ORGANIZATIONAL MODELING WITH A SEMANTIC WIKI: FORMALIZATION OF CONTENT AND AUTOMATIC DIAGRAM GENERATION

by

António Ferreira

Project/Dissertation

Software Engineering

University of Madeira

Mathematics and Engineering Department

Oriented by

Prof. Pedro Campos

and Co-oriented by

Prof. David Aveiro

November 2008

UNIVERSITY OF MADEIRA

ABSTRACT

Organizational Modeling with a Semantic Wiki: Formalization of Meanings and Automatic Diagram Generation

by António Ferreira

A key to maintain Enterprises competitiveness is the ability to describe, standardize, and adapt the way it reacts to certain types of business events, and how it interacts with suppliers, partners, competitors, and customers.

In this context the field of organization modeling has emerged with the aim to create models that help to create a state of self-awareness in the organization.

This project's context is the use of Semantic Web in the Organizational modeling area. The Semantic Web technology advantages can be used to improve the way of modeling organizations. This was accomplished using a Semantic wiki to model organizations. Our research and implementation had two main purposes: formalization of textual content in semantic wiki pages; and automatic generation of diagrams from organization data stored in the semantic wiki pages.

Keywords: Organizational Modeling, Semantic wiki, WordNet, Diagrams.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1 - INTRODUCTION

1.1 Motivation

Semantic wikis are tools easy to use by people without much software knowledge. We intend to be possible that all the collaborators of a certain organization can contribute to the creation of an organizational awareness through the use of semantic wikis. This tool allows an organic and coherent collection of organizational knowledge in the form of elements and relations in models that represent several facets of an organization, being possible to create organizational models aligned with the organization reality, which in turn allows capturing it and following its evolution.

1.2 Objectives

Analysis and development of a prototype based on MediaWiki and Semantic MediaWiki that allows the modeling of organizations and their business processes. First objective was integration of the semantic wiki with an external source to increase formalization of content from the organizational artifacts modeled. Other objective was the generation of various types of engineering diagrams to improve understanding of Organizational models.

1.3 Project description and context

The project context is semantic web and its use to model organizations. All the project work was related with a Semantic wiki. The first part was the integration of some type of dictionary for English language with Semantic MediaWiki, to give meanings to words in wiki text. The main part was using Semantic wiki advantages; from wiki stored semantic data, generate several types of diagrams from parts of the Organizational model.

1.4 Content

After Introduction follows Chapter 2 - Research context and problems definition, where we study the theoretical basis and the software used that was the point of departure of our

research. In this chapter we also identify the problem and define our research strategy. In chapter 3 - Related work, based on objectives and problems identified, we analyze projects partly addressing problems identified. Chapter 4 – Solutions and Contributions, shows the solutions found to address the problems we raised. A more detailed and technical explanation about the elaborated work follows in Chapter 5 - Implementation. In Chapter 6 - Future work, we see some interesting research directions that can be followed next. To finalize, in Chapter 7 - Conclusion, we draw the conclusions derived from the project.

CHAPTER 2 – RESEARCH CONTEXT AND PROBLEMS DEFINITION

With the general ideas of the project recognized, the present chapter is about the fundamental theoretical concepts needed in order to understand our project. To start, we introduce you the Semantic Web, a growing extension of the World Wide Web; next we take a detailed look at Semantic wikis and their advantages. Following is a study of a Framework for Organizational Engineering. Then we take a look at the applications used as a starting point for our research. In the end of the chapter we identify the problems we intend to solve with this project and set the research strategy.

2.1 Semantic Web

We begin with an introduction with a text from Sir Tim Berners-Lee. Next we study Semantic web main concepts, its definition, purpose, the limitations of hypertext web and how semantic web gives solutions to them. Then we list the semantic web components and take note of the importance of data modeling in current Organizational context. Ending this section is some interesting examples of semantic web applications.

To date, the World Wide Web has developed most rapidly as a medium of documents for people rather than of information that can be manipulated automatically. By augmenting Web pages with data targeted at computers and by adding documents solely for computers, we will transform the Web into the Semantic Web.

Computers will find the meaning of semantic data by following hyperlinks to definitions of key terms and rules for reasoning about them logically. The resulting infrastructure will spur the development of automated Web services such as highly functional agents.

Ordinary users will compose Semantic Web pages and add new definitions and rules using off-the-shelf software that will assist with semantic markup. [1]

Next three sections include texts from Wikipedia Semantic web page [2].

2.1.1 Definition

The Semantic Web is an evolving extension of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content. It derives from World Wide Web Consortium director Sir Tim Berners-Lee's vision of the Web as a universal medium for data, information, and knowledge exchange.

At its core, the semantic web comprises a set of design principles, collaborative working groups, and a variety of enabling technologies. Some elements of the semantic web are expressed as prospective future possibilities that are yet to be implemented or realized. Other elements of the semantic web are expressed in formal specifications. Some of these include Resource Description Framework (RDF), a variety of data interchange formats (e.g. RDF/XML, N3, Turtle, N-Triples), and notations such as RDF Schema (RDFS) and the Web Ontology Language (OWL), all of which are intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain.

2.1.2 Purpose

Humans are capable of using the Web to carry out tasks such as finding the Finnish word for "monkey", reserving a library book, and searching for a low price on a DVD. However, a computer cannot accomplish the same tasks without human direction because web pages are designed to be read by people, not machines. The semantic web is a vision of information that is understandable by computers, so that they can perform more of the tedious work involved in finding, sharing and combining information on the web.

In 1999, Tim Berners-Lee originally expressed the vision of the semantic web as follows: “I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.”

Semantic publishing will benefit greatly from the semantic web. In particular, the semantic web is expected to revolutionize scientific publishing, such as real-time publishing and sharing of experimental data on the Internet (later we have a perfect example of this in the Semantic Web applications section).

Tim Berners-Lee has described the semantic web as a component of Web 3.0.

2.1.3 Relationship to the hypertext web

Limitations of HTML

Currently, the World Wide Web is based mainly on documents written in Hypertext Markup Language (HTML), a markup convention that is used for coding a body of text interspersed with multimedia objects such as images and interactive forms. Metadata tags, for example:

<meta name="keywords" content="computing, computer studies, computer">

<meta name="description" content="Cheap widgets for sale">

<meta name="author" content="Billy Bob McThreeteeth">

Provide a method by which computers can categorize the content of web pages.

With HTML and a tool to render it (web browser software or another user agent), one can create and present a page that lists items for sale. The HTML of this catalog page can make simple, document-level assertions such as "this document's title is 'Widget Superstore'". But there is no capability within the HTML itself to assert unambiguously that, for example, item number X586172 is an Acme Gizmo with a retail price of €199, or that it’s a consumer product. Rather, HTML can only say that the span of text "X586172" is something that should be positioned near "Acme Gizmo" and "€ 199", etc. There is no way to say "this is a catalog" or even to establish that "Acme Gizmo" is a kind of title or that "€ 199" is a price. There is also no way to express that these pieces of information are bound together in describing a discrete item, distinct from other items perhaps listed on the page.

Semantic Web solutions

The Semantic Web takes the solution further. It involves publishing in languages specifically designed for data: RDF, OWL and Extensible Markup Language (XML). HTML describes documents and the links between them. RDF, OWL, and XML, by contrast, can describe arbitrary things such as people, meetings, or airplane parts. Tim Berners-Lee calls the resulting network of Linked Data the Giant Global Graph, in contrast to the HTML-based World Wide Web.

These technologies are combined in order to provide descriptions that supplement or replace the content of Web documents. Thus, content may manifest as descriptive data stored in Web-accessible databases, or as markup within documents (particularly, in Extensible HTML (XHTML) interspersed with XML, or, more often, purely in XML, with layout/rendering cues stored separately). The machine-readable descriptions enable content managers to add meaning to the content, i.e. to describe the structure of the knowledge we have about that content. In this way, a machine can process knowledge itself, instead of text, using processes similar to human deductive reasoning and inference, thereby obtaining more meaningful results and facilitating automated information gathering and research by computers.

An example of a tag that would be used in a non-semantic web page: <item>cat</item>

Encoding similar information in a semantic web page might look like this: <item rdf:about="http://dbpedia.org/resource/Cat">Cat</item>

2.1.4 Components

This and next section are made with excerpts from [3]. The semantic web is based on the idea of a "layered architecture". Much like the ISO concept of layers in data communications, the semantic web architecture is composed of the following layers:

- **URIs and Namespaces** - the names of things
- **XML and XMLS Data types** - a means of communicating data
- **RDF and RDF/XML** - a basic language

- **RDF Schema and Individuals** - an ontological primitive
- **Ontology languages, such as OWL** - the logical layer
- **Applications** - the implementation layer. [3]w3schools

2.1.5 Semantic Web and Data Modeling

Everyone knows that we are drowning in information, both from the databases in our companies as well as from the world-wide web, the media, and life in general. The information technology industry has been wrestling with this problem for years, and one is entitled to wonder if things will ever get better.

Well, there are a couple of new/old ideas on the horizon that might help: semantics and ontology. Data modeling was invented three decades ago to assist in the design of databases-in particular relational databases. As it matured, the technique has become recognized as a tool for analyzing the semantics of an organization - what is the structure of the organization's information as it's used in carrying out its mission?

Companies are beginning to recognize that semantics is important if their systems (and their people, for that matter) are going to communicate with each other, and, based on this recognition, they are also recognizing the importance of collecting "ontologies", or glossaries that describe the language they use to carry out their activities.

In other words, a couple of 2500 year-old words are becoming the hot new buzzwords in our industry. In simple words, ontology tells us what exists. Semantics tells us how to describe it.

About Data Models and Ontology Languages

Data models are to be understood by humans, with computers only serving as gateways to permit capture of "valid" data. In its latest incarnations, however, an ontology language begins with instances of actual data. Its purpose is to classify them so that computers can make inferences from them.

The data modeling mindset is based upon the closed world assumption: Only that which is asserted is known.

Ontology languages are based on the open world assumption. All assertions are assumed to be true until proven otherwise.

2.1.6 Examples of Semantic Web Applications

This section provides some semantic web example projects. These applications use Semantic Web technologies to their advantage, making them better comparing to standard similar applications.

Twine

Twine is an application that helps people organize, share and discover information around their interests. Twine can be described as a "knowledge networking" application. It has aspects of social networking, wikis, blogging, knowledge management systems - but its defining feature is that it's built with Semantic Web technologies.

At first glance it's very much like Wikipedia, but there is a whole lot more smarts to the system. Described as "knowledge networking"- i.e. it aims to connect people with each other "for a purpose". It's not based around socializing, but to share and organize information you're interested in. Using Twine, you can add content via wiki functionality; you can email content into the system, and "collect" something (as an object, e.g. a book object). [4]

And while users certainly don't need to understand the Semantic Web in order to appreciate Twine, several technologies are hard at work behind the scenes of its simple user interface.

Let's take a closer look at one of the most important - the Resource Description Framework language or RDF.

Twine and RDF

Twine's "smarts" are derived from the simplicity of three-part RDF statements, often called triples or tuples. In fact, all information in Twine - whether about a particular object,

person, note, bookmark, tag, email message, or even a video - is expressed in a set of tuples.

Figure 1: Twine page

However, if the same URL is accessed by a system that asks for data in the form "application/rdf+xml", instead of returning a page of HTML, an RDF document is returned.

RDF documents are made up of simple three-part statements in the form <subject, predicate, object>.

For example, a system will see that Jurassic Park

- Has an author: Michael Crichton
- Was released on: 9/07/2006
- And has a comment, made by /user/lew.

Processing data in this clear 1, 2, 3 format is much faster and less error-prone than "screen scraping" the web page in the hope of retrieving the correct fields. Twine's knowledge of Jurassic Park is simply the set of all tuples that have this book as the subject. When two tuples refer to the same object, they become linked and in this way start to build a semantic

graph. In short, Twine uses tuples to access a tremendous breadth and depth of information about any given subject.

RDF and the Semantic Graph

Where Twine is differentiated from the likes of Wikipedia is that its underlying data structure is entirely Semantic Web. The Semantic Web technologies used are: RDF, OWL, SPARQL, XSL.

RDF statements form a graph of arcs and nodes. Data input into Twine connects to people, places and other pieces of information. The graph below was made by RDF Gravity to virtually display the RDF description of Jurassic Park.

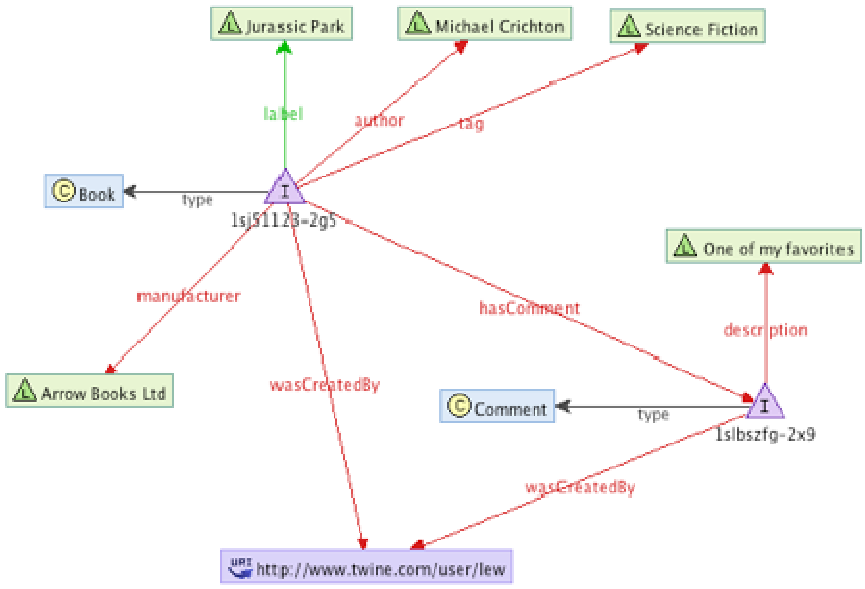

Figure 2: RDF graph of a Twine's page

The above graph shows that there is not only a book by Michael Crichton, but there is also a comment linked to the book. Both the book and the comment were made by /usr/lew. As more books are authored by Michael Crichton or published by Arrow Books, these objects continue to link together. Twine uses the data and properties in the graph to link related

information, allowing users to search along different dimensions. As more information is added, the richer and more useful the graph becomes. [5]

Analyses

We consider Twine's functionalities similar to a semantic wiki (introduced in the next section). It has some additional features like the tags system, also the user interface is much more attractive than the semantic wiki (SMW) we used in our project and that we will introduce later on this chapter.

SIOC

Reading SIOC official site [6] we discover that Semantically-Interlinked Online Communities or SIOC is a framework aimed at connecting online community sites and internet-based discussions. Currently, online communities (boards, blogs, etc.) are like islands - they contain valuable information but are not well connected. SIOC allows us to interlink these sites, and enables the extraction of richer information from various discussion services.

SIOC in brief

- The core of SIOC is the ontology. It's a vocabulary that contains concepts necessary to express information contained in online community sites
- Online community sites then provide information about their structure and contents to the outside world. This information is machine readable and structured using the SIOC ontology
- Since the information is already present inside these sites, all that is needed is to install a SIOC export plugin or extension
- This information can be used by tools that understand SIOC data to suggest related information from other community sites

Figure 3 shows various information islands and how SIOC connects them.

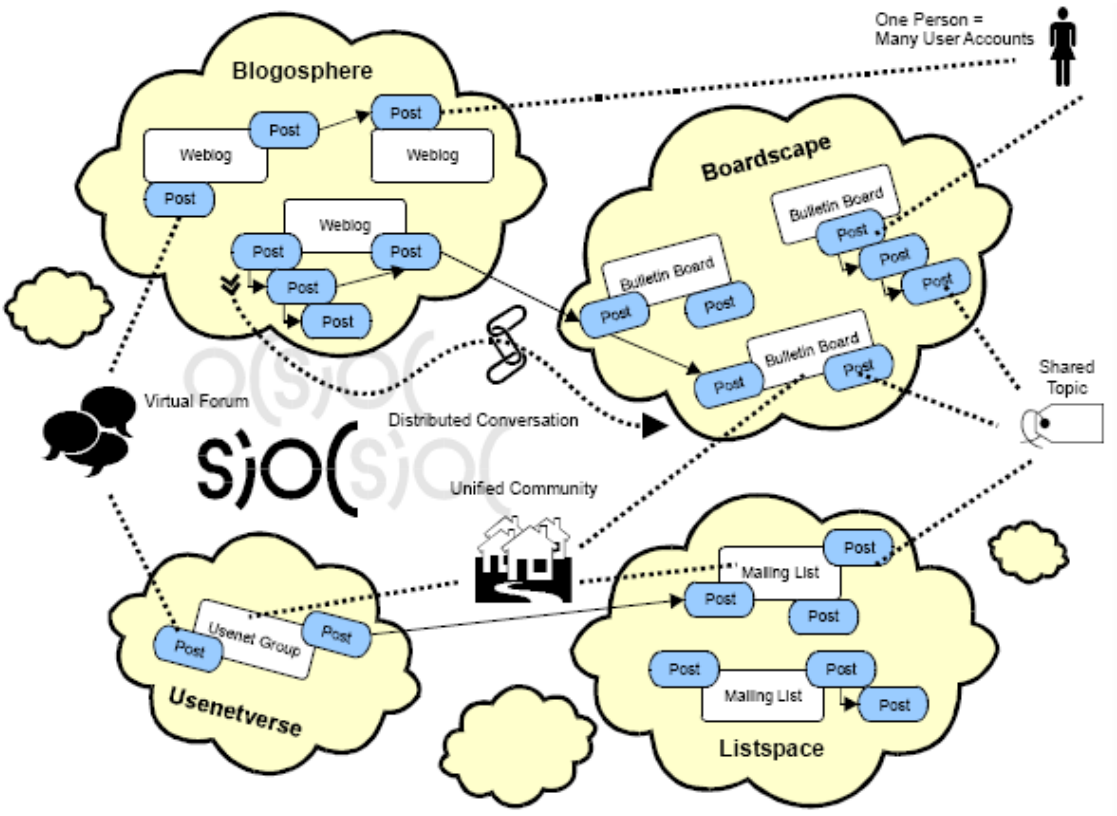

Figure 3: Creating connections between discussion clouds with SIOC

Current and Future Uses of SIOC

- Create distributed conversations across blogs, forums and mailing lists
- Use as an enhanced export/import format, with access to either the entire content or summaries
- Enable publishing and subscribing to decentralized discussion channels and communities

Next Figure shows the main concepts (and their relations) of the SIOC ontology.

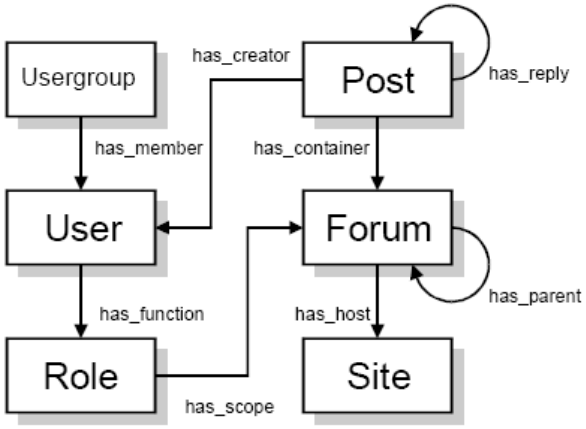

Figure 4: The main concepts in the SIOC ontology

Analyses

SIOC is a very good example of the power of semantic relations. I think it can be viewed as similar to RSS but for online communities and more powerful. It uses semantics to find relations between different and isolated sources. It's easily installed in websites, and the information can be accessed installing a Firefox extension.

Nextbio

NextBio is a privately owned interactive life-science search engine company. The search engine searches through and correlates highly complex experiments, literature and clinical data to aid researchers in making new discoveries. It provides a unified interface for researchers (biologists, clinicians and biomedics) to easily formulate and test new hypotheses across vast collections of experimental data.

