Realizing the Corporate Semantic Web: Prototypical Implementations

Technical Report TR-B-10-05

Adrian Paschke, Gökhan Coskun, Dennis Hartrampf, Ralf Heese, Markus Luczak-Rösch, Mario Rothe, Radoslaw Oldakowski, Ralph Schäfermeier and Olga Streibel

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

28 February 2010

GEFÖRDERT VOM

Realizing the Corporate Semantic Web: Demonstrators

Adrian Paschke Gökhan Coskun Dennis Hartrampf
Ralf Heese Markus Luczak-Rösch Mario Rothe
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,hartramp,heese,luczak,mrothe,
oldakowski,schaef,streibel@inf.fu-berlin.de

28 February 2010

Abstract

In this technical report, we present prototypical implementations of innovative tools and methods developed according to the working plan outlined in Technical Report TR-B-09-05 [23].

We present an ontology modularization and integration framework and the SVoNt server, the server-side end of an SVN-based versioning system for ontologies in the Corporate Ontology Engineering pillar. For the Corporate Semantic Collaboration pillar, we present the prototypical implementation of a light-weight ontology editor for non-experts and an ontology based expert finder system. For the Corporate Semantic Search pillar, we present a prototype for algorithmic extraction of relations in folksonomies, a tool for trend detection using a semantic analyzer, a tool for automatic classification of web documents using Hidden Markov models, a personalized semantic recommender for multimedia content, and a semantic search assistant developed in co-operation with the Museumsportal Berlin.

The prototypes complete the next milestone on the path to an integral Corporate Semantic Web architecture based on the three pillars Corporate Ontology Engineering, Corporate Semantic Collaboration, and Corporate Semantic Search, as envisioned in [23].

Contents

Chapter 1

Introduction

The project Corporate Semantic Web (CSW) aims at establishing semantic technologies in enterprises.

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.

In this technical report, we present prototypical implementations of innovative tools and methods developed during the third phase from March 2009 to March 2010. The prototypes complete the next milestone on the path to an integral Corporate Semantic Web architecture based on the three pillars Corporate Ontology Engineering, Corporate Semantic Collaboration, and Corporate Semantic Search, as envisioned in [23].

In chapter 2 we present our ontology modularization and integration framework which helps improving efficient using and reusing of ontologies for the Corporate Ontology Engineering pillar. Then we introduce a server-side end of an SVN-based versioning system for ontologies called *SVoNt*.

In chapter 3 we present the results achieved in the Corporate Semantic Collaboration pillar which comprise a light-weight ontology editor for modeling non-experts, a prototype implementing a novel ontology aided method for finding experts in corporate wiki systems, and a wiki system enhanced by semi-automatic and automatic link recommendation aid, both for authors and readers.

Chapter 4 covers the results in the Corporate Semantic Search pillar. We present a prototype for algorithmic extraction of relations in folksonomies, a tool for trend detection using a semantic analyzer, a tool for automatic classification of web documents using Hidden Markov models, a personalized semantic

recommender for multimedia content, and a semantic search assistant developed in co-operation with the Museumsportal Berlin.

Chapter 2

Corporate Ontology Engineering

Bringing ontologies into a running enterprise environment requires an ontology management framework which allows for a seamless introduction into the ongoing business processes as well as the existing IT infrastructure with its running enterprise applications. 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.

The main principle we followed is to keep as many established tools and methodologies as possible to allow a familiar environment. We implemented new functionalities necessary to manage ontologies on the basis of well known systems from software engineering. The Ontology Management and Integration framework presented in Section 2.1 is based on Eclipse and the versioning framework described in Section 2.2 is an extension of the Subversion versioning system.

2.1 Modularization and Integration Framework

Component-based development of large and complex software systems by small well defined building blocks improves the comprehension 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, the state of the art in ontology engineering is the usage of monolithic ontologies. For that reason the number as well as the size of available ontologies has increased with the growing utilization during the last years. In order to improve the efficient usage (e.g. through scoped reasoning for reasoners), to simplify the maintenance (e.g. through refactoring support) and to allow reusable components (e.g. through increased human understandability) there is a need to modularize large ontologies into well-sized building blocks in a (semi-) automatic way. Especially from the viewpoint of the Semantic Web reusability is a crucial issue because an agreed common semantic model allows easy data integration and interoperability.

Considering ontologies as networks of concepts connected through properties, network analysis techniques are a promising approach to analyze and modular-

ize ontologies. As a very well established discipline in science there are a lot of sophisticated methods and tools for network analysis available. We believe that these methods can be modified and applied to ontologies, so that the ontology structure can be used to analyze the content and to identify regions, which can be seen as network "communities" and can be extracted as modules. Furthermore, we are convinced that structure analysis enables a first evaluation of the usability by allowing different views, so that existing ontologies can be easier comprehended by ontology engineers. This is very important because refactoring and reusing of existing models assume that these models are understood.

2.1.1 Efficient Using and Reusing Ontologies

During the last two decades the interest in using ontologies has increased. According to the last few years this trend was mainly driven by the vision of the semantic web. Defined as a problem-relevant, explicit and formal specification of a shared conceptualization, the importance of ontologies lays in the deep problem and domain analysis to create them. Because a good analysis clarifies the structure of the domain knowledge [12]. But a good analysis as only one part of the overall ontology creation process is a very cumbersome and time-consuming activity. In order to provide some structural guidance for the ontology creation process some ontology engineering methodologies have been proposed (e.g. Cyc Method, Uschold and Kings, Gruüninger and Fox, KACTUS approach, Methontology[17], SENSUS-based Method, On-To-Knowledge, and NeOn).The newest trend in ontology engineering is to build ontologies with a community in a collaborative manner (e.g. Joshi[19], DILIGENT[25], Dogma, HCOME, RapidOWL, Ontology Maturing). In most methodologies ontology reuse is recommended, because it is expected to reduce engineering costs by avoiding re-building already existing conceptual models. Apart form reducing costs, reusing existing ontologies increases interoperability from the viewpoint of the Semantic Web, where ontologies are considered as shared knowledge [15, 7].

Even though most of these approaches mention the reuse of existing ontologies as possible starting point, none of them describe in detail how to discover and analyze candidate ontologies. This is very important, because reusing ontologies presumes availability of already existing ontologies and discovery of potential candidates for the particular use case. Even though RDF and OWL files are based upon the XML syntax, which is declared to be human readable, it takes some time to comprehend the content and the main structure and to understand the main idea and purpose of the model. In case of ontologies with hundreds and thousands of concepts it is nearly impossible for the human mind to overview the whole model. But this is essential to decide if a candidate ontology is really usefull and whether it needs some customization, which in turn presumes the ability to comprehend the existing.

For that reason we propose a framework which supports the user by analyzing the content of ontologies and allows to semi-automatically extract needed portions as modules.

2.1.2 Architecture

As a very well known integrated development environment Eclipse allows to implement functional extensions through plugins. In this regard we have iden-

tified functional components which can be implemented as Eclipse plugins so an Ontology Modularization and Integration framework can be realized. Figure 2.1 illustrates the architecture of this framework.

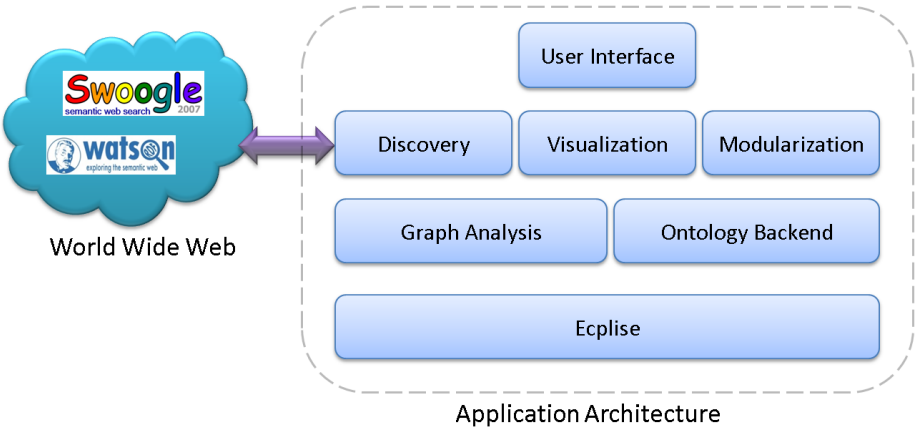

Figure 2.1: Architecture of the Modularization and Integration Framework

There are two basic components, the Ontology Backend and the Graph Analysis component. The Ontology Backend provides basic functionality to load and parse RDF and OWL documents. It allows to access different parts within the document, e.g. subject, predicate and objects of RDF statements. The Graph Analysis components provides means to anaylse the graph structure. Different algorithms for various graph metrics are part of this functional components.

The Discovery component allows to find existing ontologies. It is connected to different online ontology search engines and libraries. Basic terms of the target domain can be used to search for candidate ontologies for reuse. If the connected search engines provides meta information about the found ontologies, they are also downloaded by the Discovery component. These information are very helpful for the first decision, if a deeper analysis of the ontology is necessary. If an ontology is chosen it is downloaded and loaded through the Ontology Backend.

The Visualization component represents the chosen ontology as a graph, so its content and structure is easier to comprehend by the user. It accesses the ontology through the Ontology Backend and obtains its network structure. Through the Graph Analysis component, the Visualization component is able to get structure-based metrics about the graph. This information can be used, to optimize the visualization. E.g. centralitiy values about nodes allow to find nodes which are important according to the structure. These nodes can be highlited with the Visualization component.

The Modularization component is able to access the ontology through the Ontology Backend and to obtain information about the network structure of the ontology through the Graph Analysis component. Based upon these information the Modularization component is able to propose ways how to decompose the ontology into modules. With the Visualiaztion components this proposition can be represented visually.

The User Interface allows the user to interact with the under-laying components. Together with the visualization component it allows to navigate through the ontology and to choose different network visualization layouts and perspectives on the ontology, which highlight different aspects of the graph, e.g. the class hierarchy. It is possible to zoom into different parts of the ontology to better understand relations between concepts.

2.1.3 Realizing the System

Based upon the architecture from Figure 2.1, we made the decision to reuse the SONIVIS:Tool[1] to realize the targeted system. The SONIVIS:Tool is a network analysis software which is based upon Eclipse and allows easy extension through the Eclipse Plugin system. It provides already Graph Analysis and Visualization features and makes use of the Eclipse User Interface. Figure 2.2 represents the architecture by highlighting the SONIVIS:Tool components with blue boxes, the Eclipse platform as a green box and our extensions as orange boxes.

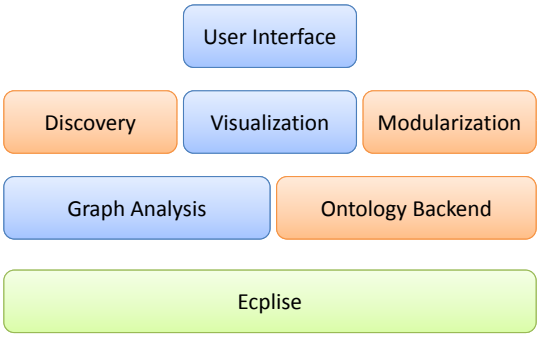

Figure 2.2: SONIVIS:Tool based Architecture

The SONIVIS:Tool is designed in a modular way and makes use of the R Project [2] for graph analysis with its implementations for calculation of the different metrics and uses Prefuse[3] for the visulization of the network for exploratory analysis. In order to realize the architecture we implemented the Discovery, Modularization and Ontology Backend as a plugin. It extends the SONIVIS:Tool and provides the necessary functionalities for the Ontology Modularization and Integration Framework.

2.1.4 The Tool

Figure 2.3 represents the Ontology Discovery Wizard which allows to search for ontologies. It represents the search result as a list and provides additional meta information obtained from the search engines. It is possible to choose to which search engine the key words should be sent. At this moment Watson and Swoogle are supported. But additional search engines can be added to this list.

