

**Univerzita Pavla Jozefa Šafárika v Košiciach
Prírodovedecká fakulta
Ústav Informatiky**

**Experience Management based on
Ontology and Text Notes**

The EMBET System

Rigorózna práca

Košice 2005

Michal Laclavík

Abstract

In this work we describe ontology based knowledge representation and its use for experience management. It is well known that knowledge has an immense value in all kinds of businesses and people's every day life.

In this work a solution for the **e**xperience **m**anagement **b**ased on **t**ext notes (EMBT) entered by a user is described. The key idea is that a user enters notes in a particular situation/context, which can be detected by the computer. Such notes are later displayed to others or the author in a similar situation/context. The context of a user is detected from computerized tasks performed by the user. Not entire detected context is relevant to the entered note and the system with user's assistance needs to detect the relevant part of the context based on the text of the note using semantic annotation and ontology based knowledge model. Problem or application domain is also modeled using ontology.

The solution has been used and evaluated in the Pellucid and K-Wf Grid project and it is continuously developed in the K-Wf Grid project and the Znalosti project.

The CommonKADS methodology and Protégé ontology editor have been used for knowledge modeling, and the OWL ontology for the knowledge representation. The Jena library was used for knowledge and data manipulation and the Java language was used for implementation of the architecture.

Table of Contents

1	Introduction	6
1.1	Motivation and Background.....	6
1.2	Overview of the Approach.....	7
1.3	Structure of the work.....	7
1.4	Research History.....	8
2	State of the Art	9
2.1	Outline of Research Environment	9
2.1.1	Knowledge and Experience Management Overview.....	9
2.1.2	Ontology Overview.....	10
2.2	Knowledge and Ontology.....	11
2.2.1	Role of Semantic Web in Knowledge and Ontology Field.....	11
2.2.2	The Ontology Related Methodologies.....	12
2.2.3	Ontology Representation and languages.....	12
2.2.4	Semantic Annotation.....	14
2.3	Experience management.....	17
2.3.1	Representation of Cases.....	18
2.3.2	The similarity measures.....	20
2.4	Knowledge Presentation.....	21
2.5	Conclusion of State of the Art.....	22
2.6	Objectives of the work.....	23
3	Used Methodologies and Tools.....	24
3.1	Methodologies.....	24
3.1.1	Unified Modeling Language (UML).....	24
3.1.2	CommonKADS Methodology.....	25
3.1.3	Formal Methods Used for Ontology model description.....	26
3.2	Tools.....	28
3.2.1	Protégé Ontology Editor.....	28
3.2.2	Jena library.....	29
4	Ontology Model and Modeling Methodology.....	30
4.1	Specification of Ontology Model for Experience Management.....	30
4.1.1	Extension of model related to Semantic Annotation and text processing.....	32
4.1.2	Workflow and Organizational related Extensions of Model.....	33
4.2	Defined Modeling Methodology.....	34
4.3	Conclusion.....	36
5	Design & Development of EMBET System.....	37

5.1	Overview of the approach for Experience Management.....	37
5.2	Design of the EMBET System.....	40
5.2.1	Use Cases.....	40
5.2.2	Architecture and Technology.....	43
5.3	Knowledge Manipulation Algorithms	44
5.3.1	Algorithm for User context Updating.....	44
5.3.2	Context Matching Algorithm for User resource Updating.....	45
5.3.3	Semantic Annotation Algorithm.....	46
5.4	Specification of EMBET Software Library.....	48
5.5	Interfaces to GUI and External Systems.....	49
5.6	Presenting Ontological Knowledge to Users.....	52
5.6.1	Transformation Solution used for presenting knowledge in experience management ..	52
5.6.2	Knowledge Presentation Conclusion.....	54
5.7	Conclusion on EMBET Design & Development.....	54
6	Use and Evaluation of Results in Pilot Operation.....	55
6.1	K-Wf Grid IST Project.....	55
6.1.1	Flood Forecasting Application.....	57
6.1.2	Domain Ontology Model.....	59
6.1.3	Example of Use.....	61
6.1.4	Conclusion.....	65
6.2	Pellucid IST Project.....	65
6.2.1	Pellucid Pilot Sites.....	66
6.2.2	Ontology Model.....	67
6.2.3	Conclusion.....	70
6.3	Znalosti Project.....	70
6.3.1	Semantic Annotation.....	70
6.3.2	Conclusion and Future work in Znalosti Project.....	72
6.4	Conclusion.....	72
7	Conclusion	73
7.1	Summary of work.....	73
7.2	Results	74
7.3	Scientific Contribution.....	74
7.4	Future Work.....	75

List of Figures

Figure 2.1: W3C architecture of Semantic Web.....	13
Figure 3.1: CommonKADS Models.....	25
Figure 3.2: OWL Constructors.....	26
Figure 3.3: OWL Axioms.....	27
Figure 3.4: Example of Parents Ontology.....	27
Figure 4.1: Pattern Ontology.....	32
Figure 4.2: Basic Ontology for Knowledge Modeling.....	35
Figure 5.1: Experience Management Cycle of EMBET.....	38
Figure 5.2: EMBET UML use cases diagram.....	40
Figure 5.3: EMBET Architecture.....	43
Figure 5.4: Fragment of EMBET ontology with example of ontology individuals.....	46
Figure 5.5: UML Class Diagram of Main EMBET classes.....	48
Figure 5.6: EMBET GUI.....	52
Figure 5.7: Pellucid GUI window with Active Hints (right) and opened resource window (left).....	53
Figure 6.1: K-Wf Grid Knowledge Cycle and Objective.....	56
Figure 6.2: Main Elements of Knowledge Model in K-Wf Grid.....	57
Figure 6.3: Sample flood application workflow, using ALADIN/NLC.....	58
Figure 6.4: Service Ontology with several grid services	59
Figure 6.5: Time Data Ontology.....	60
Figure 6.6: Location Ontology.....	61
Figure 6.7: Entering new Note EMBET Window.....	62
Figure 6.8: Note Context Detection.....	62
Figure 6.9: Problem Definition EMBET Window.....	64
Figure 6.10: Main Domain Concepts of CDG Ontology.....	68
Figure 6.11: Main Domain Concepts of MMBG Ontology.....	69
Figure 6.12: Main Domain Concepts of SADESI Ontology.....	69
Figure 6.13: Job Offer Ontology.....	70
Figure 6.14: Job Offer Individual Created by EMBET Annotation.....	71
Figure 6.15: Fragment of Pattern Individuals.....	71
Figure 6.16: Example of Job Offer.....	72

1 Introduction

This work deals with ontology based knowledge and its use in experience management system. It is well known that knowledge [DAVEN00] has an immense value in all kinds of businesses and people's every day life.

The experience management solutions are focused on knowledge sharing and collaboration among users in organizations. A lot of companies have recognized knowledge and experience as the most valuable assets in their organization. Experience is something that is owned by people only, not obtainable by computer systems. Anyhow, according to the state of the art in the area we can create an experience management system, which will manage (not create) experience and will be able to capture, share and evaluate experience among users.

1.1 Motivation and Background

Knowledge is really a key asset in the world today. The world economy is moving towards the so called knowledge economy, where knowledge is what is valued more than any other resource. Companies value their knowledge and try to rebuild themselves towards knowledge enterprises. They run knowledge or experience management projects to discover and manage valuable knowledge and experience in a company [DAVEN00]. This effort includes also creation and maintenance of knowledge or experience management systems, thus the focus of our effort is on creation of such reusable experience management systems.

The main objective of the solution is to provide a simple and powerful experience management infrastructure, which can be applicable in many areas with users sharing and collaborating through experience. The idea is to return experience to users when they need it. Therefore it is crucially important to model and capture a user context and the described solution can be used only in applications where actions/tasks performed by a user are computerized and can be captured and reported to the system in the form of events.

1.2 Overview of the Approach

Ontology based knowledge management is used, where uncertain knowledge is not present. This is a suitable model for many applications, especially where experience management is needed.

The key idea is that a user enters notes in a particular situation/context, which can be detected by the computer. Such notes are later displayed to others or the author in a similar situation/context. The context of a user is detected from computerized tasks performed by the user. Not entire detected context is relevant to the entered note and the system with user's assistance needs to detect the relevant part of the context based on the text of the note using semantic annotation and ontology based knowledge model. Problem or application domain is also modeled using ontology.

The CommonKADS methodology and Protégé ontology editor have been used for knowledge modeling, and the OWL ontology for knowledge representation. The Jena library was used for knowledge and data manipulation and the Java language was used for implementation of the architecture.

1.3 Structure of the work

Chapter 1 *Introduction* gives a brief overview of the work.

Chapter 2 *State of the Art* explains the main terms and areas such as knowledge, experience or ontology and our understanding and limitations of these terms and discusses the current research state in areas of semantic web, ontologies, knowledge management, experience management and semantic annotation. The last subchapter explains goals of the work.

Chapter 3 *Used Methodology and Tools* depicts methodologies and tools used in the work to obtain the presented scientific results

Chapter 4 *Ontology Model and Modeling Methodology* describes ontology knowledge model for Experience Management and methodology for extending such model for different applications.

Chapter 5 *Design & Development of EMBET System* presents EMBET Experience Management System built on model described in chapter 4 – its approach for experience management, use cases, architecture, developed software libraries and algorithms, interfaces and graphical user interface.

Chapter 6 *Use and Evaluation of Results in Pilot Operation* discusses use of such models and architecture in several projects: K-Wf Grid and Pellucid IST projects and Znalosti national project.

Chapter 7 *Conclusion* concludes the results of the work.

1.4 Research History

The work is written as a monograph, but some of the results have already been presented or published.

Several articles have been published on the theme of Ontology based knowledge & experience management and knowledge related architectures:

- Pellucid Agent Architecture for Knowledge Management in Public Organizations [BAL03B]
- Use of Ontology in Virtual Organizations for Environment Risk Management [BAL03A]
- Distributed Knowledge Management based on Software Agents and Ontology [LAC03C]
- Knowledge management for organisationally mobile public employees [PELL03]
- Pellucid Agent Architecture for Administration Based Processes [LAC03A]
- Knowledge Management for Administration Processes [LAC04A]
- Reusable Agent-Based Experience Management and Recommender Framework[BAL04A]
- Model of Experience for Public Org. with Staff Mobility[KMGOV04]
- AgentOWL – Library which supports OWL agent memory model in JADE based on Jena[LAC05A]
- Methods for Presenting Ontological Knowledge to the User[LAC05B]
- Experience Management Based on Text Notes (EMBET)[LAC05C]
- e-Collaboration and Knowledge Sharing based on Text Notes[LAC05D]

The presented work was developed, used and evaluated in several international and national projects dealing with Semantics and Knowledge:

- K-Wf Grid IST Project[KWFGGRIDWEB] of the 6th Framework Program. The project focuses on knowledge management in a Grid environment. Results of our work were also used for writing the successful project proposal. The project was launched in September 2004 and some ideas and models described in this work are being used for development of several components in the project. EMBET architecture also represent one of the main components of the project.
- Pellucid IST Project [PELL02] Agent based knowledge management project. Most of our work was evaluated on this project. This project was successfully finished and evaluated in December 2004.
- Znalosti Project started in September 2004. This project is focused on creating a semantic search engine for chosen application domains. The infrastructure and libraries presented in this work as well as presented similarity based algorithms are used in this project.

2 State of the Art

This chapter describes the current state of the art in areas of interest:

- Semantic Web & Ontologies
- Knowledge & Experience management
- Semantic Annotation
- Knowledge Presentation

It also discusses the motivation and background for the research presented in the work. The last sub-chapter explains the goals of the work.

2.1 Outline of Research Environment

In order to have a comprehensive view of the presented work, it is necessary to examine terms such as experience, knowledge or ontology, and our understanding and limitations to this terms. These terms have been used very widely and it is necessary to make them more precise.

2.1.1 Knowledge and Experience Management Overview

Information is a fact or knowledge about a specific event or subject.

Knowledge means having information and understanding it through experience.

Human knowledge is based not only on facts, which are true or false but also on uncertain knowledge, which is partially true or false. Several methods can be used to represent such knowledge, e.g. probability measures, fuzzy logic or computing with words [WANG01]. Some methods are known to represent uncertain knowledge in agent systems by e.g. extended FIPA-SL language [FIPASL]; however, uncertain knowledge is still quite complicated and not directly understandable especially for computer systems. When using uncertain knowledge or knowledge where true and false facts are not strongly defined, computer systems cannot discover new facts in existing knowledge

base using basic logical operations. This is known as a fundamental problem of contradictory knowledge in computer systems [WOOL02].

This is why knowledge discussed within this work focuses only on strongly true facts. Such facts are structured and can be defined by ontologies. Using this solution, it is easier and straightforward for computer systems to understand knowledge and to discover new knowledge from existing knowledge. One could say that using such knowledge is sufficient for very limited area of applications, but by evaluation of different applications [PELLD4], where experience knowledge management is needed, it is useful to focus on knowledge based on facts rather than on uncertain knowledge.

2.1.2 Ontology Overview

Ontology is often explained in different ways. It can be understood as vocabulary, thesaurus or taxonomy. In the work ontology is understood in the way how the semantic web community defines it.

Ontology is description of problem domain, where entities of the domain, its properties and its relations are described.

A well known definition of an ontology is:

An ontology is a specification of conceptualization

Ontology has become a very important aspect in many applications to provide a semantic framework for knowledge management. Ontology is a set of definitions of content-specific knowledge representation primitives (classes, relations, functions and constants). Ontology represents the hierarchical structuring of knowledge about things by subcategorizing them according to their essential qualities.

The huge advantage of ontology is not in processing, but in sharing meaning, emergence and discovery of gaps and for improving a tacit knowledge transfer. Ontology may contain information in a specified declarative language, but it may also include unstructured or unformalized information expressed in a natural language or a procedural code.

Computer-based ontology provides formal and structured representation of domain knowledge. It is designed to serve as a raw material for computer reasoning and computer-based agents. The ontology provides a formally defined specification of the meaning of those terms, which are used by computer systems during their interoperation.

Ontology can be represented by UML[UMLWEB], any object oriented language, RDF[RDFW3C], DAML+OIL[DAML], OWL[OWLWEB] or any other representation which can define objects, properties and its relations. OWL – Web Ontology Language is considered by the semantic web community and also within the work as the best ontology representation. In the work OWL subset OWL-DL is used for knowledge representation.

2.2 Knowledge and Ontology

This chapter describes methods, techniques and representations used in the Knowledge Management area. Under knowledge we understand the knowledge built on facts, events which happened in the environment of problem domain. Knowledge in this form can be represented by ontologies. Especially focus is given on ontologies as defined in the Semantic web area.

2.2.1 Role of Semantic Web in Knowledge and Ontology Field

The Semantic Web is envisioned as an extension of the current web where, in addition to being human-readable using WWW browsers, documents are annotated with meta-information. This meta-information defines what the information (documents) is about in a machine processable way. The explicit representation of meta-information, accompanied by domain theories (i.e. ontologies), will enable a web that provides a qualitatively new level of service. It will weave together an incredibly large network of human knowledge and will complement it with machine processability. Various automated services will help the user achieve goals by accessing and providing information in machine-understandable form. This process may ultimately create extremely knowledgeable systems with various specialized reasoning services systems that can support us in nearly all aspects of life and that will become as necessary to us as access to electric power.

Ontologies offer a way to cope with heterogeneous representations of web resources. The domain model implicit in an ontology can be taken as a unifying structure for giving information a common representation and semantics. The ontology representation in semantic web started with XML, RDF and continued with DAML and currently is ending up with the OWL – Web Ontology Language.

HTML=> XML => RDF => RDFS =>DAML+OIL => OWL

Ontology in XML form thus comes from the semantic web community. Different knowledge representation comes from Artificial intelligence (AI) – Logic programming and Experts Systems. While in the semantic web you describe domain as objects related to problem domain, in logic you describe formulas or rules valid for such objects. The AI field is based on Lisp structures and Prolog like expressions.

2.2.2 The Ontology Related Methodologies

In recent years, some research groups have proposed methodologies guiding the ontology development process. Ushold's skeletal methodology was the first methodological outline proposed in 1995 on the basis of the experience gathered in developing the Enterprise Ontology. On the basis of the Toronto Virtual Enterprise (TOVE) project, Ushold and Grueninger (1996) described ontology development steps. A method to build an ontology in the domain of electrical networks was presented from Bernaras et al. (1996) as part of the Esprit KACTUS project. At the same time Methontology appeared (1996), extended in later papers. In parallel, the philosophical discipline of ontology is evolving towards an engineering discipline. In the following, we give a brief overview of these methodologies[TSWEB].

Interesting is also Methodology for Application Driven Ontology Management which is described in last sub chapter. The semantic structuring achieved by ontologies differs from the superficial composition and formatting of information (as data) afforded by relational and XML databases. With databases virtually all of the semantic content has to be captured in the application logic. Ontologies, however, are often able to provide an objective specification of domain information by representing a consensual agreement on the concepts and relations characterizing the way knowledge in that domain is expressed. This specification can be the first step in building semantically-aware information systems to support diverse enterprise, government, and personal activities.

