Realizing the Corporate Semantic Web: Concept Paper

Technical Report TR-B-09-05

Adrian Paschke, Gökhan Coskun, Marko Harasic, Ralf Heese, Markus Luczak-Rösch, Radoslaw Oldakowski, Ralph Schäfermeier and Olga Streibel

Freie Universität Berlin
Department of Mathematics and Computer Science
Corporate Semantic Web

18 May 2009

STI · BERLIN

Freie Universität Berlin

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Realizing the Corporate Semantic Web:

Concept Paper

Adrian Paschke Gökhan Coskun Ralf Heese
Markus Luczak-Rösch Radoslaw Oldakowski
Ralph Schäfermeier
Olga Streibel

Freie Universität Berlin
Department of Mathematics and Computer Science
Corporate Semantic Web
Königin-Luise-Str. 24-26
14195 Berlin, Germany

paschke,coskun,heese,luczak,oldakowski,schaef,streibel@inf.fu-berlin.de

18 May 2009

Abstract

In this concept paper, we outline our working plan for the next phase of the Corporate Semantic Web project. The plan covers the period from March 2009 to March 2010.

Corporate ontology engineering will improve the facilitation of agile ontology engineering to lessen the costs of ontology development and, especially, maintenance. Corporate semantic collaboration focuses the human-centered aspects of knowledge management in corporate contexts. Corporate semantic search is settled on the highest application level of the three research areas and at that point it is a representative for applications working on and with the appropriately represented and delivered background knowledge.

Each of these pillars will yield innovative methods and tools during the project runtime until 2013.

We propose a concept draft and a working plan covering the next twelve months for an integrative architecture of a Corporate Semantic Web provided by these three core pillars.

Contents

Chapter 1

Introduction

The transition from manufacturing to information economies and the progressive globalization of markets pose new challenges to enterprises. The amount of information that companies have to produce, acquire, maintain, propagate, and use has increased dramatically. While technologies and tools that help collaborating and structuring content, such as tagging, wikis, blogs, and collaboration platforms are in place, companies seek more capable approaches for gaining, managing, and utilizing knowledge required for their business processes. The amount, the heterogeneity, and the multimodality of data remain problematic aspects in the integration of data sources, presentation of information, and the extraction of knowledge.

In this regard, the Semantic Web offers promising solutions but also poses new challenges. The principal aspect of the Semantic Web is a shift away from the focus on data and documents towards their actual informational content and a machine-readable representation of it by the means of ontologies. Being a formal specification of a conceptualization an ontology allows inference engines to derive new information that is implicitly contained in it. As a result of the need for better knowledge management in corporate settings Corporate Semantic Web (CSW) aims at establishing semantic technologies in enterprises. Focusing on the controlled environment in contrary to the global Semantic Web it avoids facing unresolved problems like scalability, broader adoption of commonly shared ontologies, and trust issues. Therefore the potential in many industrial application scenarios and the short-term practicability in closed enterprise settings merits continued work, although semantic technologies are relatively young and gaps in standards and implementations exist.

After the Corporate Semantic Web project took up its work in February 2008, we introduced our initial vision of a Corporate Semantic Web as the next step in the broad field of Semantic Web research. Starting from interviews with regional industrial partners, we were able to develop a number of real world application scenarios and to identify requirements of the corporate environment and gaps between current approaches to tackle current problems facing ontology engineering, semantic collaboration, and semantic search.

In the second phase of the project runtime, we were able to enforce our outreach to and scientific cooperation with Berlin and Brandenburg based enterprises with the aim to establish a knowledge transfer channel between scientific institutions and enterprises in the local area. The results from the applicability

studies and requirement analysis from the first project phase yielded a sound and robust architecture for the upcoming Corporate Semantic Web.

This report introduces our Corporate Semantic Web architecture and outlines our working plan for the next project phase, covering the period from March 2009 to March 2010.

Our architecture integrates the three pillars that we consider the building blocks of a Corporate Semantic Web: corporate ontology engineering, corporate semantic collaboration, and corporate semantic search.

1.1 Corporate Ontology Engineering

In highly dynamic markets, enterprises depend on rapid integration of newly-acquired knowledge into their business processes. While corporate knowledge multiplies and evolves, companies seek knowledge management tools supporting this process in an efficient and cost-effective way.

Corporate ontology engineering tackles this problem from two perspectives:

Ontology modularization and integration focuses on the optimization of discovery, acquirement and use of corporate knowledge. In chapter 2.3, we propose an ontology discovery and integration component that discovers ontologies based on requirements which can be formulated by the aid of a requirements editor. Further, we propose a methodology for efficient modularization and retrieval of ontology parts while omitting unneeded parts.

Ontology versioning focuses the evolutionary aspect of corporate knowledge. While tools like "semantic diff building" and versioning on the semantic level are in place, none of these are currently able to provide commit and rollback actions on the concept level. In chapter 2.4, we propose an architecture and a technical concept for SVoNt, a version control system for OWL ontologies, built upon the well-known Subversion (SVN) system.

1.2 Corporate Semantic Collaboration

While knowledge workers are the principal actors in the process of knowledge evolution in enterprises, current tools mainly provide support for direct ontology editing and thereby require ontology engineers. This complicates the process of ontology evolution can become cost intensive. It would be desirable if knowledge workers could modify ontologies without having to be adept in their underlying formalisms.

In chapter 3.1, we propose a design for a web-based light-weight ontology editor that enables domain experts without knowledge about ontologies to contribute to the corporate knowledge corpus.

1.3 Corporate Semantic Search

Corporate semantic search investigates information discovery in both semantic as well as non-semantic data utilizing innovative semantic search techniques to facilitate deep analysis of available information by analyzing complex relationships in non-semantic data as well as providing users with personalized access to corporate information. The semantic web strongly depends on ontologies as

means for the formal representation of knowledge, whereas most documents lack formal semantic annotation. As a solution for searching in non-semantic data we introduce a trend analyzing tool that combines statistical methods with human-driven tagging approaches for recognizing semantics in and deriving trends from unstructured information sources (see chapter 4.1).

Another important aspect in the domain of search is personalization. Companies with large products scope and high customer diversity seek solutions for personalized search and recommender systems. In order to guarantee scalability and integration, we find a formalization of user profiles and application domains indispensable. In chapter 4.2 we propose a flexible framework for matching formalized personal profiles and corporate resources.

In the following report, we will cover the above approaches in detail. We will depict our technical concepts for tackling the above mentioned problems and show how they integrate into our Corporate Semantic Web architecture.

Chapter 2

Corporate Ontology Engineering

2.1 Introduction

In this section we describe the conceptual design for the work packages "Ontology Modularization and Integration" and "Ontology Versioning". Figure 3.1 highlights the components of the Corporate Semantic Web architecture that are involved in realizing both work packages.

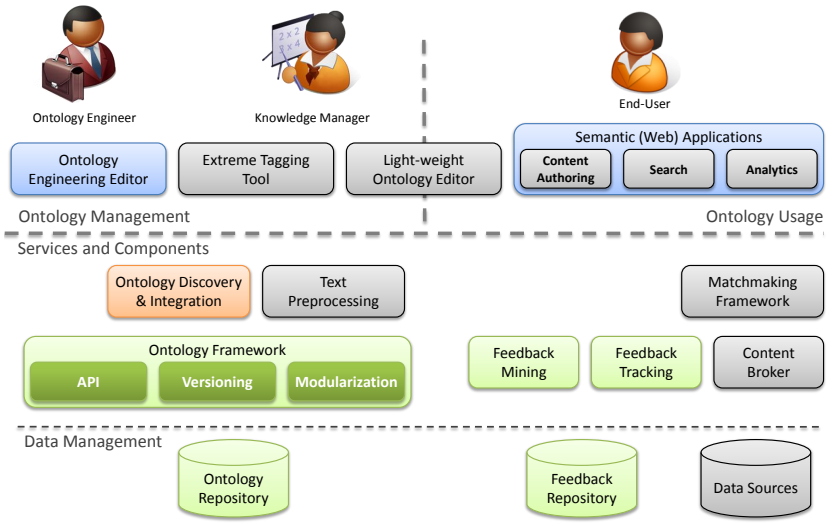

Figure 2.1: Overview of the architecture with focus on ontology engineering

2.1.1 Fundamentals

In modern economy the success of companies strongly depends on their ability to rapidly adopt new achievements into their business processes. The quick creation of economic value from knowledge is essential. But, classic enterprise applications and software systems are based upon static information systems and database solutions. They are not able to reflect dynamicity of knowledge and cannot enable the agile adoption of new achievements into the business processes. They are mainly developed for very special purposes aiming at meeting some functionalities and requirements of the company.

Knowledge-based applications are built upon knowledge bases (KB), which allow to manage declarative knowledge, that is to create, modify, and delete knowledge stored in the KB. Such applications are able to take new knowledge immediately into consideration and are therefore a very promising approach to displace classic data based and application dependent software systems. Being able to handle dynamicity and power of knowledge they are the key for flexible business processes.

In the Semantic Web ontologies are an important means for representing knowledge. They provide a shared understanding of a domain of interest. In a corporate environment, ontologies describe terms and their interrelation relevant to business context. However, there are the following four gaps between industrial needs and currently reached status of ontology engineering research as we identified in our previous work [Report 1]: (G1) the academic orientation gap, (G2) the application maturity gap, (G3) the process gap, and (G4) the cost-benefit-estimation gap. We stated that there is a need for a new methodology to create ontologies for corporate settings, which we call corporate ontology engineering (COE). Such a methodology has to close the aforementioned gaps by keeping the corporate setting in view. That is, it has to meet the following requirements

- The influence of the ontology development and maintenance process on the work flow of domain experts have to be minimized to avoid negative influence on their productivity. (R1)
- The already existing and running system must not be disturbed. (R2)
- The ontology has to evolve with the progress of the company. (R3)
- The need for ontology engineers have to be minimized to reduce costs. (R4)

As a solution approach we proposed the early stage of the Corporate Ontology Lifecycle Methodology (COLM). Being focused on corporate settings it inherently closes gap G1 and will prepare the ground for new knowledge based applications (G2). It reflects the agility of knowledge engineering (R3) processes and brings application dependency through the concrete definition of the application environment and the business needs (G3).

COLM consists of two different cycles, namely the usage cycle and the engineering cycle, and 7 phases. By splitting the development process into the mentioned parts it allows an intuitive understanding of raising costs and provides a cost-sensitive evolution of knowledge in form of ontologies. Enabling cost-benefit-estimation it closes the gap G4 and fulfills the last requirement listed above (R4).