[1] http://www.sonivis.org
[2] http://www.r-project.org/
[3] http://www.prefuse.org/

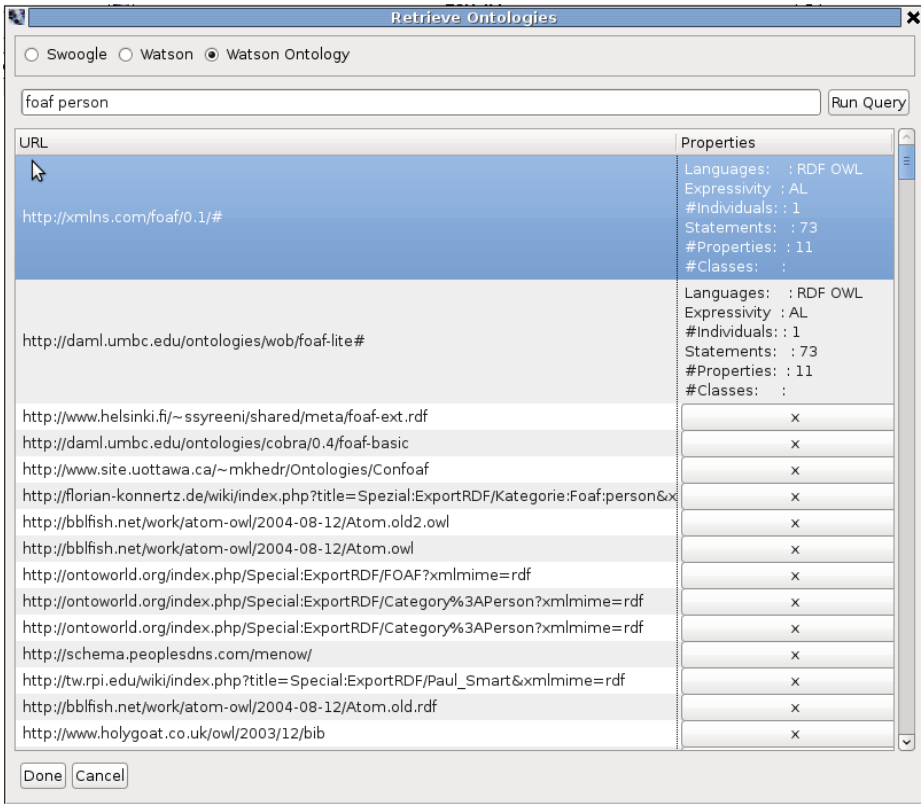

Figure 2.3: Ontology Discovery Interface

If an ontology in the list seems to be useful, it can be chosen for a deeper analysis. It will then be downloaded and represented as a network as shown in Figure 2.4.

There are two different ways to highlight concepts in the graph. The first one is to modify the size of the node, and the second one is to use different colors. Visualizing central concepts with bigger boxes enables the human eye to detect those central components easier. On the other hand, coloring similar nodes with the same color, simplifies the observer to detect those groups. Applying these two different highlighting methods based on the values of the graph properties of the nodes simplifies the exploratory analysis of the ontology. The class hierarchy, the concepts and their position within the whole graph can be understood faster. By reducing the cost to decide whether a candidate ontology is suitable for the targeted domain or not an efficient way to analyse an existing ontology lowers obstacle of ontology reuse.

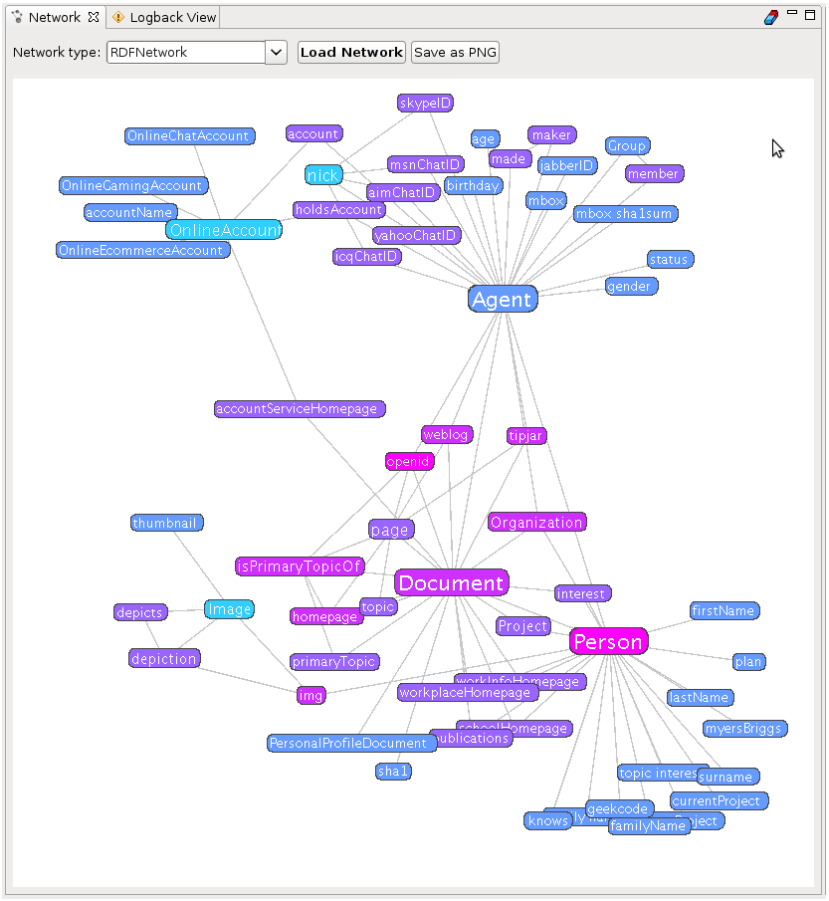

Figure 2.4: Visualization of the FOAF Ontology

2.2 SVoNt - An SVN-based versioning approach for ontologies

The SVoNt system is a versioning system for the collaborative development of ontologies based on the Subversion system. Formal knowledge is driven by a process, because it's being changed by the evolution of mankind corresponding to naturally evolutionary processes. This change of the knowledge base also concerns ontologies which are therefore not to be considered static. Examples for this can be found in medicine, where the knowledge base is altered steadily by new diseases, rapidly mutating viruses and changing healing methods. For this reason maintenance of ontologies is fundamental for the realization of schemes of a knowledge base.

Up to now user friendly approaches which take account of the evolutionary process of ontology engineering were missing. The aim of SVoNt is to provide a lightweight tool for versioning of ontologies to ease the development of evolutionary ontologies and in this way lower the inhibition threshold for using semantic technologies. Subversion is used as basis and is extended and adapted

for the use with ontologies. Thus it is guaranteed that you have recourse to existing knowledge bases of a user and also non-experts can participate in the development of ontologies.

2.2.1 Ontology Versioning Fundamentals

Klein and Fensel[1] define the versioning of ontologies as "Possibility to handle changing of ontologies by creating and maintaining different versions". Thereto it is necessary to monitor and save changes between different versions. They see the reason for changing an ontology in the changing of the domain, the changing of the common conceptualization, and the changing of the specification. For SVoNt a concept-based versioning is used to track changes of the concept level of an ontology. The lifecycle of a concept begins with its creation. In the course of the evolution of the ontology changes occur time and again, which concern this concept. These can have different reasons like modifications of annotations, changes of the taxonomy, or renaming of the concept. In the end of the lifecycle the concept is deleted, whereas its version history is preserved.

Ontologies consist of a set of elementary statements which are called axioms or triples. The structural difference between two ontologies is a set-theoretic symmetric difference of the axioms. As opposed to the generation of a structural difference the logic of a statement is considered for a semantic diff to determine changes of the semantic. Therefore the complexity of a semantic diff exceeds the complexity of a structural diff. According to Völkel and Groza [31] a semantic diff is independent of the power of the used ontology language. Therefore there is no universal algorithm to calculate the semantic diff of an ontology that is independent of the language. For the versioning of ontologies it must be possible to merge two versions of an ontology, which evolved in different development branches. This happens among others for every update command of a versioning system where changes to a local copy must be merged with a revision committed by another developer. The easiest way of merging two ontologies is the structural merge. This means that the set of triples of both ontologies are merged by a set-theoretical union. In doing so structural and semantic conflicts may arise which must be detected and treated.

2.2.2 SVoNt Basics

The SVoNt system consists of an extendable Subversion server and a special SVoNt client. The extension of the server uses functionality like logging, authentification and versioning features of Subversion as far as possible. This way it can be addressed by classic SVN clients, whereas the ontology specific functionality is not available. These can only be used by a modified client. Thereby the SVoNt server can be integrated into an environment in which Subversion is already used for versioning of ontologies without much effort. In the SVoNt system the Web Ontology Language (OWL) is used as basis for versioning. Each SVN repository represents the evolution of an ontology. This ontology is located in a specific OWL file in the repository.

Regular versioning of Subversion is very unsuitable for the use of versioning ontologies because changes of an ontology can be tracked only partly by file based versioning. An evolving history only leads to an overview of the changes

of the whole ontology but does not express changes of the structure or the semantic of an ontology. For this reason an additional versioning mechanism is used, which versions the ontology file with the Subversion system and uses a separate system for concept-based versioning of the ontology. For that conceptual changes of the ontology must be created on the SVoNt server, because Subversion only submits changes of the file lines to the server. These semantic changes are generated on the server by performing a diff between the updated ontology and the base ontology. This way of generating a difference is not trivial and only works if the ontologies are syntactical and semantical consistent. Therefore the updated ontology must be checked for that. These changes are deposited persistently in the conceptual change log and are provided via an additional external interface. An SVoNt client can use this specific information to perform version specific tasks on the ontology level.

2.2.3 SVoNt System Architecture

Because Subversion uses a client-server-architecture, SVoNt does as well. The overall system consists of an extended SVN server and a special SVoNt client. The server stays connectable with a regular SVN client, because the extended SVN interface of the SVoNt system is fully compatible with Subversion.

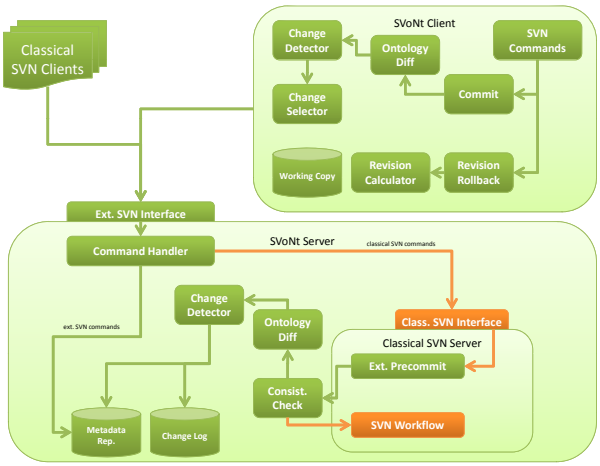

Figure 2.5: Design of the SVoNt System Architecture

Let's have a look at the server of the SVoNt system, which is depicted in Figure 2.5. It is an extension to the SVN server, offering additional functionality for versioning ontologies. E.g. there is a check done through a precommit-hook to see, if an ontology was changed and needs to be handled in the special ontology versioning. This process follows a specific procedure running through the following modules:

1. Consistency Check: Checking the updated ontology for its syntactic validity and semantic consistency

2. Change Detector/Ontology Diff: Detecting the changes done to the ontology by calculating the difference between the updated ontology and the current version in the repository

3. Change Log/Metadata Repository: Writing the changes to the change log to create a concept-based history of the ontology

To allow access to the additional versioning information generated by this process, the interface of the server was extended. That way an external access apart the classic SVN interface is granted.

There are two kinds of clients in the SVoNt system. The classics like TortoiseSVN, RapidSVN and Subclipse stay untouched and can use the regular SVN information of the SVoNt server only. The extended client on the other hand is a specialized software tailored for versioning ontologies. In addition to the classic SVN revision information it uses ontology-based concept changes from the change log of the SVoNt server. This information is used to display versioning specific metadata in the concept view. A rollback module checks, if a particular revision of concepts can be undone without generating incompatibilities.

Furthermore the client keeps a local change log. Changes done since the last update of the ontology are stored in it. These, in common with the SVoNt server, are detected by a local Change-Detection module. The Change-Selector module allows for the user to choose specific changes to either undo or commit them to the versioning server.