Existing methodologies and practical ontology development experience have in common that they start from the identification of the purpose of the ontology and the need for domain knowledge acquisition. More information on ontology related methodologies can be find in Appendix.

2.2.3 Ontology Representation and languages

Ontology can be represented by UML[UMLWEB], any object oriented language, RDF [RDFW3C], DAML+OIL[DAML], OWL[OWLWEB] or any other representation which can define objects, properties and its relations. OWL – Web Ontology Language is considered by semantic web community and also by us as the best ontology representation.

Ontologies are not all built the same way. A number of possible languages can be used, including general logic programming languages like Prolog. More common, however, are languages that have evolved specifically to support ontology construction. The Open Knowledge Base Connectivity (OKBC) model and languages like KIF[KIFWEB] or FIPASL[FIPASL] (and its emerging successor CL -- Common Logic) are examples that have become the bases of other ontology languages. There are also several languages based on a form of logic thought to be especially computable known as description logics. These include Loom and DAML+OIL[DAML], which is

currently being evolved into the Web Ontology Language (OWL)[OWLWEB] standard. When comparing ontology languages, what is given up for computability and simplicity is usually language expressiveness, which is not always a bad deal. A language need only be as rich and expressive as is necessary to represent the nuance and intricacy of knowledge that the ontology's purpose and its developers demand[DENNY02].

The wide array of information residing on the Web has given ontology use an impetus, and ontology languages increasingly rely on W3C technologies like RDF Schema as a language layer, XML Schema for data typing, and RDF to assert data.

The W3C has put forward a very clear architecture for the SW (see Figure 2.1), described by Berners-Lee[BSWEB]. This architecture is cleanly layered, starting with the foundation of URIs and Unicode. On top of that sits syntactic interoperability in the form of XML, which in turn underlies what is like the data interoperability layer, RDF[RDFW3C] and RDF schemas. Those layers sum up most of the SW that is presently available in the implementation form.

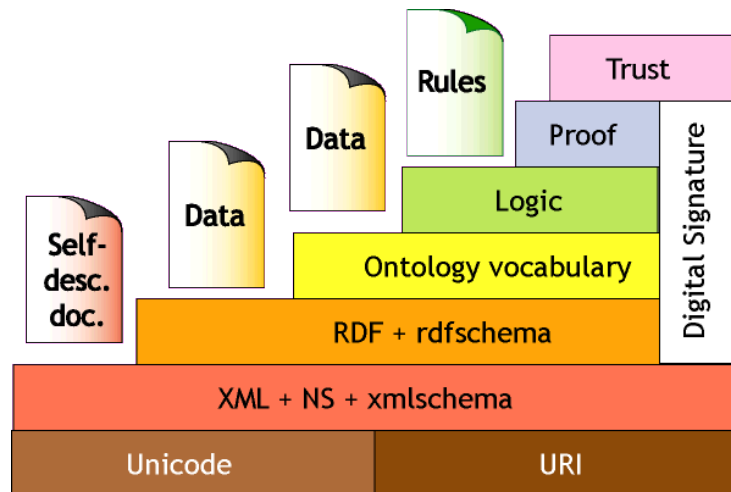


Figure 2.1: W3C architecture of Semantic Web

On top of RDF lie ontologies, which allow further description of objects and their interrelations. The W3C in conjunction with DARPA and the European Union is pursuing the development of languages in this area right now. Ontologies provide the ability to say "my world is like this" and are the foundation that will enable programs to reason about different worlds and environments and make connections between them. In the work all described models follow OWL-DL recommendation.

2.2.4 Semantic Annotation

Annotating is a writing-to-learn strategy for use while reading or rereading. Annotated text helps readers/processors reach a deeper level of understanding.

As described in the next paragraph, the annotation could be done by an annotation service (such as in the Annotea project) that stores and provides annotation without modifying an annotated document or annotation tags that reside in the same document as annotated text. The first approach enhances collaboration via shared metadata based Web annotations, bookmarks, and their combinations. In either case, the client must be aware of such description.

By using the appropriate standard for describing the annotation, the annotation should be utilized by web browsers, the AI agents, full-text search engines, inference engines, etc.

2.2.4.1 Annotation approach in Annotea Project

Annotea [ANOTEa] is a system for creating and publishing shareable annotations of Web documents. Built on HTTP, RDF, and XML, Annotea provides an interoperable protocol suitable for implementation within Web browsers to permit users to attach data to Web pages so that other users may, on their own choice, see the attached data when they later browse the same pages. The Annotea protocol works without modifying the original document; a user is not required to write access to the annotated Web page. The Annotea protocol is suitable both for annotation data primarily intended for human viewing as well as for annotation data expected to be consumed by other application programs, such as automatic classification tools, search engines, and workflow applications.

The Annotea project is part of the project Semantic Web Advanced Development (SWAD). Annotea enhances collaboration via shared metadata based Web annotations, bookmarks, and their combinations.

Annotea protocols

This document describes a protocol for manipulating sharable annotations of Web documents. The protocol specifies how data are to be exchanged between a conforming client - an application that produces and consumes annotation data - and a service - an application that stores the data and supports query capabilities to return some or all of the stored data.

While clients are generally expected to present annotations visually in a graphical user interface (such as Amaya client), the Annotea protocol does not require nor does it assume that the data is generated or consumed by a visual tool. The Annotea protocol is equally suitable for use by applications to exchange data for non-visual uses by 'attaching' it to Web pages where no

write access to the Web page is required.

An annotation service is an instance of an HTTP server that responds to the Annotea protocol at a specific URI. A single HTTP server might support multiple annotation services at the same host address; each such annotation service would have its own URI with the same 'host' portion.

An Annotea capable client, such as Amaya, interacts with one or more annotation services using a set of protocols that are built on top of the HTTP protocol as described by this document.

Annotea currently specifies three kinds of protocols:

- **Annotation protocols.** Describes the client-server interactions related to posting, downloading, updating, and deleting annotations.
 - Posting a new annotation: a client publishes a new annotation
 - Querying an annotation server: a client sends a query to a server and gets back annotations
 - Downloading an annotation or its body: after identifying an annotation, typically as the result of a query, a client retrieves all the data describing that annotation
 - Updating an annotation: a client modifies an annotation and publishes these modifications back to the annotation server
 - Deleting an annotation: a client deletes an annotation from the server
- **Reply-to-annotation protocols.** Describes the client-server interactions that support replies to annotations and thus promote threads of discussion.

To facilitate discussion about Web documents through the use of annotations, the Annotea protocol includes a separate class of resource called Reply. Replies are a mechanism that allows people to publish replies to annotations; for example, they allow someone to reply to a comment. Replies can also be made to other replies and thus promote threads of discussion. Moreover, as each reply is identified with a unique URI, the Annotea protocol also permits a client to annotate a reply.

- Posting a new reply to an annotation or to a previous reply: a client publishes a new reply
- Querying the annotation server: a client sends a query to the server and gets backs the replies
- Downloading a reply or its body: after identifying a reply, typically as the result of a query, a client retrieves all the data describing that reply
- Updating a reply: a client modifies a reply and publishes these modifications back to the annotation server

- Deleting a reply: a client deletes a reply from the server
- **Optimizing query requests.** Describes special URI query parameters to combine queries in a single HTTP request.
When a client is viewing a specific annotation, however, it may prefer to issue a single query to the annotation service to return all the information about annotations and replies of that annotation.
If a client knows that it is making a query for statements regarding an annotation, it may combine both query parameters in a single request. The query parameters that follow the Annotation fragment are indented below for better readability.

2.2.4.2 Annotation approach in Ruby Annotation

Ruby is the term used for a run of text that is associated with another run of text, referred to as the base text. Ruby text is used to provide a short annotation of the associated base text. It is most often used to provide a reading (pronunciation guide). Ruby annotations are used frequently in Japan in many kinds of publications, including books and magazines. Ruby is also used in China, especially in schoolbooks.

Unlike the Annotea system, the Ruby annotation is stored along with the text that has to be annotated as a XML tags. Some user agents might not understand ruby mark-up or may not be able to render ruby text correctly. In either situation, it is generally preferable to render ruby text, so that information is not lost [RUBY].

2.2.4.3 RDF Annotations

Blackbox content model

The idea here is that an annotation object is created with data about who made the annotation, when the annotation was made, and what the annotation is annotating (in Annotea's case, this can be an html/xml document or part of that document). The content of the annotation is invisible to RDF and represented as an html snippet.

RDF Blackbox content model

This is similar to the Annotea model but the comment fragment is a literal rather than (x)html.

Annotation as reification

Reification is process of creation the reified statements, where reified statement is an object of type `rdf:Statement` and also a resource representing the reification of a triple. A resource of type `rdf:Statement` must have exactly one each of `rdf:subject`, `rdf:predicate` and `rdf:object` properties.

An annotation is equivalent to a reification of a statement. As reification enables you to make the stating of an RDF triple a first class resource, the annotator can say anything about a resource in RDF and then attach 'creator' or some of such predicate to the reified statement node to represent the creator of the annotation [RDFANNOT]. Here, the problem of fake reification arises, because some RDF triples whose subject is the anonymous content node and whose predicate and object refer to the thing annotated.

2.2.4.4 Approach used in the work

While most of annotation solutions try to find and create object in the text or documents, in EMBET we need to detect ontology elements within existing application/domain ontology knowledge model. It means that by simple EMBET annotation engine we need to achieve the following objectives:

- Detecting Meta data from Text
- Preparing better structured data for later computer processing
- Structured data are based on application ontology model

2.3 Experience management

This chapter briefly summarizes bases of experience management theory and shows the bases and direction for approach used in the work. This summary is based on Bergmann book[BERG02] and [EXPEXT].

Experience management is simply the capability to collect lessons from the past associated to cases. There is a person who has a problem p which is described in a certain space, called the Problem Space (P). In the experience Management system there is Case-Lesson pairs, that is (c,l) , where is a Case Space (C) and a lesson space (L). We could have a single multidimensional vector in which we distinguish a case part and a lesson part. To be able to collect a lesson we must first devise a problem transformation function, that is a function that maps problem space to case space.

$$c = f(p)$$

This function should be mono valued. For the most simple cases this function is the identity function, because the developer of the Experience Management System (EMS) need to characterize the problem with the same attributes of the case. However, even in this situation, the important fact is that we are facing a semantic bridge. We do not store problems (which are infinite and cannot be predicted) but cases, which are a *formal representation of problems solved in the past, or situations happened in the past.*

To be able to re-use a particular lesson we should:

1. *Characterize a problem* (which is a separate process requiring effort)

2. *Transform the problem* from the space P to the space C.
3. *Choose* from the cases the most “useful” *lesson* from the case-lesson pairs stored in the database
4. *Apply* that lesson.

The point "Choose" introduce the function of utility that, for each problem (transformed) outputs a number which says the utility of the use of that lesson for that problem. Unfortunately this function is not known in advance, but could be verified only after the lesson is applied to the problem at hand. To overcome this Bergman introduces the similarity function and postulates that the most useful lesson will come from the most similar case. The problem, of course, is to develop this similarity function which is a quite complex problem.

In EMBET the described steps are recognized as:

1. User context detection from the environment which describes problem P
2. Our Model is described by ontology and Notes are stored with an associated context, which describes space C
3. Notes represent learned lesson L which is associated with space C (note context). The note context is matched with a user problem described by the detected user context. The user context is wider than the note context and as a result all applicable notes are matched and returned.
4. Applying the lesson is left to the user by reading appropriate notes.

2.3.1 Representation of Cases

There are three well known kinds of case representation:

- Unstructured Text
- Question and answer
- Structural

Unstructured text

This is the most simple case. The experience is simply stored in a form of documents (they could be manuals for call center operators, or procedure manuals). The experience retrieval is left entirely to the end users which will use simple text search utilities (Google like) to retrieve the article they want. Keywords search is the most popular technique for retrieval in this situation. There is no use to build this case base but it is naturally of little utility, unless the users are well experienced, because textual search approaches are mostly unable to capture semantics of the text. This approach is well studied when there are not too many cases at a time, and when each case has a short description with quite discriminating words occurring in the text. E.g.: frequently asked questions.

Conversational approach

In this way the experience is gathered in the form of Conversations. A case is represented through a list of questions that varies from one case to the other. The case author must define the order in which the user is asked to answer the questions during the consultation. Approach is very useful for domains where a high volume of problems must be solved over and over. However, the case base is organized manually which is a complex and costly activity therefore maintenance costs are high! Good for applications in which only a few questions are needed for decision making e.g. call center. Nevertheless, adding a new case could change the order or the meaning of the questions abruptly.

Structural

This is the most used and useful case representation. Cases are represented objectively inside the computer, in a "mathematical" form, that can be used by a computer. This approach is useful in domains where additional knowledge, beside cases, must be used in order to produce good results.

- *Attribute-value representation:* This is the most simple case representation. Every case is an entity with a set of attributes and all information contained in the case is represented through sets of attribute-values. No effort is used to put an "order" in cases: all cases are equal. No "ontology", no classes, no extras.
- *Object oriented representation:* This is the classical representation, there is an "ontology" of cases (even if the author does not use the word ontology but prefers to use the most common "taxonomy"). Every case belongs to a class and classes are related using the usual object oriented relations (is-a, has-a). Cases are represented as collections of objects, each of which is described by a set of attribute-value pairs. The structure of an object is described by a class that defines the set of attributes together with a type for each attribute. Classes are arranged in a class hierarchy that is suitable for complex domains in which cases with different structures occur.
- *Graph representation:* This is similar to the second, but we have cases which are described as topological entities, that is the shape of the case is important. This is for applications requiring the similarity in the "physical sense" too.
- *Predicate logic representation:* This is the most complex. Simply stated each case is represented with a series of predicates as is_Expensive() or Near(). The predicates can be composed and could model relations which are not simply modeled in the object oriented or in the topological representation.

We represented cases by object properties of the relevant object. This is based on the structured representation.

2.3.2 The similarity measures

To measure similarity we simply could use the distance. The less the distance (in the space C) the more the similarity. There are several distance measures, the classical: (Euclidean, Hamming...) to the most modern (fuzzy, sigmoid, step...) This naturally could be used if the attribute to be compared is numerical. If the attribute is not numeric something new must be found.

- Symbolic attributes
- Hierarchical attributes
- Object similarities
- Graph similarities
- Predicate similarities.

Symbolic attributes: For symbolic attributes the only alternative is to order the symbols (like in the ASCII code). The similarity is simply the numerical distance. If the attributes cannot be ordered, the only solution is to have a bidimensional array of distances.

Hierarchical attributes: we can distinguish similarities between the “leaves” of the taxonomy (which is a tree) and the nodes (which are similar to classes in the object oriented sense). Similarity between leaves is dependent on the similarity of the Least Most Common Ancestor, which is stored in a table. Similarity between inner nodes is dependent on the similarity on the semantic of the query which is given by the user. There are three semantic interpretations:

- *Any value:* the user wants a value and it does not matter which particular leaf node is retrieved.
- *Any value in case:* This is the opposite, in the case base a case is stored which is valid for each of the leaves in the tree, a kind of generalized case.
- *Uncertain:* the case (or the query) is related to a particular leaf node which we cannot specify. In this case the system must make an assumption on the particular case which is inside the tree (optimistic, pessimistic, average).

Object similarities are almost the same as what was given for the hierarchical attributes. The semantic is a bit different, though: in a taxonomy the leaves are “concrete” case of the abstraction of the tree, instead in an object oriented representation the leaves are “concrete classes” which are potentially an infinite set of cases (each with its own attributes).

Graph representation: the similarity could be only topological. In literature there are some algorithms that try to transform a graph into another. We could count the number of elementary operations to transform it and this would be a measure of similarity. The fact is that the cost of this operations is usually NP, and so it is only valid for small graphs.

Predicate similarities: Also for predicates we could have an engine that tries to

transform a case (that is, a list of predicates) into another list (another case) or that tries to infer the second from the first, counting how many elementary passages must be done. Again, these algorithms are very complex and costly.

We have used simple similarity algorithms in matching the resources context to the current context. If a resource context is found in a current context then contexts are similar. For certain applications other similarity measures can be more appropriate based on weight of context elements.

2.4 Knowledge Presentation

If knowledge is represented by ontology, naturally there are several ways how to present it. It is important in Knowledge Management Systems not only to capture, store or discover knowledge but also to return appropriate knowledge in a human understandable form. The personalization of knowledge is not covered in the overview. Focus is mainly on how to present knowledge in an ontological formalized form for the user.

Object Tree

The Object Tree is the base of ontology. A hierarchical object tree with properties can be presented to the users that is widely and commonly used method, for example in ontology editors such as Protégé[PROTWEB] or OilEd [OILED]. However, this approach is mostly understandable to experts who understand ontology. As an evidence of this fact can serve our experience with developing the so called “browse window”[LAC03C], which was powerful in returning knowledge from knowledge base but was not understandable by end users. As a result we can say that this approach is valuable indeed for people who understand the structure of knowledge in ontology, they can read it fast and get familiar with it. Browse window was developed as a part of the first Pellucid Prototype[LAC03A]. Regular users who do not know ontology and knowledge representation in computer systems are lost in such presentation and are not able to gather any information. Subsequently we can say this kind of presentation can be used for end users of knowledge system only in limited ways.