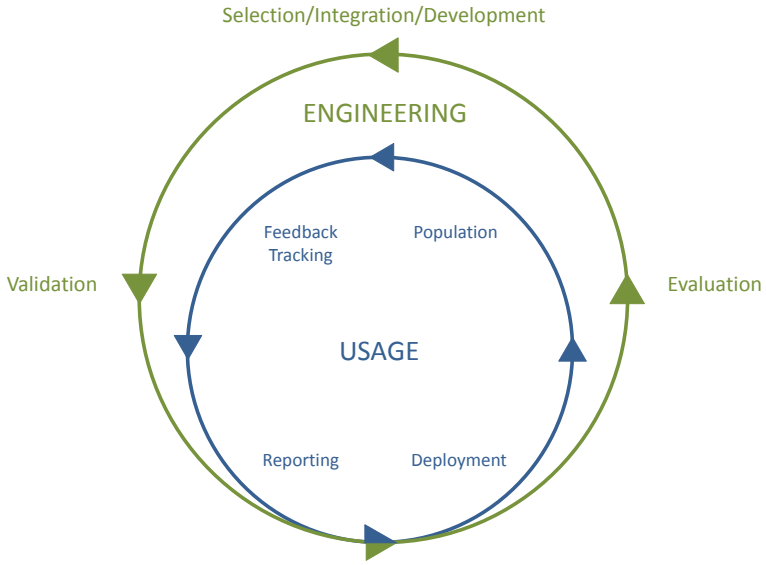

Figure 2.2: The Corporate Ontology Lifecycle Methodology COLM

2.2 Scenarios / Use Cases

Aiming at the adoption of semantic technologies to corporate environments it is very essential to have concrete scenarios and use cases. Especially for ontology engineering tasks it is very hard to find representative real world scenarios which are relevant for companies. For that reason we are going to introduce two scenarios from our industrial partners which demonstrate two main aspects which we focus in corporate ontology engineering during this phase of the project CSW.

In order to enable an agile and evolutionary development of corporate ontologies which reflects the progress of the company and allows for better cost estimation by separating between engineering and usage phases of the ontology lifecycle, the first of the two investigated aspects is ontology versioning. The second is ontology modularization and integration allowing less investment costs by modular reusability of existing ontologies and easy management as well as maintenance by improving understandability through well sized ontology modules.

2.2.1 Ontonym

The Ontonym GmbH based in Berlin is a provider of ontology-based services which supply the appropriate background knowledge for the client's applications and services. Thus, Ontonym addresses the gap that the missing expertise of ontology development and management is a key barrier for internal adoption of ontology-based systems in infrastructures especially of small and mid-sized enterprises. Figure 2.3 shows how the clients of Ontonym access an ontology-based system via Web service interfaces. To facilitate an early access to the

service the evolving ontology prototypes underrun an agile lifecycle which is hidden from the client who just benefits more and more from the improved ontology.

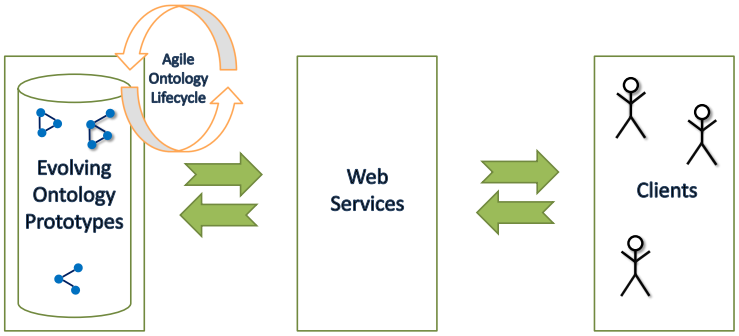

Figure 2.3: Web services hide the agile ontology lifecycle from the user

Ontonym needs an efficient ontology versioning strategy since the company has to handle an often and rapidly changing ontology which is accessed and maintained by several people internally. On the one hand, these people are the ontology engineers. But, on the other hand software engineers have to access the ontology as well from time to time. The former prune and refine the ontology depending on new knowledge they observed by evaluating queries and responses of their services as well as time consuming research in the customer's domain. The latter have to align the service functionalities depending on the capabilities of the ontology. In summary, Ontonym needs a distributed ontology version control system for users with disparate viewpoints and skills. Ontology engineers as well as software engineers have to use this system in an intuitive way.

Ontology versioning also comes into play for Ontonym since the company needs support for branching and merging of ontologies which satisfy the individual needs of different customers. That is, it is possible that the ontologies of various customers of the same economic sector only differ in certain specific details which depend on each customer's requirements while they have a common base model. In this case Ontonym has to handle depending branches of ontologies and merge parts of them in case of an update on the common base model. More, it is possible that the requirements of a customer change that much over time, that he wants to create branches of the primary base version. Again, in the case of an update related to common parts of the branches or to build a completely new version which integrates the modeled knowledge of all branches, a merge is necessary. Figure 2.4 depicts a representative branch and merge scenario of an evolving ontology.

2.2.2 FIZ Chemie

The information center for chemistry Fachinformationszentrum Chemie (FIZ Chemie) in Berlin provides networked information infrastructure and state-of-the-art, scientifically sound information services. It develops information sys-

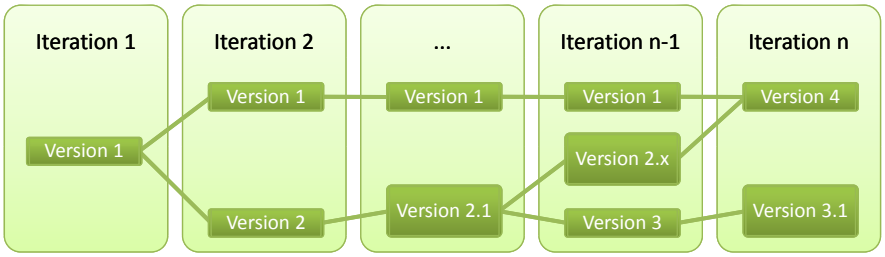

Figure 2.4: A representative versioning workflow

tems and supply customized content upon request for public authorities as well as companies in private industry. It has access to a large chemistry library with ebooks from various fields within chemistry. For an easy access to that information FIZ Chemie wants to provide a personalized semantic search through an intuitively usable interface to its customers. For that reason semantic description of the domain and the profile of the user is necessary. Due to the huge size and complexity of the domain there is the need for modular ontologies for specific fields within chemistry to represent the expert's vocabulary and guide the search in an efficient way by flexible and modular composition of necessary knowledge.

As a very basic natural science chemistry is a very suitable domain for the reuse of already existing public ontologies in order to decrease the investment cost and reduce the time to deployment. In fact, there are numerous publicly available candidate ontologies, which can be used as a basis for the development of an applicable ontology. Relevant parts can be extracted as modules and customized for the own requirements. During the application different modules can be integrated in order to compose a knowledge base in a flexible and application-dependent manner.

2.3 Ontology Modularization and Integration

Divide and conquer is a very well established idea in the field of software engineering. Component-based development of large and complex software systems by small well defined building blocks simplifies the understandability as well as the management and leads to reusable software modules and a scalable overall system. Accordingly, designing ontologies in a modular way is intuitively promising in order to benefit from the same advantages. However, modeling knowledge represented by ontologies is fundamentally different than software modules, which are mainly describing processes. Thus, software engineering methodologies and tools cannot be transferred easily to ontology engineering, which necessitates further investigation. The overall goals of modularization are the following: (1) creation of components, which can be reused in a flexible way and lead to (2) scalable and (3) personalized utilization; (4) increased understandability for humans by well-sized modules avoiding too large ontologies, in such a way that the (5) complexity can be managed easily.

In order to clarify the need for ontology modularization and integration and its advantages for especially corporate ontology engineering it is very impor-

tant to have a close look at its impact on the previously introduced gaps and requirements. Keeping the corporate context in view, especially the gap G4 (the cost-benefit-estimation gap), modular reuse of existing ontologies is very important to reduce the capital expenditure and the time to deployment of ontologies. Additionally, well-sized modules are increasing the understandability for humans and allow complexity management and easy maintenance, which leads to a reduced need for ontology engineers (R2) and enables evolving ontologies (R3). Modular ontologies are also beneficial for efficiency, scalability, and personalization. During the usage only modules which are necessary with respect to the application and the user can be selected and integrated so that loading needless parts and wasting storage as well as computing power can be avoided.

Optimizing ontology modules for users and applications presumes that there is meta information available about them. Besides the structure and the semantic content of the ontology itself the closed and controlled environments of companies allow obtaining additional information beneficial for the modularization and integration of ontologies. In contrast to open and large-scale systems as the World Wide Web, corporate settings make it possible to take information about the usage and evolution of ontologies, as well as the structure of the environment comprising departments, workflows and users into consideration.

By observing the usage and having knowledge about the users, their workflows and departments it is possible to conclude the relevant parts of an ontology for specific use cases. This allows to create modules which are optimized for such use cases and leads to personalized and efficiently usable ontologies. Furthermore, observing the evolution of the ontology helps to identify reliable and vague parts. Concepts and relations which are used frequently but are not changed can be marked as reliable, while others which are changed very often with respect to their usage can be seen as vague needing more investigation by ontology engineers and / or domain experts.

2.3.1 Technical Concept

In the following we will demonstrate functional components within a corporate environment which are necessary to realize modularization and integration of ontologies in order to meet the aforementioned goals. Figure 2.5 illustrates the relevant parts of the overall system architecture and shows the interrelation between them.

The Ontology Engineering Editor is a front end application for ontology engineers and knowledge managers. It is a tool to create, load, visualize and modify ontologies directly. The Ontology Discovery & Integration component has access to the local ontology repository as well as the publicly available repositories as Watson, Swoogle etc. According to ontology requirements and based upon module descriptions it is able to find and integrate ontology modules. These requirements can be defined manually by ontology engineers using the Ontology Engineering Editor or automatically by Semantic (Web) Applications. The Modularization component of the Ontology Framework is capable of module extraction and ontology partitioning. According to feedbacks from the Feedback Repository it can partition the ontology automatically. Furthermore, it is possible to use its module extraction algorithms manually through the Ontology Engineering Editor. For a better understanding of the components and

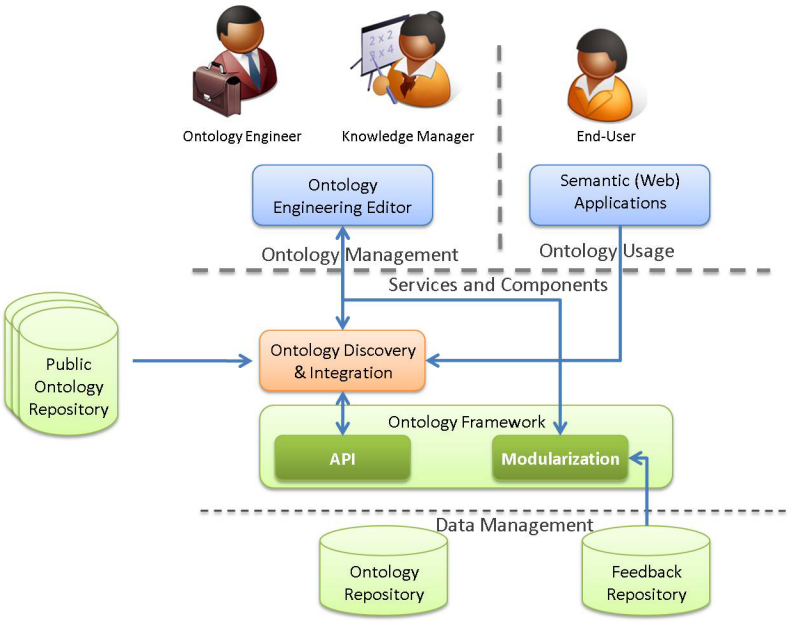

Figure 2.5: Technical Components for Ontology Modularization and Integration

their interrelation we will describe a specific scenario. It is about the modular reuse and integration of existing ontologies.