2.2.4 SVoNt Commit Workflow

The server-sided extension of Subversion consists of an ontology specific processing of the commit of an ontology, which runs through the three modules sequentially as it is depicted in Figure 2.6. On a commit to the SVoNt server a precommit hook is executed on the server to check, if the commit contains an ontology. In such a case the versioning process is triggered. For that the ontology is passed on to an OWL validator, which scans the ontology for specific properties. If that is not the case the execution of the commit is cancelled and the client receives a specific error. In a positive case the Change-Detection module is activated, which generates the difference between the basis ontology and the updated ontology. If no changes between the ontologies are detected, the precommit is succeeded and the ontology is being versioned in Subversion. This is possible if textual comments in the ontology or the order of concepts are changed, leaving the semantic of the ontology untouched. In the case of detecting changes with the ontology diff, these changes are passed to the Change log module, which writes them to the change log of that revision. With that the ontology specific processing is completed and the ontology can be added to the SVN repository and being versioned.

2.2.5 Prototype Implementation of the SVoNt Server

The implementation of the server is largely done in Java. A reason for that is among others the widely spread libraries being available for the processing of semantic technologies while being matured to a certain degree. Thereby many existing implementations can be built on, keeping the degree of in-house development low. As ontology backend the OWL-API was used. Figure 2.7 shows

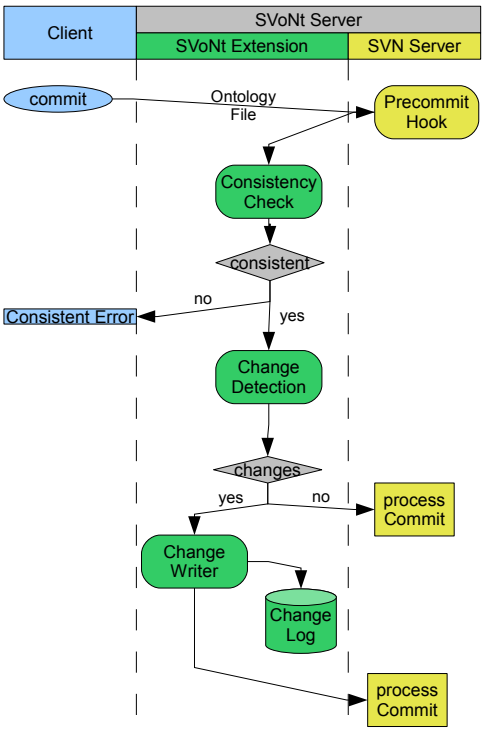

Figure 2.6: Workflow of the SVoNt Commit Operation

the rough structure of the server implementation. The SVN and precommit side are coloured light blue, the Java parts of the server green. The connection to the SVN server is done via precommit hook scripts provided by Subversion, what enables to interfere with the commit process. The SVoNtRunner, whose `run()` method triggers the ontology specific versioning process, is the heart of the implementation. For the standard implementation of the validation module Pellet with the OWL-API is used. Pellet is an OWL reasoner, which historically supports the OWL-DL specification and to large parts the OWL2 specification. Thereby the syntactic and semantic validity of the ontology is checked.

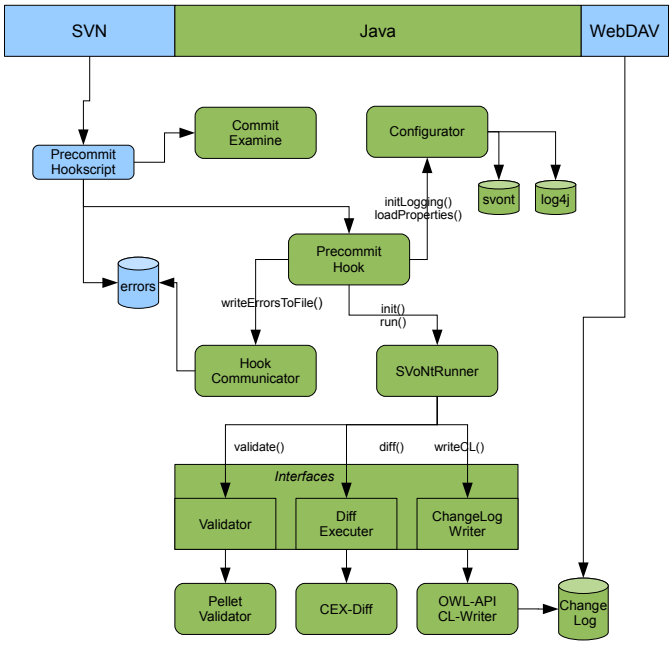

Figure 2.7: Implementation of the SVoNt Server

The Change-Detection (Diff) module implements algorithms of the OWLdiff library, which can be used in adapted form.

1. Basic Ontology Comparison Algorithm
2. CEX Logical Diff

The first algorithm is a native implementation of an axiomatic difference generation, which detects the structural changes between the ontologies and returns those axioms that were added or removed. With an entailment checking with Pellet it is possible to determine the semantic equivalence of both ontologies. The OWLdiff implementation of the CEX Logical Diff Algorithm is based on the work of Boris Konev, Dirk Walther and Frank Wolter [21] and is able to detect pervasive changes in the semantic of both ontologies. This implementation is used to identify concepts as being changed in marking all concepts as

semantically changed, which are returned by CEX-Diff. The semantic changes of the ontology being recognized are brought into a persistent state by generating a file in the Ontology Metadata Vocabulary (OMV) format for each commit, which represents a revision.

Chapter 3

Corporate Semantic Collaboration

In this section we give details about three demonstrators being developed in the work package "Corporate Semantic Collaboration". The first one considers the collaborative development of ontologies in the form of a light-weight ontology editor as it is outlined in [23]. The other two demonstrators are located in the area of wikis: The first illustrates how the edit history of a wiki site can be exploited to identify domain experts. The other, supports authors in linking their content to other sites of the wiki by recommending link targets.

The ontology editor is described in Section 3.1 while the wiki demonstrators are described in Section 3.2 and Section 3.3, respectively.

3.1 Light-weight Ontology Editor

The development of the light-weight ontology editor is the following scenario occurring in a company: A company wants to improve some functionality of an application by using semantic technologies. Typically, such a scenario includes the development of an ontology modeling the application domain. Since the development of an ontology is a complex process – comparable to processes in software engineering – a first version of the ontology has to be constructed by ontology engineering experts in collaboration with domain experts. A domain ontology is generally not static but has to be adapted to changes in the application domain, e.g., an ontology on chemistry has to be modified periodically due to new research results. In most cases, the changes at a time will be rather small, e.g., rename a concept or add a new concept, and, thus, the costs of a hiring an ontology engineer will be out of all proportion. As a consequence, an employee of the company should be responsible for keeping the ontology up-to-date. However, in the absence of appropriate tools for non-experts it is currently a challenge for the employee to handle this task. With the light-weight ontology editor we address the need for a simple and easy-to-use ontology editor which can be operated by non-experts[1].

[1]We refer to non-experts as people who are domain experts but do not have knowledge in semantic technologies

In [23] we give an overview of operations on ontologies. Developing the ontology editor we decided to focus primarily on basic operations such as searching, adding, removing, and editing concepts and relationships. Additionally, we planned to provide tagging and discussing mechanisms to users.

In the following we first give a short introduction to the domain model on which the editor is based. Afterwards, we describe the architecture and their functionalities. The last two sections cover requirements on the management of RDF data and the user interface.

3.1.1 Domain Model

Figure 3.1 shows the key entities of the domain model of the ontology editor and the relationships between them. For the orange entities, e.g., *User* and *Tag*, we omitted the relationships because these are related to almost any other entity. In the following we shortly explain each entity of the domain model.

A *repository* contains a collection of projects which describe the model of an application domain; we refer to this model as the ontology of that domain. There may exist multiple repository in an RDF store, i.e., multiple ontologies. A *project*, in turn, contains concepts and relationships of a part of the application domain. Thus, we refer to it as a sub ontology or an ontology module. To facilitate the reuse of (sub) ontologies a project can import another project and make use of its concepts and relationships. Within a project, *concepts* and *relationships* are the smallest entities of an ontology and are identically defined as in the Semantic Web.

The afore mentioned entities form the basics for modeling an ontology. Moreover, we implement concepts of *design patterns*, *discussions*, and *tags*. A design pattern is a template for modeling a certain aspect of an ontology, e.g., something has an location. Thus, it can be interpreted as a small reference ontology. Design pattern are modeled using a controlled vocabulary and are managed also as RDF in a project. In the context of the ontology editor, a discussion is an exchange of opinions about the modeling of projects, concepts, relationships, or design patterns. They may be useful to decide on the representation of a part of the application domain and later document the rationale of the modeling. Finally, we included tags into the domain model as simple and easy means for describing the contents of repositories and projects as well as the meaning of concepts, relationship, and design pattern with arbitrary terms (e.g., not necessarily originating from some controlled vocabulary).

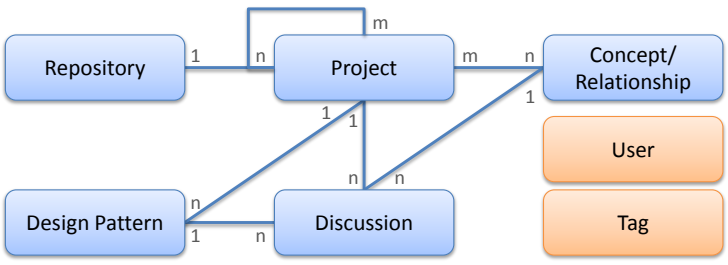

Figure 3.1: Domain model of the editor

Last but not least, the domain model contains the entity *user* which is a person who is allowed to access the ontology editor and (s)he may execute certain operations on repositories, projects, etc. depending on the granted access rights (e.g., an employee of a company).

3.1.2 Architecture

Having introduced the domain model we now give an overview of the components involved in the currently developed ontology editor (cf. Figure 3.2). In the figure we omitted standard components such as user administration and authentication.

At the lowest level of the architecture an *RDF store* is responsible for storing and querying the data produced by the ontology editor. Accessing functionality of the component *repository and project manager* a person with some knowledge in semantic technologies, e.g., an ontology engineer, can setup the ontology editor for the use by domain experts. For example, he has to create and configure a repository containing the projects belonging to an ontology and import the initial ontologies and design patterns. The component *design pattern manager* supports the administrator of the ontology editor to create, delete, or modify design patterns. Moreover, it generates input forms for inserting data by ordinary users based on a design pattern. The *discussion manager* handles user discussions about concepts and relationships of the ontologies, keeps track of arguments, and supports the process of decision-making. The discussion process will be structured according to a well-known methodology, e.g., DILIGENT [26] supports controlled discussions besides others. Finally, the component *ontology support* helps the user to edit an ontology appropriately, e.g., avoiding inconsistencies and redundancy or placing a new concept in the ontology. For example, inconsistencies may occur if a user deletes some concept of a project which is needed by other ones and redundancy if he creates a concept that already exists in the project with a similar label (e.g., "alcohol" vs. its German translation "Alkohol").

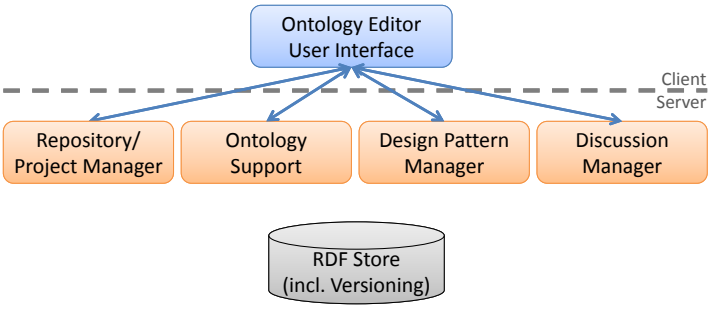

Figure 3.2: Components of the editor

3.1.3 Managing the RDF Data

In our implementation, we use Sesamé[2] as RDF store. We chose Sesamé, because it is open source software and offers a good performance, even for larger datasets. Moreover, it supports named graphs and the management of multiple RDF repositories. Both are very useful for managing several ontologies easily within a single Web application. Another possibility was to use a SPARQL endpoint at this point, but we did not, because the update language of SPARQL is still in the process of standardization and is not supported by Sesamé at the moment.