Graph

If we talk about ontology languages based on RDF[RDFW3C] such as DAML, DAML+OIL[DAML], OWL[OWLWEB] or RDF itself, the natural way how to present an ontology to the user is drawing of graphs, where arches and points can have assigned certain textual, colored or other values and names. The best example is UML[UMLWEB] itself, but other representations exist too, like Ontoviz Tab[ONTOVIZ] in the Protégé editor (the diagrams of ontology models in the work are developed by Ontoviz) or Cartoo Technologies products[CARTOO]. A graph structure can express almost all information

encoded in ontology because values could be assigned to arches and points and other data such as the width of arch or the color can be displayed. If KM system discovers new facts, users usually require explanation of this new explored fact, this was confirmed in most of knowledge management projects. A graph sometimes can explain to a person the reason of creating a new fact, by connections to other facts, or reason for returning of knowledge to the user. The KM system is usually not able to create good sentences, which would explain the reasons and ways of thinking but by a graph this can be understood, however reading of graphs require flexible users and not everybody can understand graphs. In addition as far as we know there is no general tool available for such graph creation, but we see this as an area of future improvement in knowledge presentation.

XSLT Transformation

RDF based ontology languages are also XML[XMLW3C] based and thus ontology information can be simply presented by XSLT[XSLW3C] transformation sheets which transform XML to HTML. This approach was used and evaluated in the Pellucid and K-Wf Grid project and it is described in chapter 5. In XSLT, any ways for graphical representation of knowledge encoded in ontology can be defined. On the other hand it involves a quite extensive customization effort. XSLT style sheets must be created for each ontological element or at least for each type of ontology element. XSLT is a commercial standard which is used in many e-commerce and e-business applications, accepted by a community and thus customization of presentation can be done by regular developers. Such implementation does not require knowledge experts. The core of the KM system just needs to return proper XML of a requested ontology element.

2.5 Conclusion of State of the Art

An increasing number of companies are realizing that their own intranets, email communication or documents are valuable repositories of corporate information, but without understanding of how to apply it effectively this information is likely to be useless. Knowledge management is concerned with the acquisition, maintenance and evaluation of the knowledge of an organisation, but demands tools that foster productive collaboration while capturing, representing and interpreting the organisation's knowledge resources. This kind of knowledge can enhance adoption of the best practice, highlight new business opportunities, and speed up the identification of produced savings market dynamics and sales opportunities. At present, companies employ largely manual processes, though initial applications are being developed.

When developing any computer based system, it needs to work with different kinds of data. Recently Relational Database Management Systems (RDBMS)

are used for most of the information systems. RDBMS are the most appropriate, when problem domain of information system is well described and does not change over time. Object Database Management Systems (ODBMS) are more appropriate in the information system where all kinds of data extraction combination are necessary. However, updating indexing and other DB operations are much slower in OODBMS compared to RDBMS and such solutions are not too stable. Using ontology and ODBMS makes information system flexible and customizable. Such information system represents well the knowledge data and their information structure and can be built on ontology or RDF based solutions e.g. RDF stores or Jena library.

When developing Knowledge Management or Experience Management System, a developer need to create a knowledge model and design the system. Methodologies such as CommonKADS exist but they are not tied with any concrete tools. Knowledge or Experience Management Systems are mostly developed directly for a particular application and are not applicable on different problem domain or they are applicable with a lot of customization effort.

For these reasons we have developed an extensible model, modeling (also customization) methodology and Experience Management System, which can be applicable in many problem domains. The work objectives are thus concentrated on the vision, which we have based on the current state of the art and trends in the field. This vision is to support theory, modeling and software libraries and tools for Knowledge and Experience based information systems.

2.6 Objectives of the work

The concrete objectives of the work are:

- Design of Generic Ontology model for Experience management with extension for different application domains
- Design of Modeling Methodology for extension of the Ontology based Knowledge Model for different applications
- Design & Development of generic and customizable EMBET Experience Management software with the Ontology Knowledge Model
- Evaluation of results on a real pilot operation

3 Used Methodologies and Tools

In this chapter main methodologies and tools used within the work are described.

3.1 Methodologies

The chapter discusses methodologies used in the knowledge management system design and a life cycle. Some of them focus strictly on Ontological approach. The formal methods for describing ontology based models are presented.

3.1.1 Unified Modeling Language (UML)

The Unified Modelling Language – UML[UMLOMG] - is OMG's [OMGWEB] most-used specification, and the way the world models not only application structure, behaviour, and architecture, but also business process and data structure.

The OMG's Unified Modelling Language UML helps to specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements. (UML can be used for business modelling and modelling of other non-software systems too.) Using any of the large number of UML-based tools on the market, future application requirements and design a solution that meets them can be analyzed, representing the results using UML's standard diagram types.

UML defines twelve types of diagrams, divided into three categories: Four diagram types represent a static application structure; five represent different aspects of dynamic behaviour; and three represent ways you can organize and manage your application modules.

Structural Diagrams include

- Class Diagram, Object Diagram, Component Diagram, Deployment Diagram

Behaviour Diagrams include

- Use Case Diagram (used by several methodologies during requirements gathering); Sequence Diagram; Activity Diagram; Collaboration Diagram; Statechart Diagram.

Model Management Diagrams include

- Packages; Subsystems; Models.

Use case and class diagrams were for design of presented EMBET System.

3.1.2 CommonKADS Methodology

The CommonKADS methodology[CKADS] comprises a collection of structured methods for building KM systems, concentrating on the modeling activity and proposing the development of a set of models in its concrete organizations and application contexts:

- the organizational model – describes the organizational function and structure;
- the task model – describes tasks required to operate;
- the agents model – describes the capabilities required from agents;
- the communication model – describes communication between agents;
- the knowledge model – describes expertise required to operate focusing on knowledge of the domain and inference processes;
- the design model – describes design of the KM system.

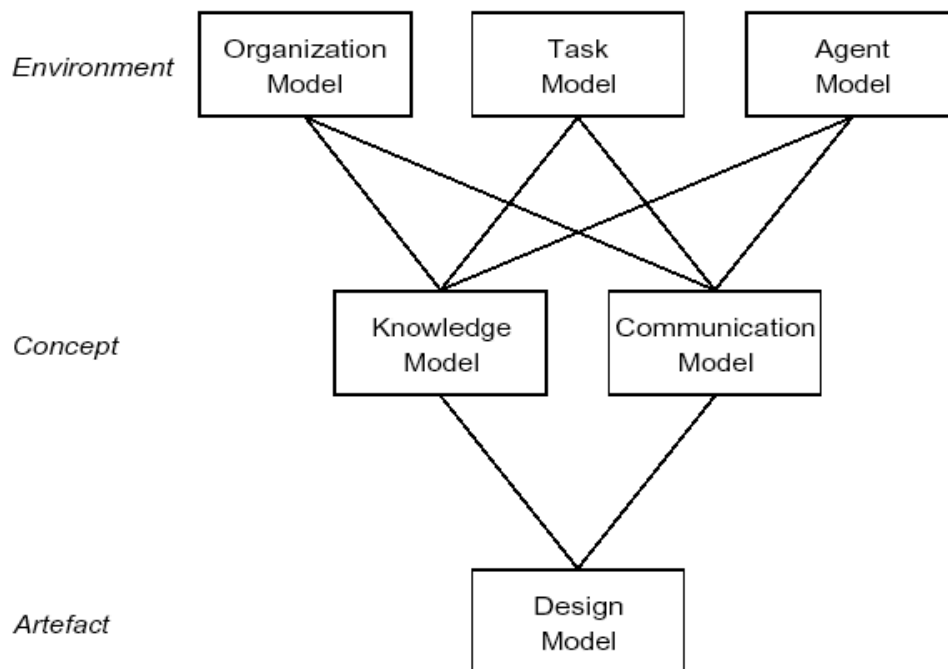


Figure 3.1: CommonKADS Models

The CommonKADS modeling process is of a great use in developing KM system within a particular domain. Presented modeling methodology is build on CommonKADS models.

3.1.2.1 Protégé as a Tool for CommonKADS

CommonKADS methodology is general methodology which is not linked with any concrete tool. Such tool also does not exist but it was recognized [PROTKADS] that a protégé ontology editor can support most of the modeling methodology. Protégé thus can be used as a tool for constructing CommonKADS knowledge models. The CommonKADS knowledge model is specified as an ontology in the Protégé specification formalism, and define a number of visualizations for the resulting types. The studies show that usage of Protégé as a "meta CASE" tool is to a large extent feasible. Moreover use of Protégé can show how the concrete, operational approach of Protégé and the highly methodological approach of CommonKADS can be combined successfully to provide the middle-ground tool that reduces the gap between a conceptual model and a usable knowledge model. The work applied the methodology approach of CommonKADS with Protégé tool.

3.1.3 Formal Methods Used for Ontology model description

In this chapter we specify formal methods for ontology models description based on graph representation and Description Logic (OWL-DL). Such model description was used in the work to depict developed ontology models.

3.1.3.1 Ontology models Using Description Logic

Ontologies in the semantic web area can be described by Description Logic.

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male	$C_1(x) \wedge \dots \wedge C_n(x)$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer	$C_1(x) \vee \dots \vee C_n(x)$
complementOf	$\neg C$	\neg Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} \sqcup {mary}	$x = x_1 \vee \dots \vee x = x_n$
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor	$\forall y.P(x, y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	\exists hasChild.Lawyer	$\exists y.P(x, y) \wedge C(y)$
maxCardinality	$\leq_n P$	≤ 1 hasChild	$\exists^{\leq n} y.P(x, y)$
minCardinality	$\geq_n P$	≥ 2 hasChild	$\exists^{\geq n} y.P(x, y)$

Figure 3.2: OWL Constructors

Not all ontology models can be specified using Description Logic (DL) but within the work we used only OWL-DL based models for the reasons explained in State of the Art chapter. Figure 3.2 and Figure 3.3 show conversion tables[HORR04] between OWL-DL definitions and DL. In the work we use such formalism[DLBOOK] to describe ontology models.

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human \sqsubseteq Animal \sqcap Biped
equivalentClass	$C_1 \equiv C_2$	Man \equiv Human \sqcap Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$ Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} \equiv {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter \sqsubseteq hasChild
equivalentProperty	$P_1 \equiv P_2$	cost \equiv price
inverseOf	$P_1 \equiv P_2^-$	hasChild \equiv hasParent ⁻
transitiveProperty	$P^+ \sqsubseteq P$	ancestor ⁺ \sqsubseteq ancestor
functionalProperty	$T \sqsubseteq \leq 1P$	T $\sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$T \sqsubseteq \leq 1P^-$	T $\sqsubseteq \leq 1$ hasSSN ⁻

Figure 3.3: OWL Axioms

3.1.3.2 Graph Ontology representation

On Figure 3.4 is shown a well known example of an ontology use. Ontology defines parent-child relations. Based on an ontology model, information “Mary is mother of John” is stored. If one will ask the system who is ParentOf John system can answer Mary, even if such information can be only inferred and it is not directly stored in the system.

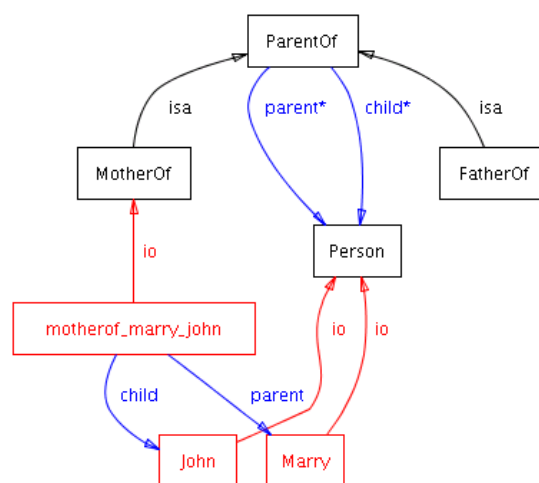


Figure 3.4: Example of Parents Ontology

The work applies similar ontology diagrams as on Figure 3.4. This is understood as a formal representation of ontology described by graph. Black boxes represent ontology classes, black arrows stand for relations between classes, mostly inheritance relations, expressed by words “is a”. Property relations are represented by blue arrows with name of property and cardinality that is mostly multiple “*”. Red boxes denote ontology individuals and red arrows relations of an individual to ontology class with associated letters “io”. In addition such ontology models can be formally described using OWL-DL or Description Logic as explained above, but it is more understandable for readers if it is in a form of graphs. For this reason such approach is used within the work. Such graphs can be generated using Ontoviz plug-in [ONTOVIZ] for Protégé.

3.2 Tools

The chapter contains a tool list with description, which had been used for modeling and developing software depicted in the work.

3.2.1 Protégé Ontology Editor

Protégé is an open-source development environment for ontologies and knowledge-based systems. It is a tool supporting the construction of ontologies and it also provides an application platform for knowledge based systems and libraries for application building. Protégé is developed at the Stanford University School. It is the best-known ontology editor with plug-ins that supports OWL and enables [PROTWEB]:

- loading and saving OWL and RDF ontologies,
- editing and visualizing OWL classes and their properties,
- defining logical class characteristics as OWL expressions,
- executing reasoners such as description logic classifiers,
- editing OWL individuals for Semantic Web markup.

Protégé has flexible architecture and is easy to configure and extend. Protégé has an open-source Java API for the development of custom-tailored user interface components or arbitrary semantic web services. There are several other ontology editors such as OilEd, OntoEdit or DUET. From our experience Protégé with its plug in architecture gives much wider possibilities. Protégé has many plug-ins and several are important for work done within the work.

- OWL Storage plug-in allows to create and manipulate OWL ontologies.
- OntoViz plug-in [ONTOVIZ] was used for visualization of ontology to graphs. All ontology structures (figures) in the work are created using this plug-in. Result graphs are similar to UML diagrams.

3.2.2 Jena library

Jena is a java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine. Jena is open source and grown out of work with the HP Labs Semantic Web Program.

The Jena Framework includes:

- A RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- RDQL a query language for RDF

The idea of RDQL is to provide a data-oriented query model so that there is a more declarative approach to complement the fine-grained, procedural Jena API.

RDQL is an implementation of the SquishQL RDF query language[SQISHQL], which itself is derived from rdfDB [RDFDBWEB]. This class of query languages regards RDF as triple data, without schema or ontology information unless explicitly included in the RDF source.

RDF provides a graph with directed edges - the nodes are resources or literals. RDQL provides a way of specifying a graph pattern that is matched against the graph to yield a set of matches. It returns a list of bindings - each binding is a set of name-value pairs for the values of the variables. All variables are bound (there is no disjunction in the query).

The jena/db module provides an implementation of the Jena model interface but with the ability to store and retrieve RDF statements using a database. It currently supports MySQL, Oracle and PostgreSQL for persistent storage and runs under Linux and WindowsXP.

The Jena2 inference subsystem is designed to allow a range of inference engines or reasoners to be plugged into Jena. Such engines are used to derive additional RDF assertions which are entailed from some base RDF together with any optional ontology information and the axioms and rules associated with the reasoner. The primary use of this mechanism is to support the use of languages such as RDFS and OWL which allow additional facts to be inferred from instance data and class descriptions. However, the machinery is designed to be quite general and, in particular, it includes a generic rule engine that can be used for many RDF processing or transformation tasks.

The Jena is used in presented work as API for OWL manipulating and storage.

4 *Ontology Model and Modeling Methodology*

In this chapter we will cover the following goals of the work:

- Design of Generic Ontology model for Experience management with extension for different application domains.
- Design of Modeling Methodology for extension of the Ontology based Knowledge Model for different applications

4.1 *Specification of Ontology Model for Experience Management*

Our model is based on six main elements: Resources, Notes, Actions, Actors, Context and Events. Model is specified using description logic (DL). Figure 4.2 shows formal graph representation using the same terms as model described in this chapter.