All imported data within an enterprise is cross-correlated to previously uploaded internal data and to the public data. Scientists are using NextBio to improve our ability to mine and identify relevant prognostic (survival) and predictive (drug efficacy) molecular signatures which are significant in their research. [7]

At NextBio, search is all about the science. NextBio's scientific foundation consists of a robust framework that connects highly heterogeneous data and textual information. Our semantic framework is based on gene, tissue, disease and compound ontologies that are

leveraged for both the data and literature search functionalities. Within this framework, information from diverse organisms, platforms, data types and research areas is seamlessly integrated into and correlated within a single searchable environment using our proprietary algorithms. NextBio correlates gene ontology, pathway and other functional information within the context of the world's experimental data. [8]Official site

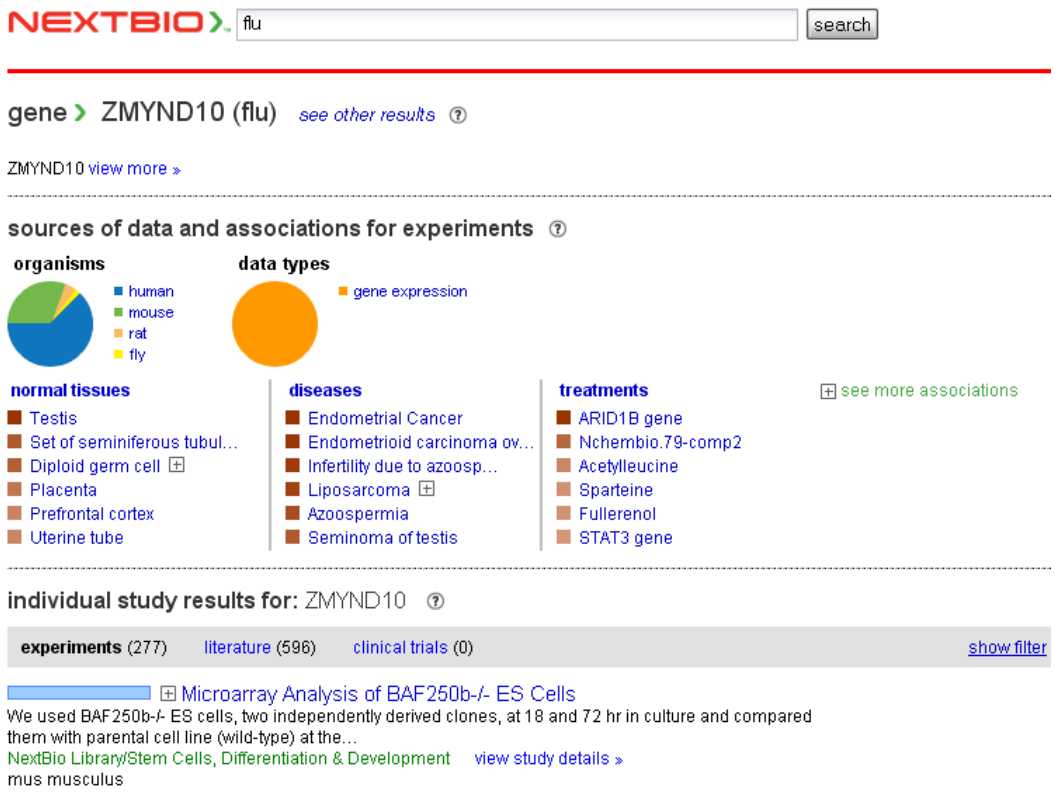

Figure 5: Example of a Nextbio search

Analyses

Nextbio is a good example of how semantics improve data search and navigation, also provides an always interesting graphical view.

2.2 Semantic wikis

This section contains excerpts from Wikipedia article [9] about a semantic wiki. A semantic wiki is a wiki that has an underlying model of the knowledge described in its

pages. Regular wikis have structured text and untyped hyperlinks. Semantic wikis allow the ability to capture or identify further information about the pages' (metadata) and their relations.

2.2.1 Key characteristics

Reliance on Formal Notation

The knowledge model found in a semantic wiki is typically available in a formal language, so that machines can process it into an entity-relationship or relational database.

The formal notation may be included in the pages themselves by the users, as in Semantic MediaWiki. Or, it may be derived from the pages or the page names or the means of linking. For instance, using a specific alternative page name might indicate a specific type of link was intended. This is especially common in wikis devoted to code projects.

In either case, providing information through a formal notation allows machines to calculate new facts (e.g. relations between pages) from the facts represented in the knowledge model.

Enables Semantic Web

The technologies developed by the Semantic Web community provide one basis for formal reasoning about the knowledge model that is developed.

2.2.2 Example

Imagine a semantic wiki devoted solely to foods. The page for an apple would contain, in addition to standard text information, some machine-readable semantic data. The most basic kind of data would be that an apple is a kind of fruit – what is known as an inheritance relationship. The wiki would thus be able to automatically generate a list of fruits, simply by listing all pages that are tagged as being of type "fruit." Further semantic tags in the "apple" page could indicate other data about apples, including their possible colors and sizes, nutritional information and serving suggestions, and any other data that

was considered notable. These tags could be derived from the text but with some chance of error - accordingly they should be presented alongside that data to be easily corrected.

If the wiki exports all this data in RDF or a similar format, it can then be queried in ways a database might - so that an external user or site could, for instance, submit a query to get a list of all fruits that are red and can be baked in a pie.

2.2.3 Use in knowledge management

Where wikis replace older CMS or knowledge management tools, semantic wikis try to serve similar functions: to allow users to make their internal knowledge more explicit and more formal, so that the information in a wiki can be searched in better ways than just with keywords, offering queries similar to structural databases.

Some systems are aimed at personal knowledge management, some more at knowledge management for communities. The amount of formalization and the way the semantic information is made explicit vary. Existing systems range from primarily content-oriented (like Semantic MediaWiki) where semantics are entered by creating annotated hyperlinks, via approaches mixing content and semantics in plain text (like WikSAR or living ontology), via content-oriented with a strong formal background (like IkeWiki), to systems where the formal knowledge is the primary interest (like Platypus Wiki), where semantics are entered into explicit fields for that purpose.

Also, semantic wiki systems differ in the level of ontology support they offer. While most systems export their data as RDF, some even support various levels of ontology reasoning.

To conclude, we can make a comparison: semantic wikis extend and improve regular wikis like semantic web extends World Wide Web.

2.3 Enterprise Architecture Modeling

This section contains excerpts from *Enterprise Architecture Modeling with the Unified Modeling Language* [10], an article that defines a conceptual framework developed for

organizational engineering. The concepts and rules explained next were the basis used to model organizations with a semantic wiki.

2.3.1 Enterprise Architecture Views

The enterprise architecture model comprises five architectural components: Organizational Architecture, Business Architecture, Information Architecture, Application Architecture, and Technological Architecture. Each of these sub-architectures is individually represented and organized as a UML package as depicted in Figure 6. Each package owns its model elements and its elements cannot be owned by more than one package. The relationships, depicted as dotted arrows, represent the dependencies of each package.

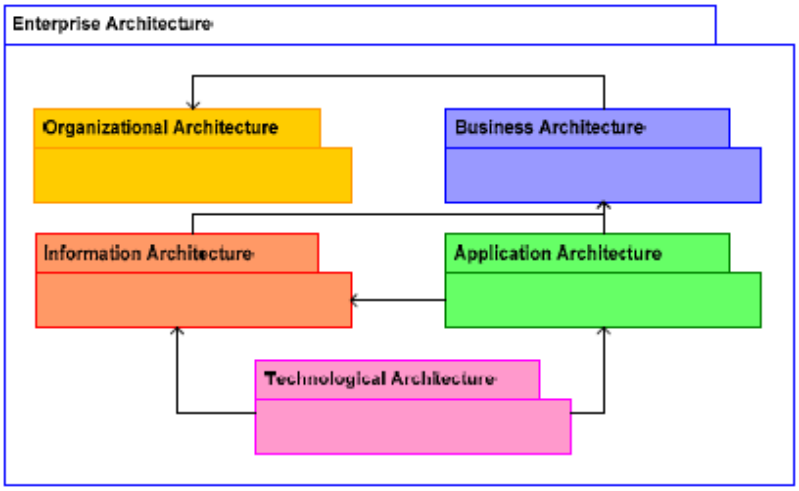

Figure 6: The five enterprise architecture components.

2.3.2 The Enterprise Architecture Model

The architectural views describe and relate the fundamental concepts that, as a whole, describe the enterprise architecture. Each is represented as a class within a specific package, as depicted in Figure 7. This section details the fundamental concepts and their relationships that are required to represent the enterprise architecture according to the five views that were defined in the previous section.

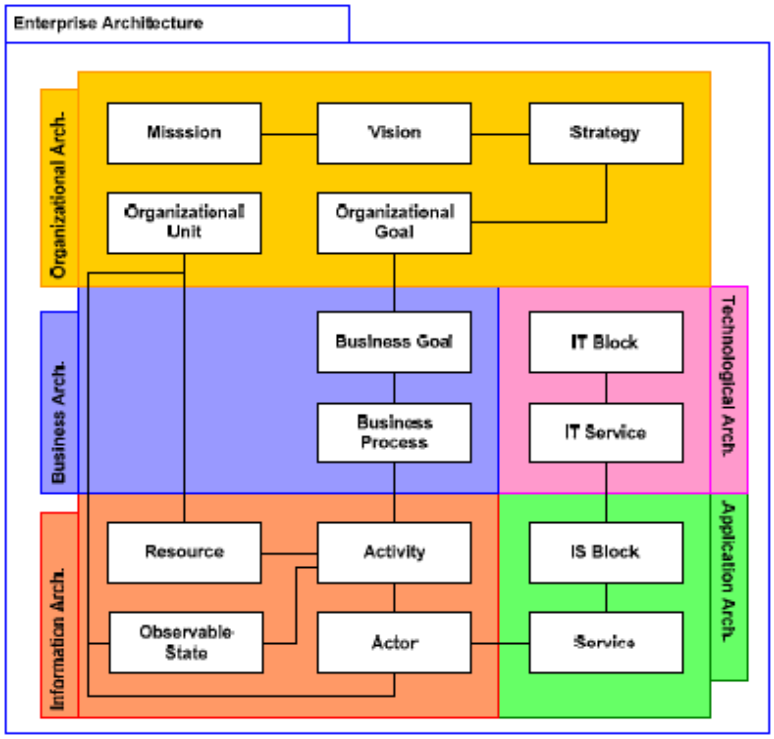

Figure 7: The fundamental concepts within each of the enterprise architecture views.

Fundamental Concepts

An organization can be modeled as a collection of business nouns that interact as described by a number of verbs. The nouns represent things within the organization that are of interest regarding the purpose of the model. The verbs stand for the enterprise activities that define how work is done and how value is added, thus describing its business processes and activities. Here we define the fundamental concept of entity and activity and that of role. These three concepts allow complex interactions of entities to be abstracted. The relationships between these three elements are depicted in the next Figure.

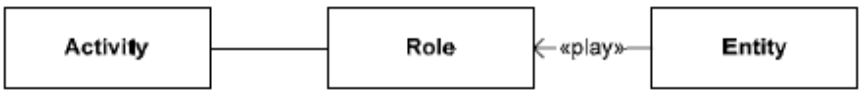

Figure 8: Relationships between Activity, Role and Entity.

Entity

An organization is composed of entities. Entities are nouns that have a distinct, separate existence, though it need not be of material existence. There is also no presumption that an entity is animate. An animate entity is able to exhibit active behavior. In enterprise modeling, an entity can be a person, place, machine, concept or event that has meaning in the context of the business, and about which some information may be stored because it's relevant for the purpose of the model.

Entities can be classified according to its attributes and methods. Entities may relate structurally to other entities, as in the case an entity is composed by other entities (e.g. an inventory is composed of products). An entity may also be specialized to restrict the features of a more general entity.

An entity is characterized by its attributes and methods. These features can be either intrinsic or extrinsic.Intrinsic features describe the entity in isolation, while extrinsic features arise from the relationships with other entities. For example, the entity Person has intrinsic features such as Age and Sex, and extrinsic features such as Job Position and Salary, which derive from a transitory relationship between the Person and the Organization. The state of the intrinsic features may change over time (e.g. Age) but always characterize the object. Extrinsic features only manifest themselves while a relationship is valid and may become unsuitable when the relationship is no longer valid.

Role

A role is the observable behavioral of an entity in the scope of a specific collaboration context. Hence, a role represents the external visible features of that entity when it collaborates with a set of other entities in the context of some activity. An entity relates to zero or more role classes through the stereotyped «play» relationship.

Roles aim at separating the different concerns that arise from the collaborations between the entities fulfilling an activity. A role may be bound to multiple entities via the «play» relationship.

Activity

An activity is an abstraction representing how a number of entities collaborate through roles in order to produce a specific outcome. Similarly to an algorithm, an activity aims accomplishing some task which, given an initial state, will always end in finite time and in a recognizable end-state. An activity may also be functionally decomposed into a finite set of further activities, thus add detail to the specification.

An activity specifies what entities are required to realize a task. As seen, roles are used to separate the description of the actual entity features from the features required by the collaboration in context of the activity. In this way, activities and entities are described separately, and roles may be reused in different activities.

Business Processes and Activity Coordination

Coordination means integrating or linking together different parts of a system to accomplish a collective set of tasks. In the case of activity coordination, it means describing how activities are linked together so that they define a business process. An example definition of business process is: A collection of activities that takes one or more kinds of inputs and creates an output that is of value to the customer.

Analyses

This framework had inconsistencies that needed to be addressed. One of them is stating that entities only relate through roles in the context of activities, while at the same time referring that “Information Architecture (...) provides a high level logical representation of all key entities as well as the relationships among them”. One key issue, while specifying an information architecture, is the ability to explore different levels of abstraction (specialization) or composition (aggregation) of certain entities. This cannot be done by using roles as, per its definition; it connects entities collaborating in an activity and not entities directly, as in an aggregation relation. Section 2.4.3 covers previous research work that shows how these inconsistencies were addressed

2.4 Review of used applications

This section introduces the software used as basis for our project; they served as starting point for what we developed. We start with MediaWiki - the wiki-system powering Wikipedia, then the study of Semantic MediaWiki, a MediaWiki extension to make it semantic. Following is a description of the Organizational Modeling extension for Semantic MediaWiki. Concluding this section is diagram drawing software Graphviz and our project's starting architecture or applications architecture, showing how the reviewed applications interact.

2.4.1 MediaWiki

MediaWiki is free server-based software which is licensed under the GNU General Public License (GPL). It's designed to be run on a large server farm for a website that gets millions of hits per day. MediaWiki is an extremely powerful, scalable software and a feature-rich wiki implementation, that uses PHP to process and display data stored in its MySQL database.

Pages use MediaWiki's wikitext format, so that users without knowledge of XHTML or CSS can edit them easily.

When a user submits an edit to a page, MediaWiki writes it to the database, but without deleting the previous versions of the page, thus allowing easy reverts in case of vandalism or spamming. MediaWiki can manage image and multimedia files, too, which are stored in the file system. For large wikis with lots of users, MediaWiki supports caching and can be easily coupled with Squid proxy server software.

Originally developed to serve the needs of the free content Wikipedia encyclopedia, today it has also been deployed by companies for internal knowledge management, and as a content management system. Notably, Novell uses it to operate several of its high traffic websites. [11]

2.4.2 Semantic MediaWiki

The Semantic wiki software our project uses is the Semantic MediaWiki, in the end of this section we explain why the choice of SMW. The following text was from Semantic MediaWiki official documentation [12].

Semantic MediaWiki (SMW) is a free extension of MediaWiki. While traditional wikis contain only texts which computers can neither understand nor evaluate, SMW adds semantic annotations that bring the power of the Semantic Web to the wiki.

Introduction

Wikis have become a great tool for collecting and sharing knowledge in communities. This knowledge is mostly contained within texts and multimedia files, and is thus easily accessible for human readers. But wikis get bigger and bigger, and it can be very time-consuming to look for an answer inside a wiki. As a simple example, consider the following question a user might have: «What are the hundred world-largest cities with a female mayor? »

Wikipedia should be able to provide the answer: it contains all large cities, their mayors, and articles about the mayor that tell us about their gender. Yet the question is almost impossible to answer for a human, since one would have to read all articles about all large cities first! Even if the answer is found, it might not remain valid for very long. Computers can deal with large datasets much easier, yet they are not able to support us very much when seeking answers from a wiki: Even sophisticated programs cannot yet read and «understand» human-language texts unless the topic and language of the text is very restricted. The wiki's keyword search does not help either in discovering complex relationships.

Semantic MediaWiki enables wiki communities to make some of their knowledge computer-processable, e.g. to answer the above question. The hard problem for the computer is to find out what the words in a wiki page (e.g. about cities) mean. Articles contain many names, but which one is the current mayor? Humans can easily grasp the problem by looking into a language edition of Wikipedia that they do not understand (Korean is a good start unless you are fluent there). While single tokens (names, numbers,

…) might be readable, it’s impossible to understand their relevance in the article. Similarly, computers need some help for making sense of wiki texts.

In Semantic MediaWiki, editors therefore add «hints» to the information in wiki pages. For example, someone can mark a name as being the name of the current mayor. This is done by editors who modify a page and put some special text-markup around the mayor's name. After this, computers can access this information (of course they still do not «understand» it, but they can search for it if we ask them to), and support users in many different ways.

Where SMW can help

Semantic MediaWiki introduces some additional markup into the wiki-text which allows users to add "semantic annotations" to the wiki. While this first appears to make things more complex, it can also greatly simplify the structure of the wiki, help users to find more information in less time, and improve the overall quality and consistency of the wiki. To illustrate this, we provide some examples from the daily business of Wikipedia:

1. **Manually generated lists.** Wikipedia is full of manually edited listings such as this one. Those lists are prone to errors, since they have to be updated manually. Furthermore, the number of potentially interesting lists is huge, and it’s impossible to provide all of them in acceptable quality. In SMW, lists are generated automatically like this. They are always up-to-date and can easily be customized to obtain further information.

2. **Searching information.** Much of Wikipedia's knowledge is hopelessly buried within millions of pages of text, and can hardly be retrieved at all. For example, at the time of this writing, there is no list of female physicists in Wikipedia. When trying to find all women of this profession that are featured in Wikipedia, one has to resort to textual search. Obviously, this attempt is doomed to fail miserably. Note that among the 20 first results, only 5 are about people at all, and that Marie Curie is not contained in the whole result set (since "female" does not appear on her page). Again, querying in SMW easily solves this problem (in this case even without further annotation, since existing categories suffice to find the results).

3. **Inflationary use of categories.** The need for better structuring becomes apparent by the enormous use of categories in Wikipedia. While this is generally helpful, it has also led to a number of categories that would be mere query results in SMW. For some examples consider the categories Rivers in Buckinghamshire, Asteroids named for people, and 1620s deaths, all of which could easily be replaced by simple queries that use just a handful of annotations. Indeed, in this example Category:Rivers, Property:located in, Category:Asteroids, Category:People, Property:named after, and Property:date of death would suffice to create thousands of similar listings on the fly, and to remove hundreds of Wikipedia categories.

4. **Inter-language consistency.** Most articles in Wikipedia are linked to according pages in different languages, and this can be done for SMW's semantic annotation as well. With this knowledge, you can ask for the population of Beijing that is given in Chinese Wikipedia without reading a single word of this language. This can be exploited to detect possible inconsistencies that can then be resolved by editors. For example, the population of Edinburgh at the time of this writing is different in English, German, and French Wikipedia.

5. **External reuse.** Some desktop tools today make use of Wikipedia's content, e.g. the media player Amarok displays articles about artists during playback. However, such reuse is limited to fetching some article for immediate reading. The program cannot exploit the information (e.g. to find songs of artists that have worked for the same label), but can only show the text in some other context. SMW leverages a wiki's knowledge to be useable outside the context of its textual article.

User manual introduction

This section is just to introduce SMW syntax and explain the basic way of making a semantic annotation with SMW.

Properties and types

Properties are the basic way of entering semantic data in Semantic MediaWiki. Properties can be viewed as «categories for values in wiki pages». They are used by a simple mark-up, similar to the syntax of links in MediaWiki: [[property name::value]]

Existing links can be directly augmented with such property information, while other types of data (such as numbers or calendar dates) need an additional editing step.