Within Sesamé, a project is realized as a named graph in a repository. Thus, any project is identified by an URI and it is possible to make statements about a project itself, e.g., to handle access rights on projects. Since any project exactly belongs to a repository, we decided that all project URIs share the same prefix and, thus, can be easily queried. Concepts and relationships, in turn, reside in a project. All newly created concepts and relationships within a project also share the same, project specific URI prefix which is generally different to the one of the corresponding project. At the moment the local part of a concept is randomly generated.

3.1.4 User Interface

The design of the user interface has a great influence on the usability of the ontology editor. Thus, we tried to transferred principles of interaction design proposed by Bruce Tognazzini [30] to the ontology editor, namely anticipation, autonomy, efficiency for the user, consistency, and track state. In the following we shortly describe how we applied them.

Anticipation means that the application attempt to anticipate the user's wants and needs. Thus, the editor hides the complex and technical issues of semantic technologies from the user. For example, the user will not see URIs if it is not necessary (e.g., for administrating repositories and projects). Moreover, the component ontology support instead of the user decides if a concept is a class or an instance, because non-experts will not be aware of the differences. In contrast, the system aims at presenting all information on a site that a user needs to perform a task, e.g., the complete hierarchy of concepts is not shown if not needed.

The next principle, *autonomy*, has the goal to keep the user informed by presenting up-to-date status information in their visual field. In the context of the ontology editor it means that the user gets immediately feedback about the result of performing an operation, e.g., in form of flash messages.

Efficiency of the user aims at improving the productivity of the user. For example, we try to reduce mouse movements by providing shortcut keys or placing the action elements (buttons and links) for the next click at the place of the previous one. We also eliminate any element of the application that is not helping the user. Last but not least, we keep messages short and precise.

In this context, we have to mention *consistency* of the user interface. For a user it is important that the application behaves in a way he is used to, e.g., shortcut keys maintain their meanings. Often it is referred to as look & feel. But it also means to be visually inconsistent when things act differently, e.g., to use different icons for different operations.

[2] `http://www.openrdf.org/`

Finally, we realize functionality that a user can resume his work on the ontologies quickly when he logins next time (*track state*). This feature is implemented as dashboards showing the recent changes on all projects which the user has access to. Moreover, we also maintain a dashboard for each project, e.g., showing recently changed concepts and relationships. To enable a user to continue his work his changes are highlighted respectively displayed separately.

Figure 3.3 gives an impression of the realized user interface. It depicts the website for editing the properties of a concept.

Figure 3.3: Screenshot of the website for editing a concept

3.2 ExpertFinder for Wikis

Wikis have emerged as important collaboration and knowledge management tools in large, locally distributed communities and corporate environments. However, not all knowledge can be documented. Especially tacit knowledge which enables individuals to solve complex problems, and which is the result of personal experience and training, is hard to make explicit. Moreover, a number of studies conducted among co-workers in medium-sized and large companies suggest that people often prefer talking to an expert rather than referring to a document when they need help with coping with a task [18, 16].

Besides the above mentioned reasons, social aspects play an important role. A person can, for instance, act as an intermediary to other important persons by directly introducing the two parties to each other or by revealing information about the other person that helps lowering the barrier to initiate communication [16].

In general, a person with expertise can deliver more practical knowledge than a document can do because persons with expertise can apply their general experience to broader classes of problems, while documents tend to remain focused on a rather tight problem context or completely lack such context [35]. Most importantly, a person with expertise can adjust her or his vocabulary to that of the inquirer who may only have basic knowledge (or even none at all) about the problem domain and the appropriate terminology, making person-to-person knowledge transfer more efficient than document-to-person knowledge transfer.

The following sections describe a prototypical system that identifies experts among wiki authors using formalized domain knowledge in the form of a light-

weight domain ontology. The system has been developed as part of a diploma thesis at the Corporate Semantic Web working group. It has been tested and evaluated using the project wiki of the Eclipse Foundation[3] and the Software Engineering Ontology (SEOntology) developed at the Digital Ecosystems and Business Intelligence Institute (DEBII), Curtin University of Technology, Perth, Australia [33].

3.2.1 Expert Model

The expert model developed in this work is based on the cognitive model of experts described by Bransford [8]. According to Bransford, experts store knowledge in a way that enables them to recognize basic structures and relevant dimensions of a problem, enabling them to find a more efficient approach to solve the problem than laymen. Likewise, experts are able to name concepts from the problem domain using domain specific terminology. The underlying assumption in this work is that wiki users who are experts in a specific topic write about this topic, either by using topic specific vocabulary, or by contributing content to topic specific wiki sections denominated by corresponding vocabulary.

Relevant feature extraction is backed by a domain ontology denoting terms in natural language using `rdfs:labels`[4]. While this yields a significant reduction of the features vector, domain concepts are hierarchically ordered by `owl:subclassof`[5] relations and otherwise semantically related by `rdf:Properties`[6], which in turn allows for augmenting the features vector with further relevant terms that are not explicitly used by the author. The relevance of additional terms is computed using ontology based similarity measures like node distance or self-information based measures until a defined threshold is reached (see figure 3.4), and each value in the features vector for a revision is weighted accordingly.

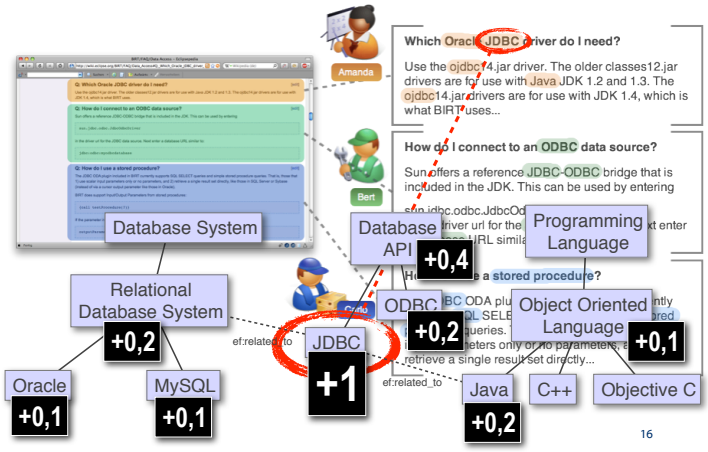

Figure 3.4: weighting of detected and related features

[3] `http://wiki.eclipse.org/Main_Page`
[4] `http://www.w3.org/TR/rdf-schema/#ch_label`
[5] `http://www.w3.org/TR/owl-ref/#subClassOf-def`
[6] `http://www.w3.org/TR/rdf-schema/#ch_property`

According to Ehrlich, reputation is another important factor when it comes to judging experts [14]. In this work, the reputation of an author with respect to a certain expertise topic is assessed using the quality control processes accomplished by peer wiki users. When authors contribute content to the wiki, their contributions become subject to a perpetual revision process by other wiki authors. This process is reflected in the wiki's revision history. The wiki principle encourages authors to change or even delete passages they object to. In turn, if a reviser considers a contribution to be relevant and correct, he can decide to keep or restore it if it has been deleted before. The contributions that survive over time can be considered public consensus. The expertise model developed in this work takes this revision process into account by considering a person the more as an expert the more revisions his or her contributions have survived, i. e. the more persons presumably have agreed with what the person has contributed. Algorithms calculating the overall reputation of wiki authors have been proposed by Chatterjee et al. [13] as well as by Adler and de Alfaro [2]. As an analysis of the Eclipse project wiki revealed, contributions that have survived five or more revision can be considered stable and will probably not be deleted later on. Thus, a trust index function is deployed which converges on one very quickly after the first revisions (see figure 3.5).

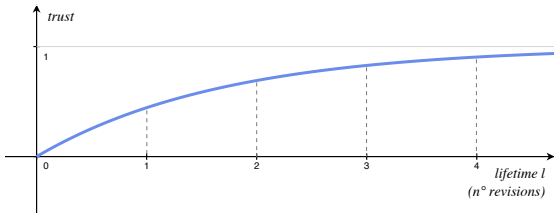

Figure 3.5: growth of trust in a user contribution over time

3.2.2 Architecture of the Prototype

The architecture of the prototypical expert finder system is depicted in figure 3.6.

The system connects to an external wiki system using a unified API that makes the system independent of a special wiki type. For each wiki page version, wiki markup must be identified and textual content must be extracted by the wiki parser. The authorship component relates contributions to their authors, and the concept matcher identifies relevant terms by referring to the external domain model and build the feature vectors accordingly.

3.2.3 Prototypical Implementation

The system was implemented using the Java language. The text analysis workflow was implemented using the Apache Unstructured Information Management (UIMA) framework [7]. Because the internal diff algorithm used by the Mediawiki

[7] `http://uima.apache.org/`

system is insufficient in accurately detecting authorship down to word level, a superior diff algorithm has been implemented.

3.3 Link Recommender

In companies wiki systems are increasingly used for knowledge management because they allow creating websites in the Internet and intranet fast and collaboratively. An important feature of wikis is that they allow authors to insert links to other pages to point to related pages or to indicate missing content. However, current wiki systems fail to support authors in creating these links, e.g., the author needs to know the title of the target page. As a result, the authors have to be familiar with the whole content of the wiki in order to create useful links to related pages. If they do not know the wiki then they are unable to provide useful links and the wiki is less beneficial as a knowledge management tool. Moreover, it may happen that authors create two pages on the same topic without knowing of each other.

We developed two extensions to existing wiki systems that offer authors seamless support for creating links between wiki pages. As a basis, we followed a fully automatic and a semi-automatic approach being explained in the following.

3.3.1 Automatic Approach

With the automatic approach, links are created on the fly, as a wiki page is rendered. The developed wiki extension uses an OWL ontology where each relevant domain concept is annotated with one or more corresponding `rdfs:labels`. Then, based on a similarity metric (see section 3.2.1), similar terms are identified. Then a link is inserted, leading to the wiki's search page, using all identified relevant terms as search terms (see figure 3.7).

3.3.2 Semi-Automatic Approach

The automatic approach has several downsides. For pages that contain many relevant terms wrt. the domain model, many links are created, distracting the reader from the important related pages. Furthermore, authors have no control over the creation of links. Thus, we developed a second semi-automatic approach. With this approach, relevant links are suggested to the author on demand. While an author edits a page, she can select a term or phrase and request link recommendations for that phrase (see figure 3.8).

The approach is based on a search index constructed of domain relevant terms that are used in page titles and section headings. Each term is extended by further relevant terms in the same way is described in section 3.3.1. When the author of a page request a link recommendation for a selected phrase, the index is searched for terms contained in the phrase, and links to the corresponding pages are recommended.

A demonstrator is available at `http://demo.corporate-semantic-web.de/xwiki/bin/view/Main/`.