Resource class stands for all the resources in the problem domain environment. Many subclasses representing resource types can be defined as a part of customization of the model. Important subclass of *Resource* is *Actor*.

$$Actor \subseteq Resource$$

$$\{actor\} \in Actor$$

Actor class denotes actors in the environment. *Actor* individuals can take an actions $\{action\}$ which are individuals of *Action* class.

$$\{action\} \in Action$$

Task class symbolizes done tasks or tasks need to be done in the environment but depending on application it can represent problems too. *Domain* class stand for application domain extension of ontology, while all extensions should be subclasses of this class. *Context* class represents context of actors, environment or else.

$$Resource \subseteq Context$$

$$Action \subseteq Context$$
$$Domain \subseteq Context$$
$$\{domain\} \in Domain$$
$$Task \subseteq Context$$
$$Context \supset Resource \cup Action \cup Domain \cup Task$$
$$\{context\} \in Context$$

Important property of *Task* is *domain*. This symbolizes domain related application concepts. It was identified that such connection is useful for setting appropriate knowledge management algorithms for actor context and resource updating.

$$Task \subseteq domain.Task(\{domain\}) \cap Context$$

Event class represents events in the system. *Event* individual $\{event\}$ is $\{action\}$ taken by $\{actor\}$ on particular $\{resource\}$ in the situation described by $\{context\}$. Properties of *Event* class are *context.Event*, *resource.Event*, *action.Event*, *actor.Event*.

$$Event \subseteq$$
$$action.Event(\{action\}) \cap$$
$$resource.Event(\{resource\}) \cap$$
$$actor.Event(\{actor\}) \cap$$
$$context.Event(\{context\})$$
$$\{event\} \in Event$$

A special type of *Actor* is *User*. The user is used for Software representation of the user of the system.

$$User \subseteq Actor$$
$$\{user\} \in User$$

When actions such as creating, updating or deleting of resources are performed in the system, events containing those kinds of action are stored and evaluated in the system.

$$\{aUpdate, aDelete, aCreate\} \in Action$$

Other actions relevant to the experience management model are confirming or voting of related resources.

$$\{aConfirm, aVote\} \in Action$$

Important subclass of *Resource* is *Note*, which represents experience gathered by user experts of the system. Properties of *Note* class are *context.Note*, *title.Note*.

$$Note \subseteq$$
$$title.Note(String) \cap$$

$$\begin{aligned} & context.Note(\{context\}) \cap \\ & Resource \\ & \{note\} \in Note \end{aligned}$$

Actor class consists of important properties: *context.Actor*, *resource.Actor*. The *context.Actor* represents actor current context. The system or application environment can be found based on stored events. Events model the environment state. The current state of the environment or actor related to environment/context is thus effected by relevant new events. The *resource.Actor* property stands for all current resources of the actor. This resources (mostly Notes representing experience) are results of actors intentions or objectives. Such resources are thus dependable on current actors environment state (*context.Actor*). Functions/algorithms for context and resource updating are specified bellow. Actor resources are in case of experience management filled with *{note}*, which represents experience.

$$\begin{aligned} Actor &\subseteq \\ & resource.Actor(\{resource\}) \cap \\ & context.Actor(\{actor\}) \cap \\ & Resource \\ context.Actor(\{actor\}) &= \\ & f_C(\forall \{event\}; actor.Event(\{actor\}) \in \{event\}) \\ resource.Actor(\{resource\}) &= \\ & f_R(contextActor(\{actor\})) \end{aligned}$$

This is where the brain of the system is located. An advantage of such model is that it enables to achieve better results when such algorithms are changed in the future, using the same model and data. Due to storing all events we can model the environment in any moment from past and process it later from any starting point with improved algorithms for context and resource updating.

4.1.1 Extension of model related to Semantic Annotation and text processing

Important part of the model is semantic annotation extension (Figure 4.1). This extension contains one class Pattern with several properties.

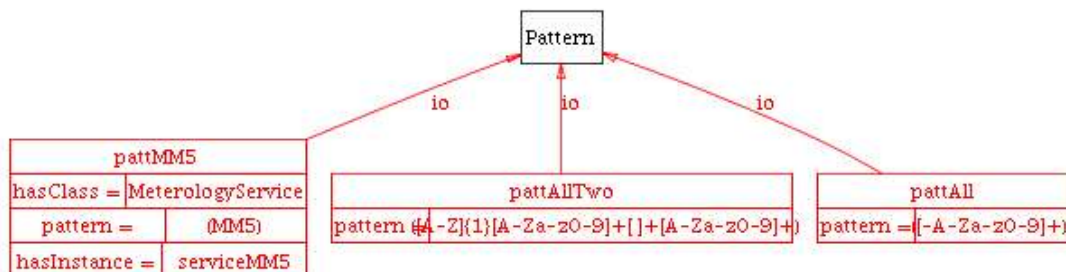


Figure 4.1: Pattern Ontology

Pattern class represents regular expression patterns which are used to annotate plain text with ontology elements. *Pattern* individual $\{pattern\}$ is evaluated by a semantic annotation algorithm. On Figure 4.1 we can see several simple patterns which can detect ontology individuals by matching String properties of such individuals. Properties of *Pattern* class are *hasClass.Pattern*, *hasInstance.Pattern*, *pattern.Pattern*.

$$\begin{aligned}
 &Pattern \subseteq \\
 &\quad hasClass.Pattern(Thing) \cap \\
 &\quad hasInstance.Pattern(\{Thing\}) \cap \\
 &\quad pattern.Pattern(String) \\
 &\{pattern\} \in Pattern
 \end{aligned}$$

The main property is *pattern.Pattern*, which is string representation of regular expression.

4.1.2 Workflow and Organizational related Extensions of Model

This chapter discusses formal specification of the Workflow related extension of Experience Management model using the description logic.

A new top element *Organization* was added representing the environment for the experience management and has properties *resource.Organization* organizational resources, *task.Organization* standing for organizational tasks and *actor.Organization* symbolizing organizational actors.

$$\begin{aligned}
 &Organization \subseteq \\
 &\quad resource.Organization(\{resource\}) \cap \\
 &\quad task.Organization(\{task\}) \cap \\
 &\quad actor.Organization(\{actor\})
 \end{aligned}$$

New subclasses of the *Actor* are modeled. The *BusinessEntity* represents organization(s) or contacts. The *Person* stands for any human contact. The *User* denotes an employee in an organization, which is a user of the system and the main actor.

$$\begin{aligned}
 &BusinessEntity \subseteq Actor \\
 &Person \subseteq BusinessEntity \\
 &User \subseteq Person
 \end{aligned}$$

New subclasses of resources representing workflow in an organization were introduced because such a model was proved to work well in organizations with workflow administration processes. *WfInstance* represents a Workflow Process Instance, can be understood as a problem solved in an organization and it is related to the *Domain* with its domain *WfInstance* property. The *WfActivity* is a task within the business process.

$$WfResource \subseteq Resource$$

$$WfInstance \subseteq WfResource$$
$$WfActivity \subseteq WfResource$$
$$WfActivity \subseteq Task$$
$$WfInstance \subseteq \\ domain.WfInstance(\{domain\}) \cap \\ WfResource$$

Such extension of the model was successfully used in the Pellucid IST project and can be viewed as extension for organizations where business processes are performed via the workflow management system.

4.2 Defined Modeling Methodology

Developed methodology integrates several parts of different methodologies. It follows the CommonKADS methodology. However CommonKADS is not tied with any modeling tool, knowledge representation or ontologies. CommonKADS is divided into two main parts:

- knowledge model based on other three models:
 - organizational or environmental model
 - agent model
 - task model
- and design of system. In our case design is presented in chapter 5.

We present knowledge based on OWL ontology and we model it in the Protégé ontology editor. Thus defined methodology for knowledge model is similar to [PROTKADS]. Ontology modeled with Protégé reflects CommonKADS models.

When using this methodology, good results can be archived only after several iteration of the process and remodeling after first developing, use and evaluation of the first system version. The iteration method is common and used in all knowledge management methodologies.

As already mentioned, when developing knowledge model for an application we follow first three CommonKADS models:

- Organizational or in our case Environment Model
- Task Model
- Agent or Actor Model

When modeling the knowledge model we have to extend the generic experience management model (Figure 4.2) with new elements and relations. This model is the same model as described using description logic in the previous chapter. For better understanding of models we will use such graph representation of models within the work.

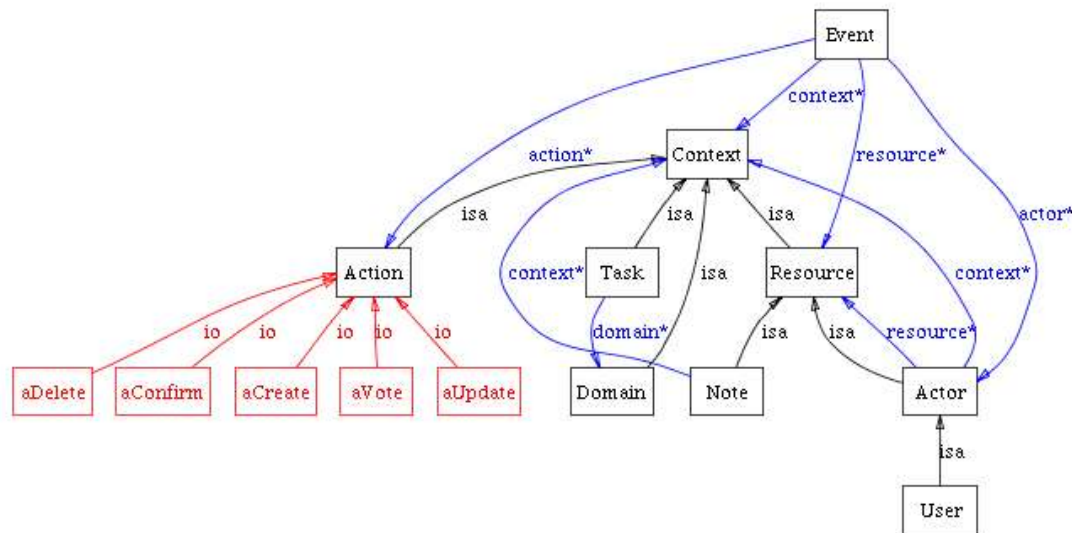


Figure 4.2: Basic Ontology for Knowledge Modeling

Environment Model models the environment of application and the environment of actors. Thus modeling a resource element needs to be extended with new types of resources such as documents, contacts, goods or services. Domain elements need to be extended with all application/domain specific elements (possibly including resources), which models problem domain and actors can use them to make decisions or accomplish a task. The resources considered as results of actor goals have to be modeled as well. Usually results of actors goals can be considered as resources mostly in case of experience management. These results are relevant experience in the form of the text notes. Properties of defined entities and their relationships need to be established as an important part of the model.

Actor Model models actors performing tasks and actions. The actor can be a human, a software or another entity performing actions and being able to be monitored by the system. An important part of the actor model is actor's context which defines current actor's environment and actor's resources which are the results of accomplished goals. Thus a part of the Actor model are definitions of 2 algorithms for actor context f_C and resources f_R updating defined in actor/agent model. Relevant algorithms were developed and described in chapter 5, but can be changed based on application needs.

Task Model models tasks, processes and actions relevant to actors. An important property of task is Domain. Tasks are often related to some resources or other domain entities and we need this relation to define functions for updating the actor context and resources mentioned above.

After defining elements, properties and their relations – defining of ontology of

mentioned models should be iteratively refined to include all needed elements. Refining the model requires consideration whether combination of actions, resources and actors can be captured as events and whether all possible events can be created from defined ontology elements. Results of models is knowledge model which consists of

- Ontology developed in Protégé which can be exported in OWL format.
- Algorithms for each actor (user) based on preferences (often algorithms are similar or same) which updates actors' context f_C and resources f_R .

4.3 Conclusion

Originally we have developed the presented model and methodology as a knowledge model of agent in multi agent systems and then it was applied for Experience Management in the Pellucid project. Within the K-Wf Grid project this model was extended and redesigned and it is in use as presented in this work We believe that this model is powerful for experience management which was proved in mentioned projects.

5 Design & Development of EMBET System

This chapter covers the following objective of the work:

- Design & Development of generic and customizable EMBET Experience Management software with Ontology Knowledge Model

In EMBET experience is understood through notes entered by the user. Such form of experience is the most understandable for humans, but it can be partially understood by a computer system. A computer system needs to return experience in a context where experience is relevant. Thus we need to model a context of environment and capture and store a context of each entered note. From the text of the note we need to detect what kind of current context of a user is important for the note and what kind is not. We can do this with user's assistance, where the user will get a pre-checked list of the detected context and will change it or submit the detected context unchanged. When the context is properly assigned, the note can be displayed to the same or other users in the same or similar situation.

5.1 Overview of the approach for Experience Management

Building an Experience Management (EM) solution, it is necessary to capture, store, capitalize and reuse experience:

- Capture: it is necessary to capture information about what (people, computer systems, an organization – any entity performing actions or tasks) are doing. Thus we need to capture Events, and also notes entered by users. Events are composed of actions, resources, actors and a context. By modeling these entities an important part of the environment or an organizational model is created. CommonKADS methodology is followed and a Protégé ontology editor is used for modeling the following standards models: an organizational model (mainly resources), the agent model (Actors) and a task Model (tasks, e.g. workflow tasks, problems to solve)
- Store: It is important to store information: captured events as well as meta data about all entities affected or affecting this events. *Store* can be

performed using the EMBET memory with connection to the Jena storage back end. Main stored data are event instances captured in the previous phrase. A special kind of events are “create” events, when a new resource instance is introduced into the model.

- Capitalize: To have appropriate intelligence in the system it is necessary to find patterns in the captured events, to explore or better define knowledge. Capitalization often means not exploring new knowledge but exploring new patterns, which can be then transformed by a knowledge engineer into experience or knowledge. In our case capitalization is used to detect context of notes and user context in a text problem definition using semantic annotation.
- Reuse: To reuse experience or knowledge, experience must be defined first. It is impossible to define experience as it is something as a point in geometry, which is undefined and can remain undefined. However, it is useful to have a partial understanding of experience, which is suitable for most of related application domains. Thus we use Text Notes entered by a user, which can be taken as conveyor of experience in a developed model. *Reuse phrase* is based on algorithms for user context f_C and resources f_R update. If an actor requires experience, experience must be returned in certain context, thus actor's current context needs to be updated to know the actor's current environment and actor's resources need to be updated too. In the experience management we define a special type of resources called Note.

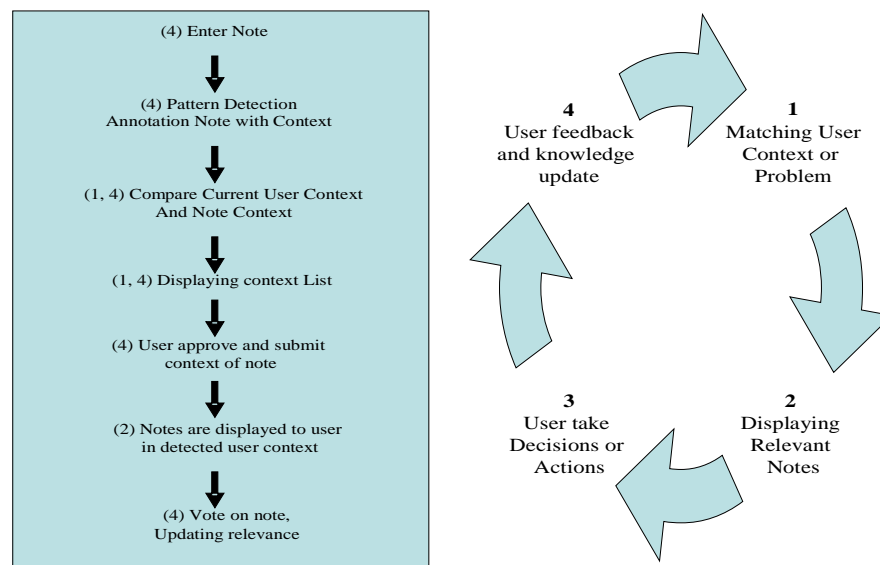


Figure 5.1: Experience Management Cycle of EMBET

For the proper experience or knowledge management we need to have a closed knowledge cycle (see Figure 5.1, on right side). The most crucial point in experience and knowledge management systems is step 4 – “User feedback on knowledge and knowledge update” (Figure 5.1 on the left side).

Updating of knowledge and experience in EMBET consists of the following steps:

- A user submits a Note, if s/he thinks s/he has relevant knowledge for the current or past situation
- The text of the note is processed and patterns of a semantic annotation are matched and notes are annotated with knowledge concepts. Context concepts are detected.
- Compare the current user context with the Note detected context.
- Displaying a Context List to a user with pre-selected context items detected in the Note.
- The user approves the context and submits the context of the note.
- The notes can be later displayed to users in a similar detected user context
- Users can Vote on Note relevance. The relevance of the note is then updated. If a user does not get relevant knowledge in any situation, and s/he gathers experience, s/he may enter a new note with his/her learned lesson.

The closed knowledge cycle means that experience is managed in the system the way that system together with users controls all phrases: capture capitalization and reuse of information and knowledge and that all the phrases are closed in the system and managed by a regular user of the system. The knowledge management systems often allow a user to see or search for knowledge but introduction of knowledge and the user feedback on knowledge is complicated and can be managed by knowledge experts only.

5.2 Design of the EMBET System

This chapter discusses use cases and architecture of the system. Main functionality and components are described.

5.2.1 Use Cases

UML use case diagram (Figure 5.2) as well as text description of use cases is the content of this chapter.

