance. As previously referred, statistic vectors of queries present the same weight for every keyword. In this case, keywords *architect* and *door frame* both have the same weight of 0.5. Keyword *architect* is matched with concept "*Architect*" and keyword *door frame* is matched with concept "*Door Component*" Hence, the semantic vector for this query will be the one of Table 6.4.

Table 6.4: Example of a query's semantic vector

Ontology concept	Weight
Architect	0.5
Door Component	0.5

The first results for the document test set is very satisfactory: The first search-resultant knowledge source presents a relevance of 52% to the query, out of a total of fifty six results. The semantic vectors for the first five results, in addition to other ontological concepts, have values for concepts *door component* and *architect*, and respective relevance to the query presented in Table 6.5.

Table 6.5: First five results for query "architect, door frame"

Document ID	Weight for concept "Architect" (rounded values)	Weight for concept "Door Component" (rounded values)	Relevance to query (%)
34	0.028	0.182	52
3	0.152	n.a.	48
56	0.002	0.122	40
42	0.022	0.101	39
33	0.043	0.045	32

It is easily comprehendible that *Door Component* concept has a big semantic weight in the first result, and the knowledge source also has semantic references to *Architect*. The style and format used to present the results were not highly sophisticated, since this work represents only a proof of concept. Hence, results are presented by showing only the relevance percentage for each knowledge source the database identifier of the knowledge source and the name and type of the knowledge source file.

As previously referred, there is another form of query search: ontology concept-based knowledge source search. This process is similar in all aspects to the other query search option presented above, but, in this case, the system does not need to perform keyword matching with ontology concepts. Concepts are chosen by clicking the checkbox next to the concept or dragging the concept to the search field below the visual tree, shown in Figure 6.3. The result ranking and presentation processes are the same for both query options.

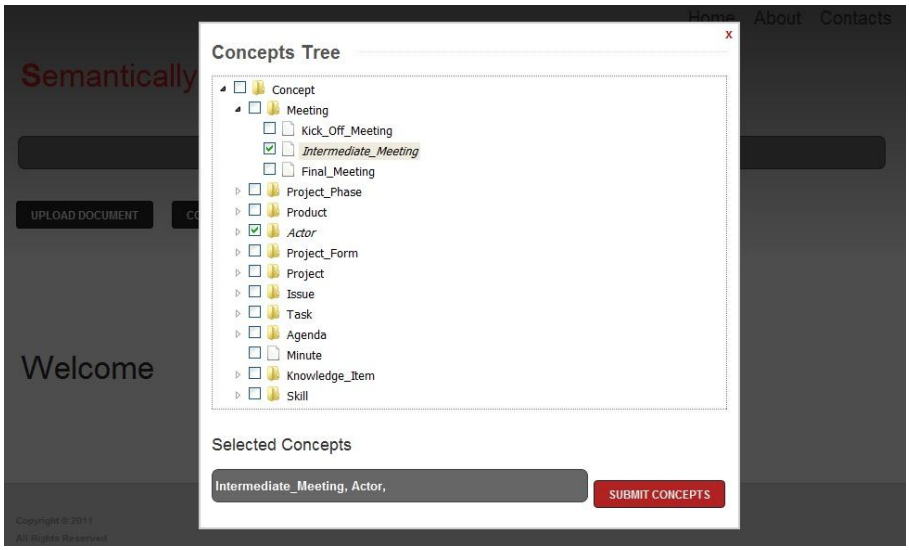

Figure 6.3: User Interface - Visual ontology tree and ontology concept-based search

7 CONCLUSIONS AND FUTURE WORK

This chapter presents an overall synthesis of the work developed here, regarding the objectives and research question enunciated in Chapter 1. Finally, research contributions and future works close the chapter.

7.1 Synthesis

This work is a contribution for the IR and KM areas, in an attempt to use Semantic Web technologies and ontologies to build semantic knowledge representation of knowledge sources. Such representations are epitomized as analytical vectors that comprise knowledge from a knowledge source, regarding the Building and Construction domain. Under this environment, and having as test stage collaborative engineering projects, users can subsequently search and discover knowledge sources that present some semantic relevance within their context of interest. Results presented on Chapter 6 show that the research question was properly approached and answered. Looking at specific goals for this work, also presented in Chapter 1, the following achievements were realized:

- *Extract underlying knowledge from knowledge sources' main contents and associated metadata, through the utilization data- and text-mining tools, in order to build a statistic vector of term occurrences and positioning in knowledge sources;*

The utilization of RapidMiner has proven to be a satisfactory choice in terms of knowledge extraction and data-mining techniques needed. Created statistic vectors fulfilled all of the system's requirements, even though the stem word issue has occurred. Possible future works on this module are introduced in Subchapter 7.3.

- *Given a set of keywords or terms that occur frequently in a knowledge source, or that have a position of relevance in the source (e.g.: term in the document's title), build keyword-based semantic vectors by matching these keywords with ontological concepts;*

SEKS system has its own Building and Construction domain-specific ontology, with ontological concepts and keywords associated to it. The mechanism for matching statistic vectors' keywords with ontological concepts' keywords has also been implemented.

- *Given a set of ontological concepts and respective statistical weights, build a keyword-based semantic vector based on such concepts' semantic relevance, not only within a particular knowledge source, but also their relevance across all knowledge sources within the knowledge source corpus;*

Mathematical functions to apply the TF*IDF algorithms, as presented by (Castells, et al., 2007), and vector normalization were developed in order to account for term relevance over the

entire knowledge source corpus' universe. Keyword-based semantic vector creation is based on such mathematical methods.

- *Given a keyword-based semantic vector, analyze the relevance of concepts present in such vector in terms of their taxonomic relationship, and build a semantic vector according to concepts related by a hierarchical relation (taxonomy-based semantic vector). This taxonomy-based vector takes into account the families, or bags, of terms. For instance, "*Design Actor*" and "*Architect*" may be seen as taxonomically related, being "*Architect*" a child concept of "*Design Actor*", much in the same way as concepts "*Dog*" and "*Animal*".*

Using the mathematical processes introduced by (Li, 2009) and (Nagarajan, et al., 2007), taxonomic relations are accounted for, and their semantic importance within the domain's context and occurrence throughout the knowledge sources' search space are considered when building taxonomy-based semantic vectors.

- *Given a keyword-based or taxonomy-based semantic vector, analyze the relevance of concepts present in such vector in terms of their ontological relations, and build an ontology-based semantic vector. This ontology-based vector is based on the underlying semantic relationships between ontology concepts. For instance, concepts "*Architect*" and "*Building*" may be considered linked by a semantic relation, such as "*designs*" or "*is designed by*".*

Again considering research made by (Nagarajan, et al., 2007), and using semantic relations contained in the domain-specific ontology and a TF*IDF family approximation algorithm, the underlying meaning given by such relations further enriches the semantic knowledge representation provided by keyword- and taxonomy-based semantic vectors, by analyzing the occurrence and relevance of such relations.