2.3.2 Modular Reuse

A four step methodology for modular ontology reuse is depicted in Figure 2.6.

Figure 2.6: Modular Reuse of Existing Ontologies

According to these steps the components of the technical concepts can be used in the following way.

The Ontology Discovery & Integration component allows ontology engineers and knowledge managers as well as end user applications to find relevant ontologies. At the beginning of the ontology development process ontology engineers together with knowledge manager and domain experts need to analyze the domain and identify requirements. They can define concepts and properties which need to be comprised by the targeted ontology.

Given these requirements the Ontology Discovery & Integration component searches for relevant ontologies in different ontology repositories and search engines as Watson, Swoogle etc. and returns a set of candidate ontologies. Domain experts and ontology engineers control such candidate ontologies and decide if they have to redo the search with modified requirements or if the results are acceptable and they can go to the next step.

In the next step the candidate ontologies together with the requirements are sent to the Modularization component. Only really relevant parts of the candidate ontologies are identified based upon the requirements. These parts are extracted by using module extraction mechanisms as closed independent modules. Each module is associated with a part of the requirements which it meets. The modules are controlled by the ontology engineers and domain experts. If necessary the Modularization component is used again with the same candidate ontologies but with modified requirements. As soon as the modules are acceptable the integration phase is entered.

The ontology integration components is responsible for integrating the modules extracted from very different candidate ontologies into one overall ontology and to remove redundant and overlapping parts. The created ontology has to fulfill all requirements. It is checked by the ontology engineers and domain experts. If they are not satisfied with the integration they can redo this step with new requirements.

Following the integration process which provides an acceptable ontology the Modularization component creates different well-sized building blocks with metadata by using partitioning mechanisms. The metadata expresses information about the content of the building blocks and their interrelation to other building blocks, i.e. information about how to integration them.

These building block together with metadata about them are stored in the local repository and can be downloaded and put together in a flexible way by different applications in various situations.

2.4 Ontology Versioning

The versioning of ontologies is important when continuously evolving ontologies are in focus. For our work we state two aspects as the most relevant reasons why the continuous evolution of ontologies is indispensable in context of corporate ontology engineering. First, there will ever be artifacts which change necessarily over time and influence the knowledge represented in corporate ontologies, e.g. the product portfolio or the department structure. Second, using or re-using a rudimentary existing ontology in the domain of discourse can lessen the costs for initial ontology development. Furthermore, we also adopt two base perspectives in which ontology versioning has to be applicable. The first one is the perspective of the vendor of ontologies and ontology-based services or software and the second one is the perspective of the user of ontologies as part of his information systems infrastructure.

Existing approaches yielded valuable results with focus on tracking of changes, change detection and calculation of the so called ontology diff (or semantic diff) to facilitate efficient and conflict free versioning mechanisms. However, an approach which provides commit and rollback of ontology versions on the concept level is missing. Such an approach can upgrade the distributed development and evolution of ontologies effectively and thus it is in focus of our work.

2.4.1 Requirements of a Flexible Ontology Versioning System in the Corporate Context

As we mentioned it in the introduction of this chapter, we raised requirements on corporate ontology engineering in general during the first phase of the project Corporate Semantic Web. Now we adopt the requirements R2 (the already existing and running system must not be disturbed) and R4 (the need for ontology engineers have to be minimized to reduce costs) and derive the following specific ones for the design and implementation of an ontology versioning solution which is applicable in the corporate context flexibly.

1 The system has to deploy the actual ontology version(s) to various applications directly. (related to R2)

2 The system has to be usable for non-experts in the field of ontology engineering. (related to R4)

3 The system has to be usable with other tools than dedicated ontology engineering applications. (related to R4)

2.4.2 User Groups

With reference to requirement (2) we state that other people beside *ontology engineers* have to access the ontology version history and derive the users which form the relevant target group for our system. From a technical perspective, these people are *software engineers* and in some cases *IT administrators* or *IT managers*. They check out the actual ontology versions to verify that the modeled knowledge is conform with the requirements of the software systems which use it. That means they search for necessary ontology primitives and structural aspects of the conceptualization, e.g. essential properties or depth of the hierarchy.

2.4.3 Design of the SVoNt Version Control System for OWL Ontologies

We decided to set up on top of the well-known version control system SVN which is typically used for versioning of source codes in software engineering contexts. Thus, the selection of SVN fulfills requirement (2) perfectly from a non-technical point of view. To facilitate ontology versioning in the sense of all of the mentioned requirements, the text-based approach of SVN has to be extended to act on semantic structures of OWL ontologies and finally integrated into an ontology management framework which handles the access to appropriate ontology versions for distributed applications.

We implement an extension of the classical SVN approach for the server and the client side as it is depicted in Figure 2.7.

On the server side we wrapper the SVoNt server architecture around the classical SVN server architecture and keep all functionalities of the classical SVN system working as before. Internally, we add two major components which facilitate the additional functionality, namely consistency checks and generation of the ontology diff. The former supports the detection of of semantic inconsistencies when a new version is committed to the system and results an alert

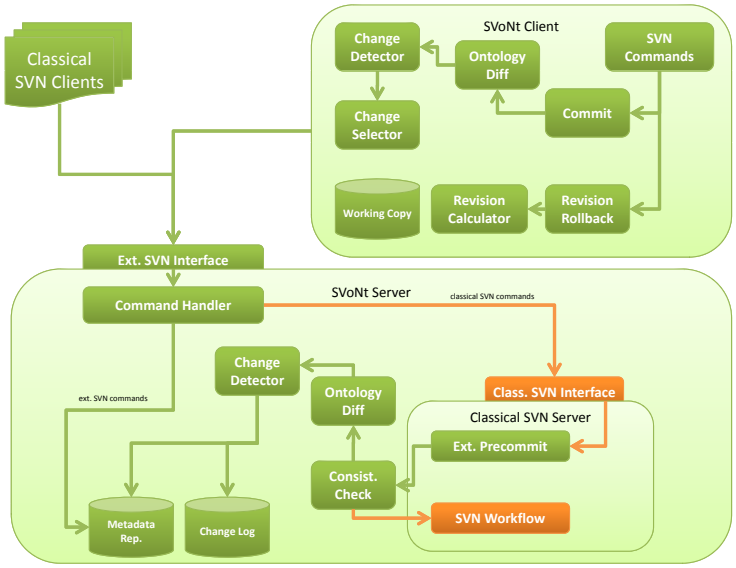

Figure 2.7: The SVoNt architecture

and avoids the commitment in case of a failure. The latter is applied after the consistency of the committed version is approved and provides the semantic distinction of the endmost and the new ontology version. From the set of differences we calculate a set of granular change operations and store them into a change log.

Our server side approach allows the use of any classical SVN client since we kept all actions and commands working as before. However, we implement a special SVoNt client as well which will provide

- the visualization of revision numbers on concept level,
- the partial commit of selected sets of changed concepts and
- concept-oriented rollback to former revisions.

To do that the SVoNt client accesses the SVoNt server change log and retrieves information about the atomic changes. A special tree view will provide an intuitive visualization as it is commonly familiar from the SVN visualization of source code repositories.

2.4.4 Incorporation into the Overall Concept

Our SVoNt approach is a core component of a COLM ontology framework which will provide an integrative set of functionalities to store, access and maintain OWL ontologies within an information systems infrastructure. Our first draft proposal of this architecture is shown in Figure 2.8.

This depiction describes the various systems which access an ontology, respectively an ontology repository, for certain tasks, e.g. the ontology editors for

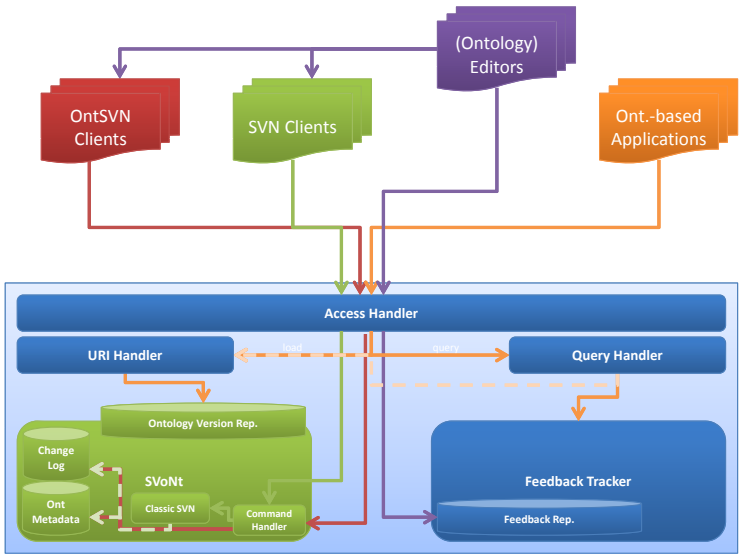

Figure 2.8: Proposal for an ontology management framework architecture including the SVoNt system

manipulating an ontology or ontology-based applications for querying them. A central component is the feedback tracker which observes each action performed on the ontologies and supports the detection of new knowledge or potentially weak parts of the model.

2.4.5 Working Plan

Our working plan for the development of an ontology versioning system based on SVN is divided into four packages (WP 10.2.1 - 10.2.4) which are performed from 04/2009 until 03/2010. At the end of this process we expect a version 0.9 of both, an SVoNt server and an SVoNt client, which are ready for beta testing in enterprises. The SVoNt server will available as a standalone component as well as part of an integrated COLM framework. During the last year of this phase of the project we are planning to test and mature the COLM framework including the SVoNt server so that it is ready for stable productive use in enterprises.

2.5 Conclusion

In this chapter we introduced a technical concept to realize corporate ontology engineering. Based upon the Corporate Ontology Lifecycle Methodology (COLM) we investigated in two important aspects, namely ontology versioning and ontology modularization and integration. Both are essential to enable an agile and evolutionary development of corporate ontologies which reflects the progress of the company and allows for better cost estimation.

2.5.1 Modularization and Integration Dimensions of COLM

Ontology modularization and integration allows less investment costs by modular reusability of existing ontologies and easy management as well as maintenance by improving understandability through well sized ontology modules. It is in different ways integrable into the lifecycle. As illustrated and mentioned previously COLM facilitates cost estimation in the run-up of cost-intensive evolution steps. The modularization and integration dimensions of COLM helps to decrease investment costs as well as the operational costs and supports to realize ontology adoption to the corporate environment by keeping the incentives of a company in view.