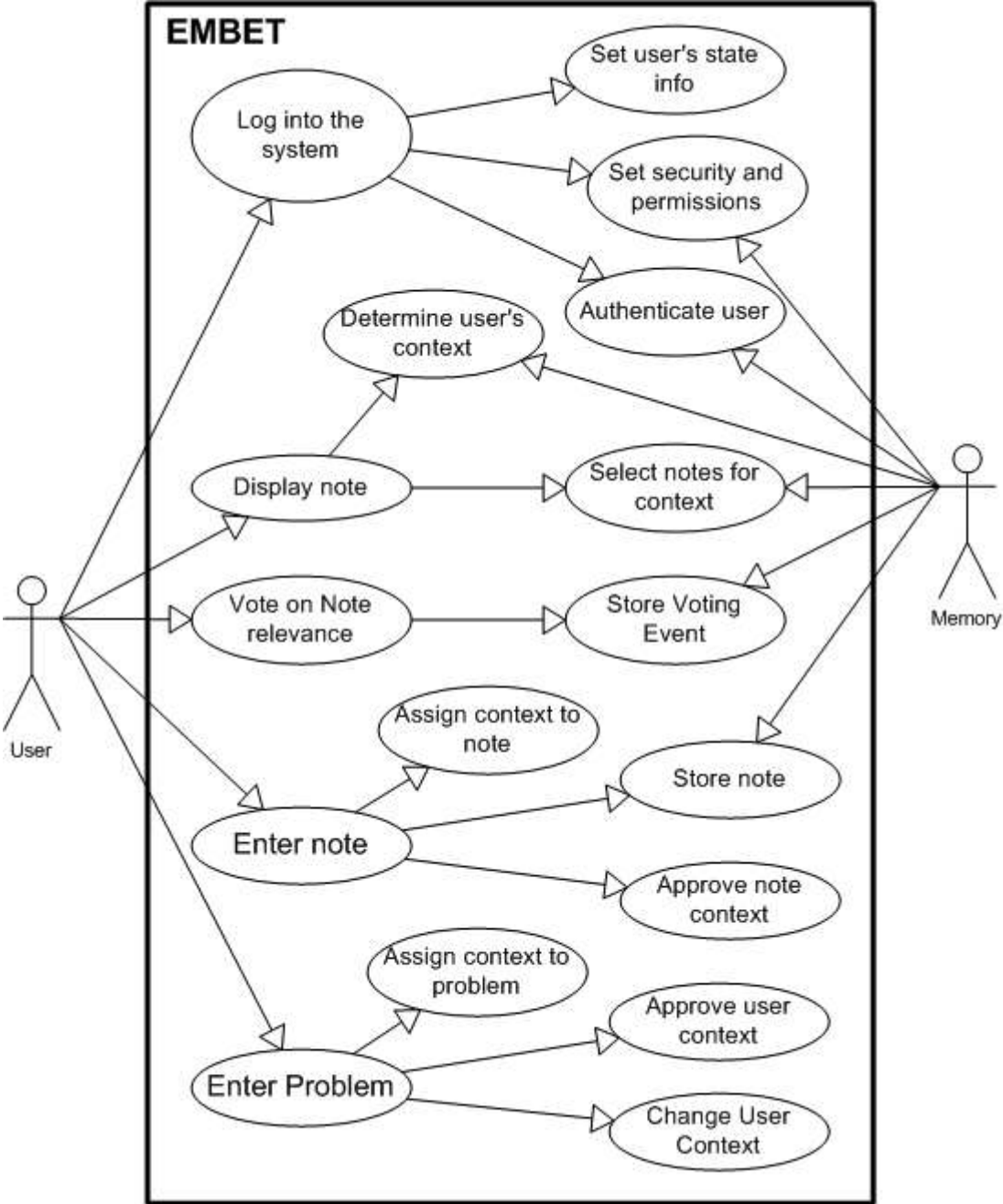


Figure 5.2: EMBET UML use cases diagram

5.2.1.1 UML Use Case 1 – Log into the System

ACTORS: User, Memory

GOAL: Log into the System

- PRIMARY FLOW:
- User enters and submits his login/password
- <<uses>> Authenticate a user by checking privileges in authorization database
- <<uses>> Read security settings and permissions for the user in the authorization database
- <<uses>> Set user's status information

5.2.1.2 UML Use Case 2 – Problem/context specification

ACTORS: User

GOAL: Define user context

PRIMARY FLOW:

- A user wants to start a new task so he needs to specify his context if it is not already detected by external systems.
- A user enters a textual problem description
- <<uses>> EMBET core tries to determine what is the context of such problem and detects an application related objects/elements relevant to text problem description
- <<uses>> A user inspects and approves the problem context proposed by EMBET Core, possibly modifies the context and approves the new user context
- <<uses>> Change User context according to approved problem context

5.2.1.3 UML Use Case 3 - Display a Note for a Context

ACTORS: User, Memory

GOAL: Display relevant notes (experience) in user context

PRIMARY FLOW:

- User context had changed
 - Because changing of context was detected by external applications

- Because a user described his/her problem (see previous Problem specification use case)
- <<uses>> Determine User's Context from the Memory
- <<uses>> EMBET Core Selects Notes for the current User Context
- EMBET GUI Displays the Notes for the Context

5.2.1.4 UML Use Case 4 – Voting on Notes

ACTORS: User, Memory

GOAL: Voting on displayed notes (experience) relevance

PRIMARY FLOW:

- A user reads notes for his/her problem context and thinks that some of displayed notes are not relevant or are very relevant
- A user votes on notes relevance
- <<uses>> EMBET Core store the Note/Relevance pair to Memory which is used in future note display

5.2.1.5 UML Use Case 5 - Submit a Note to a Context

ACTORS: User, Memory

GOAL: Attach a note (experience) to application related context

PRIMARY FLOW:

- A user has finished an action and potentially has finding experience which he wants to share with other users
- A user enters a note
- <<uses>> EMBET core tries to determine what is the context for which the User wants to write a note
- <<uses>> A user inspects and approves the context proposed by EMBET Core, possibly modifies the context and approves the context
- <<uses>> Store Note submits the Note/Context pair to Memory

5.2.2 Architecture and Technology

Architecture of EMBET consists of 3 main elements:

- Core of the system
- Graphical User Interface (GUI)
- System Memory

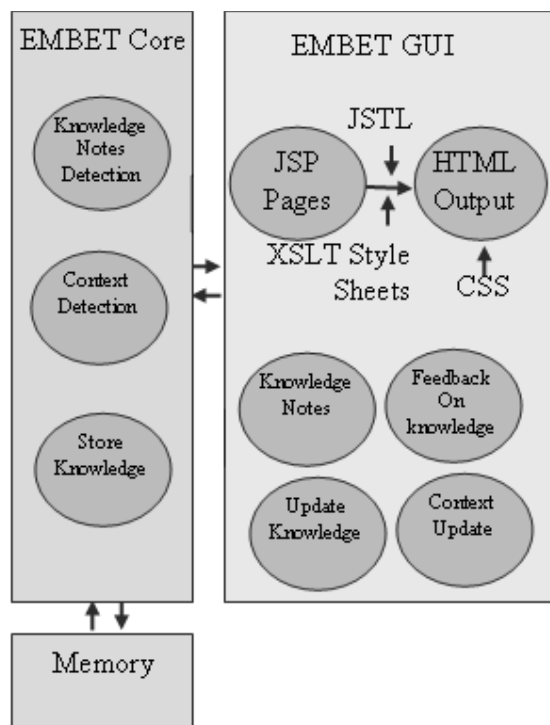


Figure 5.3: EMBET Architecture

EMBET Core provides the main functionality of EMBET. It determines a User context and searches for the best knowledge (in a form of text notes) in its Memory. The knowledge is subsequently sent through XML-RPC [XMLRPC] or SOAP to EMBET GUI. When a user enters a note, the EMBET Core processes the note and determines the context of it from the current user's context and the context detected in the note. When the user confirms the context, the EMBET Core stores the note and user's feedback. The core also handles a user the state (context).

EMBET GUI visualizes the knowledge and the user's context information to the user. Furthermore it informs the EMBET core about user context changes. The context can be reported also directly to the EMBET core from external systems (e.g. from workflow systems, received emails, or file system monitors). EMBET GUI visualizes knowledge based on XML [XMLW3C] transformation to HTML through XSL [XSLW3C] Templates processing. Moreover EMBET GUI has an infrastructure for a note submission and context visualization. It further provides a user with feedback (voting) on knowledge relevance. In addition, it contains a user interface for knowledge

management by experts where an expert can change a note and its context.

EM BET Core - EM BET GUI interface is used for an XML data exchange between EM BET Core and EM BET GUI. The Interface will be based on the SOAP protocol where both components act as web services; currently we use the XML-RPC protocol for an XML message exchange.

Interface to Memory is used for information and knowledge extraction and storage. It is based on RDF/OWL data manipulation using Jena API, which EM BET Core uses to extract and store knowledge.

Experience is represented by text notes, an unstructured text, entered by a user. For the context and environment modeling we use ontology, thus we use a Protégé ontology editor for ontology based modeling. Ontology is stored and managed in the Web Ontology Language (OWL) [OWLWEB]. The Jena Semantic Web Library [JENAWEB] is used for knowledge manipulation and knowledge storing. The Java technology is used for developing the system and user Interface is based on the JSP technology. The XSL templates are used to transform XML generated from OWL to displayed HTML. Since the Java technology is chosen as background for the EM BET, a choice of the web server – Jakarta Tomcat and implementation middleware is reasonable. XML is a widely used language for web development; it is regarded as a universal format for structured documents and data on the Web. Therefore in EM BET XML is used in various forms/degrees from description of ontology to the communication content language. The XML/XSLT technology permits visualization of XML documents and display of information presented to the user by EM BET. The JSTL is a native Java library for XML processing.

5.3 Knowledge Manipulation Algorithms

5.3.1 Algorithm for User context Updating

For context f_c updating a simple algorithm has been developed. This algorithm simply deletes the old actor context when a new actor related event is received and replaces it with the received event context.

Complexity of such algorithm is $O(n)$, because it just goes through each context element on new event and assigns it to actor context.

More complicated algorithms can be developed for EM applications as part of application customization if needed:

- Context can be updated and deleted according to ontology element types e.g. One individual of each type can be present in the system
- Priority is given to ontology element types.

```

Procedure updateContext(Event e)
// e - new event in the system relevant to actor
begin
  actor = e.actor; //event is related to some actor (user)
  if e.action = Create or e.context is empty then
  begin
    //when a resource is added (like note) or context of event is empty
    //we can add context of actor to the event
    while c = foreach(actor.context) do
      e.context += c;
    end
  else
  begin
    //if actor (user) performed some action or action related to user happened
    //we need to update actor context based on received event context
    remove(actor.context);
    while c = foreach(e.context) do
      actor.context += c;
    end
  end
end

```

5.3.2 Context Matching Algorithm for User resource Updating

For the actor resources f_R the algorithm updating a special type of resources representing experience - Notes was defined.

Complexity of algorithm is $O(n^2)$. The idea is that algorithm goes through all notes and all elements of actor context comparing them with note context elements. If note context is subset of actor context, the note is added to the actor model – actor resource property.

```

Procedure updateResources(Actor actor)
begin
  // unlink any notes of actor
  unlinkActorNotes(actor);
  while noteS = foreach(Note from Memory) do
  begin
    // checking defined note context elements
    matchAll = true;
    while c = foreach(noteS.context) and matchAll do
    begin
      if actor.context has not c
        matchAll = false;
      end
    //new note is added to actor
    if matchAll
      addProperty(actor.resource, noteS);
    end
  end
end

```

This is quite a simple but powerful algorithm. More complicated algorithms can be developed for EM applications as part of application customization if needed:

- Percentage of matched ontology elements is given as threshold for note relevance.
- Experience (notes) can be updated by giving weights to some types of ontology elements. Matching of note context does not have to be exact but can be sum of ontology elements weights. If required threshold is passed, note is considered as relevant.

5.3.3 Semantic Annotation Algorithm

In this chapter we describe algorithm which is related to detection of experience/note context detection. The goal of note context detection algorithm is to detect the context property of Note instance. The instances of the Pattern class are used to define and identify such context relations, where the pattern property contains the regular expression which describes textual representation of the context element. The ontology model was already described in chapter 4 and it can be seen on Figure 4.1. The text of the examined note is processed with the regular expression for every pattern and when it is matched the context of the Note individual is filled with the resources of *hasClass* or *hasInstance*. Moreover, when the *hasClass* property exists in the *Pattern*, the RDQL query is constructed and processed to find the individuals that match the condition:

- individual is the class of *hasClass*
- a *property* of individual contains the matched word

As it is illustrated in the Figure 5.4, the text of a note is analyzed with patterns and thus the property *context* of the evaluated *Note* individual is filled in with matched ontology elements. When the user's context has changed the EMBET finds the most appropriate notes by the analysis of the current context and the context of notes.

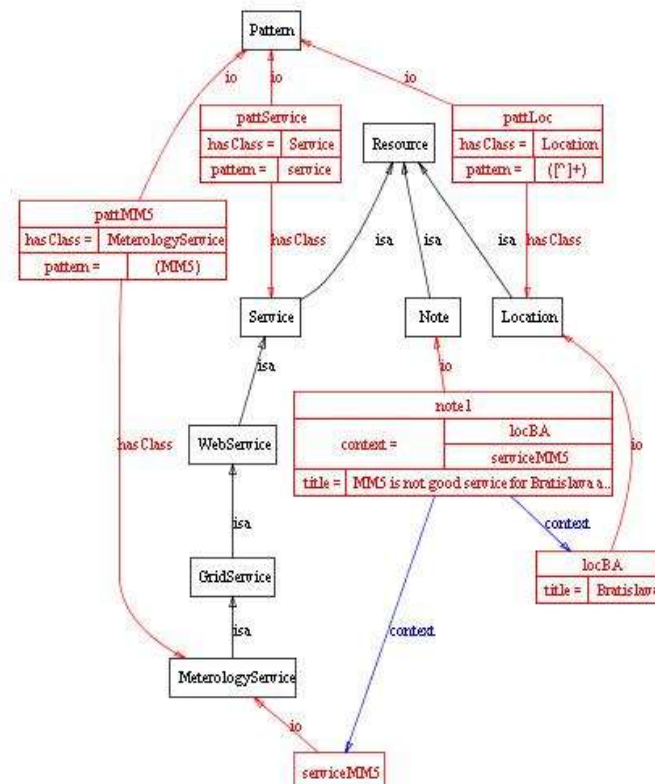


Figure 5.4: Fragment of EMBET ontology with example of ontology individuals

5.3.3.1 Illustration example

A user creates a textual note for example “MM5 is not a good service for Bratislava”. When it is submitted, the EMBET finds relations to other resources. The resources of the context are identified by the regular expression. In this case, the location of Bratislava is identified by the regular expression “([^J+])” because *locBA* individual has title property “Bratislava” and the service name is identified by the regular expression (*MM5*) because the *serviceMM5* individual contains the text MM5 in its description property. EMBET only tries to propose the most appropriate context for textual note. Then the user could be allowed to change the context of the note, which does not have to be always detected correctly. Finally, the note is stored into the memory.

5.3.3.2 Annotation Algorithm

The principle of described algorithm can be shown on the following pseudo-code.

```
note function identifyContext( note )
begin
  note_text = note.getTitle();
  foreach( pattern in patternClass )
  begin
    // regular expression matching
    foreach ( pattern.match(note_text) )
    begin
      if( patten.hasClass )
      begin
        note.addContext( hasClass );
        RDQL = "SELECT ?x "+
          "WHERE " +
          "(?x <rdf:type> <"+hasClass+">),"+
          "(?x ?y "+ matchResult +)";
        result = executeQuery(RDQL);
        if( hasResult )
          note.addContext( result );
        end
      if( pattern.hasInstance )
        note.addContext( hasInstance );
      end
    end
  end
  return note;
end
```

Complexity of the algorithm is $O(n^2)$. The algorithm goes through all patterns and matches it in all the text so as the second dimension we can take word count in the document.

5.4 Specification of EMBET Software Library

In this chapter we provide description of developed software libraries to be used for knowledge management in the EMBET system.

Library consists of the following main classes (Figure 5.5): Ontology, Memory, Anotate, DetectContext, EventHandler and Core.

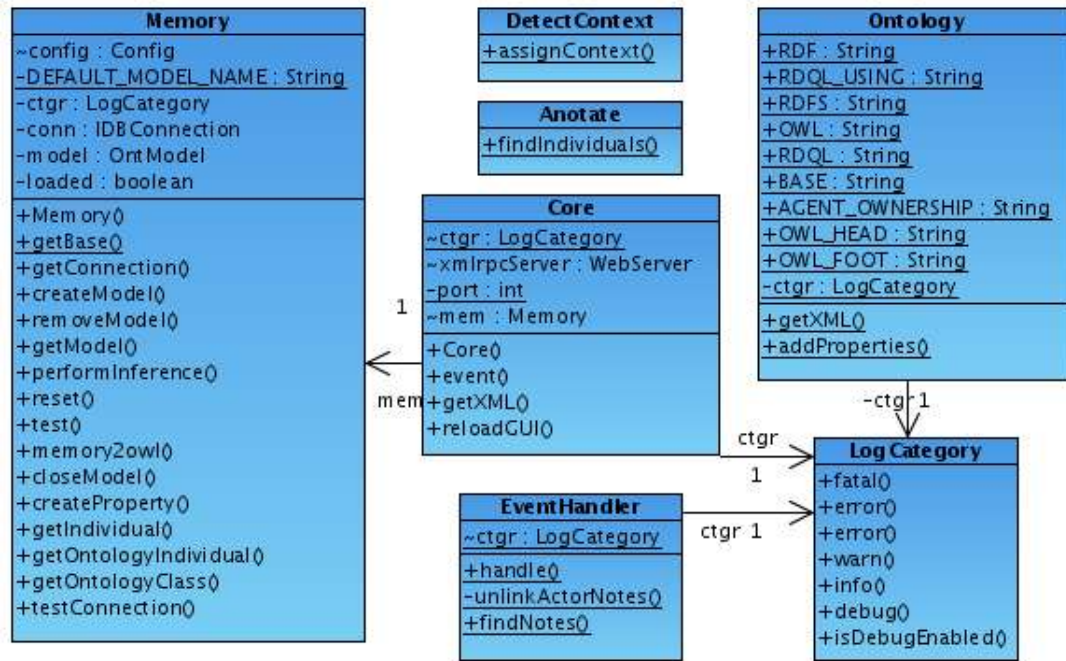


Figure 5.5: UML Class Diagram of Main EMBET classes

The class Ontology (Figure 5.5) has basic constants related to OWL ontology and the used ontology model.