- *Compare semantic vectors using a similarity measure that allows analyzing whether or not those vectors are semantically similar or not. If two vectors are semantically similar, then the knowledge sources associated to them are also similar, in terms of their meaning and context.*

Finally, semantic vector comparison is provided through the application of Euclidian distance and sparse-matrix multiplication methods. The presented work even allows users to input search queries, attributing semantic vectors to them for knowledge source search and retrieval, using SEKS User Interface.

In conclusion, this work presents a solid contribution not only to the IR field of work, through the application of semantic mechanisms in the creation of representations for knowledge

sources, and to project teams that use some kind of collaborative environment, by helping teams to choose relevant knowledge from a panoply of knowledge sources and, ultimately, saving time to both teams and organizations, in terms of knowledge search and investigation. Furthermore, the presented work can be applied to any system that has knowledge source storage and retrieval capabilities, from search engines to personal desktop file management systems. On the other hand, the application of other research works is a proof of the efforts that are being made in the semantic knowledge indexation area of study. Hence, a progressive evolution of this and other works is needed in this research field, so that the use of Semantic Web mechanisms to manage, index and retrieve knowledge sources can be commercially implemented on a global scale.

7.2 Research Contribution

Besides highlighting and presenting a contribution for the topic of unstructured data over the Web, through the creation of semantic vectors that represent knowledge within knowledge sources, SEKS system presents a proof of demonstration for Semantic Web-enabled systems that approach the above topic through the KM and IR areas' perspective, providing both client-side application and Web Services funtions to be used by individuals and systems across other contexts and areas.

As previously referred in Chapter 1.3, this work was developed under the CoSpaces Integrated Project, and during such development the following papers were published or submitted:

- Lima, C., Figueiras, P., Costa, R. (2010). A Knowledge Engineering Approach Supporting Collaborative Working Environments Based on Semantic Services. KEOD 2010 - International Conference on Knowledge Engineering and Ontology Development. Valência, Spain (ISI Web of Science).

- Costa, R., Lima, C., Antunes, J., Figueiras, P., & Parada, V. (2010). Knowledge Management Capabilities Supporting Collaborative Working Environments in a Project Oriented Context.. ECIC 2010 - 2nd European Conference on Intellectual Capital, (pp. 1-9). ISCTE Lisbon University Institute, Lisbon, Portugal and Polytechnic Institute of Leiria, Portugal (ISI Web of Science).

- Figueiras, P., Costa, R., Paiva, L., Lima, C., Gonçalves, R. (2012) Information Retrieval in Collaborative Engineering Projects – A Vector-Space Model Approach. KEOD 2012 - International Conference on Knowledge Engineering and Ontology Development. Barcelona, Spain (ISI Web of Science).

- Costa, R., Figueiras, P., Luis, P., Jardim-Gonçalves, R. and Lima, C. (2012). Capturing Knowledge Representations Using Semantic Relationships - An Ontology-based approach.

SEMAPRO 2012 - International Conference on Advances in Semantic Processing. Barcelona, Spain (ISI Web of Science).

7.3 Future Works

The presented work has functionalities that could not be tested although they are implemented, namely statistic and semantic vector union. On the other hand, some glitches and limitations could be solved by the following necessary future works:

- The Knowledge Extraction Module should be built with a knowledge extraction API, like Apache Lucene (The Apache Software Foundation, 2011), for overriding the necessity of having a system's administrator responsible for the task of manually creating knowledge sources' statistic vectors, in this case using RapidMiner (Rapid-I GmBH, 2011). RapidMiner itself also has a knowledge extraction API.
- Furthermore, the Knowledge Extraction Module should be implemented in such manner as to create stem word disambiguation mechanisms in order to match statistic vectors' keywords with the exact equivalent term in the ontology. This can be made by programmatically developing filters with the desired specifications.
- Obviously, the system's ontology does not comprise all concepts and semantic relations regarding the Building and Construction domain. Such completeness of subjects should be integrated in an ontology, so that semantic vectors could truly represent knowledge within a knowledge source. Hence, research and development of complete specific-domain ontologies is also pointed as possible future work. Furthermore, ontological concepts and relations should be inserted and managed dynamically, through learning processes, in order to make possible for the ontology to learn, capture mew concepts and relations from the knowledge source corpus' universe and update relation importance between concepts, while new sources become available.
- When users browse to the SEKS User Interface and input a search query, their contexts should be considered. CoSKS contemplated this limitation by asking users to fill out HTML forms with information regarding their role in the ontology's domain (e.g. architect, labourer), and associated projects, meetings, tasks, issues, etc. This data would then be stored in the system's database and retrieved whenever users inputted queries, enriching such queries with the ontology concepts that matched the user's data. This offered the possibility for an architect to write a query string without mentioning that he or she was actually an architect. The system simply perceived that implication according to stored data about the user and his or her context. Databases and mechanisms for this functionality were already implemented for CoSKS, but were not used in this work due to shortage of time.

- Some research should be done over the subject of taxonomy- and ontology-based semantic vector creation for queries. Particularly, if the context of the user is being accounted for, it is possible that the analysis of taxonomic and ontological relations across queries' keywords could prove to be a good contribution for the semantic indexation of queries themselves, and ultimately, for the relevance of retrieved results.
- Finally, although the presented work was developed under the collaborative engineering projects' scope, it has already some features towards flexibility, configurability and usability in other contexts. Even so a more extensive work is needed to generalize this work across other areas, such as search engines or desktop search software.

BIBLIOGRAPHY

Stanford Center for Biomedical Informatics Research, 2011. *Stanford's Protégé Home Page.* [Online]
Available at: http://protege.stanford.edu/

Alavi, M. & Leidner, D. E., 1999. Knowledge Management Systems: Issues, Challenges and Benefits. *Communications of the Association for Information Systems,* 1(7), pp. 2-37.

Antoniou, G. & Harmelen, F. v., 2004. *A Semantic Web Primer.* Boston: Massachussets Institute of Technology.

Antunes, J., 2010. *Design and implementation of an autonomous, proactive, and reactive software infrastructure to help improving the management level of projects.* Monte da Caparica: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa.

Baker, C. J. O. & Cheung, K.-H., 2007. *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences.* New York: Springer Science+Business Media, LLC.

Berners-Lee, T., Hendler, J. & Lassila, O., 2001. The Semantic Web. *Scientific American,* pp. 34-43.

Bikakis, N. et al., 2012. The XML and Semantic Web Worlds: Technologies, Interoperability and Integration. A survey of the State of the Art. In: *Semantic Hyper/Multi-media Adaptation: Schemes and Applications.* s.l.:Springer.