Turning Links into Properties

Consider the Wikipedia article on Berlin. This article contains many links to other articles, such as «Germany», «European Union», and «United States». However, the link to «Germany» has a special meaning: it was put there since Berlin is the capital of Germany. To make this knowledge available to computer programs, one would like to «tag» the link [[Germany]] in the article text, identifying it as a link that describes a «capital property». With Semantic MediaWiki, this is done by putting a property name and :: in front of the link inside the brackets, thus: [[capital of::Germany]]

In the article, this text still is displayed as a simple hyperlink to «Germany». The additional text capital of is the name of the property that classifies the link to Germany.

Why Semantic MediaWiki to support modeling in Organizational Engineering?

For a project introduced in next section, with the objective of choosing the best semantic wiki to work with, various semantic wikis were compared. In this section we report the conclusions.

SMW was the semantic wiki that better supported organizational modeling in Organizational Engineering. With the SMW was possible to create business objects and relations between them in an easy and collaborative way. The hierarchy between the concepts as all types of relations between organization entities could be created using semantic annotations. Next we see some other advantages.

Templates

The SMW provided the possibility of configure sections (templates) that appear in all pages, it was possible to specify in the SMW, the organizational views. This way, it was only necessary to define them in one place (a template), this way any change to the views would be automatically updated in all organization entities.

Definition of new functions

Another feature that revealed essential was the possibility to create new functions that could be used the same way of templates. The functions could be programmed in PHP (Semantic MediaWiki programming language), and added to the functions list. Whenever SMW found a call to one of those functions, invokes it to present its result in the page were it was specified. The big advantage of define functions is that we could create functions much more complex than the ones we could only with templates.

2.4.3 Organizational Modeling with a Semantic wiki

João Mendes developed a project called "Organizational Modeling with a Semantic Wiki" for *Instituto Superior Técnico*. This section includes excerpts from [13], an article about semantic bootstrap for organizational modeling. Follows a summary of the project:

The project presented a basic set of modeling primitives and rules with the purpose of enabling the construction of semantically rich and coherently integrated organizational models, in the context of enterprise engineering and architecture. This set, called "semantic bootstrap for organizational modeling", is operational through the use of a semantic wiki. This tool's paradigm has a number of key advantages that allow it to function as an information repository of which we can extract several different blue prints (views) that one can elicit from an organization.

Our project is the combination of the semantic bootstrap defined in SMW and an extension for SMW that provides graphical models (diagrams) of the organization.

Semantic bootstrap for organizational modeling

The modeling of organizations with SMW was made through the definition (in SMW) of a semantic bootstrap for organizational modeling. To the user model organizations and their

business processes, needs to respect and follow this set of rules. The referred rules suffered changes during our project's development work. We made some corrections to obtain a better representation of organizations, their business processes and entities relationships.

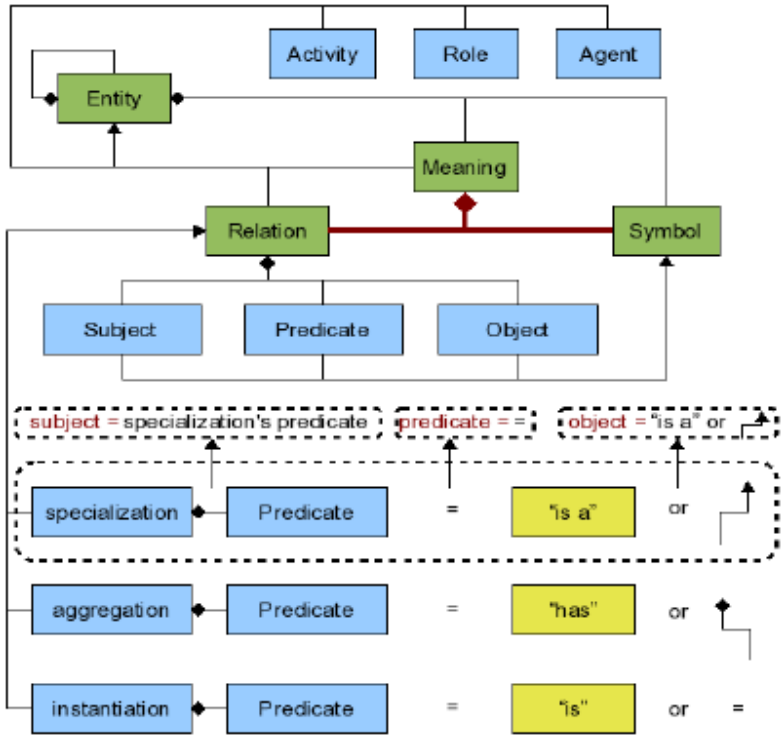

Figure 9: Semantic bootstrap

Organizational Modeling extension for SMW

This extension generates diagrams and views of the organization modeled in SMW. It was a starting point for our work, and our intention was to improve it.

We won't make an extended description of the tool, concentrating only in the main features that can be called sub components of the extension: Semantic searches and organizational views, detection of invalid relations, and activity diagrams generation.

Semantic search engine

An organization isn't a simple system. Its representation involves many business objects, even to little dimension organizations. The views facilitate the user navigation through

organization information. However don't substitute a powerful search engine. A semantic search engine, besides having the capabilities of a textual search engine, also has the possibility of searching semantic or structured contents. This search engine can find relations between various organization entities. Therefore searches are much more accurate.

Semantic searches and organizational views

A very important advantage of using a semantic wiki for organizational modeling, adapted to support their semantic bootstrap, lies in the capacity of showing rich information in an automatic way, thanks to semantic searches. To illustrate this we now use an example from the prototype, a page used to model the activity Cook an omelette. The content of this page is: "*Cook an omelette is an [[Is_a::Activity]] that can be decomposed into three activities: [[Has_activity::Beat eggs]], [[Has_activity::Heat fat in cookware]] and [[Has_activity::Fry eggs]]. It starts with the activity [[Begins::Beat eggs]]*".

They created a template that is showed automatically in every page of the wiki, which presented three organizational views: Structural, Business and Functional. Figure 3 is a screenshot of the section of the wiki page which shows these views. Each view and/or perspective utilizes one or more calls to a semantic search function, that, based on the current page (entity) being viewed returns, in a structured way, a set of pages (entities) directly or indirectly related to the current. They implemented a nested search mechanism that allows to search for a list of entities directly related to the current entity with a certain predicate (e.g. all entities that are inputs of sub-activities of current activity) and for each element of this list to do a simple search (e.g. activity that has as input that element) or another nested search.

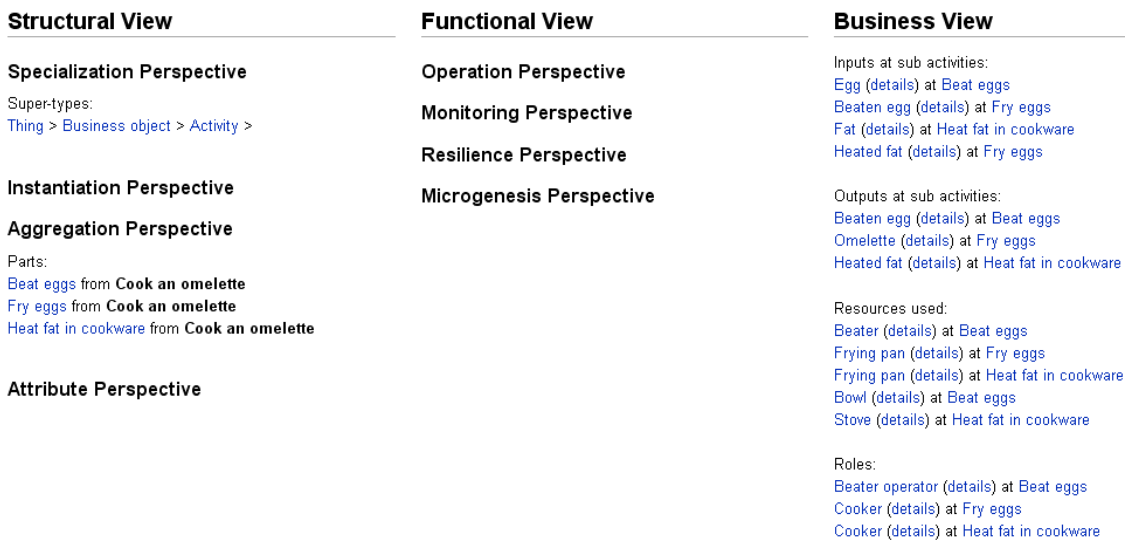

Figure 10: Organizational views template

It's easy to create relations and entities to model organizational features and also it's possible to change and/or create additional views by changing or adding new semantic searches to the template that is showed on every wiki page.

Invalid relations detection

The extension template also calls a special function that validates all semantic links present in a page being viewed. Basically the function checks, for all links present, which are the allowed types for subject and object of the respective relations and checks if the existing links respect such restrictions. For example if one would create the semantic link, in page *Cooker* [[plays::egg]] an error would be shown, indicating that by using predicate plays in entity *Cooker*, the object must be an Activity (like Fry) and not an entity (in this case, egg). Other kinds of restrictions can be defined at Relation classes and be enforced by similar validation functions.

Model as a graph and view as a sub graph

The organization model in a wiki can be considered as a graph where the nodes are pages and edges are instances of semantic relations (created whenever a semantic link is created). By using semantic searches (simple or nested) we can easily render any organizational

view (or architectural view) of an enterprise architecture, by specifying a number of parameters such as (1) type of entity that activates the search (2) and relevant predicate or predicates for rendering the desired view. A view is nothing but a sub-graph or semantic cut of the full graph that constitutes the organization, in other words, it's a projection (simpler sub-graph) of the full graph, defined by a set of predicates. By integrating software Graphviz with Semantic MediaWiki we are able to automatically generate organizational diagrams that illustrate the result of the semantic search or view that one wants to see. The diagram in Figure 15 was automatically generated on the fly, in the page containing the activity *Cook an omolette*. The diagram is built by parsing the graph that results of a semantic search that in its turn parsed through semantic links present in active page and semantic links present on other pages referred by the above mentioned links. The diagram is an SVG file and each element is clickable, containing a link to its respective wiki page, which greatly enhances ease of navigation and exploration of the models.

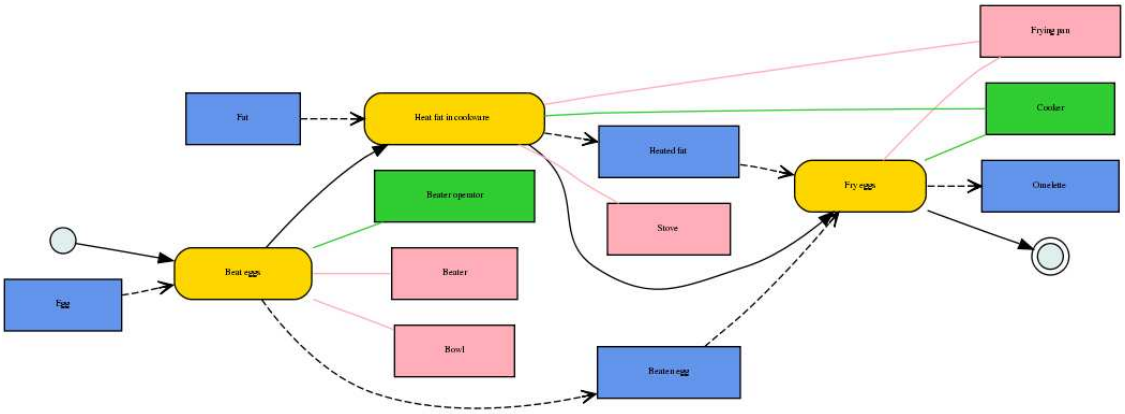

Figure 11: Activity diagram automatically generated on the fly in a wiki page

In terms of diagrams, this extension only drew activity diagrams, so as we will see more exhaustively in chapter 5 – Implementation, part of our work was to improve this extension with the creation of new types of diagrams.

2.4.4 Graphviz - Graph Visualization Software

Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. Automatic graph drawing has many important applications in software engineering, database and web design, networking, and in visual interfaces for many other domains.

Graphviz is open source graph visualization software. It has several main graph layout programs. It also has web and interactive graphical interfaces, and auxiliary tools, libraries, and language bindings.

The Graphviz layout programs take descriptions of graphs in a simple text language, and make diagrams in several useful formats such as images and SVG for web pages, Postscript for inclusion in PDF or other documents; or display in an interactive graph browser. (Graphviz also supports GXL, an XML dialect.)

Graphviz has many useful features for concrete diagrams, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes.

In practice, graphs are usually generated from external data sources, but they can also be created and edited manually, either as raw text files or within a graphical editor. (Graphviz was not intended to be a Visio replacement, so it’s probably frustrating to try to use it that way.)

Programs

Graphviz is a set of programs, dot is the most popular mainly because draws many different shapes permitting the generation of several types of diagrams.

- dot - makes “hierarchical” or layered drawings of directed graphs. The layout algorithm aims edges in the same direction (top to bottom, or left to right) and then attempts to avoid edge crossings and reduce edge length
- neato and fdp - make “spring model” layouts. neato uses the Kamada-Kawai algorithm, which is equivalent to statistical multi-dimensional scaling. fdp implements the Fruchterman-Reingold heuristic including a multi-grid solver that handles larger graphs and clustered undirected graphs.

- twopi - radial layout, after Graham Wills 97. The nodes are placed on concentric circles depending their distance from a given root node.
- circo - circular layout, after Six and Tollis 99, Kauffman and Wiese 02. This is suitable for certain diagrams of multiple cyclic structures such as certain telecommunications networks.

During our project development we used neato for Entity-Relationship models and dot for the other diagrams. [14]

2.4.5 Initial Architecture

Figure 12 shows the used applications and how they interact to each other, this was the initial architecture of the project. MediaWiki software with SMW extension installed creates a Semantic Wiki. The Organizational Modeling extension (at this point only generated activity diagrams) provided business diagrams generation using Graphviz to draw them.

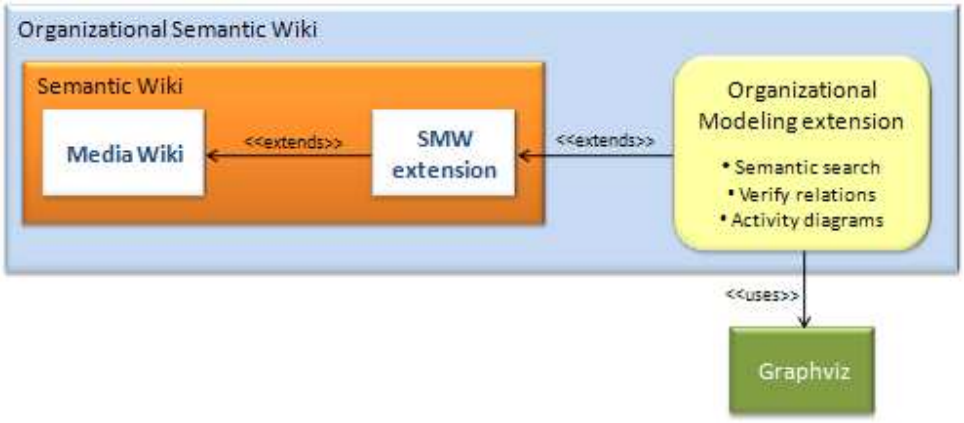

Figure 12: Used applications architecture

2.5 Problems definition

If we look only at MediaWiki it's very good tool to keep knowledge. SMW extension adds semantic relations to wiki entities improving searches and navigation. The specification of relations between wiki entities provides a better understanding of the entities themselves

and their context. Although it also has problems, especially if we use it for specific area like we did for Organizational modeling, Next we describe the problems we identified in the beginning of the project and that we worked to solve.

Possible misinterpretation due to limited formalization of meanings

There is no way to add or specify meaning to words or concepts, that are in wiki text but don't have an associated wiki page describing it, therefore without relations to other pages. This lack of relations or any type of data about the words increases the necessity to add information to them. A limited formalization of content could lead to misunderstanding of some words, or at least make the user doubtful about some words meaning.

Limited model visualization

Another MediaWiki strong limitation is that information is displayed only with text. SMW helps implementing a fact box with a summary of each page relations, but this is also textual. An essential component in modeling is the presentation of information in a graphical way, in the form of diagrams. Organizational Modeling extension already has a prototype of activity diagrams automatic generation from wiki's pages, but only one type of diagrams isn't enough. Many facets of an organization need specific engineering diagram types.

2.6 Research strategy

Based on the problems identified we define the research strategy. A solution for the limited formalization should easily add information to prevent inconsistencies in the specification and interpretation of used expressions, permitting the most rigorous possible interpretation of concepts present on wiki. A simple presentation of the saved information is also a requirement.

In model visualization the challenge was to develop different types of diagrams to help assimilation of organizational knowledge by the users. Using the semantic web as source, these diagrams should be automatically generated, without any special intervention by the user.

Once introduced the project, its context explained, applications used, the problems identified and a research strategy defined; next chapter reviews applications or projects related to ours.

CHAPTER 3 – RELATED WORK

In this chapter we study what already has been done in the project area. We didn’t find many projects directly related with semantics to model organizations, therefore we had to generalize the search. We start with a review of tools related to search of meaning. Still in same area but already related with semantics we review an application that not only searches but also adds and presents specific meaning to knowledge elements. Next is a study about the use of semantics to model business processes, and, in the end of this chapter we review some automatic bottom-up diagram creation tools. For each of these projects we present what we consider to be limitations taking in account our research aims.

3.1 Dictionary Tooltip extension for Firefox

Related to the formalization of content, this was the first application found that had similarities to what we intended, especially the search for meanings and its presentation to user.

The Dictionary Tooltip Firefox extension shows the meaning of the selected phrase in a tooltip on the same webpage. Main idea is "not" to open a new tab or window for looking definitions when you are seriously reading an article. It's a great multi-lingual learning tool.

This extension provides various sources to search within (dictionaries, wikis, translators, Google). The user also can save notes to his searched phrases.

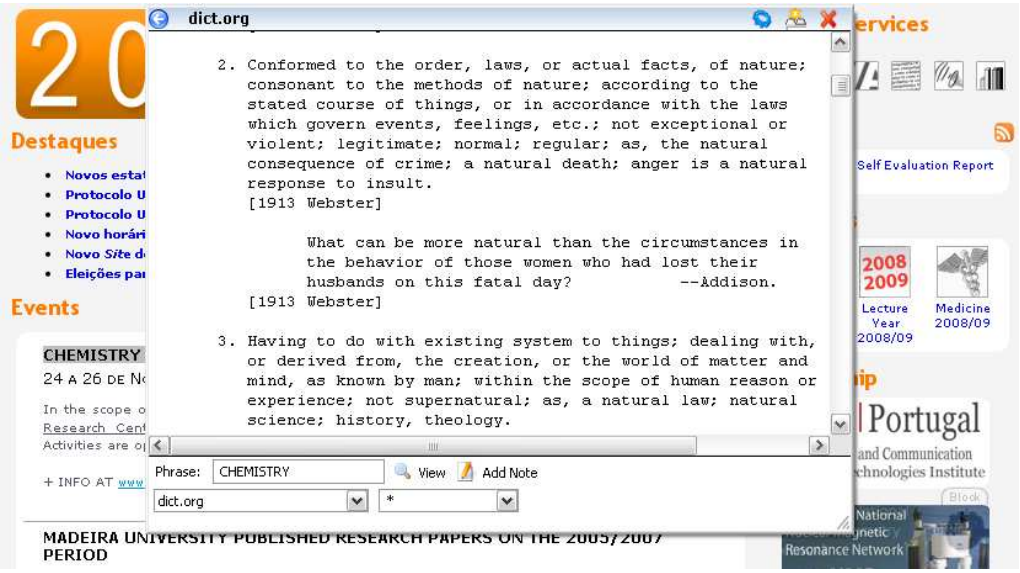

Figure 13: Dictionary Tooltip in action

Limitations

The major limitation related to want we want to develop is that this tool don't provide a way to save the information found. It's a good solution to find definitions or data related to a word, but it can't save the definition found and the user think suits better to that specific phrase. Other limitation is it only works in Firefox browser.

3.2 Inline Google Definitions extension for Firefox

Another Firefox extension, although similar to the previous one, Google inline definitions is simpler and more objective. Only returns definitions of a word or concept. The user doesn't need to choose from various sources, it returns all the definitions its search motor finds.