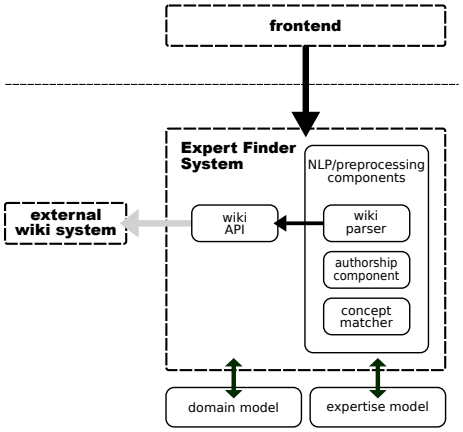

Figure 3.6: high level architecture of the expert finder system

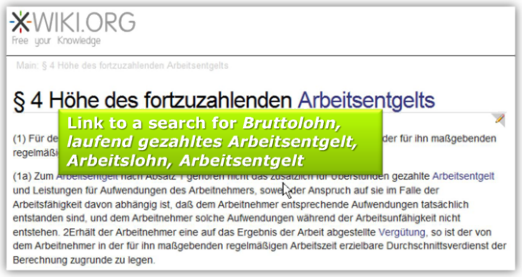

Figure 3.7: automatic link creation

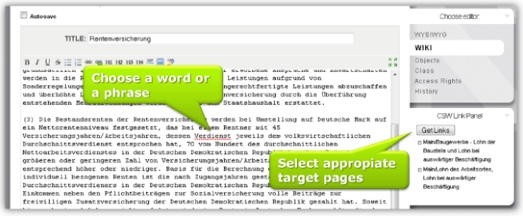

Figure 3.8: semi-automatic link recommendation while editing

Chapter 4

Corporate Semantic Search

Following sections give an overview over the implementation of concepts developed in the previous phases of the research pillar Corporate Semantic Search. In our research, we first explored the many possibilities of applying semantic search approaches to the corporate context theoretically and then concentrated on few of them in order to accomplish the proof-of-concept for our approaches. Most of the methods demonstrated below are the result of our work under the industrial cooperation. Many of them are work in progress and serve as bricks that will be put together in the next months of research and development in order to fullfill our concept of corporate semantic search.[1]

4.1 Algorithmic extraction of semantic relations out of folksonomies

Problems with "Semantics" based on Tags[28] There are two main problems that emerged in social tagging: **Ambiguity**- which means having more than one meaning for a tag- reduces the precision of a keyword based search in folksonomy tags. Therefore users searching for *atlas* retrieve relevant resources to *world atlas*, as well as results for *Atlas Mountains* in Africa. **Synonymy**- which means equal or similar meaning of tags- reduces the quantity of the resultset of a keyword based search in folksonomy tags. Users searching for *titan* should retrieve *atlas* as well, since due to mythology *Atlas* is an examplar of a *titan*. In addition to folksonomies, our concept of Extreme Tagging Systems (ETS) as an extension of common folksonomy allows to tag tags and to tag emerged relations between tags X,Y: 'X $\langle is - tagged - with \rangle$ Y'. Extreme tagging extends folksonomy graphs by adding semantics to tags. It enables the use of user's own concepts in description of the meaning, e.g. the use of their "subjective" synonyms. This is useful for generating personal ontologies from the ETS graph. However, these advantages bring obstacles in realizing ETS, which are: the high **user interaction** in the tagging process and the **user-specific language** used for tags and relation descriptions like users' synonyms. To address the problems described above, different methods for the algorithmic extraction of semantic relations have been developed. Many of these methods

[1] In the sections: 4.1, 4.2 we use excerpts from our publications. Names of contributors are given in footnotes and bibliography.

are restricted to only one type of semantic relation. From the users point of view both relation types have an equal strong influence on the retrieval experience. Also as these methods were developed for folksonomies they do not take into account the special characteristics of ETS. Hence, we focused on an integrative method that considers both types of relations and benefits from the ETS concept.

Algorithmic extraction of tag semantics (aETS) aETS is based on a four-stepped process. Using the data contained in a folksonomy, our method develops an ontology built upon semantic relations between folksonomy tags. In the **preparation phase** the Jaro-Winkler-algorithm is used on the set of tags to unify the spelling of the tag occurrences via gathering all different spelling forms of each tag. In the **disambiguation phase**, a bipartite graph is established for each tag that should be disambiguated (*dis-tag*). The bipartite graph contains *users:* u and *entities:* e as vertices. Tagging of an e with the *dis-tag* by u is represented by an edge in the graph. Applying one-mode projection to the bipartite graph results in a new graph where vertices are represented by entities e and edges between them are transformed from tagging edges of the bipartite graph. The transformation reduces the graph by omitting u (entities tagged with the dis-tag by the same user are connected with the edge). It allows for representing tagged entities as a network of vertices due to users who tagged these entities with the dis-tag. Applying the Girvan-Newman-algorithm for cluster determination to the transformed graph determines clusters in the graph; hence it determines possible meanings of the *dis-tag*. Entities that are included in one cluster are considered as possible synonyms. In the **synonym extraction** step the cosine similarity between each of the tags is used to determine synonyms. In the **generating ontology** step the extracted semantic relations are recorded as a Semantic Web ontology.

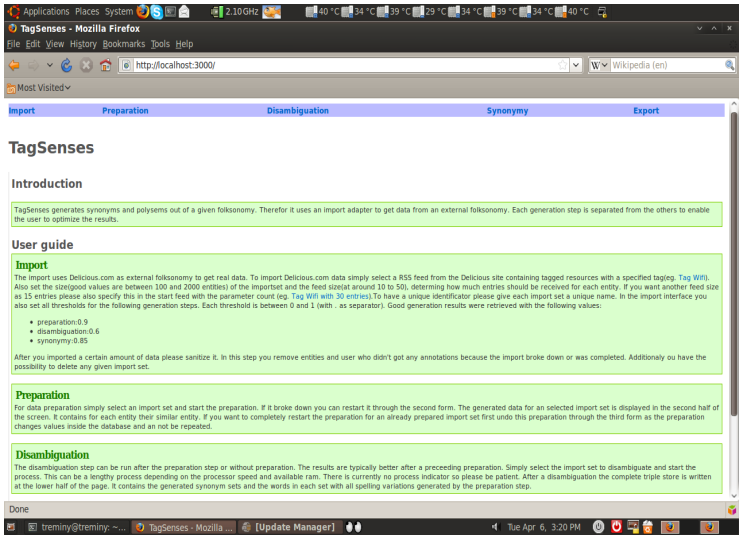

Figure 4.1: TagSenses application

TagSenses demonstrator The aETS method has been prototypically implemented in Ruby and tested relying on Delicious[2] user tags. The first evaluation of aETS shows very promising results of extracting semantic relations out of tags. In order to validate the quality of retrieved relations and the usability of the generated ontology, we additionally evaluated the aETS results with a test user group. The manual approach has shown that aETS is a very promising method for determining user-generated, language independent polysemes and synonyms. TagSenses generates synonyms and polysemes out of a given folksonomy. Therefor it uses an import adapter to get data from an external folksonomy. Each generation step is separated from the others to enable the user to optimize the results. The import uses Delicious.com as external folksonomy to get real data. To import Delicious.com data a user has to simply select an RSS feed from the Delicious site containing tagged resources with a specified tag(eg. Tag Wifi). Also setting the size (good values are between 100 and 2000 entities) of the importset and the feed size (at around 10 to 50), determines how much entries should be received for each entity. If you want a different feed size than 15 entries you have to specify it in the start feed with the parameter count (eg. Tag Wifi with 30 entries). To have a unique identificator the user has to give each import set a unique name. In the import interface one should also set all thresholds for the following generation steps. Each threshold is between 0 and 1 (with . as separator). Good generation results have been retrieved with the following values:

- preparation:0.9
- disambiguation:0.6
- synonymy:0.85

After importing a certain amount of data, it has to be sanitized. In this step entities are removed and user who didn't got any annotations (because the import broke down or was completed). Additionally users have the possibility to delete any given import set.
For data preparation simply an import set has to be selected and the preparation can start. The generated data for a selected import set is displayed in the second half of the screen. For each entity it contains its similar entities. If one wants to completely restart the preparation for an already prepared import set he first needs to undo this preparation through the third form as the preparation changes values inside the database and is not repeated.
The disambiguation step can be run after the preparation step or without preparation. The results are typically better after a preceeding preparation. The import set to be disambiguated can simply be selected and the process starts. This can be a long process depending on the processor speed and the available memory. There is currently no process indicator so one has to be patient. After a disambiguation the complete triple store is written on the lower half of the page. It contains the generated synonym sets and the words of each set with all spelling variations generated by the preparation step.
To start the synonym generation the import set can be selected and run. It has a very long runtime and there is no process bar given at the moment. It outputs the complete triplestore after the generation with the generated synsets and the

[2] `http://delicious.com/`

containing wordsenses for each synset.
The complete ontology inside the triple store can be exported as rdf/xml, n3 or as svg image. Additionally one can clear the complete triple store for another generation process. TagSenses has been developed under Mac and deployed on Ubuntu Jaunty Jackalope with kernel 2.6.28-15-generic using WEBrick 1.3.1 and ruby 1.8.7 (2008-08-11 patchlevel 72), Postgresql 1.8.4 database, Redland RDF Store (http://www.librdf.org) + Ruby Bindings. Screencast of the demostrator can be viewed on: `http://www.corporate-semantic-web.de/algorithmic-extraction-of-tag-semantics-40aets41.html`

4.2 Supporting knowledge based trend detection using Semantic Analyzer

Knowledge based trend detection[29] In order to realize the knowledge based trend mining, trend knowledge has to be identified and formalized. Referring to the Semantic Web ontology approach, we proposed the usage of trend ontology as knowledge base for the automatic trend mining. A trend, in terms of market research, is the evolution of a customer opinion referring to a specific topic that can be described by its categories or labels. Customer opinion is strictly conjoined with sentiments used by customers to express linguistically their emotional viewpoint on specific issues. In general, the automatic trend mining as for market research should allow the enhancement of process efficiency in the analysis of textual market research data that is generated in primary and secondary research described in the former section. This includes the automatic categorization and valuing process of open ended questions, the filtering of relevant information, and trends identification. A trend ontology should support the analysis process by providing knowledge regarding main market research concepts that occurs in texts of the market research project (e.g. what is *image* in terms of market research, what is *product quality*). This also includes:

- main keywords and terms used by customers in order to describe their opinion (substantives, verbs, adjectives, "This brand fits to me", "I like the nice logo"),
- customer opinion categories in terms of market research studies (overall satisfaction, level of commitment),
- categorization of customer opinions based on a given list of categories that are relevant for the respective project,
- knowledge about trend indicating features of any given keyword or term (positive, negative and neutral description keywords).

Considering the requirements, the trend ontology has to be defined as a knowledge model that contains: (i) the meta-level knowledge about market research concepts (commonly used in the market research), (ii) common keywords used in the market research projects (based on market research specific projects) and (iii) knowledge about trend indicating terms and relations in market research. Furthermore, the trend ontology should be used as a knowledge base that can be applied in different phases of the trend mining process: feature extraction, selection and learning stage of the trend mining process (cf. 4.2).

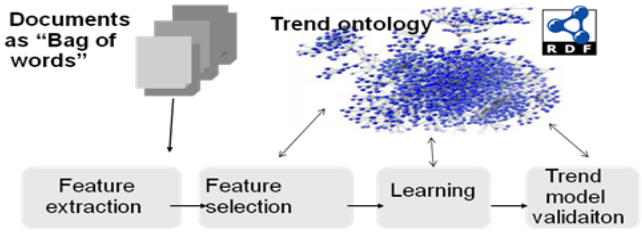

Figure 4.2: Trend ontology embedded in machine learning approach

4.2.1 Trend ontologies

Keyword/concept based trend ontology Relying on the experience of experts from the market research domain, we identified and modeled with Protégé using RDFS an initial keyword set categorized by the main concepts of the market research (our case considered only high tech market). The main set categories are: Image (image), Produktqualität (product quality), Kundenbeziehung (customer relation), Service (service), Stimmungsbild/Wahrnehmung/Entscheidung (public opinion/customer view/decision).
Each category is implemented as a class consisting of relevant concepts that describe the category. For the product quality category, the concept set consists of: e.g. Zuverlässigkeit (realiability), Performanz/Leistung (performance/power), etc. We defined the class property included-in, in order to express semantically the category membership of the given keyword/concept. In addition to the categorized concept sets, we modeled synonyms for several keywords/concepts and added the trend-indicating property to each concept that had been classified by experts as trend indicating. A keyword/concept based trend ontology is built on a very simple schema and can be easily applied, for instance, in order to extend the word based feature vector creation as for machine learning methods.