Bizer, C., Heath, T. & Berners-Lee, T., 2009. Linked Data: The Story So Far. *International Journal on Semantic Web and Information Systems,* 5(3), pp. 1-22.

Breitman, K. K., Casanova, M. A. & Truszkowski, W., 2007. *Semantic Web: Concepts, Technologies and Applications.* London: Springer-Verlag London Limited for NASA.

BuildingSmart, 2011. *IFD Library for BuildingSmart.* [Online]
Available at: http://www.ifd-library.org/index.php?title=Home_Page
[Accessed 30 October 2011].

Camarinha-Matos, L. M. & Afsarmanesh, H., 2006. *COLLABORATIVE NETWORKS - Value creation in a knowledge society.* Shanghai, Proceedings of PROLAMAT'06.

Castells, P., Fernández, M. & Vallet, D., 2007. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. *IEEE Transactions on Knowledge and Data Engineering,* February, 19(2), pp. 261-272.

CoSpaces Project Consortium, 2010. *CoSpaces: Innovative Collaborative Work Environments for Design and Engineering.* [Online]
Available at: http://www.cospaces.org/
[Accessed 2010].

Costa, R. et al., 2010. *Knowledge Management Capabilities Supporting Collaborative Working Environments in a Project Oriented Context.* s.l., Proceedings of the 2nd European Conference on Intellectual Capital.

Dalkir, K., 2005. *Knowledge Management in Theory and Practice.* 2nd ed. Burlington, MA: Elsevier Inc..

Deza, M. M. & Deza, E., 2009. *Encyclopedia of Distances.* Heidelberg: Springer-Verlag Berlin Heidelberg.

Dubin, D., 2004. The Most Infuential Paper Gerard Salton Never Wrote. *Library Trends,* 52(4), pp. 748-764.

Eclipse Foundation, 2004. *Eclipse - The Eclipse Foundation open source community web site.* [Online]
Available at: http://www.eclipse.org/
[Accessed 2008].

Goble, C. et al., 2001. *Conceptual Open Hypermedia = The Semantic Web?.* Hongkong. China, CEUR Workshop proceedings.

Google Inc., 2012. *Google Search Home Page.* [Online]
Available at: http://www.google.com

Harris, Z., 1954. Distributional Structure. *Word,* 10(23), pp. 146-162.

Ichijo, K. & Nonaka, I., 2007. *Knowledge Management and Creation: New Challenges for Managers.* Oxford: Oxford University Press.

Idehen, K., 2010. *Creating, Deploying and Exploiting Linked Data.* [Online]
Available at:
http://virtuoso.openlinksw.com/presentations/Creating_Deploying_Exploiting_Linked_Data2/Creating_Deploying_Exploiting_Linked_Data2_TimBL_v3.html#(1)
[Accessed 2012].

Java.net, 2008. *JAX-WS Reference Implementation (IR) Project.* [Online]
Available at: http://jax-ws.java.net/
[Accessed 2011].

Jones, K. S., 2004. A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation,* 60(5), pp. 11-21.

Kashyap, V., Bussler, C. & Moran, M., 2008. *The Semantic Web: Semantics for Data and Services on the Web.* 1st ed. Berlin: Springer-Verlag Berlin Heidelberg.

Kemafor, A., Sheth, A. P. & Maduko, A., 2005. *SemRank: Ranking Complex Relati onship Search Results on the Semantic Web.* Chiba, Japan, Proceedings for the 14th International World Wide Web Conference, pp. 117-127.

Kern, E.-M. & Kersten, W., 2007. Framework for internet-supported inter-organisational product development collaboration. *Journal of Enterprise Information Management,* 20(5), pp. 562-577.

Kroeger, P. R., 2005. *Analyzing Grammar: An Introduction.* 1 ed. New York: Cambridge University Press.

Larousse, 1994. *Nouveau Larousse Encyclopédique.* 3905 ed. Paris: Larousse.

Liferay Inc., 2011. *Liferay Content Management Features.* [Online]
Available at: http://www.liferay.com/products/liferay-portal/features/cms
[Accessed 2012].

Lima, C., Figueiras, P. & Costa, R., 2010. *A Knowledge Engineering Approach Supporting Collaborative Working Environments Based on Semantic Services.* Valência, Spain, s.n.

Linden, R. M., 2002. *Working Across Boundaries: Making Collaboration Work in Government and Nonprofit Organizations.* 1st ed. San Francisco(CA): John Whiley & Sons, Inc..

Lin, D. & Pantel, P., 2001. *DIRT: Discovery of Inference Rules from Text.* San Francisco, CA, s.n., pp. 323-328.

Li, S., 2009. A Semantic Vector Retrieval Model for Desktop Documents. *Journal of Software Engineering & Applications,* Issue 2, pp. 55-59.

Manning, C. D., Raghavan, P. & Schütze, H., 2008. *Introduction to Information Retrieval.* Cambridge, UK: Cambridge University Press.

Mathew, G. E., 2002. *The State-of-the-Art Technologies and Practices in Enterprise Business Collaboration,* s.l.: Infosys Technologies Limited.

Nagarajan, M. et al., 2007. Altering Document Term Vectors for Classification - Ontologies as Expectations of Co-occurrence. *ReCALL,* p. 1225.

Nonaka, I., 1994. A Dynamic Theory of Organizational Knowledge Creation. *Organization Science,* 5(1), pp. 14-27.

Nonaka, I., Reinmoller, P. & Toyama, R., 2001. Integrated information technology systems for knowledge creation. In: *Handbook of Organisational Learning and Knowledge Management.* Padstow: Oxford University Press , pp. 827-848.

Nonaka, I. & Takeuchi, H., 1995. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* 1st ed. New York: Oxford University Press.

OCCS Development Committee Secretariat, 2011. *OmniClass - A Strategy for Classifying the Built Environment.* [Online]
Available at: http://www.omniclass.org/
[Accessed 27 October 2011].

Oracle Corporation, 2010. *Java.* [Online]
Available at: http://www.java.com/
[Accessed 2011].

Oracle Corporation, 2011. [Online]
Available at: www.mysql.com

Oracle Corporation, 2012. *Java Servlet Technology.* [Online]
Available at: http://www.oracle.com/technetwork/java/index-jsp-135475.html
[Accessed 2012].

Rajman, M. & Besançon, R., 1998. *Text Mining - Knowledge extraction from unstructured textual data.* s.l., Proceedings of the 6th Conference of International Federation of Classification Societies (IFCS-98).

Rapid-I GmBH, 2011. *RapidMiner.* [Online]
Available at: http://rapid-i.com/content/view/181/190/
[Accessed 2011].