With the previous extension studied the user chooses a source to search within, this way you may need to search in various sources (one each time) before find what you want. Google inline definitions don't have sources to choose from, it already provides results from several sources, so it's easier to found the definition searched.

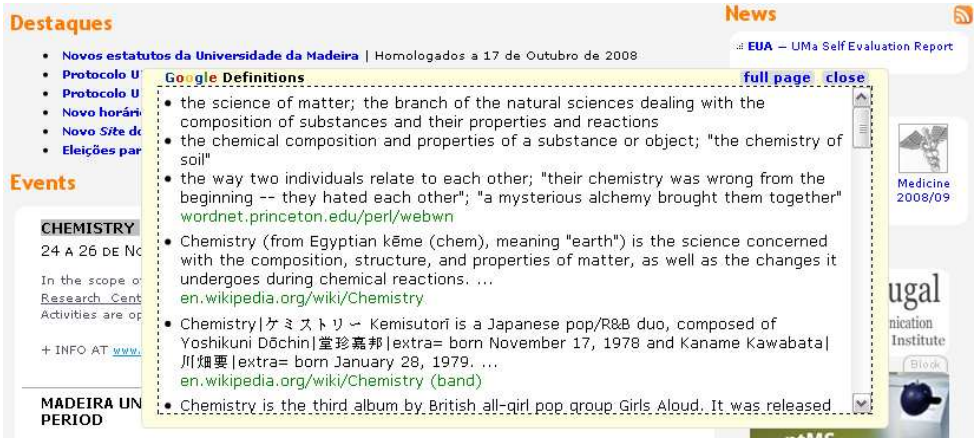

Figure 14: Inline Google definitions

Limitations

Limitations are the same as the Dictionary Tooltip. The user can't save data returned from searches and the tool it's only supported by Firefox browser.

3.3 OntoLing

Another tool related with the limited formalization of meanings problem, defined in the previous chapter. This tool is much more complete than the previous because it permits to store searched information, therefore is an application much similar to what we intend.

The OntoLing Tab is a Protégé plug-in that allows for Linguistic Enrichment of Ontologies. It features functionalities for:

- Browsing linguistic resources (thesauri, dictionaries, WordNets...)
- Linguistically enriching ontologies with elements from these linguistic resources
    - Automatic Linguistic Enrichment of Ontologies (user is prompted with suggestions on how to perform enrichment)
- Building new ontologies, starting from existing linguistic resources

Access to any linguistic resource (LR) may be obtained through implementation of a proper wrapper, called Linguistic Interface.

Currently, two Linguistic Interfaces, being related to freely available linguistic resources, have been made available on this site.

- An interface for WordNet, based on JWNL (Java WordNet Library).
- An interface for DICT dictionaries, based on JavaDICT. [15]

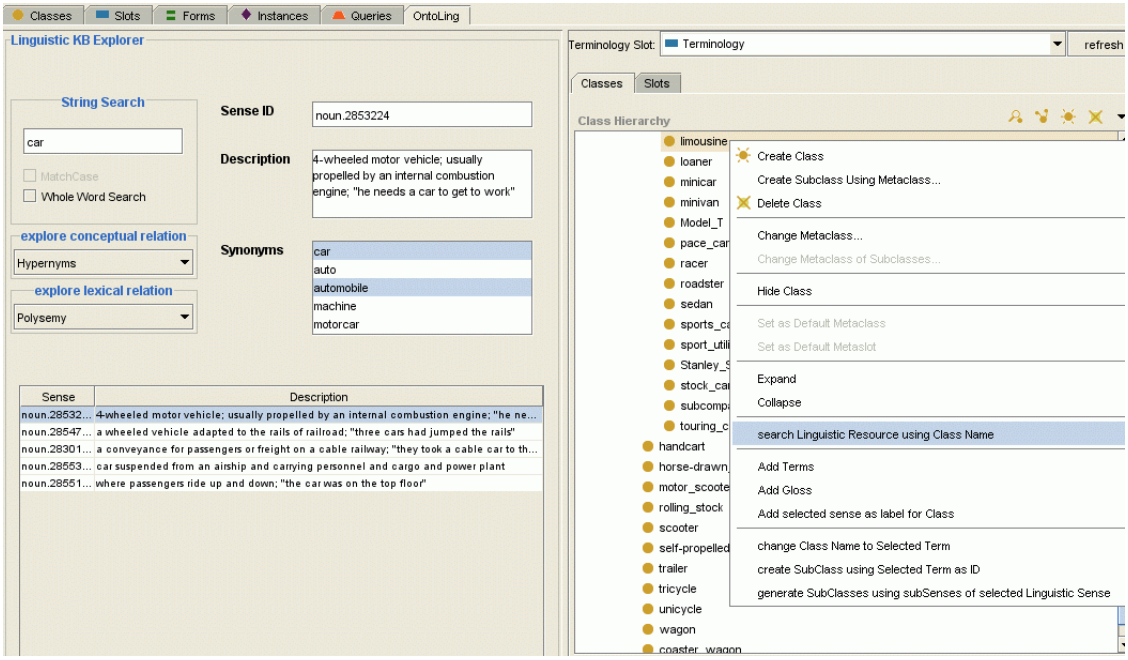

Figure 15: OntoLing

Limitations

OntoLing permits to enrich ontologies through automatic find of descriptions for ontologies elements. The big limitation for what we intend is that it only works as an extension for Protégé software, and we need to enrich the ontology defined in our platform (semantic MediaWiki). We need a solution that can be integrated in our semantic wiki. Another feature we want and that OntoLing doesn't support is a way to give meaning to words present in ontology elements description, but not defined in the ontology.

3.4 Semantic Reference - and Business Process Modeling enables an Automatic Synthesis

Related to the problem of Limited model visualization we found very few projects that used semantic web and automatically generated diagrams of the web. Although having a different primary objective, the Semantic Reference project was the one with more similarities to the work that we intended to develop. The following sections contain excerpts from [16], the paper found about this project.

3.4.1 Introduction

To enterprises today, Business processes are very important to maintain their competiveness and they invest huge efforts to describe and standardize them. Business processes are either notated only on a textual basis or graphically with models. During the last decades several graphical model standards emerged like event-driven process chains (EPC) or the Unified Modeling Language (UML2) which is more established in computer science. In particular, the UML 2.0 standard with its extended activity diagrams supports an elegant modeling of business processes.

Reference process models are the basis for many companies to develop their own business processes. Currently lots of reference processes are available, but each one uses a different language (EPC, UML, OMT, IDEF0, etc.). Additionally, it's very difficult to find a reference process which is applicable for the scope of the business area used in a company. Therefore, these reference processes should be annotated with semantic information to increase their usability and re-use.

3.4.2 Semantic Modeling of Reference and Business Processes

The semantic web service standards provide descriptions of what a service does and how it interacts with others. These descriptions can be applied to business processes and used to describe their choreography.

Each semantic web service standard has advantages when being used to annotate business processes. We are interested in a general approach for the automated synthesis of reference processes, i.e. pre-defined business processes to be customized and combined to obtain value-added business processes. In their approach they decided to combine concepts and

advantages of semantic web standards which describe web services: OWL-S, WSMO, SWSF and METEOR-S.

The ontology used to define all concepts of the company and corresponding business processes will be called data semantics and provides a basis for the modeling of processes. It defines all concepts and their relationships that describe the enterprise, the departments and their tasks which are required to annotate BPM and RPM. Additionally, (global) variables can be defined to be used in preconditions and effects for describing the change to a global state, e.g. changes in a database, etc.

3.4.3 Synthesis of Semantic Process Models

The synthesis works incremental: first, the functional semantics of each process is compared with the functional semantics of all other processes. The results are stored in a matrix that serves as source to compute bottom-up optimal composition of all processes would look like. They developed two different synthesis algorithms (Modified Prim and RandoMediaWikialk).

3.4.4 Case Study

To test the semantic process modeling and to compare the different synthesis operations, a prototypical tool was developed. It offers the modeling of a semantically-enriched UML2 activity diagram and testing the synthesis with the operations explained above. Let's assume a purchase process where a customer buys a product which has to be adapted to his needs.

A part of the semantics for this example would be the following: Figure 15 shows a part of the data semantics for one of these processes ("Order product"). The functional semantics for the same process would include (informal):

- *Precondition: ProductInStock hasValue FALSE,*
- *Output: Order,*
- *Effect: ProductOrdered setValue TRUE.*

```
<semBPM:Object rdf:ID="Order"/>
<semBPM:SemDataProperty rdf:ID="executedFor">
        <rdfs:domain rdf:resource="#Order"/>
        <rdfs:range rdf:resource="#Customer"/>
</semBPM:SemDataProperty>
<semBPM:globalVariable rdf:ID="ProductInStock"/>
```

Figure 15: Data semantics for the process "Order product"

Having modeled all processes, annotated them with semantics and started the synthesis, one gets the result of Figure 16. Both algorithms achieve the same result and the solution the user might have probably expected. If one of these processes changes or needs to be deleted, one can simply start a new synthesis to get the new optimal combination and no further action is required.

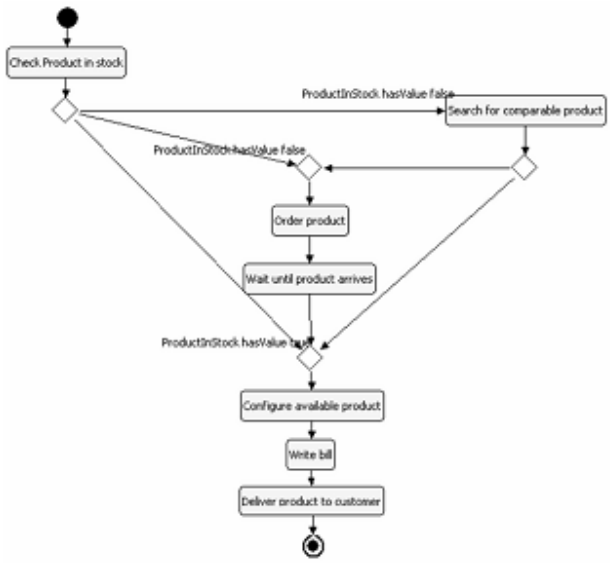

Figure 16: Result of the synthesis of our business trip example

This prototype only offers the modeling of UML2 activity diagrams; other diagrams are currently not supported. They are aiming to develop a methodology to annotate reference processes, whereas they will consider current approaches to model business processes like BPMN and approaches of business process ontologies.

3.4.5 Limitations

Semantic reference project defines the business processes using RDF which isn't a trivial language for every collaborator of an organization. One of our primary objectives is having a tool easy to use by people without much software knowledge, making possible to all the collaborators of the organization can contribute to the creation of an organizational awareness.

Other limitation is the diagram types, only activity diagrams are generated, we want to provide various types of diagrams to increase the understanding of organizations, its business processes, and entities relationships.

The diagrams generated in this project are not interactive, not allowing dynamic navigation, the possibility to navigate between diagrams and organization entities.

3.5 Automatic diagram generation applications

To solve the limited model visualization, the creation of automatic diagrams was required. This section is a selection of the most interesting tools from the many auto diagramming applications found.

Visustin v5 Flow chart generator: open up your code in this tool and it automatically creates flow charts and UML Activity Diagrams. Unfortunately it doesn't work in the reverse order, if you edit a diagram it can't create the correspondent code. Visustin supports thirty one popular programming languages.

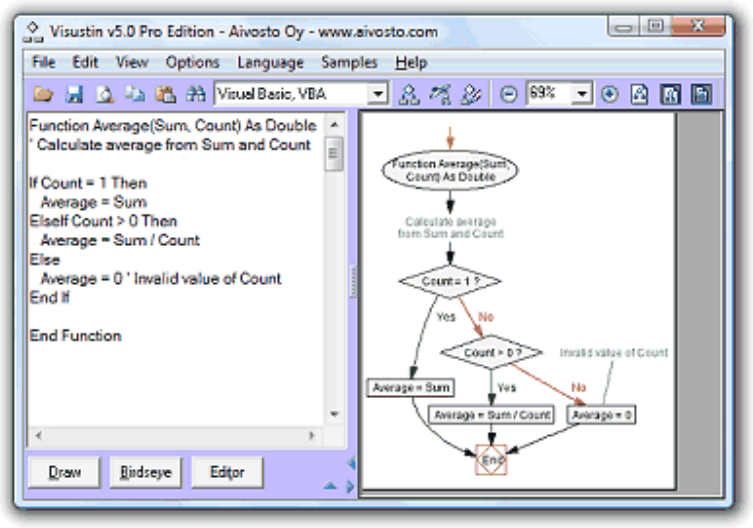

Figure 17: Visustin v5 Flow chart generator

PostgreSQL Autodoc: This is a utility which will run through PostgreSQL system tables and return HTML, Dot, Dia and DocBook XML which describes the database.

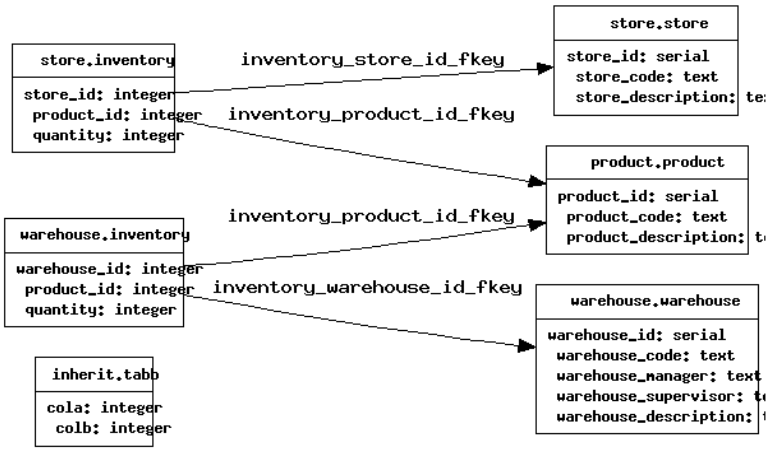

Figure 18: PostgreSQL Autodoc - Graphviz output

Linguine Maps: Linguine Maps is an open-source Java library that conducts programmatic visualization of various text files, generating from them easy-to-understand entity-relation diagrams. With a diagram it will take you and your team minutes now, instead of perhaps hours, to get familiar with new schema, object-relational mappings, or DTDs. And you can always go back to the source files when more details are needed.

All diagrams produced by the Linguine Maps are precise reflection of the source code. There is absolutely no manual work, it's fully automatic.

This tool supports programmatic visualization for: WSDL, Apache ANT build files, Document Type Definition (DTD) for XML documents, Apache ObJectRelationBridge (OJB) mapping files and Hibernate mapping files. [17]

Ragel State Machine Compiler: Ragel compiles executable finite state machines from regular languages. Ragel targets C, C++, Objective-C, D, Java and Ruby.

The core language consists of standard regular expression operators (such as *union*, *concatenation* and *Kleene star*) and action embedding operators. The user's regular expressions are compiled to a deterministic state machine and the embedded actions are associated with the transitions of the machine. Understanding the formal relationship between regular expressions and deterministic finite automata is key to using Ragel effectively.

Ragel also provides operators that let you control any non-determinism that you create, construct scanners, and build state machines using a statechart model. It's also possible to influence the execution of a state machine from inside an embedded action by jumping or calling to other parts of the machine, or reprocessing input. [18]

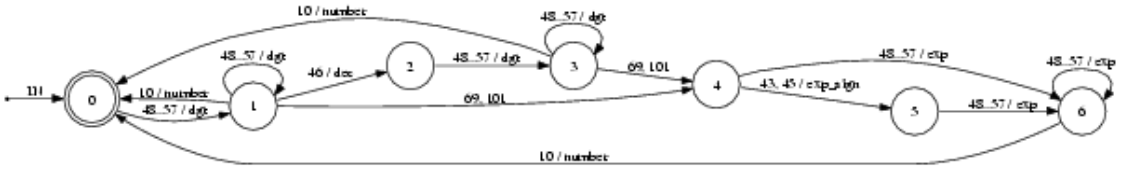

Figure 19: Ragel State Machine Compiler

OntoViz: a Protege extension, allows visualizing ontologies with AT&T's highly sophisticated Graphviz visualization software.

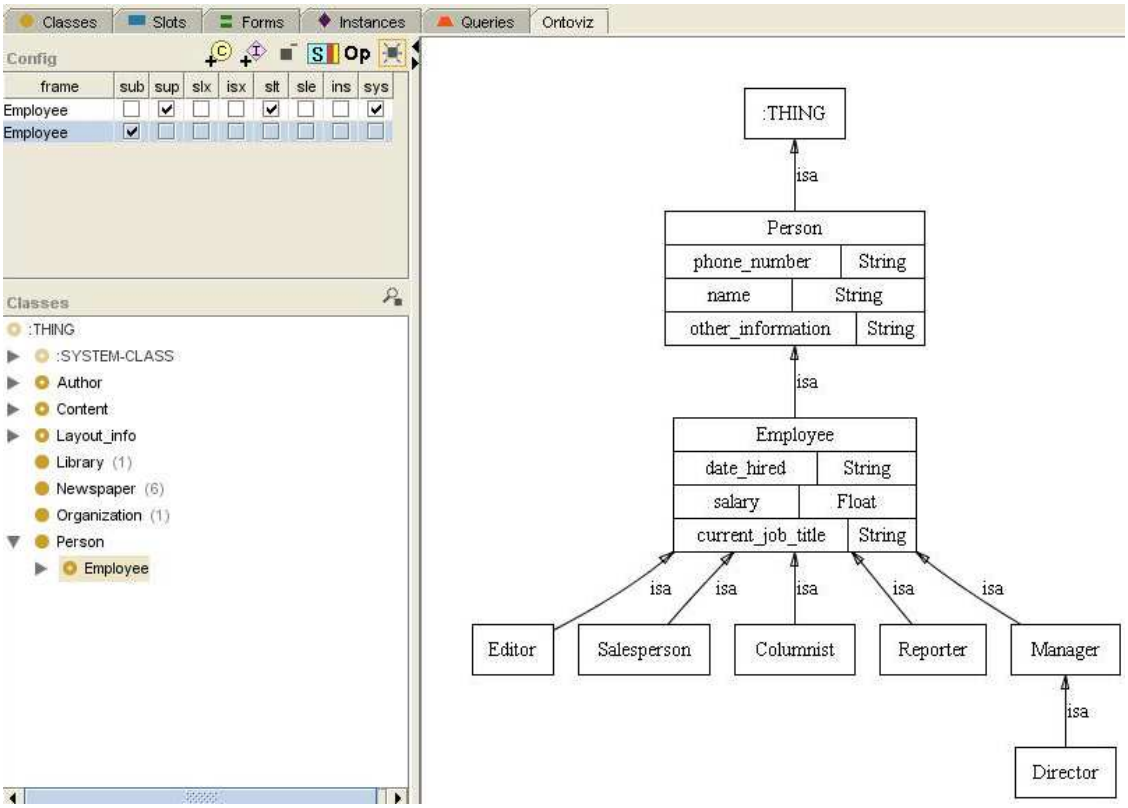

Figure 20: OntoViz

IsaViz: A Visual Authoring Tool for RDF

Is a visual environment for browsing and authoring RDF models represented as graphs.

Some features:

- creation and editing of graphs by drawing ellipses, boxes and arcs
- RDF/XML, Notation 3 and N-Triple import
- RDF/XML, Notation 3 and N-Triple export, but also SVG and PNG export

IsaViz can render RDF graphs using GSS (Graph Stylesheets), a stylesheet language derived from CSS and SVG for styling RDF models represented as node-link diagrams.

3.5.1 Limitations

Each of the applications reviewed specialize in one type of diagrams, we aim to provide various types of diagram in our application. Also these tools don't provide specific support to model organizations and provide views of their business processes and entities.

3.5.2 Conclusion

Our research revealed that the vast majority of tools for automatic generation of diagrams use Graphviz to design their output diagrams. This confirms that Graphviz is really popular and a powerful open-source software which justifies it as a good choice for the same function in our project.

Summarizing this chapter, we came to know of applications related to the project we developed: information enrichment, to semantic business processes modeling, and automatic diagram generation tools. Limitations of each of them, concerning our purposes were stated. Next chapter presents the solutions we envisioned to solve the problems.