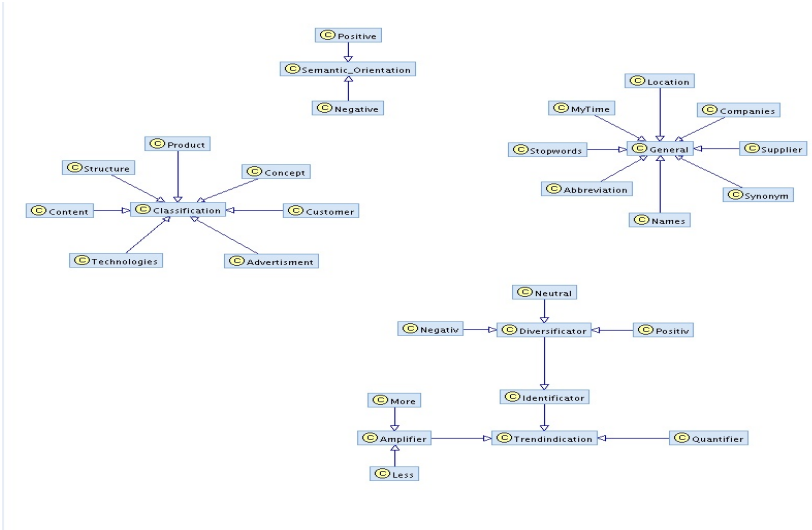

Figure 4.3: Concepts in meta ontology

Term field based trend ontology Extending the keyword-based trend ontology we observed the emergence of so-called term fields in market research, which correspond to the semantic fields from the Semantic Field Theory. Relying on the semantic field idea, the extension of concept definition by adding term fields to the concept seemed reasonable. However, defining which term belongs to the concept field and whether a given term is trend indicating or not is the more difficult the more terms are used for the term field definition; we searched for the exact definition of trend indicating features in the texts of market research. Applying statistical methods (e.g. term frequency in documents) supported by manual expertise, we identified adjectives that, according to experts, were significant for the description of customer opinion. The most relevant adjectives were: vertrauenswürdig (reliable), kompetent (competent), vielseitig (all-round), aktuell (up-to-date). Conducting the search for semantic fields of these adjectives and their relevance to the main concepts of the market research domain, we detected the appearance of so called satisfier, disatisfier and sensitive. We defined each main concept as a category with its semantic field and its own identifier that consists of diversificators. Identificators are adjectives belonging to the concept and describing its features, i.e. entertainment has entertainment identifier which is described by the adjectives: abwechslungsreich (varied), ansprechend (attractive), entspannend (relaxing), etc. A diversificator defines satisfier, disatisfier and sensitive which are adjectives grouped by the relevant meaning that refers to the positive, negative and neutral customer opinion about a given concept. Each identifier consists of a diversificator that refers to a more or less positive customer opinion. The customer (dis)satisfaction refers to a (negative)/positive trend indication. Trend ontology based on term fields adds the meta-level concepts identifier, diversificator, sensitive, satisfier and disatisfier to the keyword based trend ontology and extends concept sets in term fields.

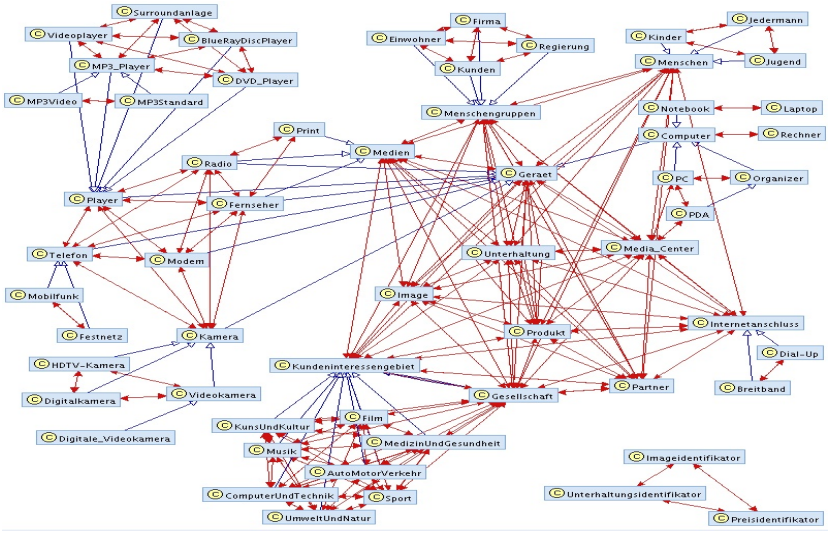

Figure 4.4: Examples of concepts in middle ontology for market research

(Temporal) invariant scheme based trend ontology The adjective groups used as satisfier, disatisfier and sensitive are important for the proper sentiment interpretation of a given set of texts. The sentiment interpretation helps for trend detection. However, the validity of diversificators often expires after some time. Assuming that adjectives used for describing customer satisfaction change with time, we searched for an invariant part of trend knowledge. The semi-automatic analysis of relevant market research news done by experts resulted in a structure that seemed to be valid for a long period of time and intuitively used by experts for analysis of market research texts. This (temporal) invariant scheme based trend ontology consists of three meta-level classes: general, quantification and classification. The class includes groups of the most important concepts like supplier and companies. Suppliers, which are important extraction features, are always used in market research projects (regarding our case study) in order to classify the relevance of the texts. The quantification part of our structure contains the idea of identifiers and diversificators, and it adds the amplifier1 as a new meta-concept. Classification consists of different categories that define the context for the quantifier. Its character is dynamic since it strongly depends on the context at a given point in time. The interesting subcategory of classification is the so-called structure that defines the basic structure for the context. We observed that this category particularly refers to the economic model of the given market. Even if we know that the trend-indicating keywords and concepts are changing in time, and that their positive or negative value differs and depends on the context, we assume that there is an invariant trend structure which contains the three main trend detection parts: general concepts, the trend value concepts, and the classification structure that models the context of the trend (cf. Fig 4.4,4.3).

TREMA Term Viewer

« Terme für Matrix auswählen | Hauptmenü

	Bilanz	Commerzbank	Banken	Kauf	Deutsche Bank	Gewinn
Bilanz	1.0	0.0	0.0	0.7225	0.0	0.5220063
Commerzbank	0.0	1.0	0.765	0.0	0.80999994	0.0
Banken	0.0	0.765	1.0	0.0	0.765	0.0
Kauf	0.7225	0.0	0.0	1.0	0.0	0.5220063
Deutsche Bank	0.0	0.80999994	0.765	0.0	1.0	0.0
Gewinn	0.5220063	0.0	0.0	0.5220063	0.0	1.0

« Terme für Matrix auswählen | Hauptmenü

Figure 4.5: Semantic Analyzer Service

Semantic Analyzer Demonstrator Our Semantic Analyzer component has been developed as a web service with an interface for a cluster method and alternatively for a feature extraction method. The main component of Semantic Analyzer are the trend ontologies with an ontology management module and with the semantic similarity function. For computing the semantic similarity we used the simple least common ancestor approach. Regarding the trend recognition process based on any statistical learning approach, Semantic Analyzer can

be used for enhancing the feature extraction step and for the simple cluster refinement. Using tomcat 6.0.20, Axis 1.4 and a very simple GUI based on ruby 1.8.6, thanks to its semantic proximity function Semantic Analyzer generates 2 WSDL files for trend ontologies: finance and mafo (cf. Fig 4.5).

4.3 Automatic classification of web document structure using Hidden Markov Models

Structure classification problems The work on Hidden Markov Models in document structure classification presents[3] a general approach for partitioning web page documents. In search engine environments, the outlined content based partitions are used to overcome problems encountered when extracting information from unstructured data sources. The challenging tasks reducing processing times and developing efficient algorithms for extracting data are crucial for processing engines. Partitioning web page content is giving the possibility to enhance the extraction process in considering only relevant page content for extraction purposes. In quality measuring the extracted data and generated models with cross-validation and precision recall method, we show the maturity of the presented approach in the information extraction domain. A demostrator for the HMM Demonstrator can be viewed on `http://www.csw.inf.fu-berlin.de/apps/hmm/Demo`

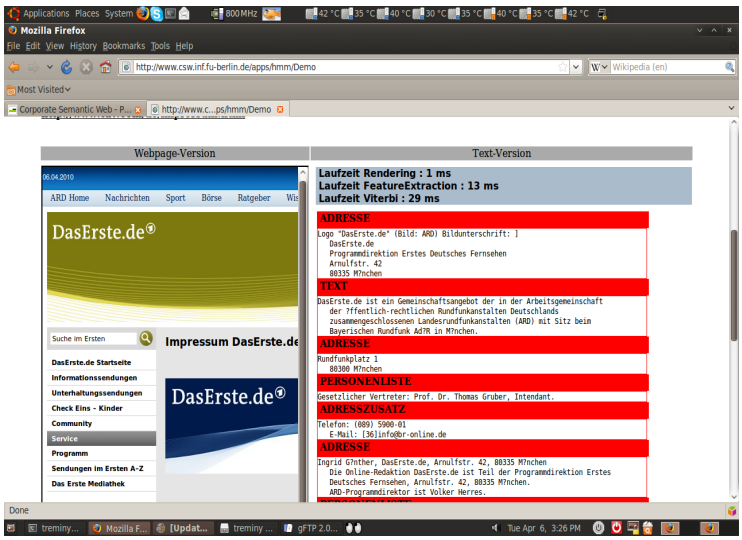

Figure 4.6: HMM demo

[3]Many thanks to Ivo Koehler and neofonie GmbH

4.4 Personalized Semantic Recommender 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) allows only a 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 to further enhance the information quality. Furthermore, the realization of this scenario requires a matchmaking component for ranking of the multimedia content with respect to user likes and dislikes. In the following section we briefly describe the prototypical implementation of a personalized semantic recommender for multimedia content.

4.4.1 Representing Domain Knowledge and User Preferences

The prerequisite for an intelligent recommender system is a proper formal representation of information it should operate on. In our case this includes the description of available resources (i.e. TV programs, films, etc.) as well as user profiles. We represent both of them in RDF [5] using concepts from a common TV ontology which we have developed in OWL [4]. The T-Box of our TV ontology is based on XMLTV[4] - an XML format for description of TV-programs with properties like *channel, title, language* etc. - additionally relying on the Time [22] and Wail[5] ontologies as well as the Escort2007[6] specification introducing further taxonomies for concepts like *content* and *intentedAudience*. The ontology was populated with a sample data set of an Electronic Program Guide covering a time span of two weeks obtained through TV Movie ClickFinder[7]. The sample data provided by the online service was available in the XMLTV format which we mapped onto our TV ontology. In order to further enhance the quality of the descriptions of films we integrated additional information from the Internet Movie Database[8] through a dedicated wrapper. The T-Box of the ontology holding information about users consists of concepts describing static and a dynamic user characteristics. The former represents personal information like *gender* and *date of birth* for which we utilized the FOAF [9] vocabulary, the latter captures user preferences (e.g. *category*, *favorite actors*, etc.) which are likely to change over time.

[4] `http://wiki.xmltv.org/`

[5] `http://www.eyrie.org/~zednenem/2002/wail/`

[6] ESCORT 2007 EBU System of Classification of Radio and Television Programmes. `http://tech.ebu.ch/docs/tech/tech3322.pdf`

[7] `http://www.tvmovie.de/ClickFinder.57.0.html`

[8] `http://www.imdb.com/`

4.4.2 Preference Matching

The task of ranking of multimedia resources with respect to user preferences belongs to the category of Multi Attributive Decision Making problems, which are characterized by [34]:

- limited number of alternatives
- limited number of decision criteria based on the values of attributes describing alternatives to be selected/ranked
- different matching methods for different kinds of attributes (e.g. depending on the datatype of attributes)
- attribute weights representing the importance of decision criteria, which are usually normalized to sum to 1,

The process of finding best alternatives for a given user profile is called matchmaking. In our scenario the process of matchmaking is carried out in three steps. First, each attribute of multimedia resources (e.g. *hasContent*) is mapped to the semantically corresponding attribute of a user profile (e.g. *interestedInContent*). As next, the values of the attributes are retrieved from the underlying RDF graphs (representing multimedia resources and user profiles) and their similarity is computed using different matching techniques depending on the datatype and semantics of the attribute in question. Finally, the attribute similarities are aggregated with the Simple Additive Weighting method, which, due to its linear form of trade-offs between attributes, generates close approximations to more complicated non-linear approaches, while remaining far easier to use and understand, as research results have shown [20]. The ranking of alternatives based on the aggregated similarity score is then generated and presented to the user.

The methods used for the computation of attribute similarities range from simple string matching (for properties like *language*, *channel*, *audio*) and numeric matching (for *contentRatings*) to matching approaches for ontology concepts. Since most of attribute values describing multimedia content and user preferences are represented by taxonomy concepts (see Section 4.4.1) we have implemented and evaluated four methods [36], [3], [10], [24] for computing concept similarity based on their respective position in a concept hierarchy (indicated by the *rdfs:subClassOf*-relation). In general, those approaches imply the specialization assumption meaning that the semantic difference between upper level concepts is greater than between lower level concepts and the similarity between parent and child concepts is greater than between brothers.