Rezende, J. & Souza, J., 2007. *Using Knowledge Management Techniques to Prove the Learning Process through the Exchange of Knowledge Chains,* s.l.: IEEE.

Robertson, S., 2004. Understanding Inverse Document Frequency: On theoretical arguments for IDF. *Journal of Documentation,* 60(5), pp. 503-520.

Sahlgren, M., 2008. The Distributional Hypothesis. 20(1), p. From context to meaning: Distributional models of the lexicon in linguistics and cognitive science (Special issue of the Italian Journal of Linguistics).

Salton, G., Wong, A. & Yang, C. S., 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM,* 18(11), pp. 613-620.

semanticweb.org, 2011. *semanticweb.org Main Page.* [Online]
Available at: http://semanticweb.org/wiki/Main_Page

The Apache Software Foundation, 1999-2011. [Online]
Available at: http://tomcat.apache.org/

The Apache Software Foundation, 2011. *Apache Jena - A Semantic Web Framework for Java.* [Online]
Available at: http://incubator.apache.org/jena/
[Accessed 2011].

The Apache Software Foundation, 2011. *Apache Lucene - Overview.* [Online]
Available at: http://lucene.apache.org/java/docs/index.html

The jQuery Project, 2010. *jQuery Home Page.* [Online]
Available at: http://jquery.com/
[Accessed 2011].

Turney, P. D., 2008. The Latent Relation Mapping Engine: Algorithm and Experiments. *Journal of Artificial Intelligence Research,* Volume 33, pp. 615-655.

Turney, P. & Pantel, P., 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence,* pp. 141-188.

Visual Paradigm International, 1999. *Visual Paradigm for UML.* [Online]
Available at: http://www.visual-paradigm.com/product/vpuml/
[Accessed 2011].

von Krogh, G., 1998. Care in Knowledge Creation. *California Management Review,* 40(3), pp. 133-153.

Widdows, D. & Ferraro, K., 2008. *Semantic Vectors: A Scalable Open Source Package and Online Technology Management Application.* Marrakech, European Language Resources Association (ELRA).

Wiig, K., 1993. *Knowledge Management Foundations: Thinking about Thinking – How Organizations Create, Represent and Use Knowledge.* 5th ed. Arlington(TX): Schema Press.

World Wide Web Consortium (W3C), 2001. *W3C Web Ontology Working Group Home Page.* [Online]
Available at: http://www.w3.org/2001/sw/WebOnt/

World Wide Web Consortium (W3C), 2004. *OWL Web Ontology Language Overview.* [Online]
Available at: http://www.w3.org/TR/owl-features/
[Accessed 28 October 2011].

World Wide Web Consortium (W3C), 2008. *SPARQL Query Language for RDF.* [Online]
Available at: http://www.w3.org/TR/rdf-sparql-query/
[Accessed 2009].

World Wide Web Consortium (W3C), 2011. *SPARQL-DL query language for OWL-DL.* [Online]
Available at: http://www.w3.org/2001/sw/wiki/SPARQL-DL
[Accessed 2012].

World Wide Web Consortium (W3C), 2011. *W3C Semantic Web Activity.* [Online]
Available at: http://www.w3.org/2001/sw/

Yahoo! Inc., 2012. *Yahoo! Search Home Page.* [Online]
Available at: http://www.yahoo.com/

Yu, L., 2011. *A Developer's Guide to the Semantic Web.* Berlin: Springer-Verlag Berlin Heidelberg.

Zhang, J., 2010. *A SOCIAL SEMANTIC WEB SYSTEM FOR COORDINATING COMMUNICATION IN THE ARCHITECTURE, ENGINEERING & CONSTRUCTION INDUSTRY.* Toronto: Univeristy of Toronto.