CHAPTER 4 – SOLUTIONS AND CONTRIBUTIONS

With project context acknowledged, problems identified and related applications studied, we now explore the solutions for the problems we elicited. The chapter starts with a review of problems and objectives which the leads us to the solutions. First is the solution for formalization of meanings, which was achieved taking advantage of other applications. The second part of the project was developing a solution to improve model visualization in a semantic wiki; this was accomplished implementing a bottom-up tool that automatically converts textual semantic data to engineering diagrams.

4.1 Problems and objectives review

Before the solutions, at this point it's essential to remind the problems defined earlier in Chapter 2, and our project's objectives.

Possible misinterpretation due to limited formalization of meanings

There was no way to add or specify meaning to words (expressions), that are in wiki text but don't have an associated wiki page describing it, and defining what that word represents in the context. This limited formalization of content could lead to misunderstanding of some words, or at least make the user doubtful about some words meaning. Naturally the referred words don't have semantic relations with other elements; this situation and the complete non information about these words, makes even more important the need to add some meaning to them.

Limited model visualization

A MediaWiki big limitation is that information is always displayed in a textual manner. Semantic MediaWiki extension implements a fact box with a summary of each page relations, but this is also textual. An essential component in modeling is the presentation of information in a graphical way, in the form of diagrams. Organizational Modeling extension already has a prototype of activity diagrams automatic generation from wiki's

pages, but only one type of diagrams isn't enough. Many facets of an organization need specific engineering diagram types.

Objectives

We had as starting point the MediaWiki application and its extensions: Semantic MediaWiki and Organizational Modeling extension. From this base we had our main objectives traced.

The objective for the formalization of content is to easily add information to prevent inconsistencies in the specification and interpretation of used words, permitting the most rigorous possible interpretation of concepts present on wiki. A good presentation of the information added is essential.

In model visualization the objective was to develop different types of diagrams to help the user's knowledge assimilation. The important feature is that the source data used to generate the diagrams is a semantic web that models an organization. These diagrams should be generated automatically and without any special intervention by the user.

4.2 Formalization of meanings

Before the explanation of solution, a brief description of the applications we used to achieve it follows. They were important because it was through the combination of them that we produced the final result. Next on this section is the solution and how we reached it. Finally we compare our solution with related work studied earlier.

4.2.1 WordNet

We had the idea of linking the semantic wiki to some type of dictionary, which should provide words and correspondent possible definitions, WordNet was our option.

WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the

browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing. [19]

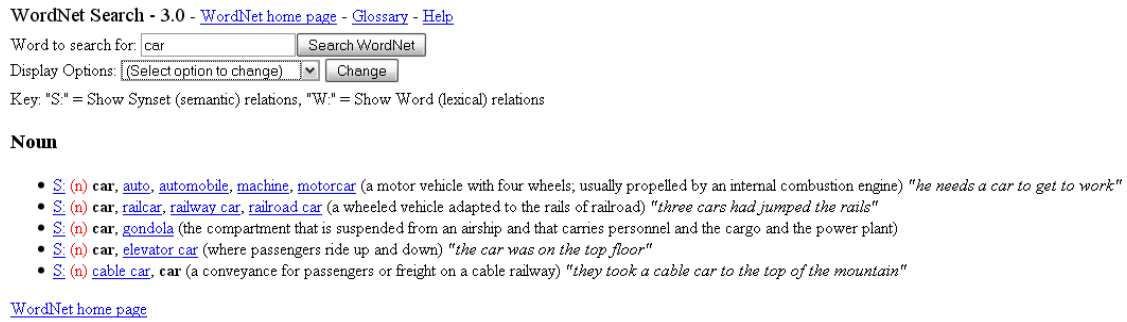

Figure 21 WordNet Web interface

4.2.2 Halo Extension

Our project's very first task was to implement an Ajax auto-complete to assist the annotation in Semantic MediaWiki. So we started looking for a solution, testing scripts similar to Google Suggestion. We didn't do more than simple suggestions because we found the Halo extension for SMW and our first objective was solved.

The SMW Project Halo Extension is an extension to the Semantic MediaWiki and has been developed as a part of Project Halo in order to facilitate the use of Semantic Wikis for a large community of users. Main focus of the developments was to create tools that increase the ease of use of SMW features and advertise the immediate benefits of semantic content.

The features of the Halo extension can be divided into four main sections:

1. **enhancing wiki navigation** - features to ease and speed up navigation and access to articles, as well as semantic data, in the wiki
2. **improving knowledge authoring** - features to allow easy and expressive addition of semantic data to the wiki
3. **simplifying knowledge retrieval** - features to query knowledge and access information stored in the wiki

4. **gardening the knowledgebase** - features that allow users to detect inconsistencies and continuously improve the quality of the authored knowledge

A demonstration video of the extension's main features is available at http://www.ontoprise.de/SMWdemo/

A demowiki with the Halo extension installed is available at http://halowiki.ontoprise.de. [20]

4.2.3 Tooltip MediaWiki extension

This extension provides the ability to add fancy tooltips to wiki text. This permits the user to save annotations of words, expressions or phrases present in wiki text. Unlike other extensions which provide similar functionality (i.e. Extension:Glossary, Extension:LinkedImage, Extension:LinkFloatie, etc), this extension allows for multi-line wiki and/or HTML syntax text for the tooltip. Additionally, the tooltip itself is displayed in a fancy semitransparent window.

4.2.4 Solution for Formalization of Meanings - WordNet integration with SMW

Although the limitations of the researched applications similar to what we aimed to develop, we got some inspiration from them (Dictionary Tooltip, Google inline definitions and OntoLing). This added with our ideas produced the successful solution.

The solution to overcome the limitations found and get a richer formalization of content objective, was to integrate some kind of dictionary with SMW. The user editing a semantic wiki page should access dictionary words definitions and add them to the words he wanted to give meaning. The add information action should be easy. An auto-complete returning a word/concept definition was one of the purposes of the solution. Finally the definition added should have a clean and easily accessible presentation.

For the dictionary, we opted to use the lexical database of English – WordNet, introduced above. WordNet also uses semantics to group related words and concepts but that wasn’t important for our objective. We just wanted to use WordNet’s words definitions to add meaning to words in wiki’s pages.

WordNet integration with SMW

First step was to access WordNet database, we found a way to store it locally. We didn't create a WordNet database, but added WordNet tables to the semantic wiki database.

To accomplish the integration we used the Halo extension for SMW as an integration layer. As Halo extended the SMW, we integrated WordNet with Halo extension and consequently achieved integration of WordNet with SMW.

The integration of WordNet with Halo was made extending Halo in order to search in WordNet database. Halo also had to be configured to trigger WordNet search and return the correspondent results, just for specific word definition case (separate from auto-complete of other SMW elements).

For the presentation of WordNet definitions feature, we adapted the Tooltip MediaWiki extension to do what we intended. It was necessary to combine this extension with Halo and WordNet part. We adapted Halo, making the bridge between previous work and the presentation of definitions. Figure 22 shows the related applications for the WordNet integration.

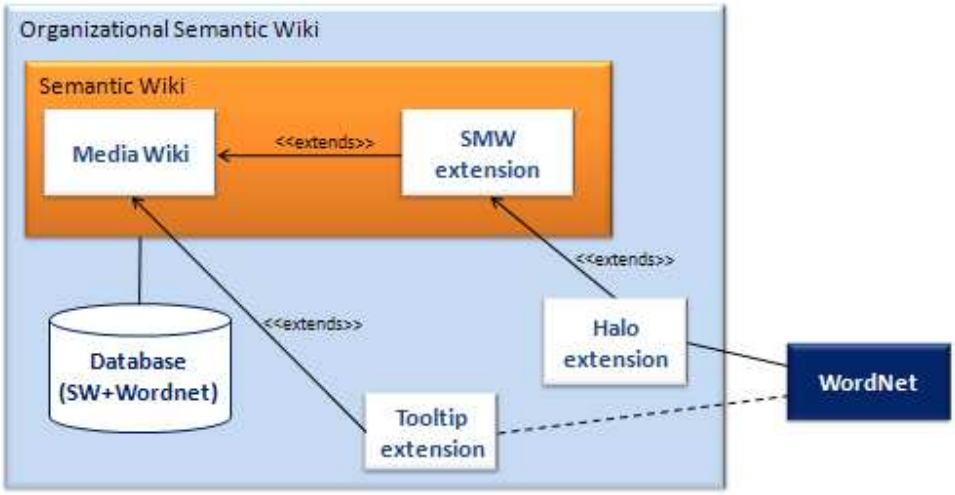

Figure 22: WordNet integration with SMW

Functionality of the solution

When the user triggers halo auto-complete for a word or concept, halo searches for the word in WordNet database, once found the matched word, its definitions are searched and returned to user, view Figure 24. The user chooses the definition he wants and after saving the page, that word appears with a different presentation in wiki text and when the mouse is over the word, the chosen definition automatically appears. In the next figure we can see the final presentation. Other defined words have a light grey background and a dashed underline.

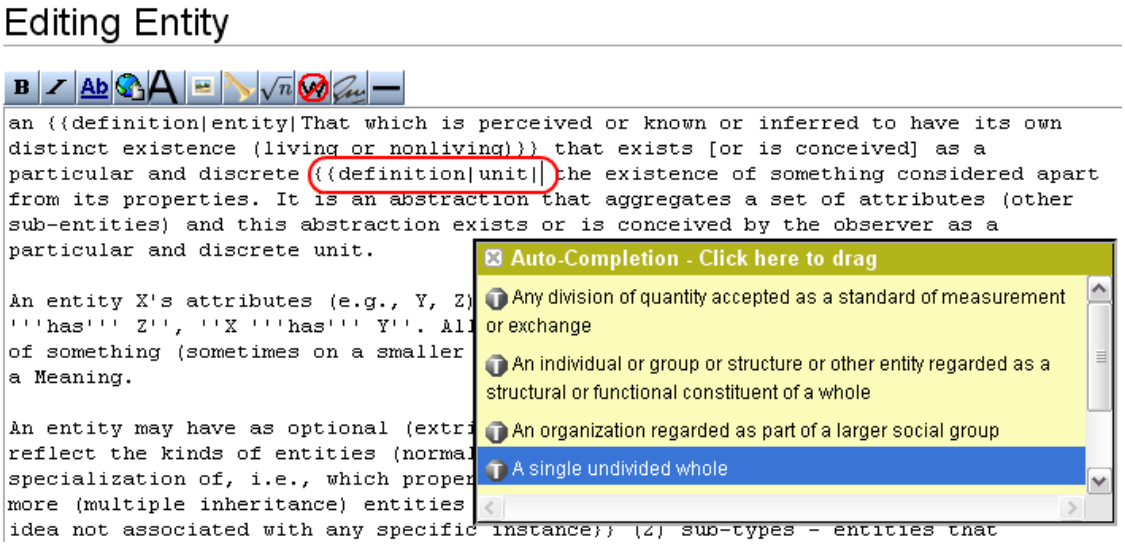

Figure 23: Auto-complete returning WordNet definitions

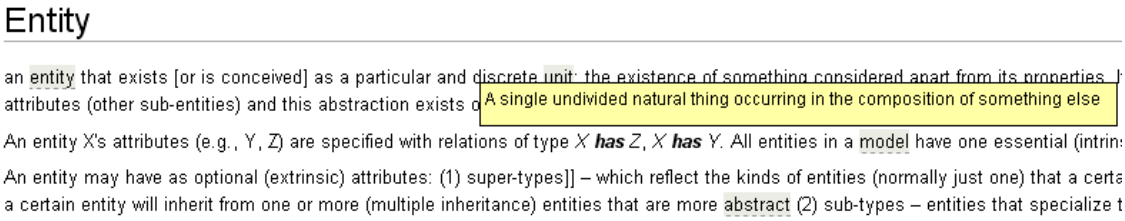

Figure 24: WordNet defined words and effect when the cursor is over a word

WordNet lexical database integrated in SMW gave us the possibility of choosing and showing meanings for words in semantic wiki pages. This functionality first objective was to work with words not defined in wiki (not semantically annotated) but present in wiki pages text. But it can also be used for words defined in wiki, giving them a WordNet

definition. In practice, its use is for most important words of the model, key words (defined or not in wiki) to understand the context and words than can have multiple meanings (giving a definition to avoid confusion).

WordNet integration with SMW is independent of Organizational Semantic wiki and respective extension Organizational Modeling. So this solution can be used in any wiki with SMW installed.

Conclusion

The final result is a tool that enriches ontologies like OntoLing, but integrated in a semantic wiki. The auto-complete feature was based in Halo extension and its data source the WordNet. A simple and accessible presentation was achieved using Tooltip extension idea, making it better.

It's common to people add notes in a non structured way to models, our wiki with semantic links between its pages (semantic relations between elements), integrated with WordNet it's a more systematic and coherent approach. The selection of WordNet definitions are simple and people can, in an organized manner, add information to a certain word present in a page. The result is increased formalization of meanings, an easier understanding of words, their meaning in the surrounding context and richer knowledge storage and presentation.

4.2.5 Comparison with related project OntoLing

Ontoling is similar to WordNet integration with SMW part of our project; it permits to enrich ontologies through automatic find of description for ontology's elements.

Earlier when studying the OntoLing in the related work chapter, we saw that one of its limitations was that it's developed to work for a specific application called Protege. Our solution was WordNet integrated in Halo extension. As Halo is a SMW extension, WordNet definitions were accessible from SMW.

As our ontology is an organization and our work platform a wiki, our solution to ontology enrichment was through an auto-complete that got definitions (from WordNet) for words present in wiki pages text. This way we granted the enrichment of the ontology.

The other OntoLing limitation mentioned earlier was not providing the enrichment of words present in ontology elements description, but not defined in the ontology. With our auto-complete solution we can also give meaning to any word present in a wiki page, enriching words not defined in the organizational ontology.

Table 1 compares differences between the two projects:

Features \ Applications	OUR	OntoLing
Enrich ontologies	X	X
Enrich words not defined in ontology	X	
Ontologies definition	SMW (simpler syntax)	RDF
Platform	MediaWiki + SMW	Protégé

Table 1: Comparison between our solution and Ontoling

4.3 Model visualization - Automatic diagram generation

Related to Limited model visualization problem, the solution was improving the Organizational Modeling extension. It only generated activity diagram with some limitations, so our aim was to improve the activity diagrams and implement new types of diagrams using Graphviz to design them. New diagram types include Entity-Relationship models, State diagrams and Use Case diagrams. All types of diagrams regularly used in Organizational Engineering area.

4.3.1 Automatic diagram generation

The Semantic wiki user introduces the organization data in wiki in the textual form, using SMW syntax studied earlier. Doing this, the organizational model takes form but the stored data is all textual.

What our bottom-up auto diagram generation tool does is convert the data (textual) introduced in SMW, to much more user friendly diagrams, producing an easier and better understanding of Organizations, their business processes, entities and how relations happen between them. By relations, we mean between elements from the same type and from different types. For example: hierarchy of employees, what activities are related to each other (same type); what activities are certain employees related to (different types).

Diagram builder algorithm

Introduced data in SMW creates a semantic graph of elements (pages). So with an Organization model defined in SMW, we already had a semantic web or graph but there wasn't a graphical way to see parts (sub-graphs) of this web. What we needed to do was, navigate through that semantic web and generate diagrams from it.

The solution was access the semantic graph by searching data stored in SMW database. To cover the SMW graph, the algorithms defined use many times the semantic search function from Organizational Modeling extension studied earlier. The principle for each diagram construction is to cover all the nodes of a graph, then for each node use the semantic search to find what nodes are related to it, and add the connections between them.

What we did was while covering the graph, as we found elements we defined them in Graphviz language, building a diagram. When the graph was complete we used Graphviz to output the final diagram images

Diagram generation methods architecture

The methods to generate different types of diagrams start by defining in what type of pages the diagrams will appear, for example activity diagrams appear in activities pages. From this point starts building the diagram in Graphviz language and have to follow its rules. The rules are the same for every type of diagram we made. First is definition of all the elements used in the diagram, it's here that the shapes of the elements are defined. Then it's the part of relations between elements, the core of each method. Here is defined the algorithm (from previous section) that, covers all the graph nodes, and connects the related nodes. To

finalize the built diagram is generated using Graphviz and the images saved in wiki database.

Unique diagram generation method

During development we implemented several generation methods, one to each type of diagram (activities, ER, state, use case). The methods architecture we just saw is common to every type of diagram. This means it should be possible to implement a unique method to generate all type of diagrams, using parameters to deal with the particularities of each diagram.

This feature would facilitate the addition of new types of diagrams but has a probable problematic part in the creation of relations. The algorithm that builds the connections between related elements is complex and varies a lot from each type of diagram. It may be possible to extract a certain pattern or set of rules that all algorithms use, and then use conditions to treat the particularities of each diagram. The problem is the number and complexity of conditions to deal with the particularities. It would be necessary many conditions and the method would be too big and complex.

Perhaps the better solution is an intermediate approach. A general method with the architecture used in diagrams generation methods and parameters to treat minor differences between diagrams (appearing pages, declare used elements, image size). Then separate methods with algorithms to build the relations part of each type of diagram.

We didn't implement this idea; it's only a contribution and an interesting feature to future work.

4.3.2 Comparison with related work

In this section we remember the limitations of work related with our project. The applications were studied in chapter 3 and now we compare them to our work, and how our solutions overcome the limitations found.

Semantic Reference

Comparing with our project, the limitation of the complex RDF language to define business processes is resolved just by the use of SMW that has a simple syntax. Modeling organizations in SMW is fairly easy; of course you need to be aware of organization architecture and its business processes. This is an advantage and one of the primary objectives, a tool easy to use by people without much software knowledge, all the collaborators of the organization can contribute to the creation of an organizational awareness.

Other big advantage is that our solution provides more types of diagrams; Semantic reference project is limited to activity diagrams. We developed automatic generated activity diagrams but also Entity-Relationship models, State diagrams and Use cases diagrams.

As for diagram navigation issue of semantic reference project, the solution was already developed by João Mendes in Organizational Modeling extension; we just had to re-use it. It works through the generation of .svg images; this type of images is clickable and we linked the graphs nodes to wiki pages through the page URL, for example, in a diagram with activity "Check registration", clicking at that activity node you are redirected to correspondent activity page. Table 2 shows the various differences between the applications.

Features \ Applications	OUR	Semantic Reference
Auto diagramming	X	X
Semantic annotation	SMW (simpler syntax)	RDF, OWL, OWL-S
Types of diagrams	4	1
Algorithms operation		
Navigable diagrams	X	

Table 2: Comparison of our project with Semantic reference

Comparison with Auto diagram generation applications

The limitations were that each application only generated one type of diagram, and the non support to model organizations and provide views of their business processes, entities and

relations. We improved the activity diagrams and implemented three new types of diagrams: Entity-relationship models, State charts and use cases diagrams. The organizational views were improved to return information related to the new diagrams. Next table compares the features of each application introduce earlier in chapter 3.

	OUR	Visustin	Linguine Maps	PostgreSQL Autodoc	Ragel State Machine	OntoViz
Automatic diagram generation	**X**	X	X	X	X	X
Types of diagrams	**4**	1	1	1	1	1
Use Semantics	**X**					X
Organizational Modeling	**X**					
Organizational views	**X**					
Navigable diagrams	**X**					

Table 3: Comparison of automatic generation diagram tools

4.4 Applications Architecture

The basic application is MediaWiki, extended with SMW to make a Semantic wiki. Then we have two MediaWiki extensions but that work as SMW extensions: Halo and Organizational Modeling, these don't work alone, their purpose is to add functionalities to SMW. WordNet connection to Semantic wiki happens through Halo. Organizational

Modeling uses Graphviz to generate the diagrams. Finally the Tooltip MediaWiki extension was installed and adapted to use in WordNet definitions presentation.