The class Memory (Figure 5.5) has a functionality to load, store and manipulate the knowledge model based on an RDF or an OWL. The Jena library is used to this manipulation. The memory can load an OWL file developed in Protégé which becomes an ontology model of the knowledge memory. The memory can be stored in a database back-end such as MySQL, other RDBS supported by Jena or into the OWL file.

The class Anotate (Figure 5.5) contain functionality to detect ontology elements in plain text based on the Knowledge model loaded by Memory.

The class DetectContext (Figure 5.5) use Anotate functionality to assign found ontology elements to Note individual as context.

The class EventHandler (Figure 5.5) can handle defined events, manipulate them and update knowledge model according to needs. Algorithms for updating actors' context f_C and resources f_R are covered in this class as well. Events are based on Event ontology class defined in the knowledge model.

The class Core (Figure 5.5) is the main class, which runs the EMBET system.

It also covers XML-RPC server and methods which can be called via XML-RPC. The event() method is used to communicate events in RDF/OWL from external systems or EMBET GUI (see next chapter). The getXML() method is used by EMBET GUI to retrieve data about the user knowledge model or other information and knowledge stored in the EMBET Memory. The reloadGUI() method is used by EMBET GUI to detect weather GUI need to be reloaded.

The developed library is built on previously developed AgentOWL library [LAC05A] and classes such as Ontology and Memory are identical.

5.5 Interfaces to GUI and External Systems

This chapter describes Interface between EMBET Core and EMBET GUI or external systems. It shows mainly an example of main Event types, which are communicated via XML-RPC method event().

All events has a structure as described in chapter 4: an action performed by an actor on resource in particular context.

Note Adding Event is sent by EMBET GUI when a new note is added by the user. In the table bellow a concrete example can be seen: note “MM5 is not a good model for Bratislava in September” is added by “misos” user, action is “aAdd”. This event is sent using XML-RPC method event() of Core class.

```
<Event rdf:ID="event20050903232243">
  <actor rdf:resource="#misos"/>
  <resource>
    <Note rdf:ID="noteTest">
      <title>
        MM5 is not good model for Bratislava in September
      </title>
    </Note>
  </resource>
  <action rdf:resource="#aAdd"/>
</Event>
```

When a note is submitted, EMBET core calls EventHandler and DetectContext and returns an XML model of Note with assigned detected context. Returned XML model is shown in the table below.

```

<Note ID="noteTest">
  <title>MM5 is not good service for Bratislava area in September</title>
  <hasAuthor>
    <User ID="misos">
      <title>Michal Laclavik</title>
    </User>
  </hasAuthor>
  <hasTime>03.09.2005 23:34:40</hasTime>
  <context>
    <Month ID="mSeptember">
      <keyword>fall</keyword>>
      <title>September</title>
    </Month>
  </context>
  <context>
    <Class ID="Location">
      <comment>Geograficke miesto</comment>
    </Class>
  </context>
  <context>
    <MeterologyService ID="serviceMM5">
      <title>MM5 Meterology service</title>
      <keyword>MM5</keyword>
    </MeterologyService>
  </context>
  <context>
    <Capital ID="locBA">
      <hasRegion>
        <Region ID="reg02">
          <title>Bratislavsky Kraj</title>
        </Region>
      </hasRegion>
      <title>Bratislava</title>
      <latitude>48.15</latitude>
      <longitude>17.116667</longitude>
      <hasRegion>
        <Country ID="countrySlovakia">
          <hasRegion>
            <Region ID="regionEurope">
              </Region>
            </hasRegion>
            <title>Slovakia</title>
          </Country>
        </hasRegion>
      </Capital>
    </context>
  </Note>

```

Note context detected in a submitted note is displayed as a check list to a user and the user reselects relevant context and submit this context. Event confirm note context is generated by EMBET GUI and sent to EMBET core as an event (example bellow). In this example a user selected as relevant context the MM5 service and Bratislava location.

```

<Event rdf:ID="event20050903232406">
  <actor rdf:resource="#misos"/>
  <action rdf:resource="#aConfirm"/>
  <resource rdf:resource="#noteTest"/>
  <context rdf:resource="#serviceMM5"/>
  <context rdf:resource="#locBA"/>
</Event>

```

A user can change context by typing text (see scenarios on problem definition) or when an external system will send "change user context" event. Example of such event can be seen bellow. In this example context is

changed for service MM5 and Bratislava location.

```
<Event rdf:ID="event20050903232345">
  <actor rdf:resource="#misoS"/>
  <action rdf:resource="#aChange"/>
  <resource rdf:resource="#misoS"/>
  <context rdf:resource="#serviceMM5"/>
  <context rdf:resource="#locBA"/>
</Event>
```

If context is changed this way, algorithm for updating actor's/user's context f_C is executed. Also resources f_R updating algorithm is executed and, thus relevant notes for new context are detected and assigned to an actor/user model. In the table below XML of the actor/user model is shown. In this example above the described note is shown as well as MM5 service and Bratislava location context is assigned. This XML model can be accessed using XML-RPC using getXML() method.

```
<User ID="misoS">
  <context>
    <Location ID="locBA">
      <title>Bratislava</title>
    </Location>
  </context>
  <resource>
    <Note ID="noteTest">
      <title>
        MM5 is not good service for Bratislava area in September
      </title>
      <context>
        <Location ID="locBA">
          <title>Bratislava</title>
        </Location>
      </context>
      <context>
        <MeterologyService ID="serviceMM5">
          <title>MM5 Meterology service</title>
          <keyword>rain</keyword>
          <keyword>meterology</keyword>
          <keyword>MM5</keyword>
          <keyword>weather</keyword>
        </MeterologyService>
      </context>
    </Note>
  </resource>
  <context>
    <MeterologyService ID="serviceMM5">
      <title>MM5 Meterology service</title>
      <keyword>rain</keyword>
      <keyword>meterology</keyword>
      <keyword>MM5</keyword>
      <keyword>weather</keyword>
    </MeterologyService>
  </context>
  <title>Michal Laclavik</title>
</User>
```

Interface contain other events regarding voting on notes and others but this functionality is still in the development stage. Events described above are implemented and functioning.

5.6 Presenting Ontological Knowledge to Users

In chapter Knowledge Presentation we dealt with the state of the art of knowledge presentation. In this chapter we explain work done in this area which can be used for presenting ontological knowledge to a user. The focus is on Transformation type of knowledge presentation. We think other alternatives such as a Graph representation need to be exploited for better knowledge presentation. We used a graph presentation to present developed formalized models in the work.

5.6.1 Transformation Solution used for presenting knowledge in experience management

The key idea of the transformation solution based on the XSL is to deliver knowledge in the HTML format. The core of the system work with the knowledge in the OWL, RDF, RDFS format using the Jena[JENAWEB] library. Knowledge is returned by Core in the form of RDF/OWL or plain XML. The main part which needs to be presented to a user are experience and user context. Experience in EMBET is represented by text Notes.

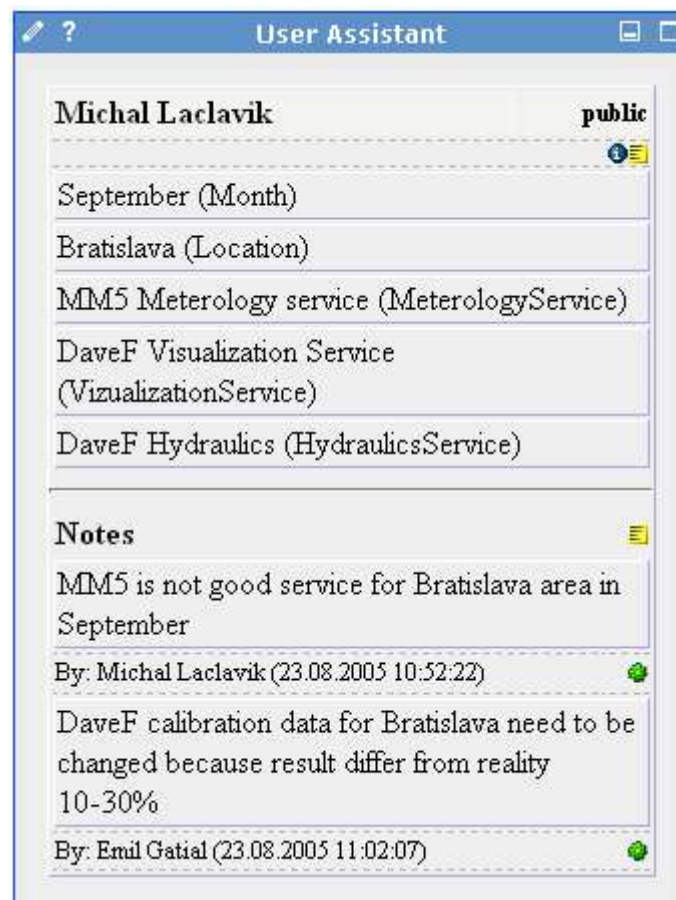


Figure 5.6: EMBET GUI

On Figure 5.6 EMBET GUI is presented. On the top elements of user context can be seen. In bottom part Notes representing experience are displayed with additional information. By clicking on icon next to “Notes” text a new note can be added. By clicking on the same icon in the context area the context/current problem can be redefined by typing text.

The system can adapt any graphical user interface, which can handle XML and supports XML-RPC communication. We have developed GUI based on the Java JSP technology, which gathers the XML from the system through the XML-RPC and transforms it to the HTML using the XSL templates.

EM BET GUI (Figure 5.6) is the result of transformation of XML returned by EMBET Core using getXML() method showed in the previous chapter.

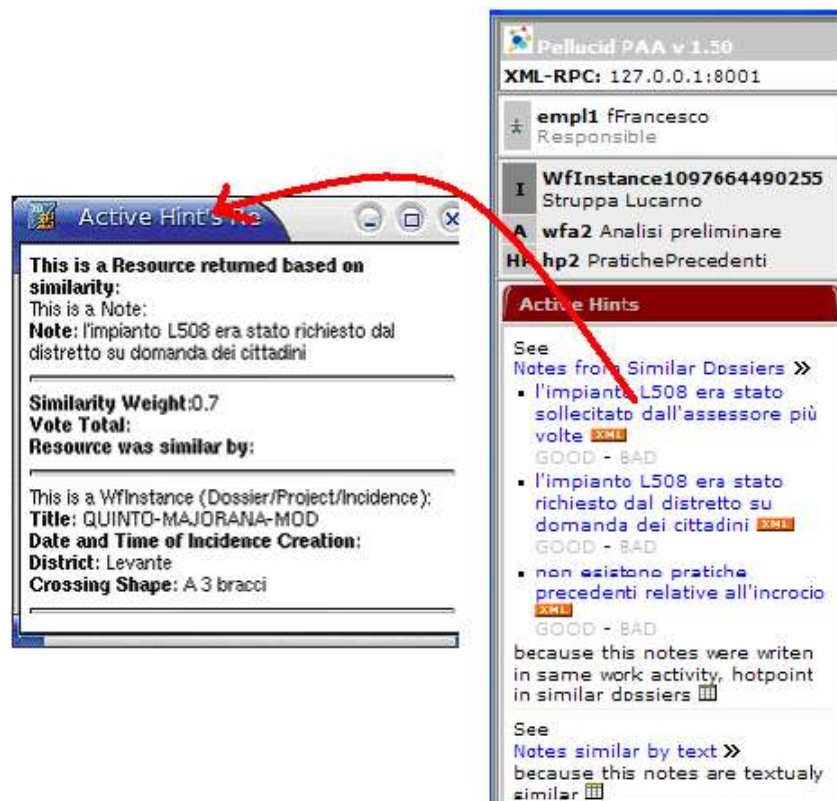


Figure 5.7: Pellucid GUI window with Active Hints (right) and opened resource window (left).

Figure 5.7 shows the GUI developed within the Pellucid Project [PELL02] with displayed active hints (AH) which are representation of experience similar to Notes but structured. The AH can contain several resources. By clicking on a resource we can see resource properties formatted by the application defined XSL templates, thus this approach was evaluated in Pellucid project and now redesigned in the EMBET system.

The system development from ontology to the HTML output covers the ontology design through the OWL structure and the XML-RDF, XML format. The ontology embraces all possible structures of output data, which a website designer needs in order to develop his/her website style. The

structure of the XSLT files is done according to the domain specific ontology. Such a structure provides a full control over the structure of all data/knowledge that may be an output from the system sent to the GUI. The XSLT files act as a conditional filter and a template for the HTML output.

5.6.2 Knowledge Presentation Conclusion

Generally there are 3 types of knowledge presentation: Object Trees, Graphs and XSL Transformation. We describe our solutions developed in the scope of the Pellucid and K-Wf Grid project based on the XSL transformation solution. The described solutions were evaluated in different types of users. The Object Tree is the best for knowledge experts, while graphs can be valuable for all users. A graph knowledge presentation was used in scope of this work presenting all ontology models as graphs output of formalized OWL-DL. The XSL transformation based on XML is probably the best solution. The XML and XSL are commercial standards and can be implemented by most of developers. In our future work we would like to focus on a Graph representation, find applications where this kind of knowledge presentation is appropriate and evaluate it. As already mentioned the work done was mainly support for EMBET core to be able to output RDF/OWL knowledge in a plain XML form by existing commercial communication protocols such as XML-RPC. The other work done with creating of the JSP pages and the XSL templates was done by other members of the Pellucid and K-Wf Grid consortium.

5.7 Conclusion on EMBET Design & Development

In this chapter the EMBET System was described, its design, development, interfaces and partially also functionality of the System. In the next chapter a use of the system in concrete applications is described and thus system functionality was tested and evaluated. The EMBET knowledge model, design, algorithms and implementation evolved and thus we believe that developed solution presents a useful model and implementation of Customizable Experience Management System.

6 Use and Evaluation of Results in Pilot Operation

In this chapter we show a use of the work described in the work in R&D projects. We discuss a use and evaluation of work results in scope of the K-Wf Grid IST project, Pellucid IST project and Znalosti Slovak national project.

6.1 K-Wf Grid IST Project

The Knowledge-based Workflow System for Grid Applications – K-Wf Grid is a project funded by EC. The main objective of the K-Wf Grid project is to enable the knowledge-based support of workflow construction and execution in a Grid computing environment (Figure 6.1). In order to achieve this objective the consortium members will develop a system that will enable users to:

- semi-automatically compose a workflow of Grid services,
- execute the composed workflow application in a Grid computing environment,
- monitor the performance of the Grid infrastructure and the Grid applications,
- analyze the resulting monitoring information,
- capture the knowledge that is contained in the information by means of intelligent agents,
- and finally to reuse the joined knowledge gathered from all participating users in a collaborative way in order to efficiently construct workflows for new Grid applications.

In the K-Wf Grid, grid services are semi-automatically composed to workflows which should solve a user problem. It was identified that even when services and input and output data are well semantically described, there is often no possibility to compose an appropriate workflow e.g. because of missing specific input data or fulfillment of a user and application specific requirements. To help user in workflow construction it is appropriate to display notes and suggestions entered by the same or different users. Thus experts

can collaborate and fill up application specific knowledge base with useful knowledge which is shown to users in the right time.

K-Wf Grid use presented ontological model (Figure 6.2) as well as several other ontology extensions, which are not related to the experience management. The main component representing the experience management in the project is User Assistant Agent (UAA). It helps a user to compose appropriate workflows for his/her problem by presenting appropriate knowledge and experience. In addition, it assists to a user in application related knowledge. The UAA architecture and model is built the same way as the model and architecture presented in the work.

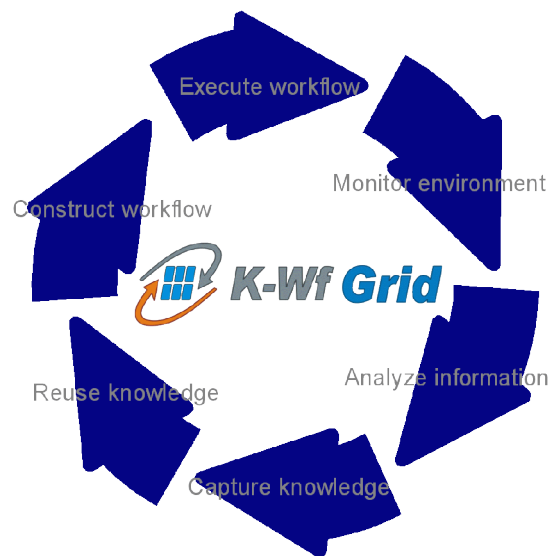


Figure 6.1: K-Wf Grid Knowledge Cycle and Objective

The K-Wf Grid uses a similar model of the environment based on resources, actors and events (see Figure 6.2) as presented in the work. The UAA manages actors' (users') context and prepare appropriate resources for actors. Resources are mainly Notes, relevant to grid services and resources, which are then composed to a workflow.