4.4.3 Architecture

Figure 4.7 shows the architecture of the prototypical implementation of the personalized semantic recommender system for multimedia content which was implemented in Java. We utilize the cf2xmltv[9] tool to retrieve descriptions of multimedia resources in xml format which are then complemented with additional information obtained from the Internet Movie Database through a dedicated wrapper and finally transformed (by means of XSLT) into an RDF rep-

[9] http://www.koetter.cc/cf2xmltv/

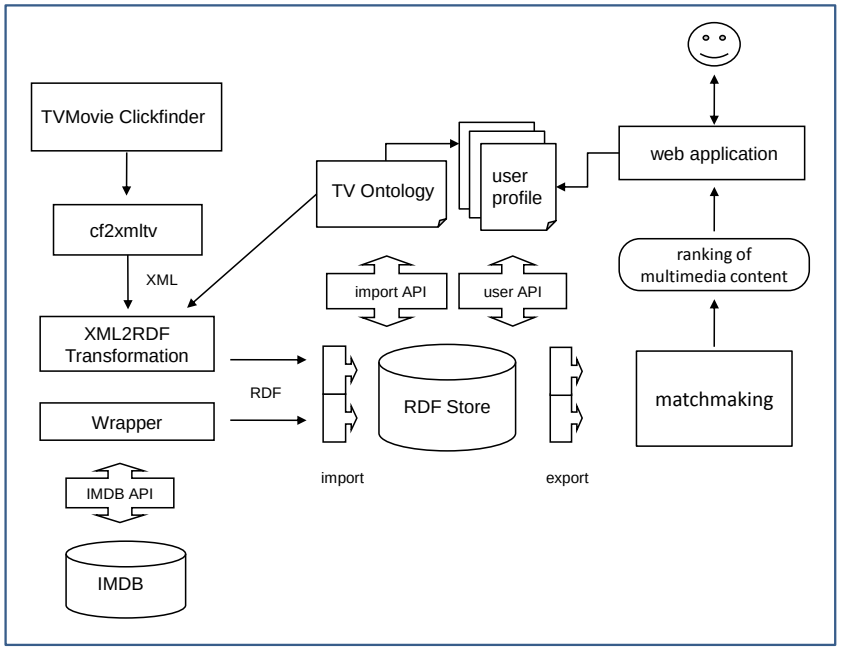

Figure 4.7: Architecture of the Recommender System

resentation with concepts from our TV ontology. For accessing, manipulating, and storage of RDF data we use Jena - the Semantic Web Framework for Java[10][11]. The key part of the architecture is the matchmaking component implemented based on SemMF - the Semantic Matchmaking Framwork[11][23] which we extended with additional attribute matchers for ontology concepts (see Section 4.4.2). The output of the matchmaker is a ranking of multimedia resources with respect to user preferences which is passed to the web application presenting the recommendations to the user.

4.5 Semantic Search Assistant for the Museumsportal-Berlin

Nowadays most of specialized portals providing information on a certain topic of interest offer users access to their information via simple keyword search. An example of such a portal is Museumsportal Berlin[12] where users can find information about over 200 museums, memorial places, castles, and other cultural institutions as well as their services and current exhibitions. In this section, on the example of the Museumsportal Berlin run by our cooperation partner x:hibit GmbH[13], we point out the shortcomings of the keyword search and demonstrate

[10] http://jena.sourceforge.net/
[11] http://semmf.ag-nbi.de/
[12] http://www.museumsportal-berlin.de/
[13] http://www.xhibit.de/

how the application of the Semantic Web technologies may lead to improved search and navigation functionality on specialized portals. The ideas presented in this section serve as a foundation for a prototypical implementation of a semantic search assistant.

4.5.1 Searching the Portal - Current Approach

Currently, the Museumsportal Berlin provides its visitors with a simple keyword based search functionality allowing users to find web pages presenting museums, exhibitions, or events in which the specified term appears either in the textual description or among tags associated with a particular page. The main drawback of the keyword search, in general, is that the results are obtained merely on the basis of a syntactic match (i.e. the exact occurrence of a given term). This problem is especially evident in cases of:

- **misspelling**
- **alternative spelling** (e.g. Sandro Botticelli vs. Il Botticello)
- **aliases** (Alessandro di Mariano di Vanni Filipepi known as Botticelli)
- **synonyms** (words having the same meaning, e.g. fix and repair)
- **homonyms** (the same word having different meanings, e.g. bank meaning either a river bank or a financial institution)

Those cases may lead to a situation where the museums of interest to the user are not found, even though they exist in the portal's database, simply because the searched keyword does not exactly match the words used in the museums' descriptions.

Another problem arises from the fact, that the search engine does not 'understand' the semantics of the search keyword and thereby cannot relate it to other terms which might also yield a valid query result. In order to illustrate this, consider the following example (depicted in Figure 4.8): If users of the Museumsportal Berlin who are looking for museums related to *impressionism*, merely perform a search for this particular keyword, they will find only two entries. There are, however, many more museums presenting paintings of impressionist artists, for example pieces by Claude Monet, Max Liebermann, or Karl Hagemeister. Unless the user performs multiple iterative searches for all those related terms, which of course is a tedious task requiring some knowledge in the arts domain, many museums of interest to the user will not be found.

Moreover, the keyword search proves rather an inefficient method if users introduce additional constraints into their queries. For example, if someone is looking for museums or events related to *impressionism*, open on Tuesday, with entrance fee less than 10 and audio guidance in English, the museums of interest can hardly be found by a simple enumeration of keywords. Instead, a mixed approach of searching and navigation is required. First, the user has to perform a query for the key concept (i.e. impressionism), then he or she has to examine each found museum or event by following links to subpages containing information on opening hours, prices, and services. In this concrete example the user of the Museumsportal Berlin would have to go through a navigation path consisting of 45 clicks, at the same time evaluating and aggregating all the information 'manually' and memorizing museums satisfying his or her preferences.

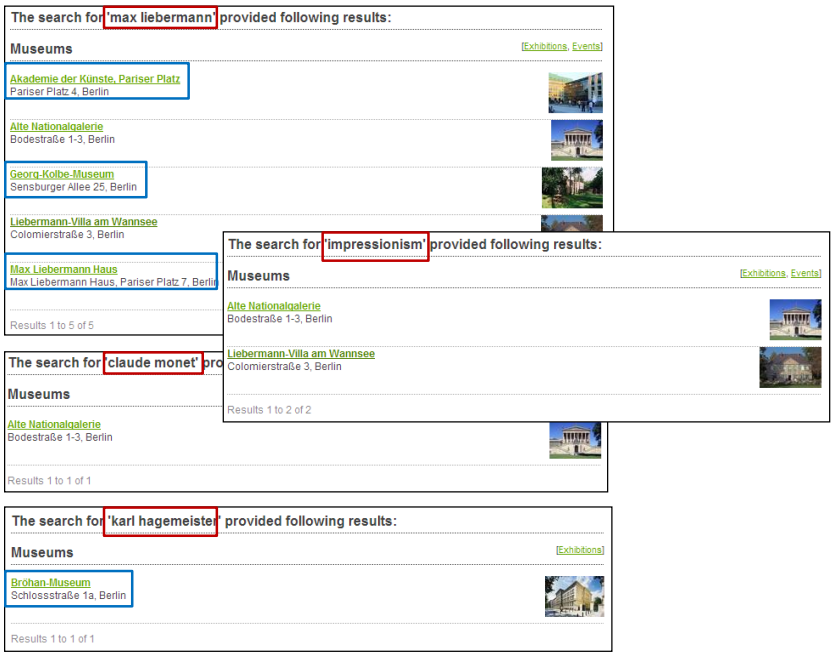

Figure 4.8: Example of a search expansion. The search for impressionism. is expanded into multiple searches for artists belonging to this art movement, like Claude Monet, Max Liebermann and Karl Hagemeister. The blue boxes represent additional search results.

4.5.2 Enhancing the Portal with Semantic Web Technologies

The problems associated with keyword search are mainly caused by the fact that most of the information available on various portals, such as the Museumsportal Berlin, is represented in form of textual descriptions designed to be read by humans. Although machines can parse web pages for layout processing, they do not understand the semantics of the data. In this paper, we propose enhancements to the portal relying on a formal representation of the information from the arts domain using Semantic Web Technologies.

Museum Ontology

In oder to capture the semantics of the portal data we developed a museum ontology (represented in OWL [4]) consisting of two sub-ontologies:

- **Museum Description Ontology** - defining key concepts used for describing cultural institutions as well as events and exhibitions offered by them.
- **Arts Domain Ontology** - capturing the knowledge from the arts domain including information on artists, art movements, etc.

The former sub-ontology is populated with instances of museums which are present on the portal. We convert all the available data about each museum into the schema of our ontology. Since most of the information is provided by those institutions themselves, through a simple input form, the data is rather weakly structured. Therefore, we additionally apply Named Entity Recognition techniques for the extraction of artist names, etc. as well as identify catchwords belonging to the arts domain. The found names and catchwords are, in turn, mapped onto concepts from the latter sub-ontology, thereby connecting the information about museums with a broader knowledge base of semantic relations from the arts domain.

Since the process of ontology development and maintenance is a rather complex and costly task, especially for such a broad domain as arts, we try to reuse already existing knowledge provided by other communities such as Wikipedia[14]. At this point, it is important to note that we utilize this particular information source only as a practical example in order to illustrate the potential benefits resulting from the application of semantic technologies. In fact, there exist several classifications and thesauri, for example the Art and Architecture thesaurus (ATT) or the Union List of Artists Names (ULAN)[15], which could be used as a foundation for our domain ontology as well.

Information Integration

There is, however, one important issue about integrating information from Wikipedia into Museumsportal Berlin, which is, that Wikipedia itself is a collection of documents represented in textual form, targeted at human readers and thus can only be queried by keywords. As argued in previous sections, we need a well-structured and semantically rich representation of data in order to overcome the limits of keyword search. This is even more important if we want to automatically integrate the relevant information from Wikipedia into Museumsportal Berlin. Fortunately, owing to the DbPedia Project [6] - a community effort aiming at extracting structured data from Wikipedia and representing it with Semantic Web technologies - we can easily perform this integration task.

For each catchword or named entity (e.g. artist name), found either in the museum description or among its tags, we perform a look up in DbPedia in order to check if the given concept belongs to the arts domain. This can be found out based on the category of the DbPedia-resource corresponding to the concept in question. For example, the catchword *impressionism* has a corresponding DbPedia-resource *dbpedia:Impressionism* which is an instance of the class *yago:ArtMovements*, as shown in Figure 4.9. If the given catchword was positively validated, additional information describing this resource (in this example: painters associated with this movement etc.) is integrated into our ontology.

By linking domain concepts on the Museumsportal with DbPedia-resources (also pointing to human-readable Wikipedia articles) we are able to enrich the content presented on the portal by embedding additional information on catchwords and entities found in museum descriptions. Consequently, visitors of the portal are provided with comprehensive information on the subject of museum

[14] `http://www.wikipedia.org/`
[15] `http://www.getty.edu/research/conducting_research/vocabularies/`

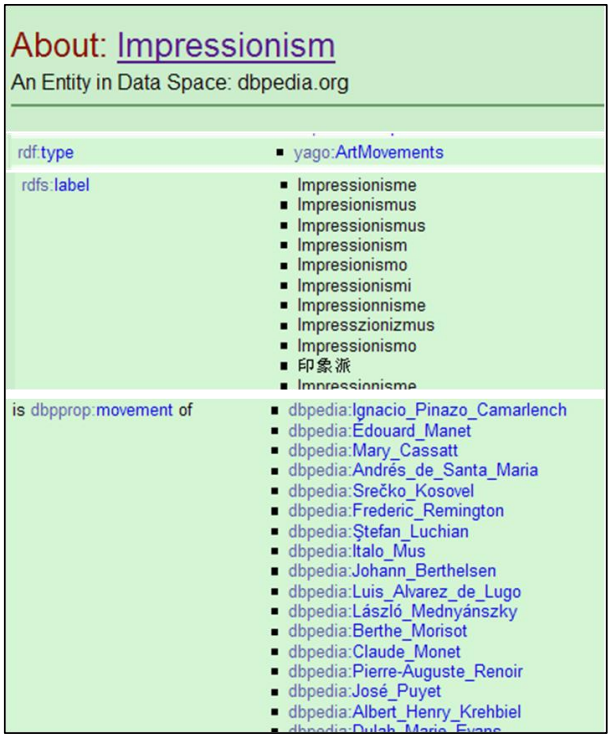

Figure 4.9: Excerpt of a semantic representation of the concept *impressionism*. Source: `http://dbpedia.org/page/Impressionism`

exhibitions without the need to leave the Museumsportal in order to consult other sources for more details on encountered keywords.