APPENDIX A

UML Sequence Diagrams

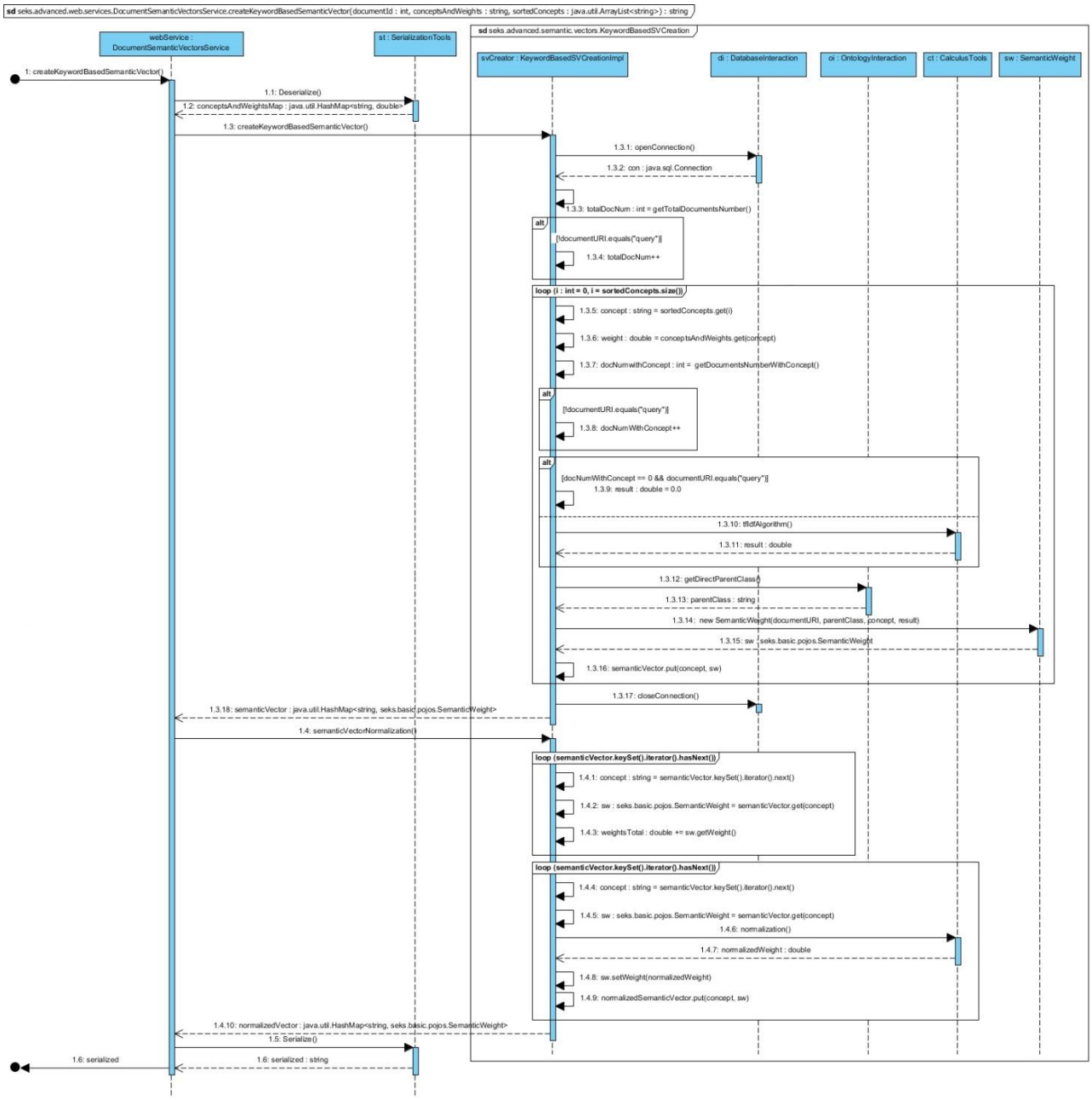

Figure A.1: USD - Create a document's keyword-based semantic vector

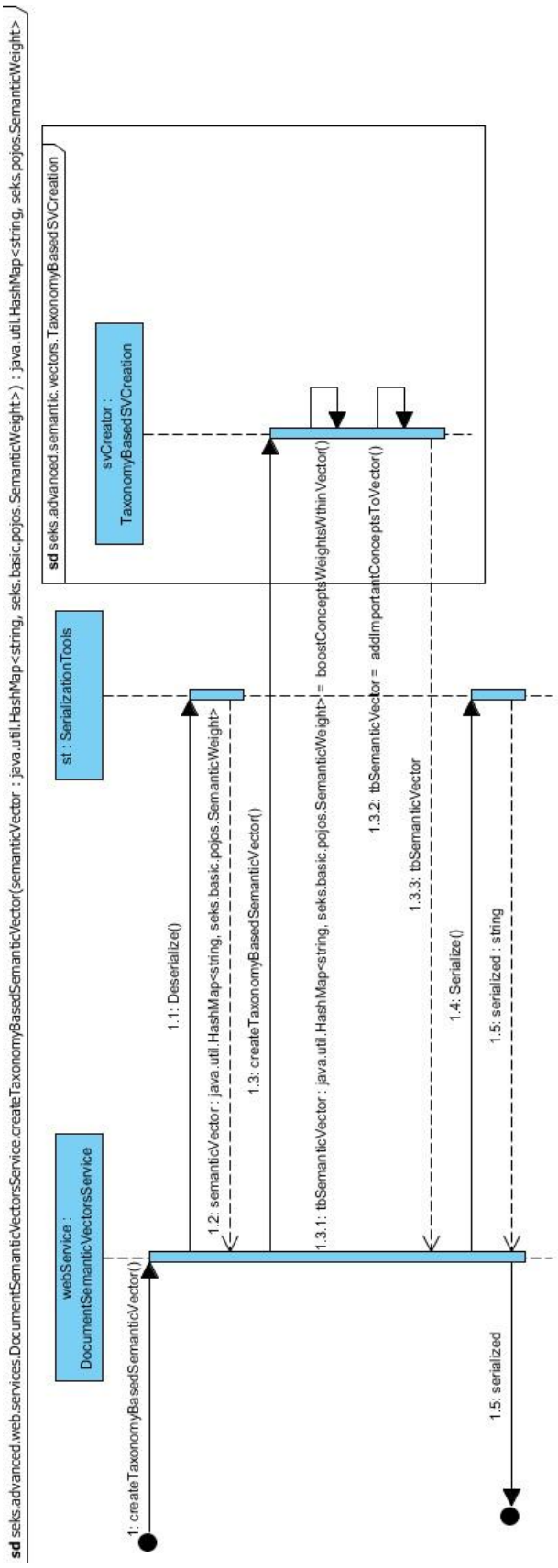

Figure A.2: USD - Create a document's taxonomy-based semantic vector

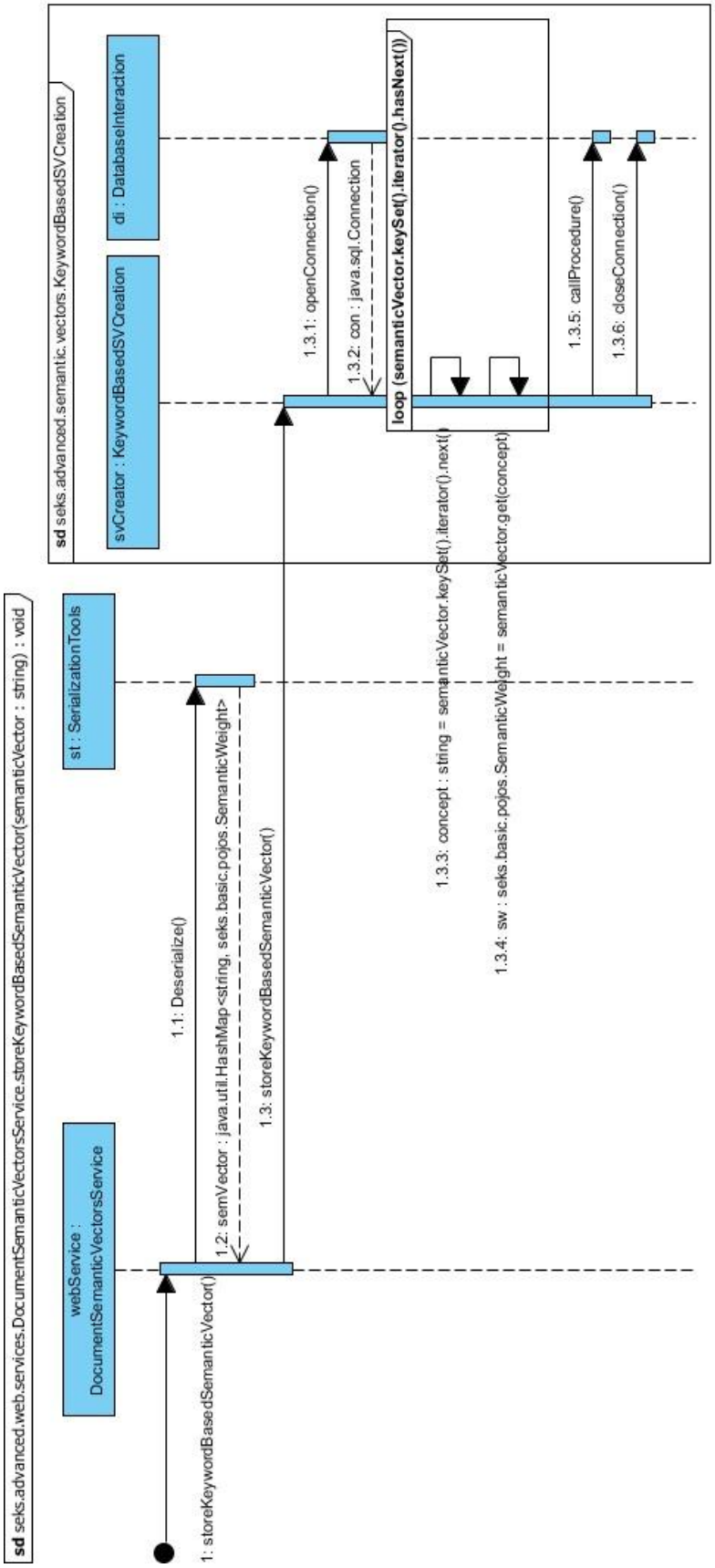

Figure A.3: USD - Store semantic vector in the system's database

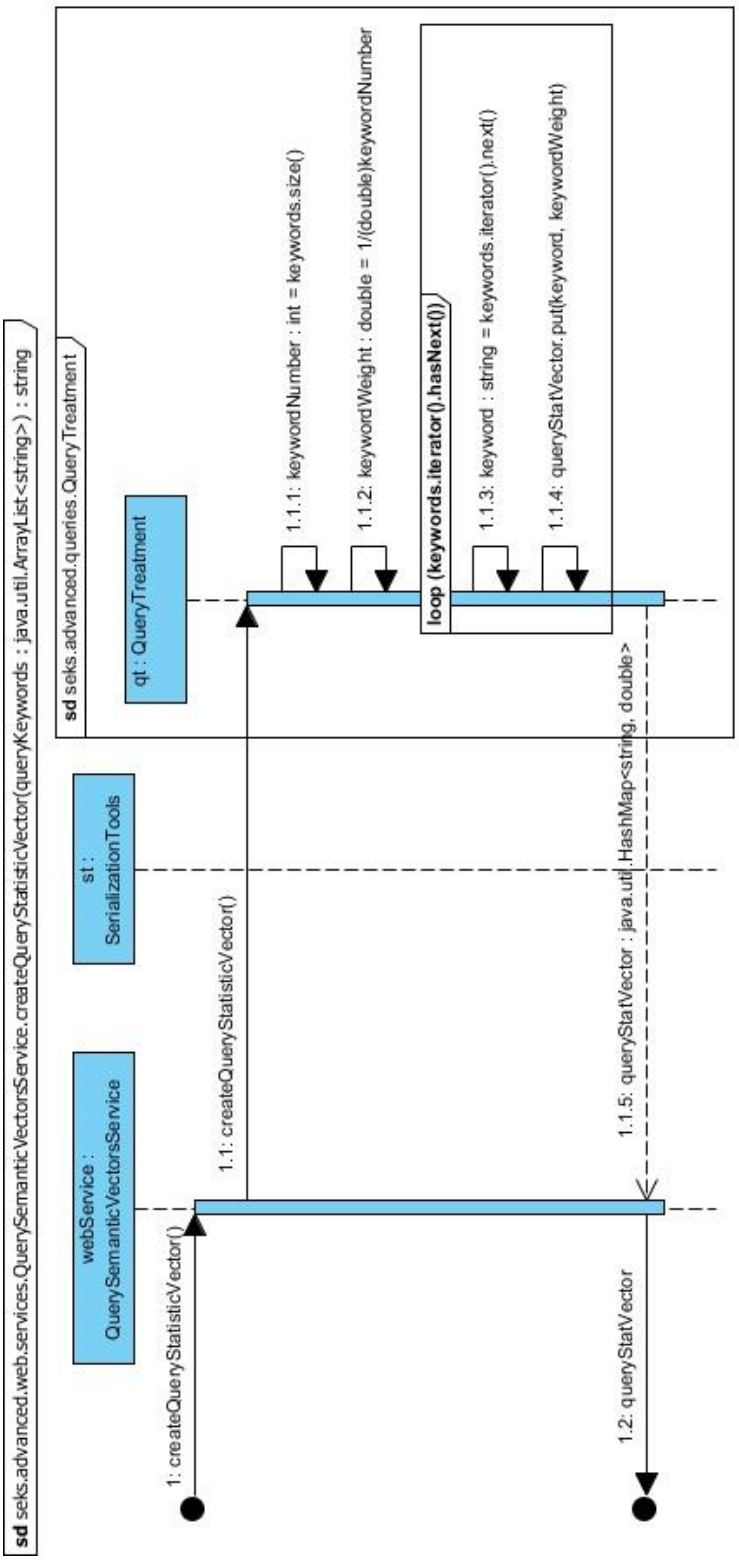

Figure A.4: USD - Create a query's statistic vector

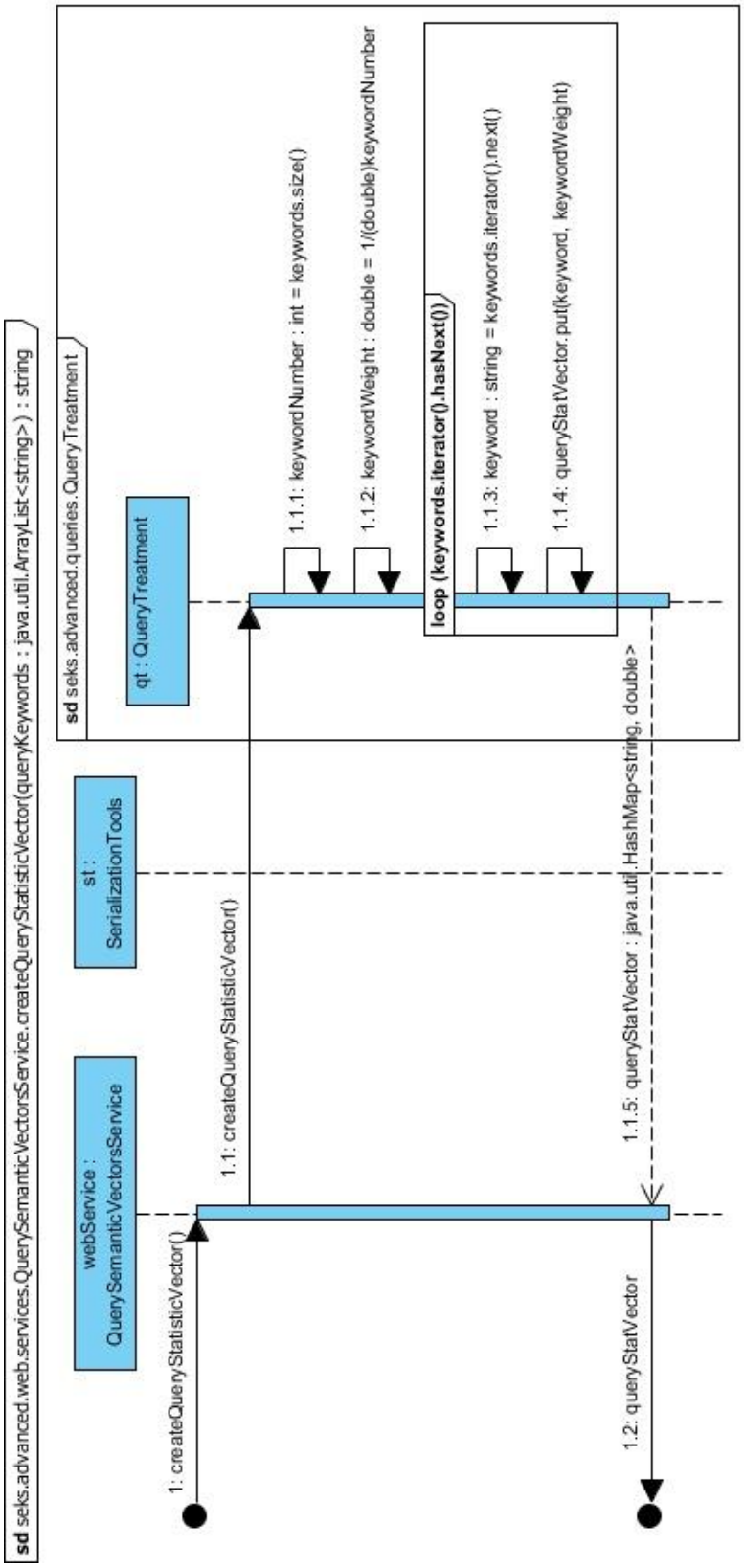

Figure A.5: USD - Create a query's semantic vector

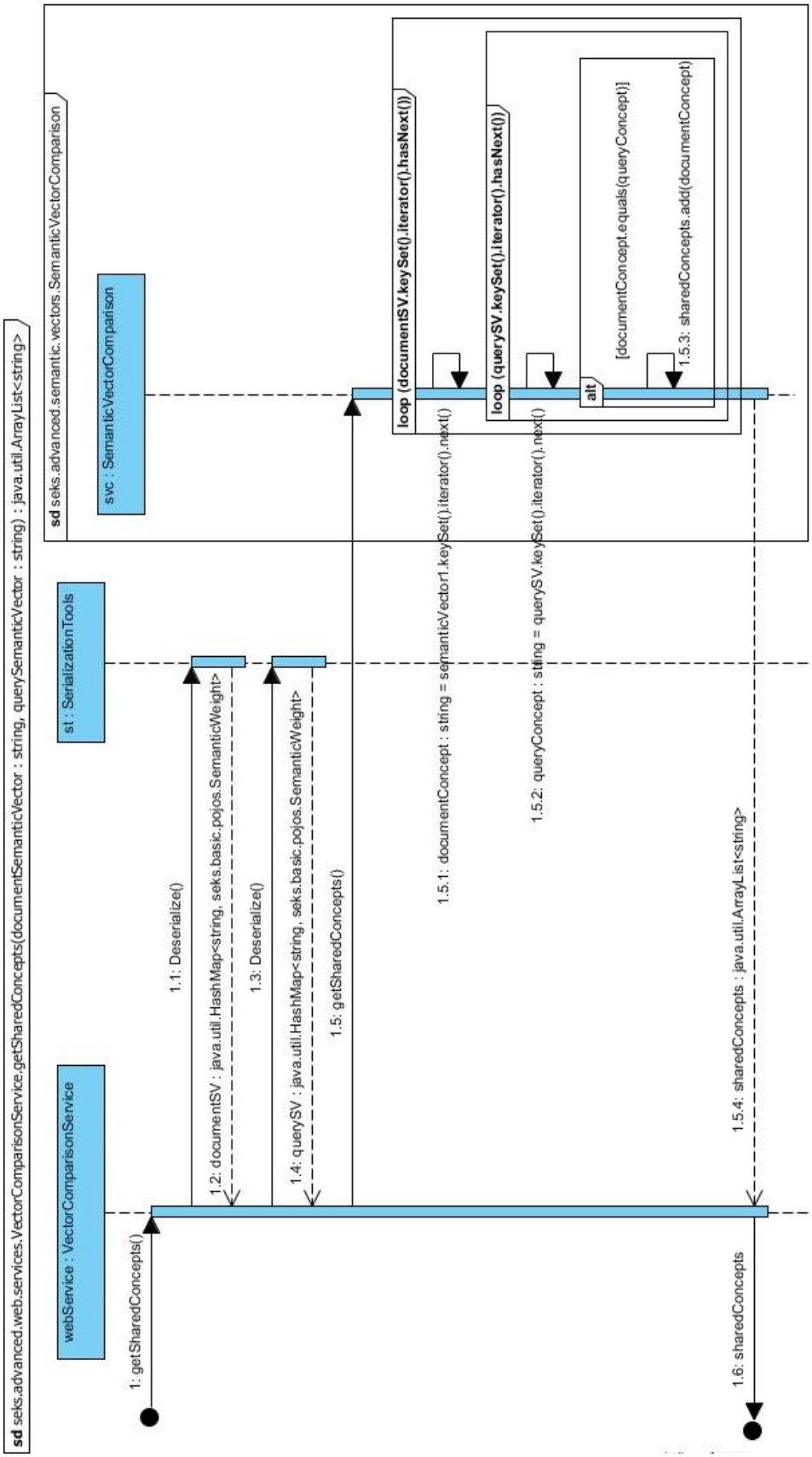

Figure A.6: USD - Get shared concepts between two semantic vectors

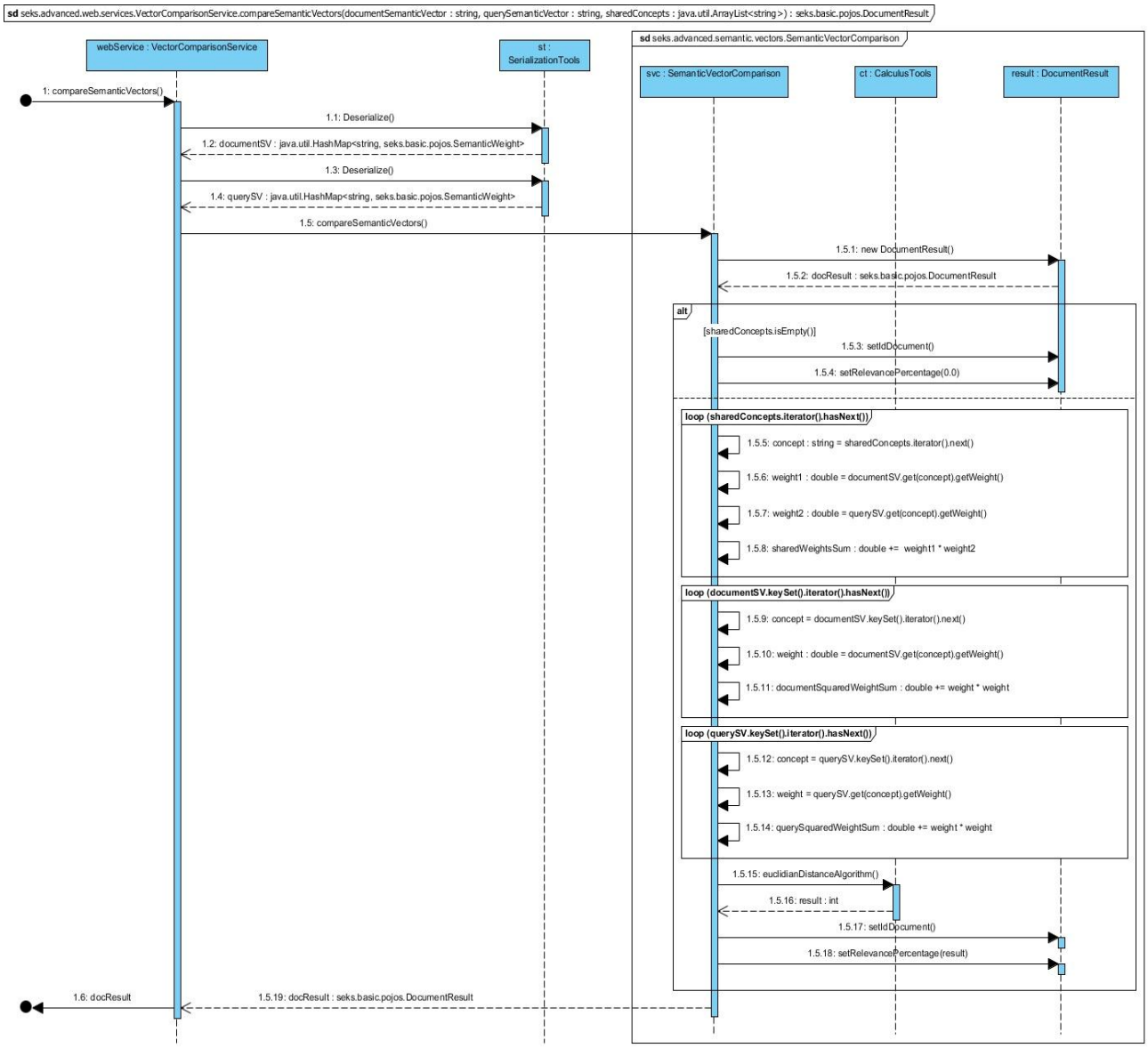

Figure A.7: USD - Semantic vector comparison

APPENDIX B

Table B.1: Adopted tools and technologies

Tool	Description	Why was it used?
Apache Tomcat 7	It is an open source servlet container that provides server-side capabilities for Java Web Applications. Apache Tomcat implements the Java Servlet specifications and provides a "pure Java" HTTP web server environment for Java code to run (The Apache Software Foundation, 1999-2011).	Implements Java Servlet 3.0 specifications. Reduces garbage collection, improves performance and scalability. Native Windows and Unix wrappers for platform integration and faster Java Servlet Http request and response parsing (The Apache Software Foundation, 1999-2011).