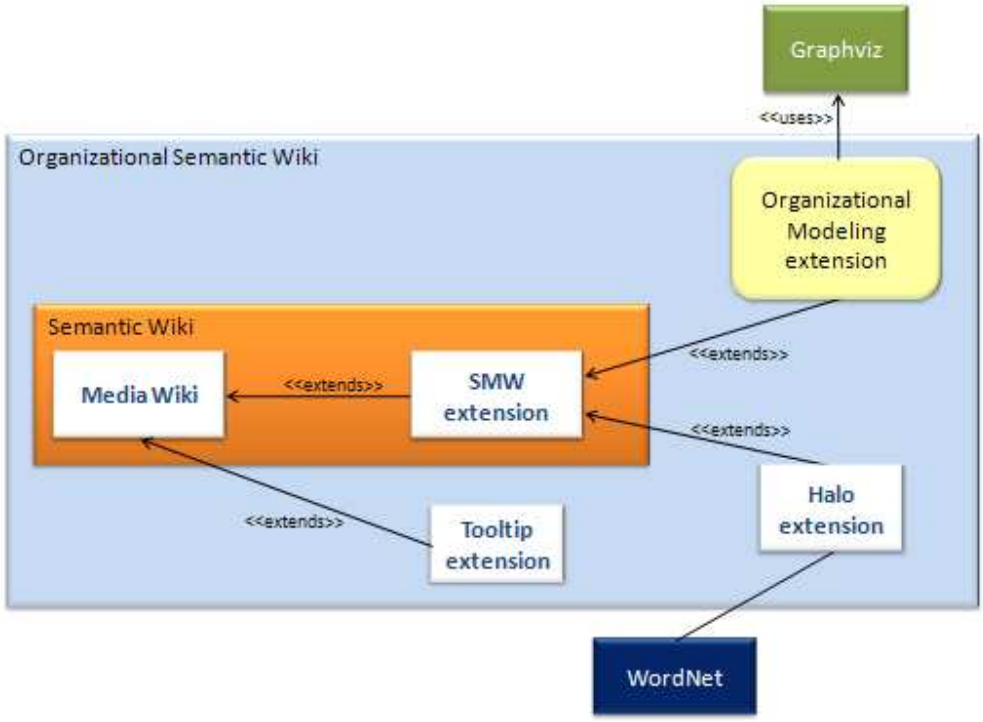

Figure 25: Project's Applications architecture

4.5 Organizational Semantic wiki concept hierarchy

We made some adjustments from the previous project wiki hierarchy. Thing it's the most general class, it permits the addition of any new class as it specialization. Relations happen between entities. All other elements of wiki are entities that related to each other, the Model entity has two specializations: Entity-Relationship Model and Use Case Diagram. The activity and State diagrams aren't defined as models because these are drawn at the respective class wiki pages.

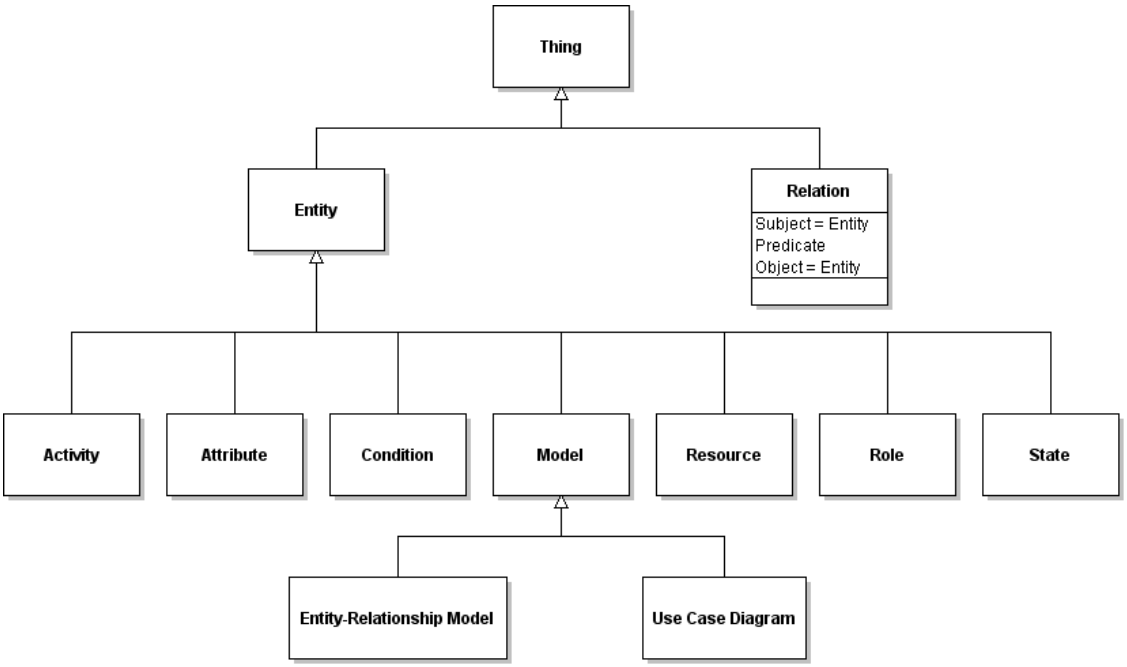

Figure 26: Organizational Semantic wiki concept hierarchy

4.6 Additional features

During our work with the semantic wiki we envisioned some others functionalities useful to improve it.

Rename for SMW

MediaWiki has a native feature to rename its pages, but called *move* because what it really does is move the page content to a new page with different name but and same content (text). The problem is that with SMW if you move (rename) a page you lose all its relations with other pages because the references (in other pages text) to the page stay with the old page name.

What SMW needed was a rename that updates all the instances of the renamed page in order to maintain the semantic relations of that page. This is a really useful feature that should be part of default SMW extension, so we decided to implement.

The solution was to make a simple interface for the rename, then scan the database renaming the page and all of its semantic relations instances presents in wiki pages text, whether it's a normal page, a Property or a Template. The rename is made in all pages versions that contain the renamed page name, safeguarding the case of an undo to an old version.

Semantic Organizational Modeling wiki for Linux / Windows

Other must have feature was the Organizational Modeling extension compatibility with Windows and Linux operating systems. The solution was to prepare the Organizational Modeling extension to recognize the server system and run the correspondent commands. This permitted compatibility with both operating systems without any user configurations.

At this point, with the solutions explained and clarified, the next chapter specifies how their implementation was made and all the important steps to reach the final result.

CHAPTER 5 – IMPLEMENTATION

This chapter is to explain in detail the implemented work, the modifications made and where the work was done. It‘s intended to explain project’s details, and especially to help those who will work with the project in the future, clearing up what was done and reporting the problems encountered.

Progressing to work done, first part was mostly research and understanding of the project context, and then we had two major topics to work around: the WordNet integration with Semantic MediaWiki, and the creation of new types of diagrams to model Organizations.

We begin the chapter with an explanation about the steps to achieve WordNet integration with SMW. Then a part not directly related with the objectives but necessary: Semantic Wiki upgrade, where the major problem was with Organizational Modeling extension adaptation to new wiki. Next section is about the automatic graph parser and diagram builder, here we detail the work developed to create each type of diagram. The explanation of each diagram also works as a user manual to make diagrams in wiki. The two additional features we found valuable and decided to implement are the following theme. In the end, some of the problems found are described.

Summary and activity diagram of the developed work

- WordNet integration with SMW
- Implementation of conditions in activity diagrams
- Semantic Wiki upgrade
- ER, State and use case diagrams for Organizational Modeling extension
- Rename for SMW
- Organizational Modeling extension for Windows and Linux

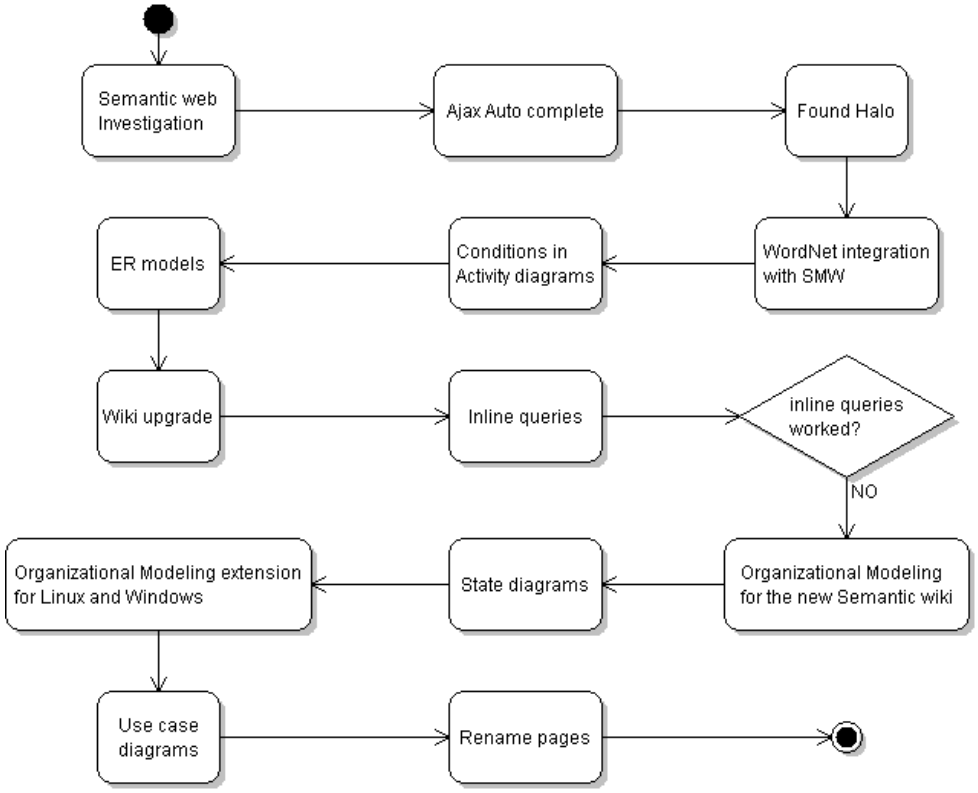

Figure 27: Developed work Activity diagram

5.1 WordNet integration with Semantic MediaWiki

The work related to WordNet can be divided in two parts: user interaction, and search for definitions at WordNet database.

The initial thought was to connect WordNet with SMW in real time connection but we didn't find way to do it because WordNet didn't provide its database online, also we didn't find other way to do it. The solution came with a ready-to-use WordNet MySQL database (the same database type MediaWiki uses) project compiled by Bernard Bou. This way we could store the WordNet database locally.

Description of the project:

- [MySQL, PostgreSQL] A ready-to-use SQL database that unifies WordNet 3.0, WordNet 2.0-2.1, 2.1-3.0, 2.0-3.0 sensemaps, VerbNet 2.1, XWordNet 1.1 compiled by Bernard Bou supports both MySQL and PostgreSQL.

The project provided SQL scripts that we had to modify (removed drop commands and foreign keys) in order to populate the database correctly. We didn't create a new database for WordNet, we added WordNet tables to our semantic wiki database.

5.1.1 User interaction

This part can also be divided, in auto-complete for when editing a page text and final presentation of the words definition in wiki pages.

Auto-complete

The first attempt of reading the WordNet database with a simple auto-complete made were unsuccessful. We pondered search an alternative dictionary but meanwhile we and had the idea of taking advantage of Halo auto-complete feature to use with WordNet.

After a study and exploration of Halo extension code we knew where to extend halo to return WordNet definitions. We needed to define the auto-complete interface part before extend halo and test the returned definitions. The logical next action was how to provide the user an auto-complete for WordNet word definitions. It had to have a syntax different from SMW ([[property::object]]), so the first idea was: in wiki page editing, after the introduction of a word followed by the "=" symbol, the user presses ctrl+alt+space (Firefox) or crtl+space (IE) to trigger the auto-complete. For example [[word=definition]], we used "=" because if we used "::" it would be like a wiki property.

Presentation of WordNet definitions in wiki

The previous solution worked for the auto-complete part but wasn't presentable because it looked the same "[[word=definition]]" in the page text.

The solution was the use of another extension: Tooltip MediaWiki extension that allowed the use of Halo because it uses a template in the wiki (halo also has auto-complete for

templates).

- This offered the possibility to use a template, with this we could have the halo auto-complete and a good presentation solution in the article text
- Creating the template:
  - Create a page with the name: “template: definition”
  - Paste the following text in it:
    - <includeonly>{{#definition: {{{1}}}|{{{2}}}}}</includeonly><noinclude>Usage: {{definition | word | word definition}}</noinclude>

We modified the tooltip extension to use the word “definition” instead of the default “tooltip”. The normal tooltip extension works simultaneously with the one we defined, but with different presentation.

Next step was extending halo to integrate it with WordNet and display its definitions. This way we integrated WordNet with SMW because halo extended SMW.

5.1.2 Search WordNet words definitions

For what we needed was necessary to use data from three tables of the WordNet database, the principal tables (word, sense, synset), light yellow colored in next figure:

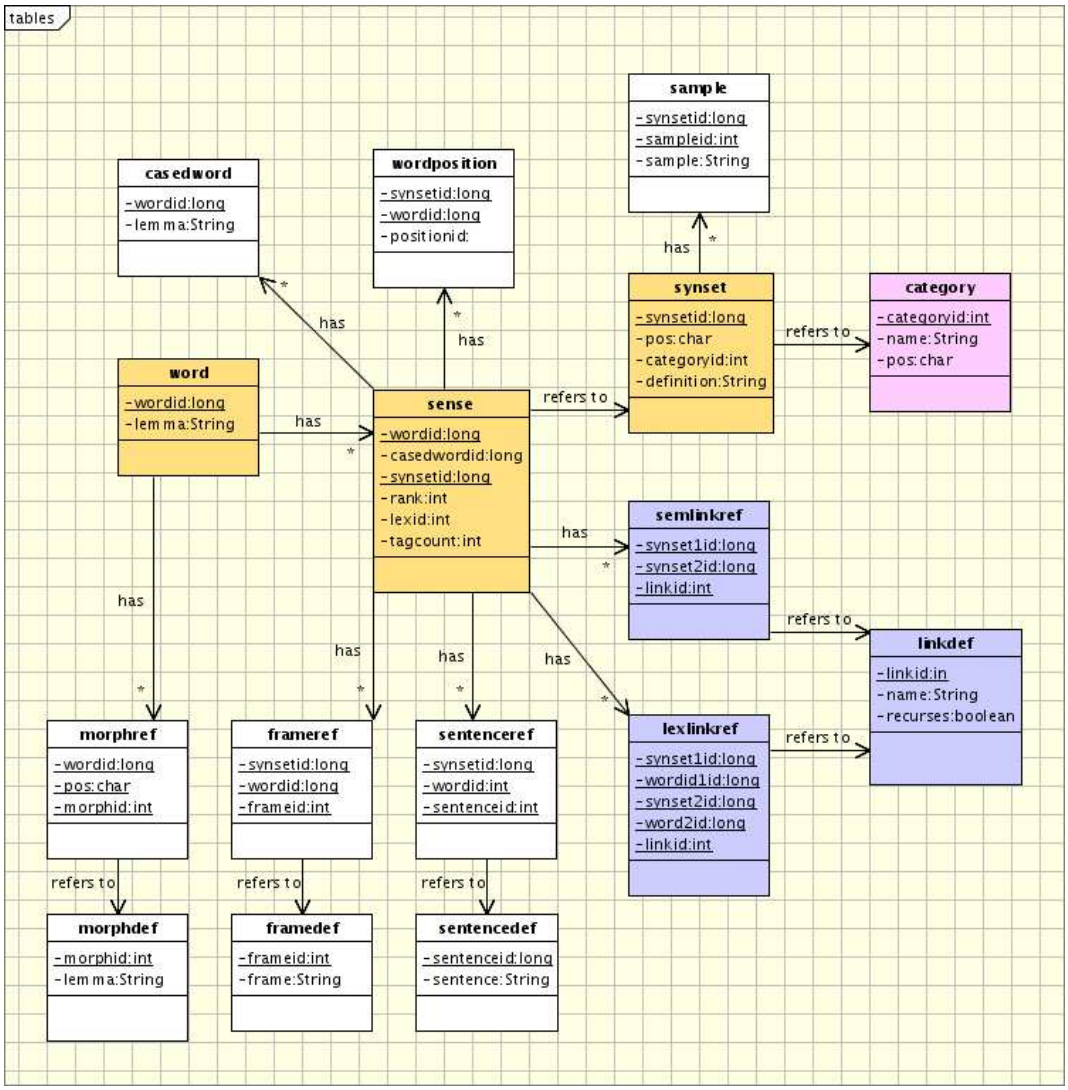

Figure 28: WordNet 3.0 Database schema [21]

Database table's explanation:

- **Word**: This table has a *wordid* and its respective *lemma*. *wordid* is the primary key, and *lemma* are words or set of words defining concepts (persons, posts, dates, objects). Some examples of these concepts: Abraham Lincoln, 1st Lieutenant, 14 July, dining-room table.
- **Sense**: Makes the mapping between tables *word* and *synset*. Responsible for the relation between words and their definitions (wordid - synsetid). Most words have

various possible definitions, so a *wordid* has various *synsetids*. This table has primary keys *wordid* and *synsetid*, and the foreign keys *wordid* (*word* table) and *synsetid* (*synset* table)

- **Synset**: Important of this table is the primary key *synsetid* and the respective textual *definition* (what we search for and is returned to user in wiki).

In queries implementation we had to search covering all three tables, the essential table is sense because connect words to their definitions.

The implementation was made in file:
\extensions\SMWHalo\includes\SMW_Autocomplete.php

We modified function SMWfAutoCompletionDispatcher() and created a new function called getWordNetDefinitionProposals()

5.1.3 WordNet integration with SMW final version

To use it, the user must type: "{{definition|word|" then press ctrl+alt+space (Firefox) or crtl+space (IE) to trigger the word correspondent definitions.

To article visualization, we created a modified Tooltip extension (file \extensions\WordNetTooltip.php) where we also changed the CSS to differentiate the WordNet defined words from the regular wiki text words.

5.2 Semantic Wiki upgrade

This section starts describing the changes of SMW 1.0, follows the tasks we had to perform in order to accomplish the Organizational Modeling extension integration in the upgraded wiki. One of these tasks was the attempt to use the native semantic searches or inline queries, that we concluded weren't useful for our objectives.

For development of the WordNet integration with SMW we used one of the latest versions of MediaWiki (1.10.*), and version 1.0beta of SMW and halo extension.

The project developed by João Mendes used MediaWiki 1.9.1 and SMW 0.6. Before starting to work in diagram construction we opted to upgrade the old semantic wiki to the latest release of both applications at the time.

MediaWiki had some changes that we will see later, but SMW had big changes. These changes included syntax, structure and consequently modifications in the SMW database.

5.2.1 SMW 1.0 changes

João's project was built using SMW 0.6, in SMW1.0 version had some changes, the introduction of categories, the attributes and relations were replaced by properties, before it was [[predicate:attribute]] and [[relation::relation_object]], now the syntax its always the symbol "::", for example: [[capital of::Germany]]. The SMW1.0 conclusions list:

- **Categories** are used as universal "tags" for articles, describing that the article belongs to a certain group of articles.
- To add an article to a category "Example category", just write [[Category:Example category]] anywhere in the article.
- A category forms a collection of articles that are considered useful or interesting for users, and categories are organized so users can browse narrower or broader groupings and find related concepts.
- How to define the predicate: David plays professor.
    - Definition of properties is free.
    - Example: in the page David we can type: David plays a [[plays::professor]] role in University of Madeira.
- Validation of relations: relations are valid since that the properties used by them are created (defined in the wiki). The user is free to create the properties he want, the same happens with the categories; even so it's advised to use the already defined.

The MediaWiki and SMW upgrade is made by installing the new version or substituting the old files for the new ones. The problem is the database (both applications use the same),

and to maintain old data (Organizational relations and concepts) we had to upgrade the old database, making it compatible with the latest versions.

The upgrade of old semantic wiki was done in collaboration with Jorge Cardoso. We upgraded MediaWiki to version 1.11.0 and SMW to version 1.0.

After upgrading, the WordNet definitions auto-complete were integrated without problem, but the Organizational Modeling extension and its respective template didn't work.

5.2.2 Adaptation of the Organizational Modeling extension

After an unsuccessful attempt to adapt the Organizational Modeling extension to the SMW 1.0, we decided to remove the inverse relation from the database and started to redefine the semantic searches of the Organizational Modeling extension. The idea was to use the inline queries of the new SMW 1.0 and try to make use of its advantages instead of our semantic search.