Improved Search and Navigation

Apart from enriching the information presented in the front-end of the Museumsportal Berlin, we also use semantic relations between concepts from the arts domain in order to overcome the limits of keyword search.

Since the domain ontology extracted from DbPedia contains information on synonyms and alternative spelling for arts concepts, e.g. *impressionism* and *impressionist art*, as well as on aliases of artist names (both indicated by the property *dbprop:redirect*), e.g. Sandro Botticelli or Il Botticello, we utilize this data by applying the mechanism of query expansion. Each search for a keyword specified by the user is complemented with queries for all its synonyms and spelling variations from our ontology.

Moreover, this simple mechanism is also applied to provide cross-lingual search. Although most of the museum and exhibition descriptions, delivered by those institutions themselves, are available in German as well as in English, there are still some exceptions where only a German version is available, especially in the case of tags. However, since the concepts in our ontology are associated

with their names in different languages (see Fig. 4.9) we are able to map the search keyword specified by the user to the same ontology concept, regardless of the language used, and expand the query into other languages. For example the search for *impressionism* (engl.) is realized by mapping this keyword to the concept *dbpedia:Impressionism* and performing the search for both the English and the expanded German (i.e. *Impressionismus*) term.

The examples so far deal with improving the keyword search for a particular concept from the arts domain by considering its different lexical representations (synonyms, alternative spelling, translations in different languages, etc.). The mechanism of query expansion, however, may go one step further by additionally taking into consideration semantic relations between different concepts. For example, based on our ontology, we are able to expand the search for an art movement into queries for artists belonging to (indicated by the property *dbprop:movement*) this particular style, or in the case of artists additionally search for their style and other artists they are related to in various ways (indicated by properties like *dbprop:influencedBy* or *dbprop:training*). As evaluated in [32] those kinds of semantic relations are the most interesting ones from the users' point of view.

Because the expansion of a query into semantically related concepts increases the recall, it is important to present the search result in such a way that it is manageable and comprehensible to users. At this point, once again, the semantic relations used in the process of query expansion might be used for generating explanations of the result set. One possible way of doing this, is to first list the exact matches of the searched keyword followed by a dynamically generated explanations of results obtained through query expansion.

Another advantage of a well-structured ontology-based representation of museum data is the possibility of the realization of complex queries, such as discussed in Section 4.5.1. Users can specify their search constraints (e.g. opening hours, entrance fee, desired services, etc.) through facets corresponding to the possible values of properties, from the ontology, describing museums and exhibitions. The preferences provided by the visitors of the portal are then translated into a formal query (i.e. SPARQL [27]). In consequence, the amount of clicks currently required to find out the desired information can significantly be reduced.

4.5.3 Architecture

Figure 4.10 shows the architecture of the prototypical implementation of the semantic search assistant for the Museumsportal Berlin. The integration component retrieves museum descriptions from the portal's database through a REST-webservice, applies named entity recognition for the extraction of relevant catchwords from textual descriptions and integrates additional information on the extracted resources from DbPedia thereby extending the museum ontology. The semantic representation of museum data is stored in the RDF backend realized with Jena Semantic Web Framework for Java[16][11]. The query extension component takes as input keywords provided by the user, maps them to concepts from the domain ontology and expands the query into related concepts using a pre-defined set of expansion rules.

[16] `http://jena.sourceforge.net/`

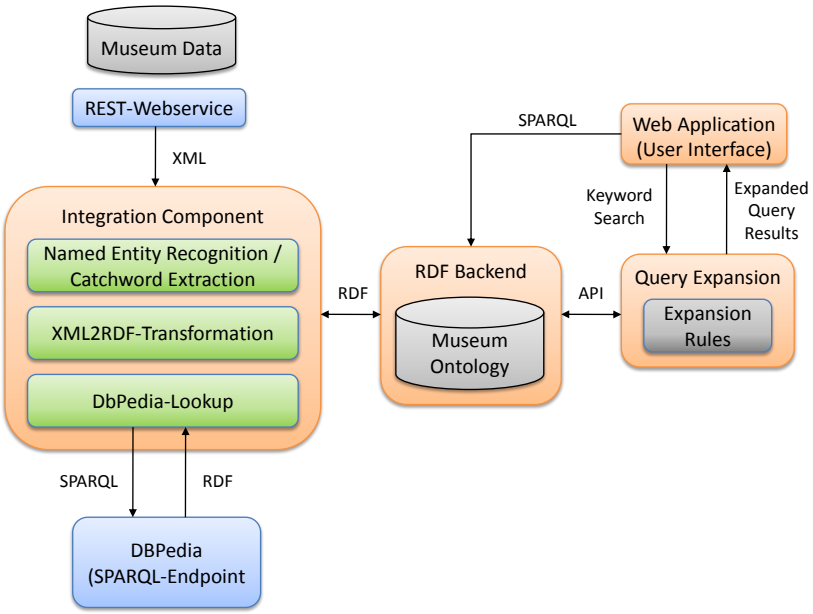

Figure 4.10: Architecture of the Semantic Search Assistant for the Museumsportal Berlin

4.5.4 Related Work

We presented how the application of Semantic Web technologies may improve search and navigation on a portal providing information on museums and exhibitions. Our analysis, however, is restricted to only this kind of information which is available on the Musuemsportal Berlin, i.e. descriptions of cultural institutions, combined with background knowledge from the arts domain.

The benefits of the deployment of Semantic Web technologies to enhance access to digital collections of museums are demonstrated by the CHIP (Cultural Heritage Information Presentation) project funded by Dutch Science Foundation program [17]. The researchers collaborating with the Rijksmuseum Amsterdam developed a content-based recommender system that recommends art-related concepts based on user ratings of artworks [32]. For example, if a user assigns the painting *Night watch* a high ranking, the user will get a recommendation for its creator Rembrandt.

[17] http://www.chip-project.org/

Chapter 5

Conclusion and Outlook

In this report, we described the progress of our work in the project *Corporate Semantic Web* during the last milestone phase. We presented several prototypical implementations as a proof of our conceptual Corporate Semantic Web architecture presented in the last report, covering the three pillars Corporate Ontology Engineering, Corporate Semantic Collaboration, and Corporate Semantic Search. The prototypes were partially developed in tight co-operation with our industrial partners.

The upcoming phase of our work will comprise an in-depth evaluation of the results of our work. Together with our industrial partners, we will assess the functional correctness and the economic gain induced by our newly developed methods.

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] M.S. Ackerman, V. Pipek, and V. Wulf, editors. *Sharing expertise: Beyond knowledge management*. MIT press, 2003.

[2] B. Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. In *Proceedings of the 16th Intl. World Wide Web Conference (WWW 2007)*, 2008.

[3] Troels Andreasen and Henrik Bulskov. From ontology over similarity to query evaluation. In *2nd CoLogNET-ElsNET Symposium - Questions and Answers: Theoretical and Applied Perspectives*, pages 39–50. Elsevier Science, 2003.

[4] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference. `http://www.w3.org/TR/owl-ref/`, February 2004.

[5] Dave Beckett. RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/`, 2004. W3C Recommendation.

[6] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[7] Elena Paslaru Bontas and Malgorzata Mochol. Towards a reuse-oriented methodology for ontology engineering. In *Proceedings of the 7th International Conference on Terminology and Knowledge Engineering TKE 2005*, 2005.

[8] J. Bransford. *How people learn: Brain, mind, experience, and school*, chapter 2, pages 31–50. National Academy Press, 2003.

[9] Dan Brickley and Libby Miller. FOAF Vocabulary Specification 0.97. Namespace document, foaf project, 2010.

[10] Henrik Bulskov, Rasmus Knappe, and Troels Andreasen. On measuring similarity for conceptual querying. In *FQAS '02: Proceedings of the 5th International Conference on Flexible Query Answering Systems*, pages 100–111, London, UK, 2002. Springer-Verlag.

[11] 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.

[12] B. Chandrasekaran, John R. Josephson, and V. Richard Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14:20–26, 1999.

[13] Krishnendu Chatterjee, Luca Alfaro, and Ian Pye. Robust content-driven reputation. *AISec '08: Proceedings of the 1st ACM Workshop on AISec*, Oktober 2008.

[14] K. Ehrlich. *Locating expertise: Design issues for an expertise locator system*, pages 137–158. In Ackerman et al. [1], 2003.

[15] Robert Tolksdorf Elena Paslaru Bontas, Malgorzata Mochol. Case studies on ontology reuse. In *Proceedings of the 5th International Conference on Knowledge Management*, 2005.

[16] T. Erickson and W.A. Kellogg. *Knowledge communities: Online environments for supporting knowledge management and its social context*, pages 299–326. In Ackerman et al. [1], 2003.

[17] Mariano Fernandez, 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.

[18] M. Hertzum and A. Pejtersen. The information-seeking practices of engineers: searching for documents as well as for people. *Information Processing and Management*, 36(5):761–778, Jan 2000.

[19] Clyde W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45(2):42–47, 2002.

[20] C.L. Hwang and K. Yoon. *Multiple Attribute Decision Making: Methods and Applications*. Springer-Verlag, New York, 1981.

[21] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Cex and mex: Logical diff and semantic module extraction in a fragment of owl. In Kendall Clark and Peter F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.

[22] Feng Pan and Jerry R. Hobbs. Time ontology in OWL. W3C working draft, W3C, September 2006. http://www.w3.org/TR/2006/WD-owl-time-20060927/.

[23] Adrian Paschke, Gökhan Coşkun, Ralf Heese, Markus Luczak-Rösch, Radoslaw Oldakowski, Ralph Schäfermeier, and Olga Streibel. Realizing the corporate semantic web: Concept papers. Technical Report TR-B-08-09, Freie Universität Berlin, April 2009.

[24] R. Pázman. Ontology search with user preferences. In P. Návrat, P. Bartoš, M. Bieliková, L. Hluchý, and P. Vojtá, editors, *Tools for Acquisition, Organisation and Presenting of Information and Knowledge*, pages 139–147, 2006.

[25] Helena Sofia Pinto, Steffen Staab, and Cristoph Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain, 2004.

[26] 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.

[27] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. Recommentation, W3C, January 2008.

[28] Mike Rohland and Olga Streibel. Algorithmic extraction of tag semantics. In *FIS2009: Proceedings of the 2nd international Future Internet Symposium, Berlin, 2009*, pages –. LNCS, Subseries: Computer Communication Networks and Telecommunications, Springer Verlag, 2009.

[29] Olga Streibel and Malgorzata Mochol. Trend ontology for knowledge-based trend mining on textual information. In *IEEE Computer Society Proceedings of 7th International Conference on Information Technology : New Generations, ITNG2010, April 2010 Las Vegas,US, TO APPEAR*, pages 00–00. IEEE Computer Society, 2010.

[30] Bruce Tognazzini. First Principles of Interaction Design. http://www.asktog.com/basics/firstPrinciples.html, 2003. Last accessed on 30.4.2010.

[31] Max Völkel and Tudor Groza. Semversion: Rdf-based ontology versioning system. In *Proceedings of the IADIS International Conference WWW / Internet 2006 (ICWI 2006)*, 2006.

[32] Y. Wang, N. Stash, L. Aroyo, L. Hollink, and G. Schreiber. Using semantic relations for content-based recommender systems in cultural heritage. In *Workshop on Ontology Patterns (WOP) at International Semantic Web Conference (ISWC)*, October 2009.

[33] P. Wongthongtham, E. Chang, and T. Dillon. Software Design Process Ontology Development. In *Proceedings of the 2nd IFIP WG 2.12 and WG 12.4 International Workshop on Web Semantics (SWWS) in conjunction with OTM 2006*, pages 1806–1813, November 2006.

[34] Chung-Hsing Yeh. The selection of multiattribute decision making methods for scholarship student selection. *International Journal of Selection & Assessment*, 11(4):289–296, 2003.

[35] D. Yimam-Seid and A. Kobsa. *Expert-finding systems for organizations: Problem and domain analysis and the DEMOIR approach*, pages 327–358. In Ackerman et al. [1], Jan 2003.

[36] 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.