At the very beginning of COLM during the selection and integration phases it is reasonable to look for existing ontologies for the sake of reusability, because developing ontologies from scratch is a very cumbersome and time-consuming task. Expecting candidate ontologies which perfectly fit into the targeted system is unrealistic, some customization will be necessary in order to adapt candidate ontologies and make them useful. At this point modularization is an important mechanisms to allow reusability even if the candidate ontologies are not usable in their original form. The possibility of extracting only relevant parts of existing ontologies and integrate them in order to achieve a useful ontology decreases investment costs drastically and makes ontology application realistic and really attractive for companies.

Modularization during the lifetime of ontologies is also possible. This can be done based upon diverse aspects. In the feedback tracking and reporting phases, the closed and controlled corporate environment allows obtaining information as relevance of concepts and relationships regarding departments and application. It also enables to observe the evolution of the ontology and allows to identify vague parts which change very often and well-established parts which change less frequently.

Finally, in case of context-sensitive and ontology-based applications, which are able to define ontology requirements, an additional aspect of ontology modularization is possible. During the deployment phase, while application need to load ontologies, optimized modules regarding the application context, can be identified and extracted in real-time. This would lead to personalization and increased efficiency of the ontology usage, because loading needless parts and wasting storage as well as computing power can be avoided.

2.5.2 Versioning Dimensions of COLM

Efficient ontology versioning is a key enabler for the agile evolution of ontologies in the setting of enterprise information system infrastructures. It facilitates the management of various coexisting representations of corporate knowledge which is used by heterogeneous applications.

Our SVoNt approach for ontology versioning incorporates into the overall approach of COLM as the central backend component which encapsulates the ontology repository. Combined with other parts of the COLM ontology management framework the whole development, update, release and access chain is handled by this system.

2.5.3 Corporate Ontology Engineering in CSW / COLM as Business Enabler

The work packages ontology versioning and ontology modularization and integration are refining the Corporate Ontology Lifecycle Methodology (COLM) towards a fundamental business enabler for utilizing ontologies in companies. By providing an ontology framework it prepares the ground for semantic applications. In corporate ontology engineering we are always keeping the real world constraints in view, through close cooperation with industrial partners.

Chapter 3

Corporate Semantic Collaboration

In this section we describe the conceptual design of a light-weight ontology editor which is part of the work package "Collaborative Tool for Modeling Ontologies and Knowledge." Figure 3.1 highlights the components of the Corporate Semantic Web architecture that are involved in building such an editor. These components allow to access ontologies (ontology framework), to keep track of user interaction (feedback tracking), and to assist a user in modeling ontology concepts (matching framework).

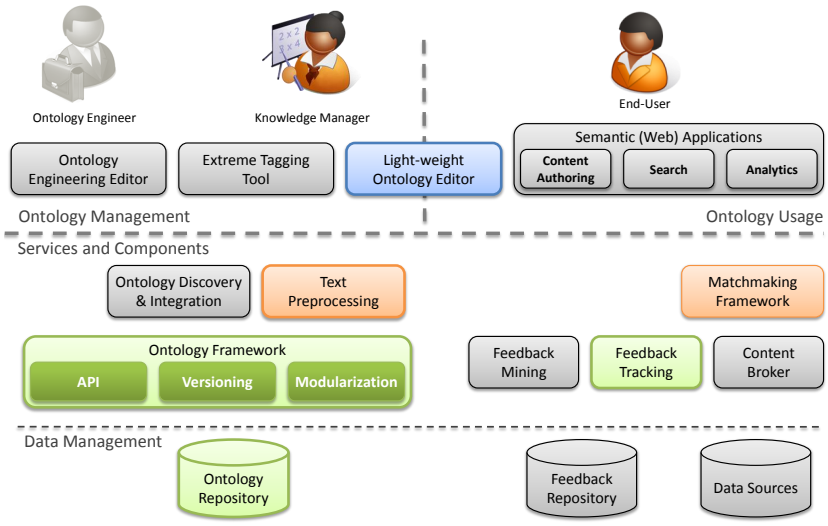

Figure 3.1: Overview of the architecture

3.1 Collaborative Tool for Modeling Ontologies and Knowledge

In contrast to existing ontology editors and our work on ontology engineering, we focus on developing an easy-to-use editor for non-experts in engineering of ontologies. Thus, we concentrate on the usage cycle of the ontology lifecycle (see Corporate Ontology Lifecycle Methodology (COLM) in Chapter 2) – the period after a first version of an ontology has been deployed – and primarily address the needs of domain experts. But the editor also supports ontology engineers as it also gathers feedback about changes in the ontology. The feedback is analyzed by ontology engineers to improve the quality of the ontologies and deploy a new release of them.

We structured this section as follows: First, we describe three real-world scenarios to illustrate the need for such an editor. Second, we identify and describe user groups and functionality of the planned editor. Finally, we specify the architecture and sketch the working plan.

3.1.1 Application Scenarios

Our cooperation partner, *Ontonym GmbH*, has made the experience that companies often carry out a project on ontology development in cooperation with ontology engineers. The ontology engineers themselves talk to domain experts to learn details about the application area. Since the involvement of external ontology engineers is expensive and the ontologies nevertheless have to be kept up-to-date, companies want their employees to be able to extend and to modify the ontologies. However, existing tools address primarily ontology experts and, moreover, do not cover and support the usage cycle of the ontology life cycle [9]. This is especially true for commercial ontology modeling tools.

In collaboration with *EsPresto AG*, we recently developed a demonstrator illustrating another application domain of a simple ontology editor: We extended a conventional wiki engine, XWiki[1], by a plug-in that recognizes concepts in the text of a wiki page and uses background knowledge, e.g., an ontology, to offer a search for similar and related terms to the wiki user. The suitability of the similar and related terms highly depend on the background knowledge. Since wikis can be applied to any application domain, the wiki users themselves should mainly be responsible to maintain the background knowledge. A light-weight ontology editor will be the instrument for this purpose. Considering semantic wikis as a related technology to our XWiki approach, we think that they also suffer from the lack of domain ontologies. Thus, our editor would facilitate the dissemination of semantic wikis.

Regarding our third scenario we cooperate with the makers of *DBpedia*[2]. DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. Their database is currently emerging to a major hub of the linked data approach. For example, the BBC developed a system that obtains unique identifiers from DBpedia to connect the content of their business units. As a result, they are able to enrich their content

[1] `http://www.xwiki.org`
[2] `http://dbpedia.org`

automatically and increase the visiting time on their website[3] by providing links to related information sources of their own website instead of external one. In a discussion a developer of the DBpedia pointed out that the vocabulary used in the infoboxes of Wikipedia are arbitrary choosen. To provide high-quality query results, they are currently developing an ontology representing the Wikipedia content consistently. Thus, they have a need for an easy-to-use editor that allows users of DBpedia to map the vocabulary of the infoboxes to their ontology.

3.1.2 Editor Functionalities

In a first step to design an ontology editor we compiled a list of user groups operating ontology engineering tools and a list of operations to handle ontologies. Compiling these lists we assumed that ontology engineers have already made basic design decisions on the ontology model and deployed a first version of it on a production system. The employees of a company are in charge of extending the ontologey. Thus, we consider in particular the usage cycle of the ontologies.

The following list gives an overview of operations on an ontology. Some of these operations may require a deeper technical understanding of ontology engineering than other one. In the list of operations we indicate the level of required technical knowledge with '+' signs, ranging from ' +' (low) to ' +++' (high). In addition, further operations are conceivable, but we omitted them, because they require a too high technical knowledge. Please note, that we do not make any assumptions on the procedure of applying changes to the ontologies. For example, the changes may be immediately permanent and visible to all or have to be reviewed by a knowledge engineer.

Search concepts A user searches for concepts in the ontologies that correspond to a given a search term. If context information of the user are available, the search engine considers them for ranking the results. (+)

Navigate Starting from some concept the user navigates through the ontology along relationships. Context information may be used to narrow the offered navigation paths. (+)

Add concepts/relationships A user adds a new concept to an ontology by typing a word into a form field, e.g., while he is annotating or tagging a document. On the one hand, he can rely on the system to find the right place for the word in the ontology. On the other hand, he may revise the suggestions of the system or add relationships manually. (++)

Remove concepts/relationships Delete concepts and relationships from the ontology. (++)

Move subgraphs Performing this operation, a user changes the parent of a concept and, thus, moves a concept together with its children to another place of the ontology. (+++)

Tag concepts Tagging of concepts aims at putting concepts into context and, therefore, it pursues a similar goal as creating relationships between concepts. Although the tags do not need to be contained in the ontology, it

[3] `http://www.bbc.co.uk/blogs/radiolabs/`

helps an ontology engineer to understand the actual meaning of a concept and decide on its placement in the ontology. (+)

Comment and discuss Users can add comments to parts of an ontology. Using this functionalities the users can collaboratively clarify the meaning of concepts and relationships and the modify the ontologies accordingly. (+)

Add data using a pattern Ontology design patterns are templates for creating new elements in an ontology. The user fills in and submits a form and the system populates the information to the ontology, e.g., enter the names of employees into a given organigram. (++)

Manage design patterns A user can create, modify, and delete ontology design patterns that can be used by other users to populate and modify ontologies. (+++)

Performing modifications on ontologies the system has to ensure that all data is still in a consistent state after applying the modifications. To add a concept, for example it has to verify that this concept does not already exist in the target ontology. If it does, the system has to execute some procederes to solve this conflict, e.g., automatically disambiguate the meaning with the help of context information. A more challenging situation arises if a user moves a subgraph of the ontology, because it may affect many other concepts, e.g., all subclasses of the moved concept.

3.1.3 User Groups

Methodologies for ontology engineering such as HCOME [7], METHONTOLOGY [10], and OTK methodology [12], refer to the user groups ontology engineers, experienced users, knowledge managers, and system designers. In the usage cycle of COLM, we distinguish between different user groups that influence the further development of ontologies. Depending on background knowledge and experience of users we consider the following main groups:

Application user Application users are not aware of the usage of ontologies in the system; for them the functionality of the application stands in the focus. Thus, they are not directly involved in the ontology engineering, but they provide implicitly feedback to ontology engineers by just using the application, e.g., the system observes the interaction between user and application.

Ontology user This group will mainly use the ontology read-only as a vocabulary, for example, to annotate content. From their point of view on ontologies they are especially able to judge the adequateness of the domain model. Their comments and the discussions between them are valuable feedback for knowledge managers and ontology engineers.

Knowledge manager Knowledge managers are responsible for maintaining the ontology in the context of a corporation. Thus, they are not only using or commenting an ontology but actively modifying it. A knowledge manager may be responsible for the complete ontology or only for a part of it, e.g., an ontology module corresponding to a particular department of

a company. The changes of ontology managers may be subject of reviews by ontology engineer.

Ontology engineer An ontology engineer has a deep understanding of modeling of ontologies and are generally not an expert of the application domain. In contrast to knowledge managers he has to consider the complete ontologies and the interdependencies between ontology modules, but has not necessarily to know the application domain in detail. Furthermore, an ontology engineer has deep knowledge about respresentation formats, technologies, and methodologies.