The K-Wf Grid Project contain several applications which are used for evaluation of project results:

- Flood Forecasting application
- Enterprise Resource Planning application
- Traffic simulation application

All applications are grid based. EMBET has been evaluated only on the Flood Forecasting Application. Customization of EMBET on other applications is in development, since domain ontologies are not developed yet.

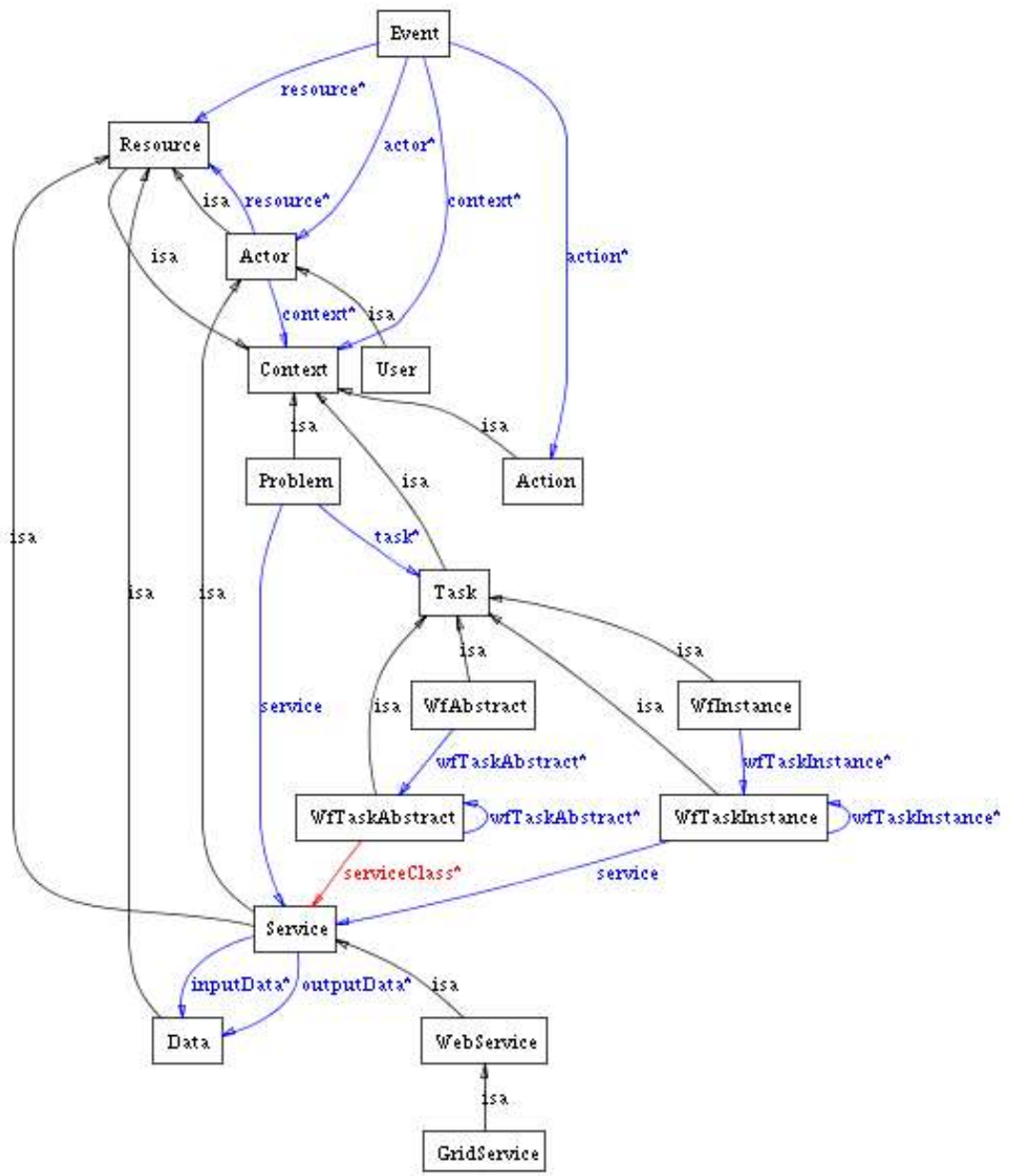


Figure 6.2: Main Elements of Knowledge Model in K-Wf Grid

6.1.1 Flood Forecasting Application

The Flood Forecast Simulation Cascade (Flood App) is a hydrometeorological application, already well tested and developed during other projects CROSSGRID. The application is a cascade (a natural workflow) of several simulation stages, which lead to the final result – a prediction of a potential flood. The cascade begins with meteorological prediction of weather for a short future period (usually not more than 48 hours). This prediction is then recomputed into possible watershed, which in turns affects water level of the target river. This water level is computed in the hydrological stage of the

application. The resulting hydrograph (time series of water level values) is fed into the final stage of the cascade – a hydraulic prediction. This – using detailed terrain model of the target flooded area – computes the water flow in the target area, water depth and flow vectors. This result may be visualized and an expert user may assess the situation.

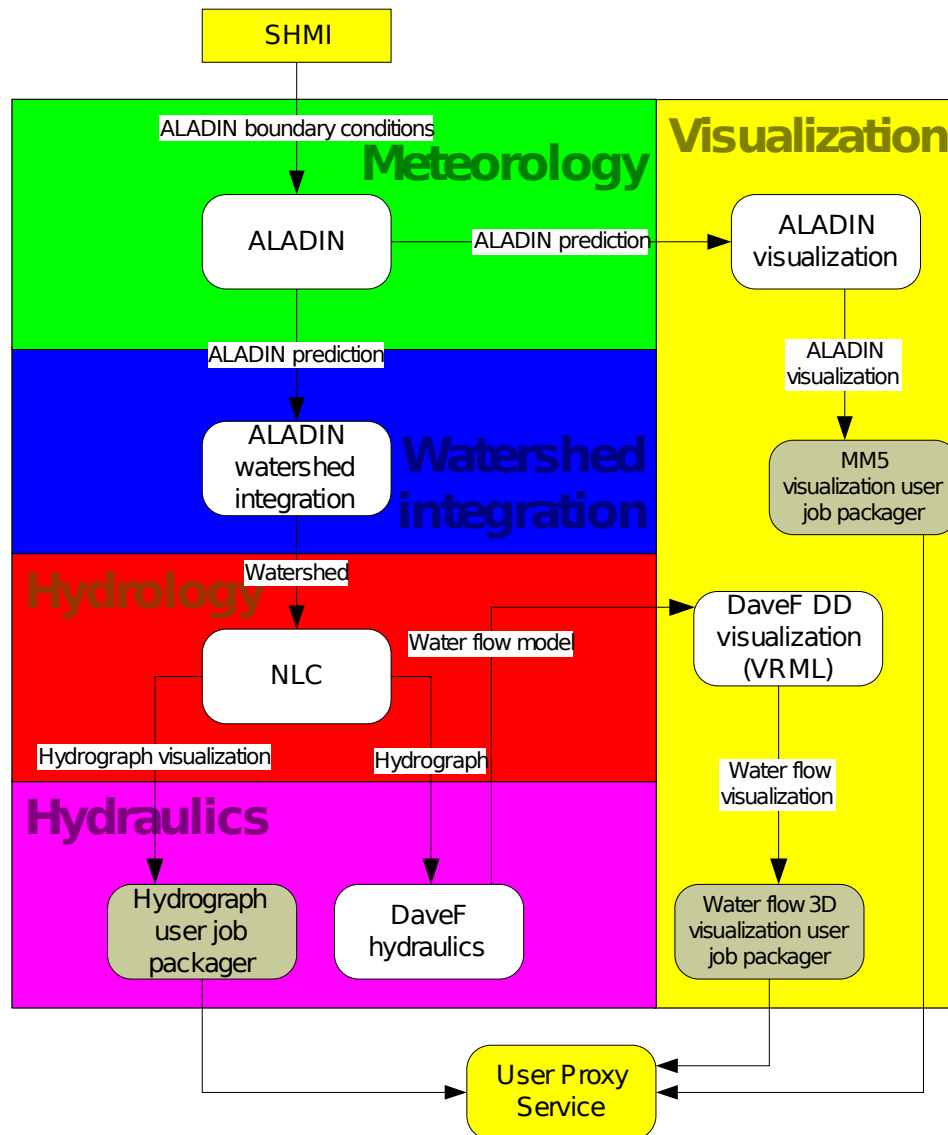


Figure 6.3: Sample flood application workflow, using ALADIN/NLC

Figure 6.3 shows a sample flood application scenario, a quite simple one. In this simulation, weather prediction is computed using the ALADIN weather prediction model, hydrology is done using the NLC model and hydraulics with the DaveF model. All outputs are visualized and shown to the user through his/her User Proxy Service. This workflow does not require any special features of the workflow engine. For the sake of readability, the figure misses certain (not essential) service calls, like data converters (job packagers) for the User Proxy Service. Different services exist in this application like MM5 meteorology service of MIKE hydraulics service. The goal is to compose an appropriate workflow of this services according to available input data or

models suitability for defined problem and other user requirements. EMBET helps to compose workflows by formalizing user requirements and also supporting a user by experience in form of notes during workflow construction, execution as well as it helps to understand and explain the execution results.

6.1.2 Domain Ontology Model

The ontology model in the Flood Forecasting Simulation Application consists of 3 main sub ontologies:

- Service Ontology
- Time Data Ontology
- Location Ontology

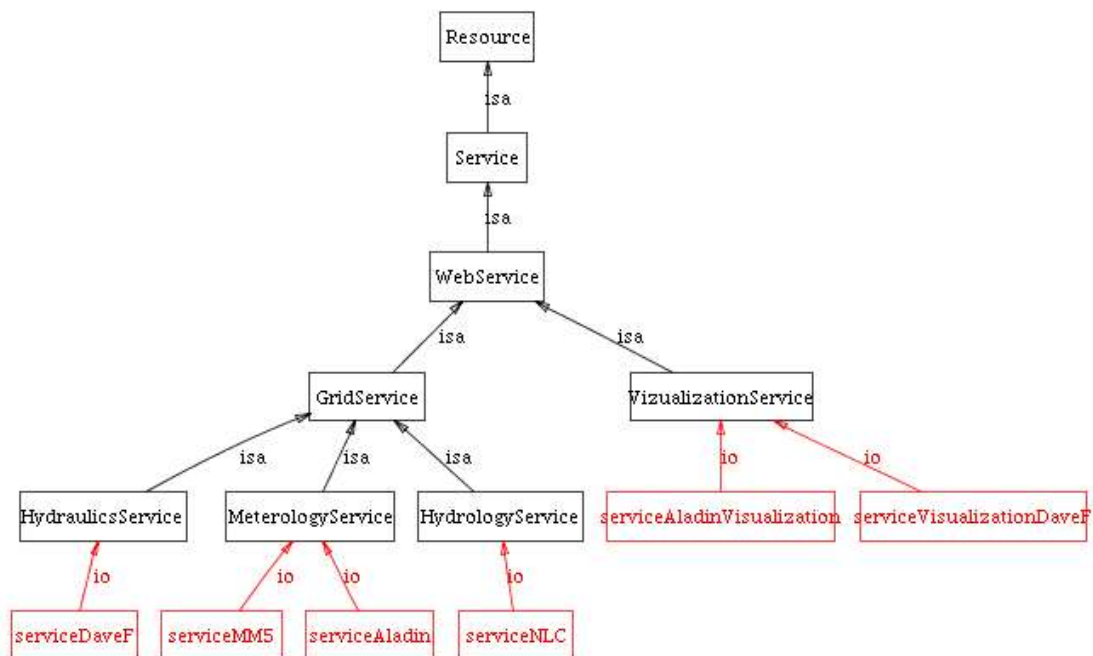


Figure 6.4: Service Ontology with several grid services

Service ontology (Figure 6.4) defines structure of Web services and grid services and several types of such services as well as its concrete individuals. Such services have input and output data showed also on Figure 6.5 which represents Time Ontology and Figure 6.6 standing for Location Ontology.

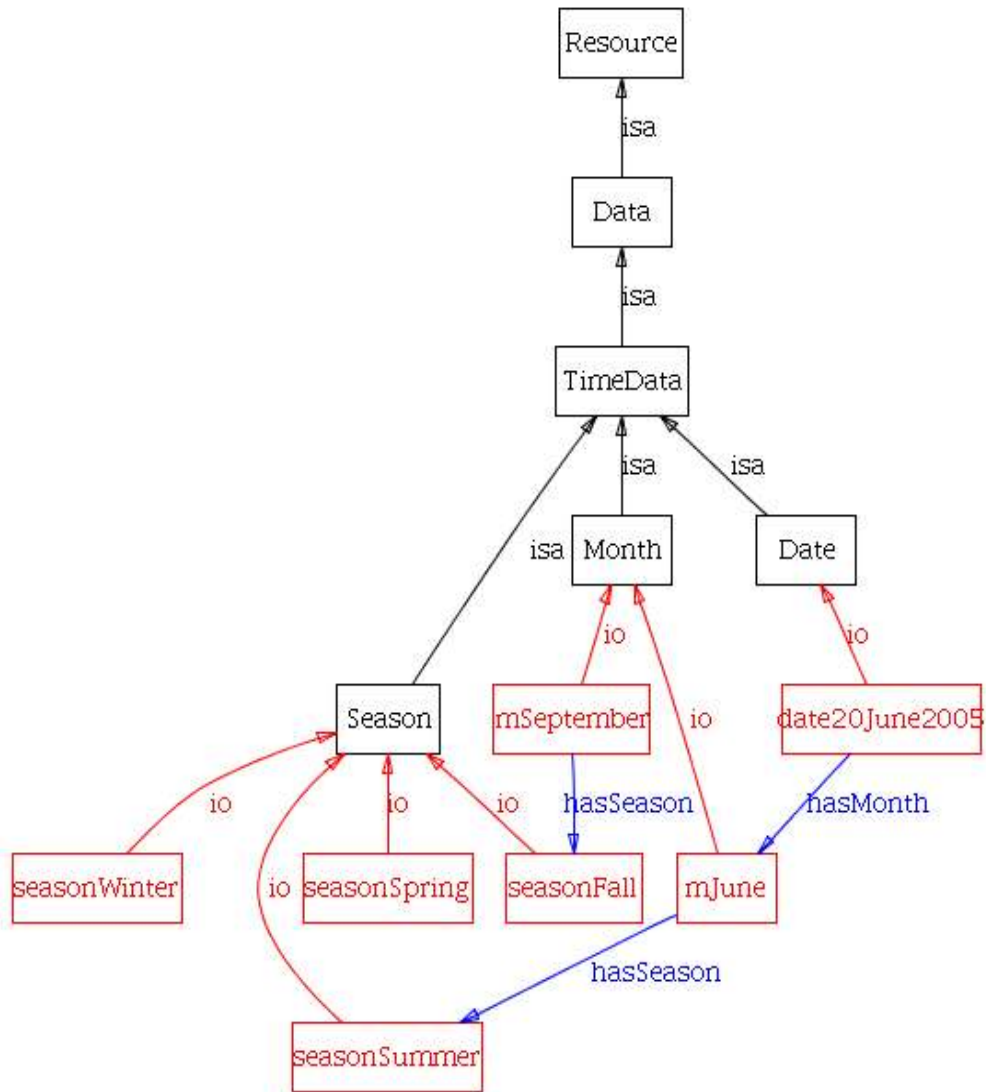


Figure 6.5: Time Data Ontology

Time Data ontology (Figure 6.5) defines time data types such as Date, Month or Season. This data are important when detected as context of Notes – for example some note need to be displayed in September or in Summer.

Location ontology (Figure 6.6) consists of individuals for whole Slovakia, and thus can be very well detected by semantic annotation. Each individual consists of latitude and longitude data which can be passed to web services. Same domain ontology is used also in the Znalosti project.