Eclipse IDE 3.7 (Indigo)	Eclipse is an open source community, whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle. Eclipse is much more than a Java IDE (Eclipse Foundation, 2004).	It is a full-featured Integrated Development Environment (IDE) that has all the needed compliances with other used technologies.
Java	Java is a programming language and computing platform first released in 1995. It is the underlying technology that powers state-of-the-art programs including utilities, games and business applications (Oracle Corporation, 2010).	The Java language provides an efficient platform to integrate several tools and technologies in a single system, using its frameworks. Other reason is the fact all Java-related tools and frameworks are open source.
Java Servlets 3.0	Java Servlet technology	Since the language of choice

	provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems (Oracle Corporation, 2012).	was Java, the server-side communication tool had to be Java Servlets. Servlets are easy to implement, as they only need three types of files: Client-side file (HTML), server-side Java class (the servlet) and a configuration file (web.xml) where the URL for each servlet is mapped to the servlet's Java class.
JAX-WS RI	It is a Web Services framework that provides tools and infrastructure to develop Web Services solutions for the end users and middleware developers (Java.net, 2008).	This Web Services framework is fully based on Java, it is compliant with the Apache Tomcat 7 server and it is very easy to integrate.
Jena Semantic Framework	Jena is a Java framework for building Semantic Web applications. Jena provides a collection of tools and Java libraries to help you to develop semantic web and linked-data apps, tools and servers (The Apache Software Foundation, 2011).	Jena framework is fully based on Java and includes an ontology API for handling OWL and RDFS ontologies. It also has a rule-based inference engine for reasoning with RDF and OWL data sources.
jQuery JavaScript Library	jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and AJAX interactions for rapid web development. jQuery is designed to change the way that you write JavaScript (The	This library was used mainly for its capabilities in terms of Asynchronous JavaScript and XML Remote Procedure Calls (AJAX-RPC), which allow client-side interfaces to send requests and receive responses from servers without having to refresh,

	jQuery Project, 2010).	giving interfaces a more dynamic look.
MySQL	It is a relational database management system (RDBMS) that runs as a server, providing multi-user access to a number of databases (Oracle Corporation, 2011).	It is an open source database application that offers fast performance, high reliability, ease to use and costs saving (Oracle Corporation, 2011).
MySQL Workbench	MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, and much more (Oracle Corporation, 2011).	This tool helps designing and developing databases from scratch, with ERD designer tools, query browser database development IDE, etc.
OWL-DL	OWL is a *Web* Ontology Language (World Wide Web Consortium (W3C), 2004). It is described in Chapter 3.	The reasons for using this language were already explained in Chapter 3.
Protégé Ontology Editor	Protégé is a free, open source ontology editor and knowledge base framework (Stanford Center for Biomedical Informatics Research, 2011).	Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development (Stanford Center for Biomedical Informatics Research, 2011).
Visual Paradigm for UML	It is a UML CASE Tool supporting UML 2.1 and the Business Process Modeling	Supports all UML diagrams used to model and design the presented work: Use Case

	Notation (BPMN). It provides business process modeling, an object-relational mapping generator for Java, .NET and PHP (Visual Paradigm International, 1999).	Diagrams, Class Diagrams and Sequence Diagrams. Furthermore, it provides an easy-to-use user interface to design such diagrams.