In literature, the role of domain experts is often mentioned. From our viewpoint, domain experts are a group of persons that may belong to all mentioned groups. Thus, we do not list them separately.

Corresponding to the tasks of the groups in the context of ontology engineering, we associate a set of operations with them that they will typically perform (Table 3.1).

User Group	**Operations**	**Level**
Application user	no operations in cause of indirect ontology usage	–
Ontology user	search, navigate, comment, discuss, tag concepts, use design pattern	(+)
Knowledge manager	all operations of ontology users, modify the ontology, manage design pattern	(++)
Ontology engineer	all operations	(++)

Table 3.1: User groups and their typical operations

3.1.4 Design of the Light-weight Ontology Editor

Designing a light-weight ontology editor we focus in the first step on the user group consisting of ontology users. In a second step we will extend the functionality of the editor to support knowledge managers. The group application users do not directly interact with the ontology, instead the system collects feedback. Ontology engineers, in contrast, need a dedicated tool to manage ontologies efficiently (see Chapter 2).

Figure 3.2 puts the leight-weight ontology editor into the context of the overall system architecture of the CSW project. The user interact with this editor which in turn interacts with the extreme tagging tool or the backend components, i.e., feedback tracking, matchmaking framework, text preprocessing, and the ontology API.

With the help of the extreme tagging tool [13] a knowledge manager can collaborate with domain experts and obtain new concepts for an application domain. The feedback tracking component is responsible for collecting feedback of the interaction between a user and the ontology editor. A knowledge manager, for example, can analyze the feedback to discover concepts that are often searched but are currently not contained in the ontology.

The matchmaking framework is a central component for the ontology editor. It is used for the following tasks:

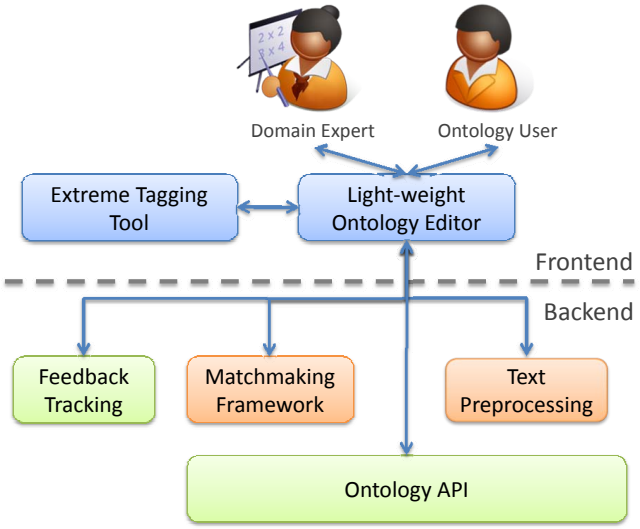

Figure 3.2: Architecture of the light-weight ontology editor

- Search (similar) concepts in the ontology to use for performing a task, e.g., annotate documents.
- Check that a concept is not already contained in the ontology when an user wants to add it to the ontology.
- Suggest a location in the ontology for a newly entered concept, e.g., a potential parent concept.

To improve the results of invoking the matchmaking framework, the ontology editor may provide context information of the user. Furthermore, the matchmaking framework may use information from external data sources, e.g., Zemanta Semantic API[4], OpenCalais[5] and DBpedia[1].

In the context of the ontology editor the text preprocessing component is an auxiliary means to help the user to recognize concepts that are already contained in the ontology or to suggest new concepts. Thus, a user saves time, because he does not try to add already existing concepts.

Finally, the ontology API allows the ontology editor to access and modify the ontology. The component implementing this API guarantees that changes to the ontology are performed consistently.

In Figure 3.3, we present the components of the ontology editor. We distinguish between user interface components (upper part) and core components (lower part). We design two kinds of user interfaces, a Web based editor and a embedded Web based editor. While the purpose of the *Web based editor* is to support the task of ontology management, the *embedded Web based editor* enables a user to make changes to the ontology on the fly, e.g., as he is doing some other task. For example, after a user finished editing the content of a

[4] `http://www.zemanta.com/`
[5] `http://www.opencalais.com/`

wiki page, he wants to annotate it with some concepts of the ontology. At this moment he does not need a full-featured interface of the ontology editor but only a few functionalities, e.g., a search field or some suggestions for annotating the text.

While we implement the Web based editor as a standard Web application, we consider to use an RDF JavaScript API as proposed in [4] for the embedded Web based editor. This API allows a seamless integration of semantic technologies into a webpage by modifying RDF models on the client side and synchronizing them with a server.

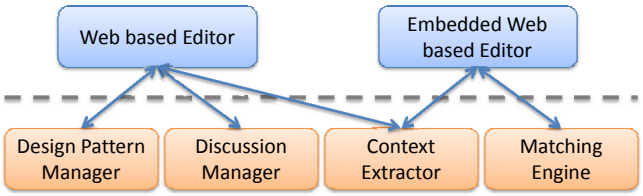

Figure 3.3: Components of the light-weight ontology editor

The editors invoke the functionalities of core components such as the design pattern manager, the discussion manager, context extractor, and annotation engine. These components in turn access services and components of the architecture depicted in Figure 3.1. The *design pattern manager* provides – as its name suggests – operations on ontology design patterns as described in Section 3.1.2. Furthermore, it extracts the RDF data from completed patterns for further processing. The *discussion manager* handles user discussions about concepts and relationships of the ontologies, keeps track of arguments, and supports the process of decision-making. Implementing this feature, we found inspiration in DILIGENT [11] which supports controlled discussions besides others. The *context extractor* is responsible for detecting the current context of a user, e.g., to improve the quality of suggestions for placing new concepts into an ontology. The *matchmaking engine* searches for concepts that are related to a given search term. This feature is used, on the one hand, to find concepts for annotating texts and on the other hand, to suggest the placement of a new concept in the ontology and its relationships to existing concepts.

3.2 Conclusion

In this chapter we presented scenarios indicating the need for a light-weight ontology editor that can be operated by non-experts. In contrast to existing methodologies and tools which only address the user groups ontology engineers, experienced users, knowledge Managers, and system designers, we explicitly address application user which have generally experience with Web browsers but not with ontologies. Furthermore, we described operations of the editor which the editor should support to be useful in our scenarios. Finally, we presented the conceptual design of our light-weight ontology editor and described its components.

Chapter 4

Corporate Semantic Search

This section presents the conceptual design for the work packages "Search in Non-semantic Data" and "Search Personalization". Figure 4.1 highlights the components of the Corporate Semantic Web architecture that are involved in realizing both work packages.

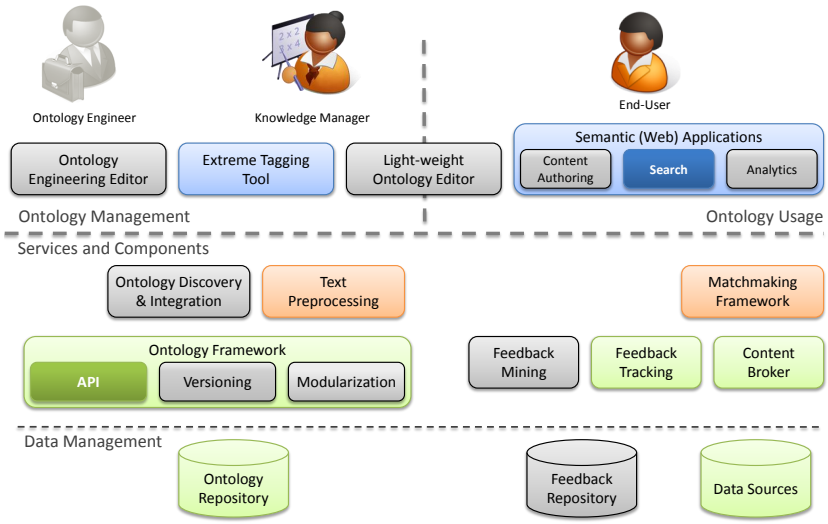

Figure 4.1: Overview of the CSW Architecture with Focus on Semantic Search

4.1 Search in Non-Semantic Data

The research pillars of corporate ontology engineering and corporate semantic collaboration as described in previous sections concentrate on formalizing enterprise knowledge in form of ontologies. The preservation and advancement

of corporate ontologies insist on ontology learning and ontology population. These crucial tasks can be accomplished by semi-automatic search for ontology concepts, complex relations between them, and instances in corporate text collections. From this point of view, the working package *search in non-semantic data* should deal with the extraction of "semantics" from the corporate text collections. Yet mostly, this "semantics" is not explicit assigned to the documents in corporate text collections. Texts are given in electronic form, in best case formatted in XML. Besides, there is a *tagging* technique which is the most common way used by many users, also by employees in companies, for annotating knowledge and information. Tagging helps in understanding the meaning of information being tagged. Combining both: search in text collections and tagging, we can satisfy the need for extracting "semantics" from texts and create an automatic approach for "understanding" given text from a text collection.

Aiming at *trend mining* as an example of the semantic search, in our work on search in non-semantic data we focus on the method that unifies statistical learning approach with semantic technologies in order to enable search for complex relations in text collections.

4.1.1 Application Scenario

One application scenario for the trend mining is trading on financial market.

The more information of a good quality a trader has, the better the base for decisions. A lot of useful information one can find in business news. Through the free news services available on the Internet, most of investors are overwhelmed with information. The process of making decisions becomes suddenly more and more difficult: information that is free is hard to avoid. Therefore, finding good quality information is the challenge. All market analysts have access to the

Figure 4.2: Trading: "real world" scenario

same information about market trends, but the best ones are able to enhance

such statistical information by bringing their own experience and their ability to make predictions from associated qualitative information into the decision making process. However, this analysis process cost time and money. In order to support human analysts, new methods are required.

A number of methods and computer programs have been developed in order to support decision makers in the investment business but most of them are based only on analysing numeric data and generating charts with stock exchange values. Nevertheless, traders still have to filter, read and analyse relevant information found in business news. The consequences of an investor missing an important piece of information, or misinterpreting a situation can be severe. The burden of success of the investors is therefore linked to their ability to collect, filter, understand, and react to reports about these information. Semantic Technologies are very promising solution for integrating expert knowledge in order to automatic analyse and "understand" important information in business news.

During our project work, we are collaborating with the JRC GmbH[1] company in order to exploit the needs for and obstacles of the automatic detection of trends in business news.

In our work, we are focusing on a general approach and developed concept that can be applied also for trend mining in market research as well as in medical diagnosis.

4.1.2 Trend Mining

State-of-the-Art in Emergent Trend Detection in Text Mining

In an overview over the research on Emergent Trend Detection in Text Mining [6] several systems that detect emerging trends in textual data are presented:

- TOA (Technology Opportunity Analysis), TimeMines, New Event Detection, Patent Miner, HDDI (Hierarchical Distrubuted Dynamic Indexing), Theme River, etc.