Inline queries

Supports subqueries, for example:

[[Category:Actor]] [[born in::<q>[[Category:City]] [[located in::Italy]]</q>]]

The inline queries were explored and tested, trying to make nested searches (like our Organizational Modeling extension) but using only inline queries we only could make searches with one level of deepness

Possible alternatives:

- Using both, inline queries and the inverse relation in the database, this way it's possible to do semantic searches on more than one level of deepness but they stay limited by a maximum deepness, the maximum deepness must be previously defined by the programmer. With the SQL semantic search using the inverse properties crated, the search is executed until the last level of deepness
    - The principal limitation comparing with SQL was not being able to make a search using a result of another search, for example a search returned an

activity and I want to search what that activity outputs.

- Continue the redefinition of the old searches without using the inverse relation
    - Would have been difficult and had the disadvantage of doing many accesses to the database and a lot more queries. This would introduce complexity and make the application slower.

Conclusion

We tried to use the inline queries to make all the semantic searches needed, with them we didn't need to create all the inverse relations in the wiki Database (table SMW_relations). But inline queries don't have the capacity of managing data as SQL and due to the limitations we decided to stand with the inverse relations in the database, using SQL (mySQL) to make the semantic searches. Inline queries are good to make a question and obtain a set of nodes (insufficient to navigate in semantic graph) as with SQL we had freedom to make nested searches and navigate through the graph. With the inverse relation we also gain performance because their creation is simple and the searches need fewer queries, improving performance.

Another strong reason to maintain inverse relations is that SMW 1.0 doesn't implement hierarchies, it uses categories, and these organize data and serves for searches inside one or more categories. Following text belongs to SMW official documentation:

"Hierarchies are in the SMW TODO list and will be implemented:

- Hierarchies of relations and attributes are obviously needed. The main challenge is to support them in inline queries, but also a well-formed RDF-export is a requirement."

The Organizational Modeling template implements hierarchies with the creation of some special properties (relations) and its correspondent inverse, for example property "Is_a" (generalization) and its inverse "inv_Is_a" (specialization).

As we decided to maintain the inverse relations we had to make the Organizational Modeling extension work in the upgraded semantic wiki to.

We discovered that to solve the Organizational Modeling extension problems we had to make the adaptations, made on the old wiki by João, but make them in the new semantic wiki. These adaptations were code changes or new functions distributed by various files of MediaWiki and SMW.

Because latest MediaWiki and SMW had many changes comparing to older versions, we had a lot of work to discover what features (like linking images directly to an image and not an image page) implicated changes and search the files and respective portions of code to change. This was caused for lack of documentation, about this part, from previous work.

Creation of the inverse relations

Inverse relations are important because they improve graph navigation and permit inverse semantic searches. This was an important requisite to make the relations between pages appear in the Organizational Modeling template.

The main problem wasn't the creation but the update of the triples (subject-relation-object), so to do the updates the solution was deleting the old triples (but not all because the subject may have relations with other pages) and save the new ones.

File changed: extensions/SemantiMediaWiki/includes/storage/SQL_Storage.php

- Functions changed: updateData (),deleteSemanticData ().
- Implementation of a new function: updateSemanticData(), only to do the update separately, resolved the problem of the deletion of the subject when we saved a page that happened due to the use of the function deleteSemanticData() called by function updateData().

Additional changes

The relations part of the Organizational Modeling extension was solve but we continued having problems.

Follows the features that needed changes and the correspondent files were code was changed.

Organizational Modeling template embedded in all wiki pages

This feature was very simple in the old wiki. In the new one was necessary some more modifications. The file used was includes\EditPage.php.

Images presentation

The image functions on the new MediaWiki changed a lot. In order to the images generated from the Organizational Modeling extension work correctly we had to copy some image functions from the old MediaWiki version. The file to copy functions from old wiki is includes\imageFunctions.php.

At the includes\Defaultsettings.php file, add svg to the permitted file extensions to the array $wgFileExtensions.

If the images still don't appear in your wiki you need to install a software called ImageMagick.

Direct link to .svg images

When uploading a image to the wiki, that image is stored in a proper image page. But we wanted that when clicking in a diagram .png image, at a wiki diagram page, the user was directly send to the .svg correspondent image. The changes wre made in file includes\OutputPage.php.

Old relations pages changed to property type pages

We had to change all old relation pages to property pages because in the new upgraded wiki the links in the fact box were to property type pages (because of the change to property to all predicates). Before the relations could have a predicate name different from the relation name, in the new wiki the relation and its predicate name must be the same. These changes were made in the wiki but we needed to adapt the Organizational Modeling extension to them. We had to update the code to, in organizational views, when appearing relations, their links follow to the new property pages.

Creation of the relations pages in the wiki and database

Once again we had problems to adapt this functionality to the new semantic wiki 1.0, more about them in section of problems found. The automatic creation of relation pages was made but later we decided to remove these pages from the wiki because SMW 1.0 provided a functionality (Pages using the property) that allowed a similar result.

The old relation pages were accessed through links from relations in the Organizational Views template and provided the subject, predicate and object of a certain relation; it also provided a way to see what pages were using that type of relation. In SMW 1.0, a property (correspondent for old wiki relation) page already has a list of pages using that property (relation).

5.3 Automatic diagrams generation

This section begins with a clarification of how MediaWiki and the Organizational Modeling extension interact. Next is this module first goal, the implementation of conditions in the current Organizational Modeling extension activity diagrams. The auto graph parser and builder for various types of diagram is section's main part, ending with instructions to create a new type of diagram.

Interaction between MediaWiki and Organizational Modeling extension

The connection happens through the use of a wiki template. The template (called Organizational views) calls the diagrams builder and semantic searches functions defined in the Organizational Modeling extension. The template was integrated in all wiki pages, so every page calls the functions with the respective page name as attribute. The Organizational Modeling extension receives the data, treats it and returns the images generated and the semantic searches results.

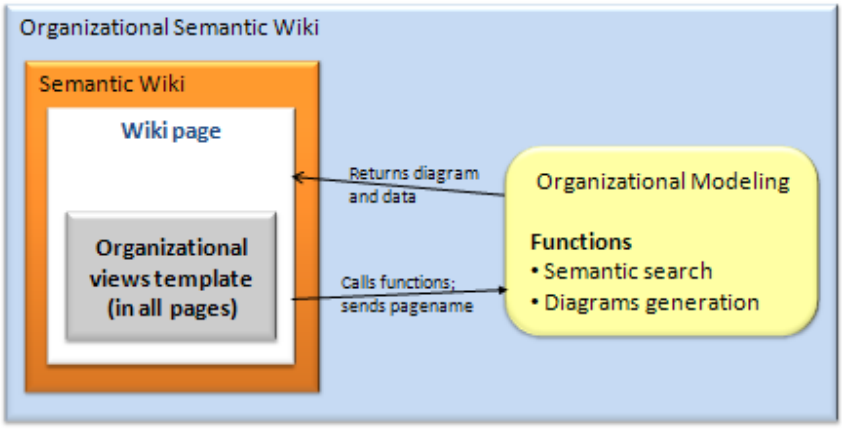

Figure 29: Interaction between MediaWiki and Organizational Modeling extension

We had the Organizational Modeling extension explained in chapter 2, section 2.4.3, so we already had a function to do the semantic search and a first prototype of Activity diagrams, the generation of the diagrams was made with Graphviz - Graph Visualization Software.

Diagram builder process

Before we describe the each diagram, it's important to understand how process used to build them. While covering the graphs and extracting its information (elements and relations), we add to a buffer the Graphviz elements necessary to build the correspondent diagram. To render the diagram we execute Graphviz commands using the buffer code as source to output png and svg image files.

5.3.1 Implementing conditions in the activity diagrams

Our task was to implement conditions in a way that the diagrams were possible to convert to and from Petri nets if needed in the future. Our wiki has two types of activity diagrams: simple (only 1 activity) and with sub-activities (shows how various activities interact forming another activity or business process). Our work was redefining the diagrams of activities with sub-activities, adding conditions.

Our redefinition resulted in a completely new algorithm to design the activity diagrams because the conditions were related to all the relations between activities. Figure 30 shows our first attempt.

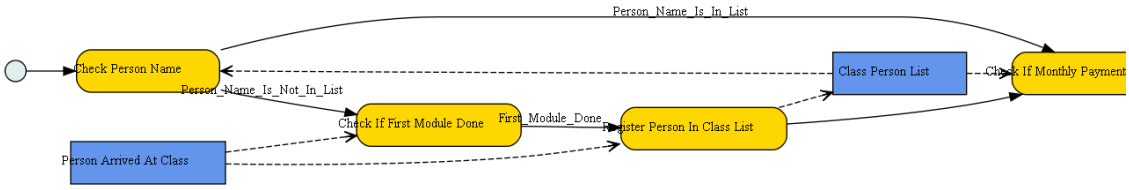

Figure 30: First version of Conditions in activity diagrams

Later we decided to improve it, adding diamonds with a question before conditions; next figure shows part of Graphviz output:

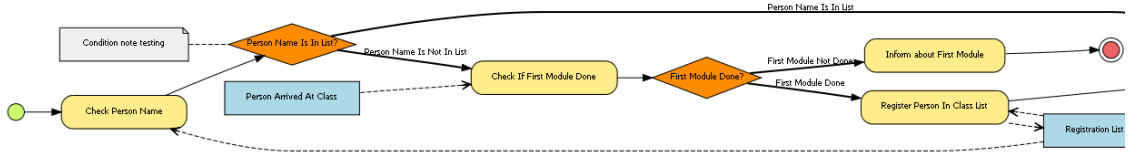

Figure 31: Final version of condition in activity diagrams

To model an activity with sub-activities (Business process) in wiki text, the user just needs to specify what sub-activities make part of the activity. For example: "*[[Has activity::Check if first module done]], [[Has activity::Inform about first module]]*" and so on. It's also need to specify the beginning and end sub-activities: "*[[Begins::Check Person Name]], [[Ends::Inform about First Module]]*".

Implementation

The function that implements this type of model is drawActivityModel() from Organizational Modeling extension.

Sub-Activities

Using a semantic search with predicate *"Has_activity"*, we start adding to our buffer the sub-activities, i.e. the activity we are modeling has various sub-activities.

PreviousSub-Activity -> Sub-Activity

This part isn't related to the conditions, it's to connect sub-activities directly connected to other without questions (conditions). For each sub-activity found above, we search for PreviousSub-Activities (executed before the current one and connected to it) through predicate *"After_Activity"* present in current sub-activity. With PreviousSub-Activities found we add (to buffer) the connection between sub-activities. Note that making this step for each sub-activity covers the all graph, connecting all the activities.

Adding Conditions

Before explaining implementation we present a figure of a part of a SMW graph to understand the relations between elements and properties used to create them:

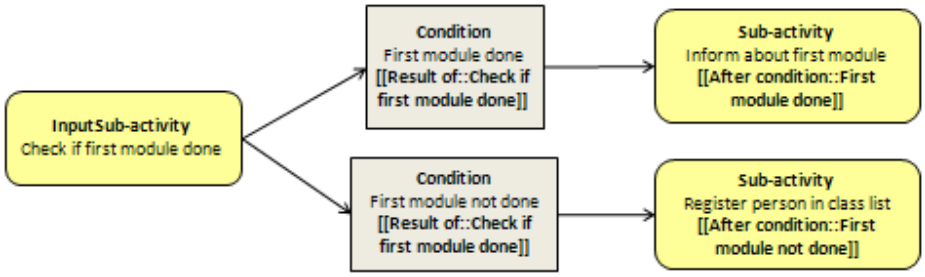

Figure 32: Activities-conditions relations graph

For the conditions implementation the connection we need to make is InputSub-Activity -> condition -> Sub-Activity. This must be made backwards, starting in the Sub-Activity. So we start with:

Condition -> Sub-Activity connection

The same way as the previous PreviousSub-Activity -> Sub-Activity connection, we search for conditions executed before the current sub-activity. Note that the search (graph navigation) is always made backwards. Here we use the predicate *"After Condition"* (present in current sub-activity) to get the conditions before the sub-activity and connect

them to the current activity. For example, following next figure diagram, supposing our algorithm was at *"Inform about first module"* sub-activity, we make a semantic search for *"After Condition"* and the result is "*First Module Not done"*. Note that the sub-activity has "*[[After Condition::First module done]]"* in its wiki text.

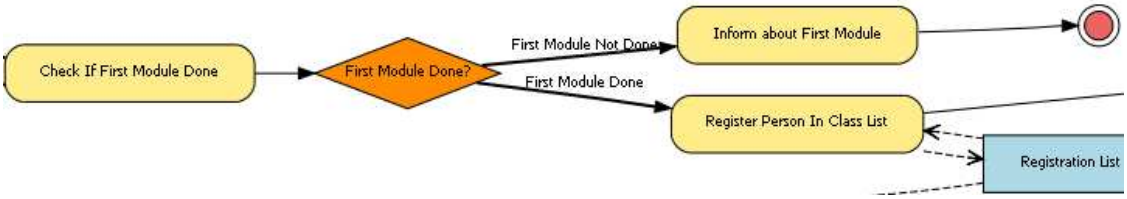

Figure 33: Conditions in our generated diagram

InputSub-Activity -> Condition connection

Then for each condition found above we search for the InputSub-Activities through predicate *"Result Of"* (Condition is a result of InputSubActivity). The found InputSubActivities are connected to the Condition.

This way we complete the InputSubActivity -> Condition -> SubActivity connections, this process is made for every sub-activity until covering all the elements of the semantic graph.

Notes

We added the possibility to take notes at activities, conditions and resources. These can be just a simple note *"[[Note::noteDescription]]"* or page with long description *"[[Note::NotePageName]]"*. All notes automatically appear in activity diagrams using searches with predicate *"Note"*.

5.3.2 Entity-Relationship Model

From this point, we could start developing new types of diagrams; the first was the ER model.

Wiki hierarchy structure for ER models:

- Entity -> Model -> ER Model -> ER Model instance

The previous diagrams (activity) were made at activity pages. For ER model we opted for the generalization of entity *"Model"*, this way, in the future can be added other types of models. Also activity diagrams had activities and conditions as elements, the ER model is composed by entities, relations and attributes.

In ER models pages the user just need to define the entities he wants to be part of the model. For example: *[[Has ER entity::Department]], [[Has ER entity::Student]]*. The diagram builder fetches and adds the relations between entities and attributes.

Implementation

The simplest possible explanation is that the algorithm covers each entity at the ER model page, and then searches and connects to all relations that entity has. For ER relations is done the same, connecting them to entities. We used the semantic search, modified by us to find all the relations that entity uses. To complete the algorithm we introduced verifications (avoid multiple arrows between same elements) and cardinality of relations.

The function that implements this type of model is drawEntityRelationshipModel() present in Organizational Modeling extension.
First we search and add to our buffer all the instances of elements that will be part of the graph:

- ER Entities (we decided to distinguish from regular entities)
- ER Attributes
- Relations

Connections

The general principle is the same of the previous diagram developed. But for the ER we have to connect two entities using a relation, for example: *University has department*, University is the *subject*, *department* the object and *has* the relation.

To help understanding an image of a semantic graph part used as source:

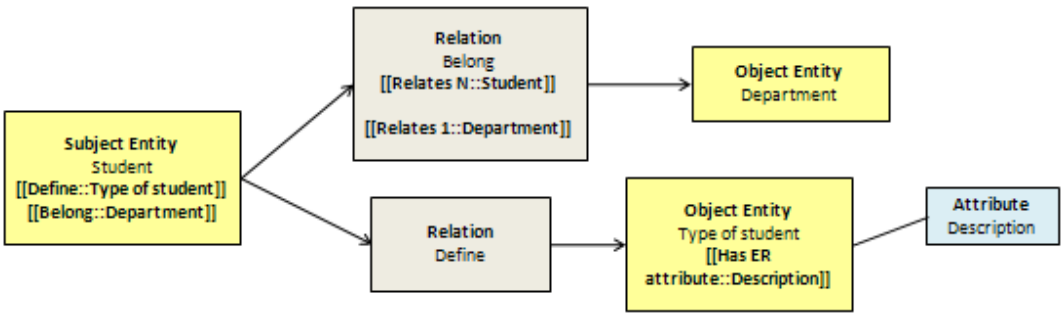

Figure 34: Entities-relationship graph

For each entity found we search just for its properties/relations (for this we modified the semantic search-instead of searching for a predicate that returned objects, we wanted just an entity's relations). The modifications were made at deepFirstSemanticSearch() function.

Subject Entity to Relation connections

Then for each relation found is made a verification to confirm that correspondent object entity is an ER entity (avoid connections to non ER entities) and a connection from subject entity to each relation added. The cardinality of the connection is discovered making semantic searches at relations (where cardinality is defined) with predicates *"Relates 1"* and *"Relates N"*, adding the correspondent cardinality symbol (1 or *) in the connection arrow.

Relation to Object Entity connections

Again for each of the relations of an entity is searched the object entity, verified if the object is an ER entity and the connection between relation and object entity added. The cardinality is the same as mentioned above.

Attributes

A semantic search with predicate *"Has_ER_attribute"* is made to find all attributes from each entity and then, the connection between them added.

SMW N-ary properties

SMW latest release provided a new experimental feature called N-ary property. So the first time we wanted to introduce cardinality in the ER models we thought of taking advantage these n-ary relations provided by SMW.

To introduce cardinality in a relation like *Student - Belong – Department (Student [[Belong::Department]])*, we used extra properties in the relation page *Belong [[Relates N::Student]] and Belong [[Relates 1::Department]]*. With N-ary relations the idea is using a unique property (in an entity) with both relation and cardinality, for example: *Student [[Belong::Department; cardinalityValue; cardinalityValue2]]*
We tried to use it to implement cardinality but because of the way these properties are defined in the SMW database (use various tables), we couldn't use the semantic search properly. We made some adaptations to the semantic search function in order to use the n-ary properties, but without success. This feature is experimental and we think isn't still implemented the best way possible, specially the database architecture part of it. We tried the N-ary some months ago, at the present they already changed name to Many-valued properties so maybe they are redefined and can be used in future.

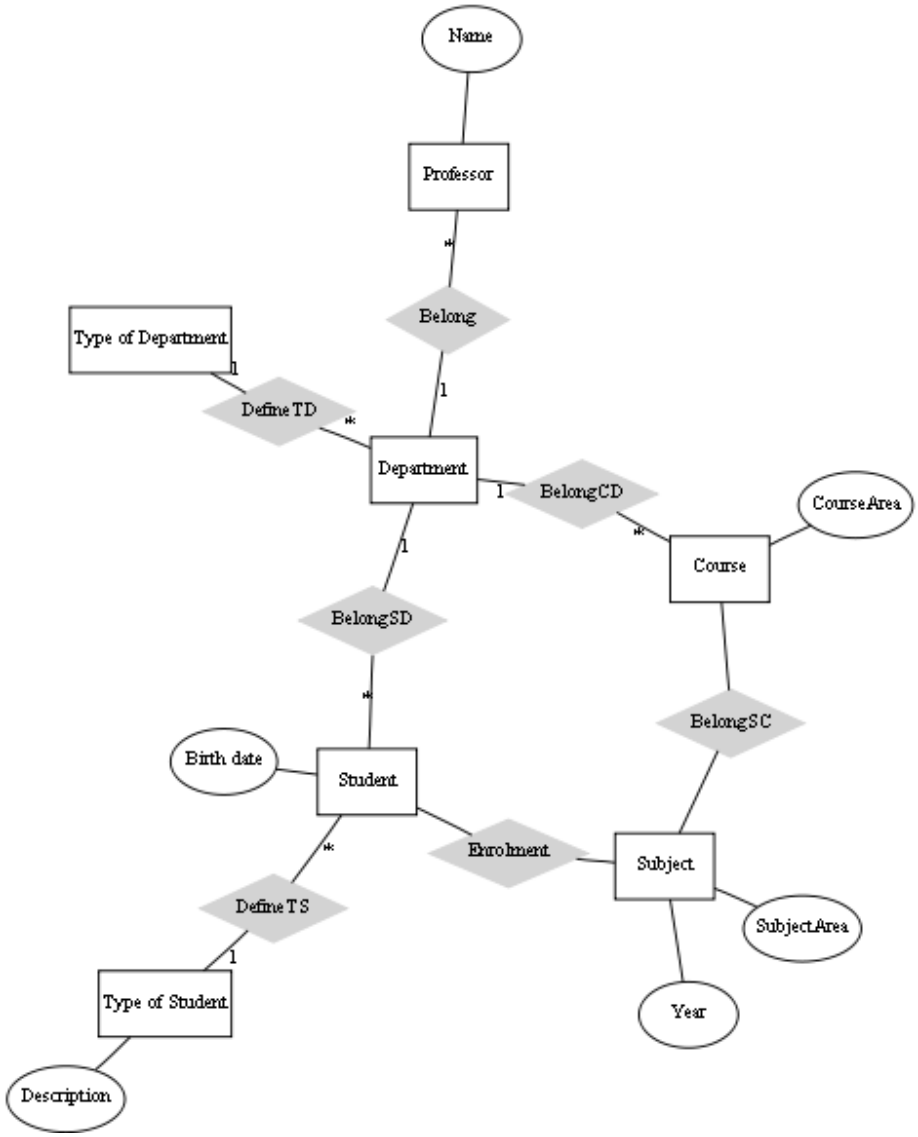

Figure 35: Example ER model of a University

5.3.3 State diagrams

The next type of diagram we found interesting were the state diagrams or state charts. They were designed for entities, to check their states and what activities lead them from a state to another.