Also the project presented in [8] includes an approach for trend detection in news. Besides scientific projects, there are few useful tools and applications that concern with detecting trends from textual information, i.e. GoogleTrends and BlogPulse. However, current solutions do not include approaches for incorporating formalized expert knowledge in the automatic trend detection. We propose an approach for automatic expert knowledge integration in the process of trend detection without involving experts in annotating text collections or in studying the content of news. Our approach is based on using Semantic Technologies and Tagging for formalizing and collecting expert knowledge.

State-of-The-Art Relevant Tools

Concerning the Trend Mining based on texts and enhanced text analysis, there are many related projects on the Internet, scientific and commercial, as well as

[1] `http://www.jrconline.com`

services that we have to take into account to during our work: GoogleTrends [2], BlogPulse[3], SystemOne[4], Connexor[5], ClearForest[6], OpenCalais[7] The last one is very relevant and useful for our work regarding text analysis.

State-of-the-Art Relevant Vocabularies and Ontologies

Several projects that concern themselves with lightweight ontologies and extended vocabularies are relevant for this part of our work, in particular: ConceptNet[8] and OpenMind[9] of Massachusetts Institute of Technology, MoaT[10], WordNet[11], SentiWordNet[12], Wortschatz Uni Leipzig[13], DWDS[14], SKOS[15], SCOT[16]

4.1.3 Technical Concept due to Learning and Correlation Approach

Based on our Learning and Correlation Approach see4.3, we developed a concept for realizing our ideas as described in [5].

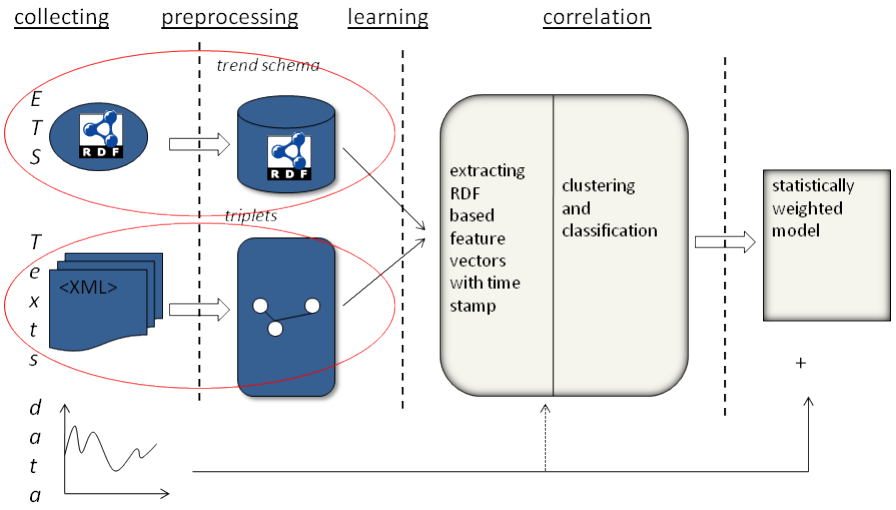

Figure 4.3: Learning and Correlation Approach

[2] http://www.google.de/trends seen on 22.01.2009
[3] http://www.blogpulse.com/ seen on 22.01.2009
[4] ttp://www.systemone.net/de/ seen on 22.01.2009
[5] http://www.connexor.com/ seen on 22.01.2009
[6] http://www.clearforest.com/ seen on 22.01.2009
[7] http://www.opencalais.com/seenon22.01.2009
[8] http://conceptnet.media.mit.edu/ seen on 22.01.2009
[9] http://commons.media.mit.edu/en/ seen on 22.01.2009
[10] http://moat-project.org/ seen on 22.01.2009
[11] http://wordnet.princeton.edu/ seen on 22.01.2009
[12] http://sentiwordnet.isti.cnr.it/
[13] http://wortschatz.uni-leipzig.de/ seen on 22.01.2009
[14] http://www.dwds.de/ seen on 22.01.2009
[15] http://www.w3.org/2004/02/skos/ seen on 22.01.2009
[16] http://scot-project.org/ seen on 22.01.2009

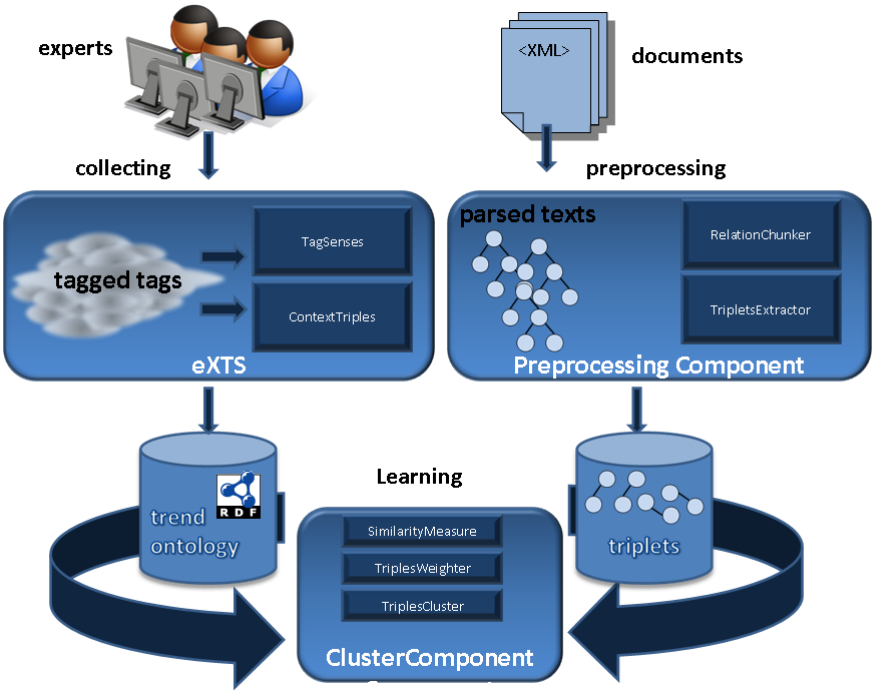

Figure 4.4: Package concept: Architecture for search in non-semantic data

In the following we describe the 2 components.

4.1.4 Collecting Knowledge with Extreme Tagging Approach

Automatic generation of relations between individual words is still a problem. The current approaches of computer linguistics do not produce satisfactory results. Therefore, people should be involved in this process. Previous tagging methods produce no precise specification of the relations. Each word is linked with another, without assigning the relation a more precise meaning. The Extreme Tagging System (ETS) approach [13] allows to tag the relations. ETS extends the common tagging method with an enrichment of the tags with a meaning. This can give the relations better semantics than just "is associated with", e.g. A "is part of" B or X "is a" Y. With the help of tags we can now use various graph algorithms to generate ontologies. Our first concept for generating semantic relations from folksonomy tags, named Tagsenses, has been developed during a master thesis[17]. In this thesis partitions of tag graphs are generated. Using these partitions ambiguities and synonyms can be found and polysems can be discovered.

[17]Mike Rohland, Master Thesis 2009: "Generierung von semantischen Relationen aus Tags innerhalb einer Folksonomie", FU Berlin

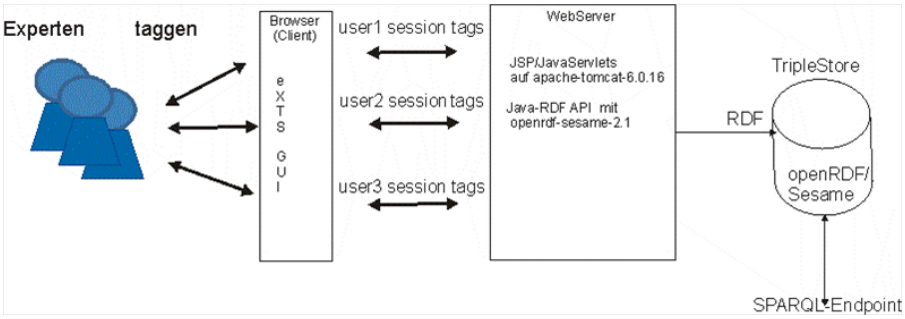

Figure 4.5: Extreme Tagging Tool: general architecture

Architecture of ETS:

ETS is realized as a web application for Apache Tomcat. In order to achieve an easy distribution of the software, we use HSQL DB as storage for the tags. This allows us to keep the database within the releases, so no external database is necessary. Furthermore, the communication between the JSP front-end and the backend is completely encapsulated by web services. So it is possible for us to use a central authority to gather the tags from various places. This can also be turned off, so eXTS can run locally and do not forward the data to the central authority. Currently, only a simple user permission system implemented. This allows only registered users to tag words and assign their tags a "sentiment". In a later phase, we plan to connect eXTS to the Tagsenses system mentioned above. With the help of domain-experts it will be easy to create a proper ontology, without teaching the experts to use an ontology creation program such as Protege.

4.1.5 Preprocessing Texts by Parsing and Chunking

The aim of the prepocessing component is to automatically generate triplets (trenary, syntactic relations) from a natural language text so that RDF-Tripels can be then generated from these triplets. We evaluated several tools and methods for their capability to fulfilling our goal.

First, we started with the evaluation of scientific reports and implementations concerning information-retrieval and concept mining from texts. These concepts use in general methods from computer linguistics. Almost all of these tools and papers apply to English which has has a different syntax than German. German is not as explored as English, but today several works are in progress. So we cannot use the approaches directly but as a lead to our goal. Today only proof of concepts exits, but with interesting approaches. They distinguish between statistic and syntactic processes and between noun and verb based processes.

The text in statistic processing is mostly classified with the TF-IDF (Salton) method to make an approximate statement about the content and to find to most significant words. After that, the text is sentence wise processed and noun-noun, noun-adjective or noun-verb tuples are extracted. If such a tuple occurs very often, you can say, these two words belong together. This method can be

improved, if you implicate synonyms for the words of the tuples. But in these methods, the syntax of the processed sentences is not regarded, so this can lead to wrong results. However, you will find many connections between the words (low precision, high recall). The biggest advantage of statistical methods is their language independence. Furthermore, entities can be recognized with regular expressions. Call numbers or salutations have usually the same structure; you can create an expression to find all entities in a specific domain. This entities gain a higher weight and you extract the tuples over relations of these words.

Syntactic methods are the other approach. Approaches from computer linguistic like POS-tagging are used here. The parsers for syntax-trees base on POS-taggers, because they have to know information about the particular words. You can identify dependencies between the words of the sentence in the created syntax trees and assign attributes to nouns (adjectives) or verbs (adverbs). Regular expressions, which are created for the particular language (e.g. X is part of Y; X, Y, Z are related to A), are also used to find relations between concepts. The biggest problem of syntactic methods is the dependency on the language to parse. A parser for English cannot be used to parse German texts. The parsers have often the problem to create the whole tree, but found relations are usually accurate. So, they have in difference to pure statistic algorithms a high precision but a low recall. The syntactic concept extraction bases on the so called tree-tagger. Our biggest problem at the evaluation was the low distribution of German in the computer linguistic. We found only two suitable parsers. First, the University of Stanford works on a project, which is available for free. This tagger was original developed for English, but there exist projects, which port it to other languages like Chinese or German. These ports do not work satisfyingly at this moment. The other result was a commercial product from Connexor. This tagger delivers far better results than the Stanford-parser and enriches the output with much more information. He finds the time and the case of a word if it is not in its baseform. Connexor provides a web based form to test its product. The results look sophisticated.

Because of the strengths and weaknesses of both approaches, both are connected to provide better results, as the use of only one of it. You can try to found entities with statistic methods and then create parse trees to find relations on the other entities.

It is very hard to test the right order of process steps or connect other sources of information like ontologies or databases. UIMA, which bases on GATE was developed to provide a framework for this. Both of them provide interfaces for which you can write your own Plug-Ins. These Plug-Ins and the built in methods can be ordered with a GUI to create a process pipeline.

Architecture of our preprocessing-component:

To generate RDF-Triples from natural language texts, these texts have to be processed in several steps. For this goal we develop the preprocessing-component, which uses several methods mentioned before.

The texts are in XML-Format. So they are read with Xerces and the significant tags are saved. We use a SQL-database, in our case HSQLDB for a later access to the texts and other caching purposes. If an actual version of our program is released, no database has to be set up, because everything necessary is stored in our version. The database is also used to store the parse trees

and the baseforms of the found words, because the creation of them takes a very long time. When we read a file, it runs through several steps. First, all abbreviations are replaced by their long form to avoid faults, e.g. the trailing point. This component is very rudimentary, because not many abbreviations are stored. In a later release several of them are stored in the database. After this first step, an other component which recognizes proper nouns and replaces them with pseudonyms processes the text. Proper nouns lead to failures in tree taggers. When the text is prepared in this way, the semantic analyzes can be started. At this state we use two different taggers, the Stanford-Tagger which is free and the demo version of Connexor. The Stanford tagger has the problem not to be very performant. He needs for a simple sentence of 20 words about 30 seconds and needs about 1 GByte RAM to fulfill its task. Even then, he can often not recognize the sentence structures properly. He cannot normalise word, we use the web service provided by Wortschatz Uni Leipzig. The results of these queries are stored in the database mentioned above. As other tagger we use the demo version of Connexor. This tagger provides far better results than the Stanford tagger, has a much better performance and normalizes the founded words. But the tagger is not free for use. Again, the calculated tree will be stored in the database for later access. Once all the texts of the corpus have been read, the IDF for each word is calculated, which later is used by us in a later processing step.

To show the results, we use Timeline and Timeplot both of them provided by SIMILE. Timeplot is useful because we want to show the documents by their dates and the share price of a company want to display. If you want a specific text to display, the TF of its terms is calculated and with the IDF of the terms a TF-IDF ranking on the content created. The parse tree belonging to the document is dynamically created using JavaScript.

4.2 Semantic Search Personalization

In the Web context, Baldoni et al. define the personalized access to Web data as *"the process of supporting the individual user in finding, selecting, accessing, and retrieving Web resources (or meaningful sub-sets of this process)"* [2]. User adapted services or applications require a user profile describing his or her preferences in order to be able to select the set of relevant information. Additionally, personalized systems require some form of representation of their application domain. Ontologies have the potential to fulfill this role by providing formalized machine-readable means for meaningful representation of both users and domain resources.

Whereas the research fields of corporate ontology engineering and corporate semantic collaboration, presented in previous sections, focus on formalizing enterprise knowledge in form of ontologies, this section concentrates on providing users within the corporate context, both internal (employees) and external (customers, business partners, etc.), with personalized access to this information. For realizing personalized search, however, there is a need for an architecture component which determines the similarity between user preferences and domain resources. In this section, we present a flexible domain-independent Semantic Matchmaking Framework for calculating semantic similarity of information objects represented as arbitrary RDF graphs, thus being applicable in

a wide range of scenarios.

4.2.1 Application Scenarios / Use Cases

Since the Corporate Semantic Web project focuses on application oriented research it is crucial to have real-world scenarios in order to evaluate our research contribution. However, the goal of the project is not to develop proprietary solutions only satisfying the requirements of one particular cooperation partner. Therefore, as far as the Semantic Matchmaking Framework, developed in this work package, is concerned, we aim at providing a flexible, generic architecture which can easily be customized to a wide range of use case. The use cases briefly described in this section represent possible application areas of the framework and serve as a proof of concept for our research.

Condat AG

Condat AG, located in Berlin, is a system integrator offering innovative IT solutions in the fields of media, mobility, and environment. In cooperation with *Condat AG* we are developing a personalized recommender system for multimedia content.

Since the emergence of Digital Video Broadcasting (DVB) viewers are overwhelmed with a huge number of TV channels (often exceeding 500) covering various fields of interest. The information about TV programs available in Electronic Program Guides (EPG) only provides limited search and filtering support (e.g. category search). Therefore, there arises a need for more sophisticated personalized search functionality. This can be realized by representing user preferences and metadata about content using domain ontologies, populating those ontologies with information from EPGs as well as by integrating external data sources for example the Internet Movie Database (IMDB)[18]. Finally, the realization of this scenario requires a matchmaking component for ranking of the multimedia content with respect to user likes and dislikes.

x:hibit

Our second use case deals with the cultural heritage domain. We cooperate with *x:hibit GmbH* - developer of the *Museumsportal Berlin*[19] where users can find information about over 200 museums, memorial places, castles, and others cultural institutions as well as their services and current exhibitions.

At present, most of the information on the portal can only be accessed via simple keyword search. Complex queries like *"find all museums and exhibitions related to Impressionism, open on Monday, with guidance in English and admission fee less than 10 Euro"* can only be realized by keyword search for *"Impressionism"* combined with long navigation paths, i.e. following every single museum/event link and subsequently opening a few more sub-pages providing information about opening hours, admission fees, services, etc.

A semantic representation of this information would allow to implement a more friendly user interface providing features like faceted browsing and timelines. Moreover, based on the background knowledge from the cultural heritage

[18] http://www.imdb.com/
[19] http://www.museumsportal-berlin.de/

domain formalized in ontologies, the system could provide further recommendations not explicitly stated by the keyword. For example, while searching for *"Claude Monet"* the user might also be shown events related to works by another (French) Impressionists.

Apart from the improvement of the search and navigation on the portal we also plan to develop a personalized recommender system based on the Push Principle. The key component of such a system would be the Semantic Matchmaking Framework for calculating the similarity between user preferences, formalized in ontology-based user profiles, and information objects representing museums and events.

4.2.2 Semantic Matchmaking Framework

The implementation of a domain-specific application architecture supporting personalized search based on user profiles requires a suitable component for ranking of resources with respect to user preferences. The process of finding best alternatives for a given user profile is called matchmaking. Such a component should offer application developers a ready-to-use tool allowing a fast implementation of the matchmaking logic, thereby reducing the cost of the overall application development. The key requirements for such a tool are:

- **domain-independent generic architecture**

  being able to handle various corporate resources and user profiles regardless of the underlying data schema (ontology T-Box)

- **flexibility**

  i.e. offer various matchmaking techniques for different kinds of object properties

- **extensibility**

  i.e. provide interfaces for implementation of new (domain specific) matchmaking techniques

- **traceability**

  i.e. deliver a detailed explanation of the matchmaking result together with the similarity ranking

Given these requirements, we designed a flexible Semantic Matchmaking Framework for calculating semantic similarity of multi-attributive and multi-dimensional information objects (as depicted in Figure 4.6) represented as arbitrary RDF graphs with concepts from an underlying corporate or domain ontology. In the corporate context, such information objects may represent, on the one hand, enterprise resources like products, services, employees, business partners, documents (including metadata), etc. On the other hand, they may represent user profiles. In general, the framework can be applied in a wide range of use case scenarios ranging from product/service recommender systems to expert finder systems.

Depending on the type and semantics of object attributes or dimensions the framework should deliver different kinds of matchmaking techniques, for example:

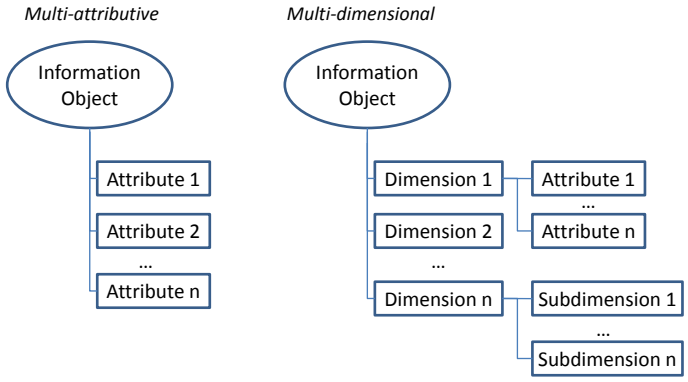

Figure 4.6: Multi-attributive and Multi-dimensional Information Objects

- **string-based**

  Calculating the similarity of two string values represented by RDF Literals. This includes comparing keywords, searching for keywords (and their synonyms) in texts, searching for Named Entities, or applying Natural Language Processing techniques.

- **numeric**

  Used to determine similarity of two numeric values. A good application example for this matching technique is the comparison of a product price some person is willing to pay (p_q) with the actual product price (p_r). For all $p_r > p_q$ the similarity shall decrease with increasing p_r. However, beyond a certain value (upper bound) where p_r would be unacceptably high the similarity shall equal 0.

- **taxonomic**

  Applied for matching attribute values represented by resources from a common taxonomy. In this case, the similarity between two concepts c_1 and c_2 can be determined based on the distance $d(c_1, c_2)$ between them, which reflects their respective position in the concept hierarchy. Consequently, the concept similarity is defined as: $sim(c_1, c_2) = 1 - d(c_1, c_2)$. For the calculation of the distance between concepts different distance functions may be applied.

  As example, consider the method presented in [14] where every concept in a taxonomy is assigned a milestone value. Since the distance between two given concepts in a hierarchy represents the path over the closest common parent (ccp), the distance is calculated as $d(c_1, c_2) = d_c(c_1, ccp) + d_c(c_2, ccp)$ where $d(c_n, ccp) = milestone(ccp) - milestone(c_n)$. The milestone values are calculated with an exponential function: $milestone(n) = \frac{0.5}{k^{l(n)}}$ where k is a factor greater than 1 indicating the rate at which milestone values decrease along the hierarchy. It can be assigned different values depending on the hierarchy depth.