This type of diagrams has a particularity compared to others. The pages where they are designed (entities in this case) don't have direct relations to other elements. The user

doesn't need to specify any relations in the entity. The relations are defined in other pages and we get them through semantic searches with the inverse predicate.

Implementation

The initial step is adding the states of the current entity; at the state pages we have *"State [[Is State Of::current Entity]]"*. So, at the entity page, trough a semantic search for predicate "*inv_Is_State_Of*" we get the entity's states.

Explaining with triplets (subject - predicate - object):

- State - Is_State_Of - Entity
- Entity - inv_Is_State_Of – State

A part of semantic graph image to help following the explanation:

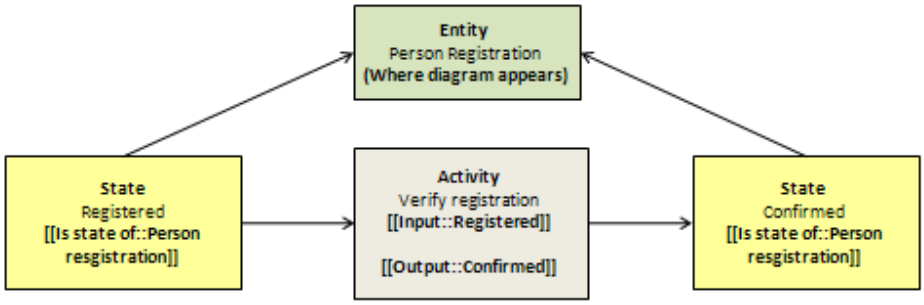

Figure 36: States relations graph

The core of the building function:

- For each of the entity's states:
    - It searches the activities that output that state (using predicate *inv_ Output*); at the activity is *[[Output::State]]*. Following Figure 37 example and supposing we're at *"Confirmed"* state, the activity found is *"Verify Registration"*.
    - Then for each returned activity, we search its input states. In example is "*Registered*" state

- Finally, each input state returned is connected to the current state, with the connection having the name of the activity that outputs that state.

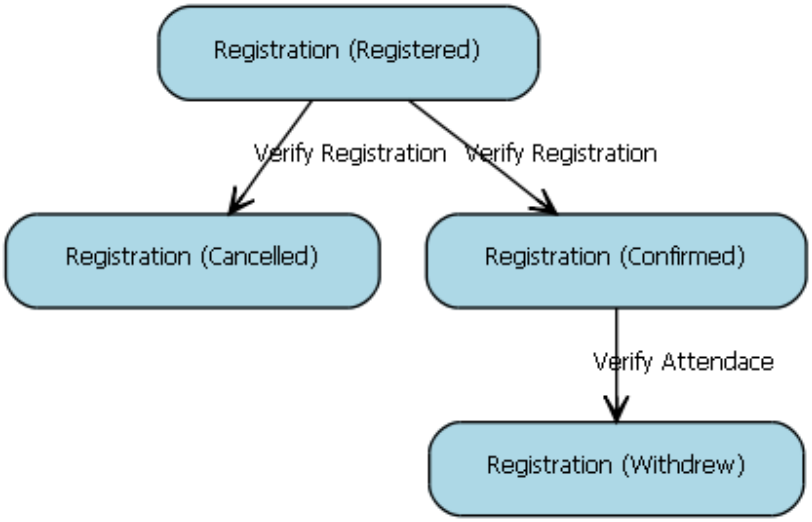

Figure 37: Example of a generated state diagram

5.3.4 Use case diagrams

This was the last type of diagrams implemented in our project; this popular diagram type defined by UML is simple but very useful to understand which actors perform the functionalities provided by a system, and the dependencies between functionalities.

In our case, what we defined as roles are the equivalent to actors, the objective was define which activities are performed by a certain role. Also important was a better understanding of the roles hierarchies, and the relations between some activities.

Wiki hierarchy structure:
Thing -> Entity -> Model -> Use case diagram

These diagrams are, like the ER model, not specified for an organizational element. To construct a Use case, the user just needs to specify the roles he wants to see the respective activities, example *"[[Role::Employee]]"* will return the activities the employee performs.

Before implementation we take a look at an example of the source graph:

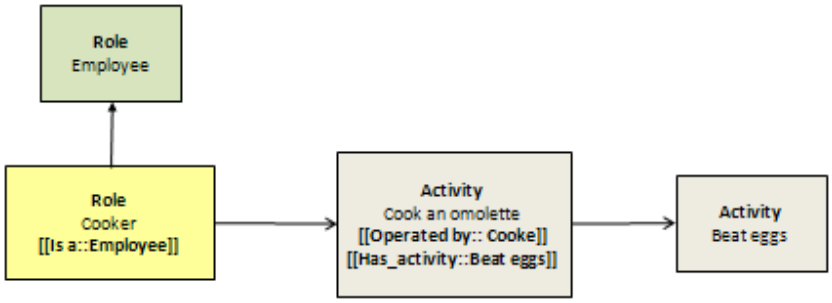

Figure 38: Roles-Activities relations graph

Implementation

First thing was drawing the users, a semantic search by predicate *"Role"* to get them. The particularity of this type of diagrams is that for each actor (role) it's designed a sub-graph.

- Then we drew the activities making a semantic search for each user activities through predicate *"inv_Operated_by"*.
    - Each activity found was connected to the correspondent user. At this point we also checked for includes and extends between activities, and each one found was connected properly. We used the predicate *"Has_activity"* for the *"<<includes>>"* and *"extends"* for the *"<<extends>>"*. Actually, the "extends" aren't used in the wiki but are an interesting predicate to use in the future so we implemented the use cases to support them.
- Also for each user we searched for generalizations between actors and connected them. Because we also used the predicate *"Is_a"* to do actors generalizations we had to add some changes in the semantic search to avoid conflictions with the normal *"Is_a"*, otherwise it will appear undesired elements in the diagram.

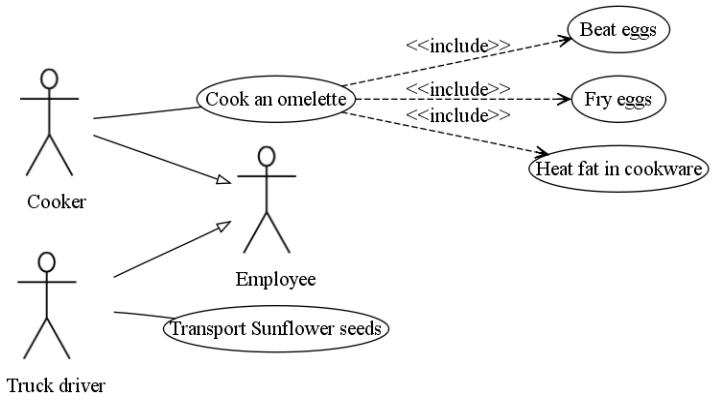

Figure 39: Example of a generated use case diagram

5.3.5 How to create a new type of diagram

Here we list the steps necessary to create a new diagram type. What files need to be modified, what functions created and where, what to modify in wiki.

Changes in OrganizationalModeling.php file of Organizational Modeling extension:

- Add a hook with the your new function name in method: wfSetupOrganizationalModeling(). Follow the pattern of the other functions defined.
- Add your new function name in function: wfOrganizationalModelingGetMagic(). Again it's just follow the pattern of the other functions defined.
- Implement you new diagram method

At the template Organizational views page in the wiki, introduce where you want to call the function defined in Organizational Modeling extension. The other diagram construction functions are all called at the beginning of the template.

5.4 Additional features implemented

Other smaller improvements were made to the wiki like preparation of the Organizational Modeling extension to work both in Linux and Windows operating systems, also the introduction of semantics in the wiki made us implement a new rename for the wiki pages.

Rename for SMW

We saw in previous chapter that the MediaWiki rename wasn't usable to SMW. Therefore we implemented functionality for renaming pages in SMW; maybe we can call it semantic rename. The file with implemented code is rename/index.php and includes all the queries performed. The used tables were: page – to change the pages' title, pagelinks – to update the links, relations – to update the semantic relations, and text – to rename all instances present in wiki text. Note that we had to update the relations and the text, if we just modified the relations, wiki semantic links would be wrong. If we just updated the wiki text it would be necessary to resave all changed pages. The rename UI is accessible through URL http://host/wiki/rename/.

Semantic Organizational Modeling wiki for Linux / Windows

The Organizational Modeling extension was only prepared to run in linux operating systems, to run in windows we created a new version, changes made were removing all "\" characters that were before a ", for example \". Also the Graphviz commands to generate the output files (.png and .svg) had to be altered.

Then we had to prepare the semantic wiki to recognize the operating system and perform different actions depending on the system. As already said the changes were the \ character necessary before the " character in linux, but mainly the Graphviz commands.

The implementation of this feature is located at LocalSettings.php and also in functions windows() and linux() of file OrganizationalModeling.php of the Organizational Modeling extension.

5.5 Problems found

Wiki upgrade

A lot of problems were found in this period of work. The problem was to integrate the Organizational Modeling extension in the latest release of MediaWiki and SMW. We had to make the old changes made in old wiki, in the new wiki. The big problem was that the new releases had many changes (especially SMW) and we didn't have documentation about the changes made in the old wiki, from previous João Mendes work. So we had to

identify where the changes were in the old wiki, we did this searching for commented code (the changed parts were commented). Because of the many changes in the new MediaWiki and SMW releases, to make the adaptations needed to Organizational Modeling work properly in the new wiki, we spent lots of time searching for what files and where to modify the code.

The MediaWiki+SMW are composed by more than three thousand files and these searches were nothing motivational.

Creation of the relations pages in the wiki and database

Once again we had problems to adapt this functionality to the new semantic wiki 1.0, mainly because of inexistent relations on the database, after the update of the SMW the database lost all the inverse relations, was needed to resave many pages to update/insert triples and inverse relations in the wiki database (table SMW_relations and SMW_attributes).

This problem was related to the wiki upgrade. This was the most difficult adaptation to make because the functions and architecture of this part were completely different from old wiki, so we had to understand the new code implemented in order to reach solution.

Beside successful end, this feature ended to be removed from our wiki.

CHAPTER 6 – FUTURE WORK

There are still some things that can be made to improve the Semantic Wiki to model organizations. First we see the ideas related with WordNet part, then the ones related with modeling and finally other general suggestions.

6.1 Related with WordNet

6.1.1 WordNet integration in Halo toolbox

Halo toolbox offers an easier way of making semantic annotations. So integrating the WordNet there would facilitate its use. The toolbox provides faster annotation of categories and properties, it would be necessary to implement the annotation of templates to use WordNet. We started exploring this features but didn't advance much.

6.1.2 Include category of word in the auto-complete

This would improve the WordNet auto-complete. When the user chooses the definition for a word, include the category of the word would increase the information about that word and its definition. The categories are: noun, verb, adjective, and adverb. The inclusion of these could help the user deciding his definition choice.

6.1.3 Include related words in the auto-complete

This would take advantage of WordNet semantic web, providing easy access to words related to the current. As a word can have various definitions, some definitions can be used in a set of words. For example: in a search for "car" the first definition set of words are: auto, automobile, machine, and motorcar.

6.2 Related with Models

6.2.1 Use of Many-valued properties in the Entity-Relationship models

We tried to use the N-ary properties (old name of Many-valued properties) new but still experimental in SMW 1.0. We didn't find a way to use them properly. This feature was

new and it seemed to need some corrections, simplifying the database architecture of it. As the name already changed maybe they are improved and can be used in future, providing an easier way to use cardinality.

6.2.2 Unique and universal method to construct diagrams

Regarding the Organizational modeling part of our project, the logical improvement is adding new types of diagrams. But more interesting is the challenge to make a general method to construct a diagram, independent of its type. We gave a contribution about it in chapter 4, proposing a general method for all diagrams common rules and then separate methods for each type of diagrams building algorithms.

6.2.3 Entity-Relationship models – entities connection through foreign key

ER entities can use a predicate “References ER attribute”, this means an entity references another entity attribute, or in other words implements a foreign key. The idea is, in ER model construction algorithm automatically connect the entities with a foreign key, with the foreign key respective entity.

6.2.4 Interaction of wiki data and diagrams with external modeling tools

Still in the diagrams area, the export of diagrams from an external modeling tool to the wiki is a good idea. The user could model easier in the external tool and then store them in wiki where he could add more detail. The addition of the inverse method, export data and diagrams from wiki to an external tool would be the ideal. The user could make his models in an external tool faster than in the wiki, export them to the wiki and make necessary arrangements. Later, if it were necessary many changes in diagram structure, was possible to export to the external tool and update them easier than in the wiki. Jorge Cardoso from University of Madeira is already developing a tool with these features in a project parallel to ours.

6.2.5 External workflow applications integration with SMW

Imagining a workflow tool that during a workflow execution could fetch data in SMW is an interesting idea. At some workflow point it could access SMW to get organized data, for

example: a certain activity will be triggered just when a determined number of people were registered in it. The tool would have to check the SMW periodically asking for the number of registered persons.

6.3 General

6.3.1 Integration of the Rename UI in MediaWiki template

Reminding one of the additional features made, we made a simple UI to the rename of pages, and this is separate from MediaWiki template. So the integration of the rename interface in the MediaWiki template is a task that can be done in the future.

CHAPTER 7 – CONCLUSION

Final chapter is to summarize and remind the important conclusions of the project. We review the project context, problems found, applications used, related work, solutions found and draw the correspondent conclusions.

After the work is done we can take conclusions about what was done. Basic conclusion is that semantic web is evolving pretty fast and its improvements comparing to the World Wide Web are remarkable. Some of these improvements include more accurate search for information, better navigation through data, and creation of relations between entities forming a semantic web.

We took advantages of semantic web technology to model organizations. This achievement was made with the use of a semantic wiki. We made a regular wiki (MediaWiki) semantic using the semantic MediaWiki extension. Then we got started from a prototype application (Organizational Modeling extension) developed to model organizations with a semantic wiki. This project just generated activity diagrams and with some limitations.

Two major problems were identified: Limited formalization of wiki content and, lack of graphical view for models specified in semantic wiki. We researched for tools similar to what we intended, identified their limitations but also got some inspiration of their good features.

First part of our work was the integration our semantic wiki with a lexical dictionary to provide a better formalization of meanings. The second part was the improvement of Organizational Modeling extension, we introduced conditions is the activity diagrams and created three new types of diagrams popular in organizational modeling area: Entity-relationship models, State diagrams and Use cases. We developed some other features to make our semantic wiki better.

We can conclude the WordNet integration with SMW was useful because offers a way to enrich content, at the same time it can eliminate doubts about some concepts. Also SMW

provides features to create models but its interface is all textual and lacks the graphical views of them. With Organizational modeling extension and our automatically generated bottom-up diagrams, we offer a graphical view much more pleasant and understandable.

BIBLIOGRAPHY

[1] “The Semantic Web: Overview / Semantic Web”, http://www.sciam.com/article.cfm?id=the-semantic-web-overview

[2] “Semantic Web - Wikipedia, the free encyclopedia”, http://en.wikipedia.org/wiki/Semantic_Web

[3] “Data Modeling, RDF, & OWL – Part One: An Introduction To Ontologies” by David C. Hay, http://www.tdan.com/view-articles/5025/

[4] “Twine: The First Mainstream Semantic Web App? – ReadWriteWeb”, http://www.readwriteweb.com/archives/twine_first_mainstream_semantic_web_app.php

[5] “The Technology | Twine”, http://www.twine.com/technology

[6] “sioc-project.org | Semantically-Interlinked Online Communities”, http://sioc-project.org/

[7] “Nextbio – Wikipedia, the free encyclopedia”, http://en.wikipedia.org/wiki/Nextbio

[8] “About Nextbio”, http://www.nextbio.com/b/corp/about.nb

[9] “Semantic wiki - Wikipedia, the free encyclopedia”, http://en.wikipedia.org/wiki/Semantic_wiki

[10] “Enterprise Architecture Modeling with the Unified Modeling Language”, by Pedro Sousa, Artur Caetano, André Vasconcelos, Carla Pereira, José Tribolet

[11] “MediaWiki - Wikipedia, the free encyclopedia”, http://en.wikipedia.org/wiki/MediaWiki

[12] “Introduction to Semantic MediaWiki”, http://semantic-mediawiki.org/wiki/Help:Introduction_to_Semantic_MediaWiki

[13] “Organizational Modeling Bootstrap with a Semantic Wiki” by David Aveiro, João Mendes, José Tribolet

[14] “Graphviz”, http://www.graphviz.org/

[15] “OntoLing Tab”, http://ai-nlp.info.uniroma2.it/software/OntoLing/

[16] “Semantic Reference- and Business Process Modeling enables an Automatic Synthesis”, by Florian Lautenbacher, Bernhard Bauer

[17] “Software Secret Weapons: Linguine Maps”, http://www.softwaresecretweapons.com/jspwiki/linguinemaps

[18] “Rangel State Machine Compiler”, http://www.complang.org/ragel/

[19] “WordNet - Princeton University Cognitive Science Laboratory”, http://WordNet.princeton.edu/

[20] “Halo Extension – ontoworld.org”, http://ontoworld.org/wiki/Halo_Extension

[21] “WordNet SQL Builder”, http://wnsqlbuilder.sourceforge.net/schema.html

ANNEX

A.1 Installation manual

1. If you're using Windows, install xampp software and copy the wiki folder present in project CD for the htdocs folder of xampp.
2. Next step is in phpMyadmin or other sql manager software; create a database named "wikidb2" or other of your choice (in this case it's needed to update the database settings in the wiki settings, LocalSettings.php). To use Wordnet extension for SMW you must also execute WordnetDB.sql script to create WordNet tables in your wiki database.
3. Still in sql manager software, create a user (username:wikiuser; password:123) with all privileges for wiki database.
4. Import our database sql script (wikidb2.sql file) to populate your database.
5. Install Graphviz and ImageMagick present in CD. You may need to restart to Graphviz work properly.
6. Access the wiki through your browser: http://localhost/wiki
7. To more details and linux installation consult the Readme Wiki installation file present in project CD.

A.2 User manual

MediaWiki, SMW, Halo, Tooltip extension manuals are accessible online, so we will concentrate in principal concepts necessary to understand and model Organizations. Starting with our semantic wiki architecture and then explaining the most important relations and concepts and how use each diagram.

Used application manual:

- MediaWiki: http://www.MediaWiki.org/wiki/Manual:Contents
- SMW: http://semantic-MediaWiki.org/wiki/Help:User_manual

- Graphviz: http://www.Graphviz.org/Documentation.php

Relations

As we already got acknowledge how to work with different types of diagrams in chapter 5 - Implementation. Here we just look at most used and important relations of the Organizational Ontology.

General use

Is a: implements generalization / hierarchies, example: Entity is a Thing.

After:

- After Activity: used at activities, to connect with other activities in activity diagrams
- After Condition: used at activities, to connect conditions in activity diagrams

Has:

- Has Activity: used in activities with sub-activities, forming a business process
- Has ER Entity: used in ER models to specify what entities are included

Input and Output: used in activities, related to input and output resources or states

Operated by: used at activities, to say what role operates it

Is state of: used in states, to relate them to entities

Plays: used in entities, to give them a role

Result of: used in conditions, to specify from which activity they result

References ER attribute: used in ER entities, to specify a foreign key