  This formula implies two assumptions: (1) the semantic difference between upper level concepts is greater than between lower level concepts

(in other words: two general concepts are less similar than two specialized ones) and (2) that the distance between "brothers" is greater than the distance between "parent" and "child". As an example, we determine the

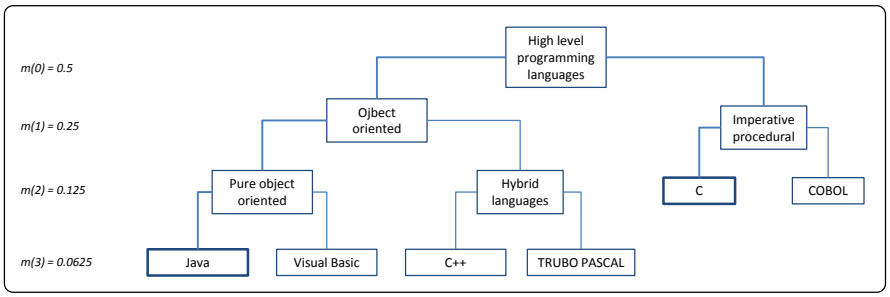

Figure 4.7: Example of a Skill Taxonomy

distance between two concepts: *Java* and *C*. Figure 4.7 shows a snippet of a simple taxonomy together with milestone values (with $k = 2$) for the corresponding hierarchy levels (in brackets). Since the closest common parent is *High level programming languages*, the distance between *Java* and *C* is calculated as follows:

$d(Java, C) = d(Java, High\ level\ programming\ languages) + d(C, High\ level\ programming\ languages) = (0.5 - 0.0.0625) + (0.5 - 0.125) = 0.8125$. Consequently, the similarity between these two concepts equals: $sim(Java, C) = 1 - 0.8125 = 0.1875$. This value is much smaller than in the case of the evidently more related concepts *Java* and *VisualBasic* for which $sim(Java, VisualBasic) = 0.875$.

- **rule-based**

  Which given a set of pre-defined rules determine the similarity between complex object dimensions. Consider, for example, an expert finder scenario in which, while searching for experienced Java programmers, only those candidates would receive a high ranking whose skill matches the concept *Java*, and additionally have already worked in projects for which *Java* skills were required.

- **(geo)location-based**

  For performing vicinity search given two locations (cities, street names, etc.) as strings or geo coordinates.

- **collaborative filtering**

  Taking into account not only a given user profile but also preferences of similar users, with respect to a particular attribute or dimension to be matched.

As depicted in Figure 4.8, the Matchmaking Framework plays a key role in realizing Web applications supporting personalized search in corporate data. In a given use case scenario, through a domain-specific Web interface , users provide their query and preferences which are represented in RDF using concepts

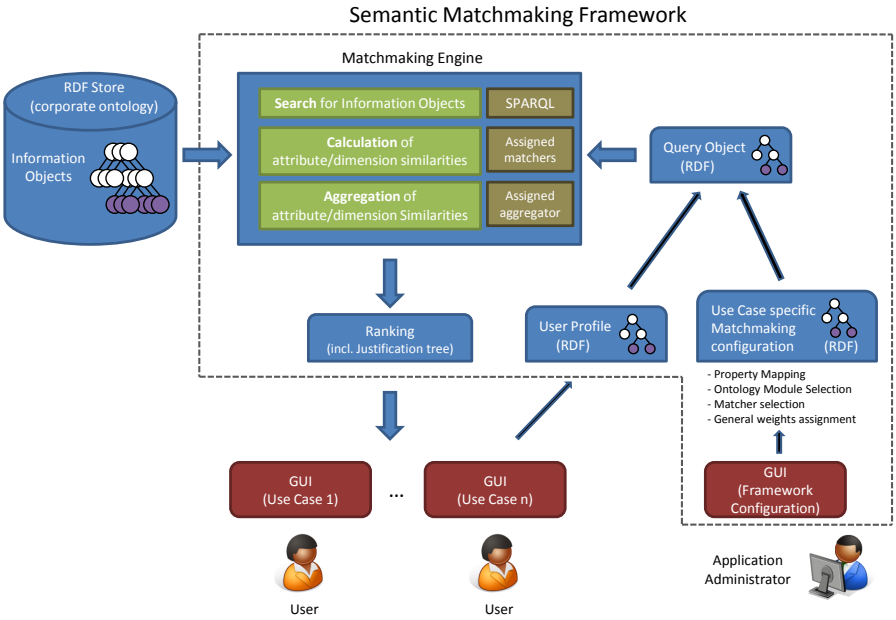

Figure 4.8: Architecture of the Semantic Matchmaking Framework

from an underlying corporate or domain ontology. As next, a user profile is merged with the use-case-specific matchmaking configuration delivered by the application administrator. It includes, among others, the selection and mapping of attributes/dimensions in user profiles with the semantically corresponding attributes/dimensions in corporate information objects to be matched, together with information about which matching techniques should be applied for computation of each attribute/dimension similarity. The aggregated RDF graph is then passed (as query object) to the Matchmaking Engine.

The process of matchmaking is carried out by the engine in three steps. First, the information objects to be matched, together with all relevant background knowledge (e.g. concept taxonomies), are retrieved from the RDF store. The access to RDF data is realized with *Jena - the Semantic Web Framework for Java* [3]. As next, for each information object, the engine computes the attribute/dimension similarities by invoking appropriate matchers implementing a certain matching technique specified by the application administrator. Finally, all attribute/dimension similarities are aggregated into an overall similarity score for a particular information object. The output of the engine is a ranking of information objects for a given user profile with additional information containing the explanation of the matchmaking process for each single object. The result is rendered in an appropriate format and presented to the user via the application-specific Web interface.

4.3 Conclusion and Outlook

In this chapter we presented our work on Corporate Semantic Search research pillar. Based on cooperation with the companies, we outlined use cases that give directions for our work. We described our concepts for realizing working packages: search in non-semantic data and search personalization. Due to the general description in [5], we introduced more detailed concepts for our research issues: Extreme Tagging, Preprocessing, Matchmaking Framework. At this time, our research is still work in progress and we are currently validating our conceptual ideas by developing first components.

Chapter 5

Conclusion and Outlook

In this document we describe the state of our work in the project "Corporate Semantic Web". While we presented the results of the requirement analysis in our first technical report, we conceptually designed an overall architecture of components that help to deploy semantic technologies in a corporate environment.

In the field of ontology engineering we investigated in two important aspects, namely ontology versioning and ontology modularization and integration based upon the Corporate Ontology Lifecycle Methodology (COLM).

Considering the usage cycle of an ontology, as described in COLM, we described operations of the editor which should be implemented to be useful in our scenarios. Furthermore, we distinguish between different user groups that influence the further development of ontologies. Last but not least, we presented the conceptual design of a light-weight ontology editor.

Finally, we presented the conceptual design for the work packages "Search Personalization" and "Search in Non-semantic Data". In the first field, we considered contextual information of a user to realize a personalized search. In the second one, we described on the one hand the extraction of "semantics" from corporate text collections, and on the other hand, we focussed on searching for complex relations in text collections.

Appendix A

Work Packages

Work package 1	**Search in non-semantic data**	02/08-01/11
WP 1 Task 1.3	Conceptual design of semantic search with knowledge extraction from non-semantic data	08/08-01/09
WP 1 Task 1.4	Prototypical implementation	02/09-01/10
Work package 2	**Search personalization**	02/08-01/11
WP 2 Task 2.3	Conceptual design of personalized semantic search based on user profiles	08/08-01/09
WP 2 Task 2.4	Prototypical implementation	02/09-01/10
Work package 5	**Knowledge extraction by mining user activities**	02/08-01/11
WP 5 Task 5.3	Conceptual design of a semantic collaborative tool for the acquisition of implicit knowledge about employees	08/08-01/09
WP 5 Task 5.4	Prototypical implementation	02/09-01/10
Work package 6	**Ontology- and knowledge modeling supported by collaborative tools**	02/08-01/11
WP 6 Task 6.3	Conceptual design of a collaborative tool for modeling corporate knowledge	08/08-01/09
WP 6 Task 6.4	Prototypical implementation	02/09-01/10
Work package 9	**Ontology modularization and integration**	02/08-01/11
WP 9 Task 9.3	Conceptual realization of ontology modularization and integration	08/08-01/09
WP 9 Task 9.4	Prototypical implementation	02/09-01/10
Work package 10	**Ontology versioning**	02/08-01/11
WP 10 Task 10.3	Conceptual realization of ontology versioning for a real-world use case scenario	08/08-01/09
WP 10 Task 10.4	Prototypical implementation.	02/09-01/10

Appendix B

Acknowledgment

This work has been partially supported by the "InnoProfile-Corporate Semantic Web" project funded by the German Federal Ministry of Education and Research (BMBF).

Bibliography

[1] Sören Auer, Christian Bizer, Jens Lehmann, Georgi Kobilarov, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 715–728, Berlin, Heidelberg, November 2007. Springer Verlag.

[2] Matteo Baldoni, Cristina Baroglio, and Nicola Henze. Personalization for the semantic web. In Norbert Eisinger and Jan Maluszynski, editors, *Reasoning Web*, volume 3564 of *Lecture Notes in Computer Science*, pages 173–212. Springer, 2005.

[3] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, New York, NY, USA, 2004. ACM.

[4] Sebastian Dietzold, Sebastian Hellmann, and Martin Peklo. Using javascript rdfa widgets for model/view separation inside read/write websites. In *Proceedings of the 4th Workshop on Scripting for the Semantic Web*, 2008.

[5] Markus Luczak-Rösch Radoslaw Oldakowski Ralph Schäfermeier Gökhan Coskun, Ralf Heese and Olga Streibel. Towards corporate semantic web: Requirements and use cases. Technical report, 2008.

[6] April Kontostathis, Leon Galitsky, William M. Pottenger, Soma Roy, and Daniel J. Phelps. *A Survey of Emerging Trend Detection in Textual Data Mining*. Springer-Verlag, 2003.

[7] Konstantinos Kotis and George A. Vouros. Human-centered ontology engineering: The hcome methodology. *Knowl. Inf. Syst.*, 10(1):109–131, 2006.

[8] Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. Mining of concurrent text and time series. In *In proceedings of the 6 th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, pages 37–44, 2000.

[9] Markus Luczak-Rösch and Ralf Heese. Managing ontology lifecycles in corporate settings. In *Proceedings of the International Conference on Semantic Systems (I-SEMANTICS)*, May 2008.

[10] Mariano Fernandez and Asuncion Gomez-Perez and Natalia Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, USA, March 1997.

[11] S. Pinto, C. Tempich, S. Staab, and Y. Sure. *Semantic Web and Peer-to-Peer*, chapter Distributed Engineering of Ontologies (DILIGENT), pages 301–320. Springer Verlag, 2006.

[12] York Sure and Rudi Studer. On-to-knowledge methodology — expanded version. On-To-Knowledge deliverable 17, Institute AIFB, University of Karlsruhe, 2002.

[13] Vlad Tanasescu and Olga Streibel. Extreme tagging: Emergent semantics through the tagging of tags. In Liming Chen, Philippe Cudré-Mauroux, Peter Haase, Andreas Hotho, and Ernie Ong, editors, *ESOE*, volume 292 of *CEUR Workshop Proceedings*, pages 84–94. CEUR-WS.org, 2007.

[14] Jiwei Zhong, Haiping Zhu, Jianming Li, and Yong Yu. Conceptual graph matching for semantic search. In *Proceedings of the 10th International Conference on Conceptual Structures (ICCS)*, pages 92–196, London, UK, 2002. Springer-Verlag.