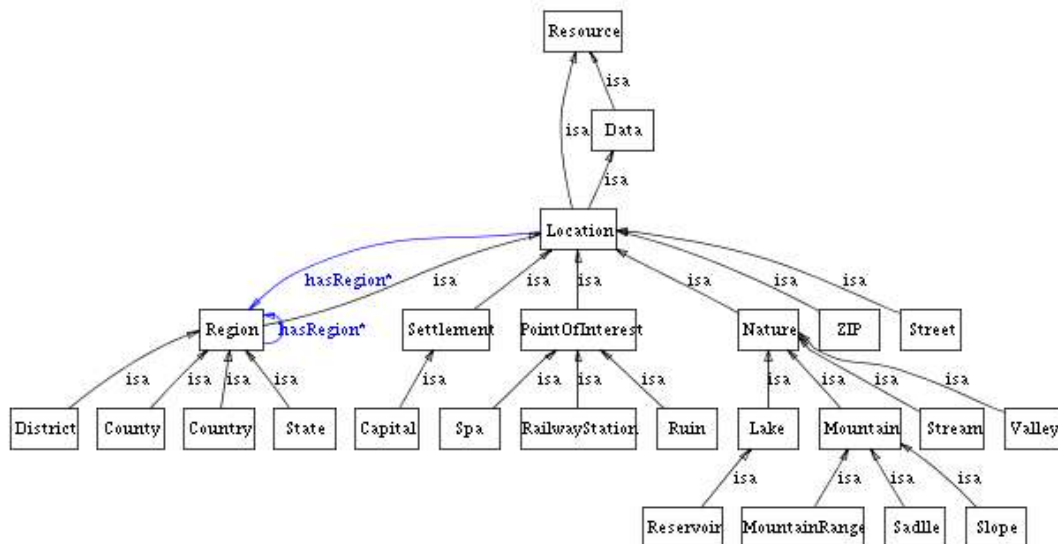


Figure 6.6: Location Ontology

Time and Location data are considered as input data for most of services in flood application because each service needs to be executed for certain time and location. Also most of the notes (experience) is related to (its context is) location, valid period and relevant model.

6.1.3 Example of Use

To better illustrate the use of EMBET in the process of user assistance, we present the following example from the K-Wf Grid project's flood forecasting application, which extends the flood prediction application of the CROSSGRID project. The application's main core consists of simulation models series for meteorological, hydrological and hydraulic predictions. The models are organized in a cascade, with each step of the cascade being able to employ more than one model. For example, the first step - the meteorological weather prediction - can be computed using the ALADIN model, or the MM5 model.

6.1.3.1 Note Adding Scenario

This scenario describes how Note - experience representation is born in EMBET system.

On Figure 6.7 Note adding window can be seen. When such note is confirmed EMBET annotate Note and detect ontology elements showed in Figure 6.8.

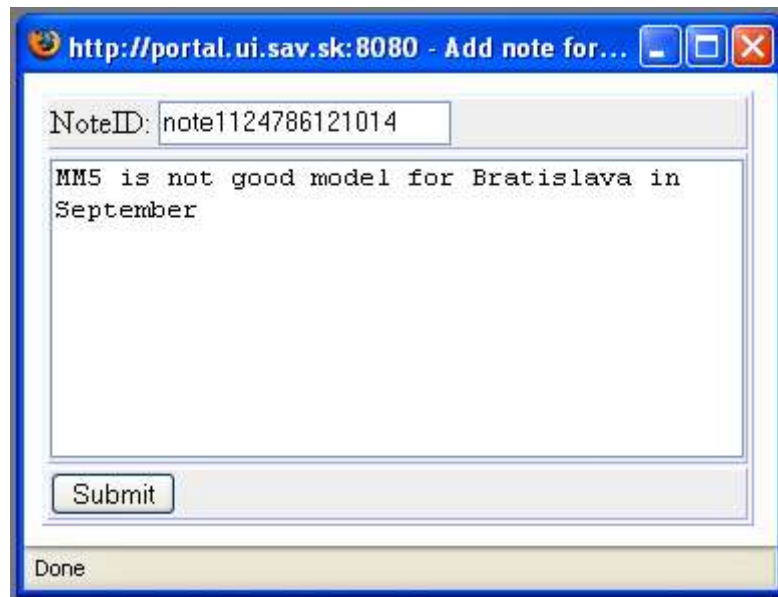


Figure 6.7: Entering new Note EMBET Window

On Figure 6.8 results of context detection can be seen. Unchecked items are current user context, checked items are elements detected from text of the note. A user selects only relevant items.

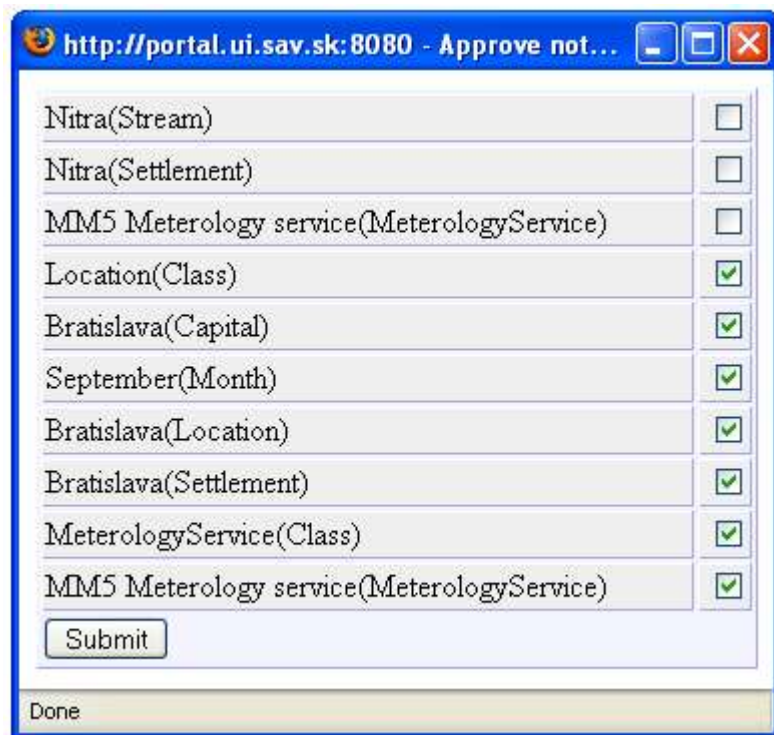


Figure 6.8: Note Context Detection

Consider that the user has used the MM5 meteorology model and he/she wants to describe its properties (gathered knowledge), which may be relevant for other users. The proposed model of interaction is as follows:

- A user enters a note through EMBET, stating that “the MM5 model is not appropriate for weather forecast in September for Bratislava area” (Figure 6.7).
- From the workflow in which the user states this note, we know directly the current user context (unchecked items on Figure 6.8)
- Some of current context can be relevant to note and some does not have to be. The note is processed and its text related to the context, as well as the relevant context items are selected (Figure 6.8). In this case, by finding the text MM5 we can assume that “Service type MM5” is the relevant part of the context. There is other context relevant information which can be detected like “September”, the time in which this note is valid.
- After the context detection, the user is presented with a checklist (Figure 6.8) where the user may select only the relevant parts of the context, which will trigger this note.
- A user gets pre-selected parts of the context, which were detected by the system as really relevant. He/she can subsequently modify the contents of the list and finally submit the note.
- Each time anyone works with the MM5 service for Bratislava area in September, the note is displayed.
- Each note can be voted by a user as being “good” or “bad” and the current results are always displayed along with the vote.

This model gives a good basis for experience management and knowledge sharing in a virtual organization as well as for application-related collaboration among users.

6.1.3.2 Problem/context Definition Scenario

The main idea of this scenario is that user context (environment/ problem definition) is changed and thus relevant experience can be showed after such change. In K-Wf grid application this can be archived by:

- reporting a change of user context from an external system. For example when a user selects a different workflow or web service in K-Wf Grid GUI event for updating user context is generated.
- Free text explanation of a user problem (Figure 6.9). Such free text is then annotated by Anotate class and detected elements are presented to the user to confirm detected user context.

Free text problem definition is important, when a user starts to work with the system and wants to define problem to be understandable for computer System. EMBET then can return relevant experience for such problem.

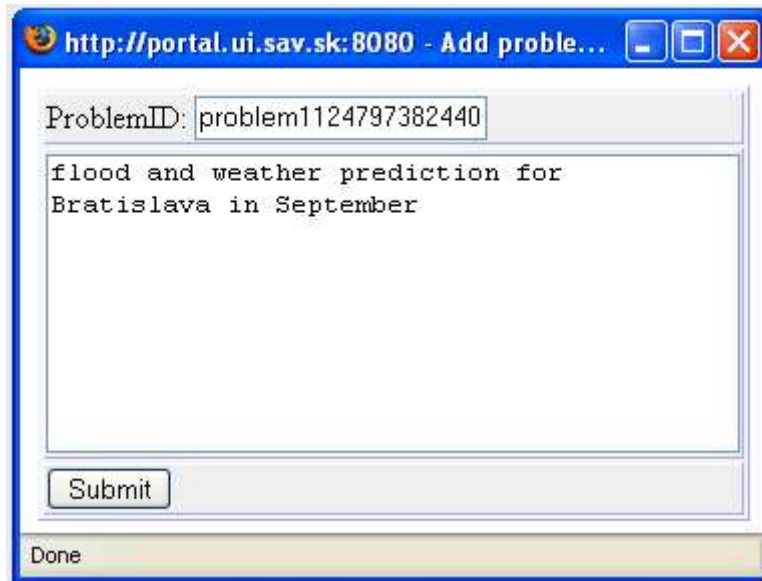
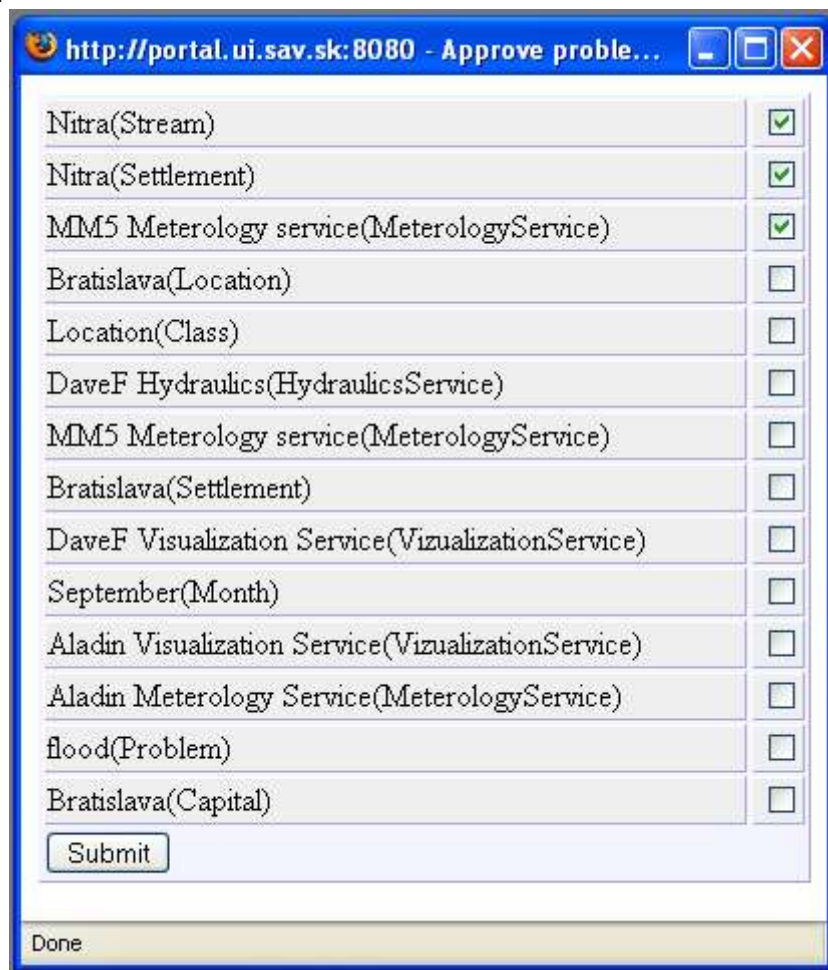


Figure 6.9: Problem Definition EMBET Window

Free text problem definition is important also because such problem definition discovers appropriate web services and data for construction of Workflow of Services.



A workflow is constructed and executed by other K-Wf Grid components but those components use detected ontology elements (web services: MM5 service, Aladin, DaveF and output data definitions: Bratislava Location, September) to compose an appropriate workflow.

When a user selects relevant problem context and submits it, even a user context change is generated and EMBET executes context f_c updating algorithm.

6.1.3.3 Context Matching and Notes Display Scenario

When context of an actor/user is changed as showed in previous scenario, resources f_R updating algorithm is executed and relevant notes are assigned to an user model. The Notes are then displayed to the user as can be seen on Figure 5.6.

6.1.4 Conclusion

The K-Wf Grid is still an ongoing project. It has provided a great opportunity to test and evaluate the results presented in the work. The EMBET architecture has been built and evaluated within this project. The work on EMBET will continue and will be applied to new applications “Enterprise Resource Planning” and “Traffic Simulation”. We hope this new applications will prove generality of the system.

6.2 Pellucid IST Project

The overall objective of the Pellucid was to develop an adaptable platform for assisting organizationally mobile employees, in effect to re-engineer their work in an organization. This will improve organization effectiveness and efficiency by formalization, recording, storage and preservation of experience and knowledge; and supporting workers during integration in a new department or a role by giving access to specific knowledge and experience accumulated in the past. At the technical level, the objective is to develop and integrate several advanced technologies in a customizable agent-based architecture. These technologies include autonomous co-operating agents; responsive interaction with the end-users; organizational memory; a workflow and process modeling; and metadata for accessing document repositories. Another objective is to obtain experience with customization and to formulate guidelines for the best practice in using the Pellucid platform in assisting organizationally mobile workers [PELLTA].

6.2.1 Pellucid Pilot Sites

Pellucid project results were evaluated on 3 pilot sites; one located in Genoa, Italy dealing with Traffic light installation and management. Other 2 are located in Seville, Spain concerning a call center and project management.

6.2.1.1 Project Management - MMBG

The Mancomunidad de Municipios del Bajo Guadalquivir is an organization created by eleven local authorities with the main objective of contributing to the social and economic development of an area with 250,000 inhabitants.

The particular problem of MMBG is the wide range of tasks that its employees must handle. They include managing of joint services (such as recycling, cultural projects), provision of innovation and technology services (such as identifying opportunities and weaknesses in the region), support to business initiatives, etc.

The MMBG objectives are as follows:

- Support activities for employment creation and improvement of living standards among the population
- Favor a change in the productive systems of local communities towards activities with comparative advantages in national and international market
- Create transport and communication infrastructures and the necessary services to support the development of local economies
- Administer economic and financial aid destined for promoting production and constructing infrastructures and services
- Coordinate initiatives and collaborate with agents involved in regional development projects taking place in the area.
- A management of projects procedure was selected as a representative workflow process in this organization.

6.2.1.2 Traffic Light Management - CDG

The pilot application of the Comune di Genova will address the support to organizational mobility among several areas of the Traffic and Mobility Management Department of a large city administration - the department managing facilities and technical systems belonging to the urban and near-urban road system. The range of services managed includes:

- planning, installation and maintenance of traffic signs and signaling systems
- planning and management of parking areas and resources

- strategic traffic planning, by study, definition and design of traffic circulation plans within the overall road network
- strategic planning of Public Transport services
- daily operation and management of technical systems and facilities controlling traffic and mobility within the road network.

A process for installation and maintenance of a traffic light plant was selected as a representative workflow process within the organization.

An analysis of the process was performed with the aim to model the process and discover the bottlenecks in the process. Knowledge of the organizational structure was considered in modeling the process.

6.2.1.3 Call Center - SADESI

The Junta de Andalucía, of which the Consejería de la Presidencia is a part, has a corporate telecommunications network. This is known as SADESI. It covers 50 departments and 1.250 buildings connected for data services, 53 departments, 550 buildings, and 47.000 lines for fixed telephony services, 53 departments, 7.300 lines for mobile telephony services.

Management and resolution of fixed telephony breakdowns by the agent of a call center was chosen as a representative workflow process within the organization.

The organization, unlike the two ones mentioned below, has a software application with some WfMS functionalities, (Vantive) implemented. The Pellucid purpose in this organization is to support a new agent in the call-center to decrease a training time, and to improve quality and efficiency of the service.

The objective of the Pellucid platform is to capture experience from other call-center agents, to classify and sort the general knowledge and support their resolution, and be of assistance during the process of training a new agent. The possibilities to improve functions of the Vantive application system were analyzed. A user interface to cooperate with this Vantive “workflow system” was created.

6.2.2 Ontology Model

In the Pellucid project, a similar Knowledge Ontology model was used as presented in chapter 4. However, the main ontology element representing experience was not Note but the so called Active Hint. The Active Hint was composed according to the following formula[BAL04A]:

[ACTIVE_HINT] = [WORK_CONTEXT] + [ACTION] + [RESOURCE] + [EXPLANATION]

Context updating f_C and resources f_R updating algorithms were thus also

different than in presented EMBET architecture. EMBET can be understood as a more advanced step in development of EM System. In the Pellucid project resources f_R updating algorithm used also special developed similarity measures for context or ontology element similarity. In the EMBET system this is not necessary any more since semantic annotation is used, but such similarity method can probably achieve better results in specific applications.

In this chapter we present extension of knowledge model for different Pellucid pilot sites. This demonstrates that the used model is generic. Pellucid application used also Workflow model extension presented in chapter 4. Described Domain Ontologies show only the most essential domain concepts.

6.2.2.1 Traffic Light Management (CDG) Domain Ontology

The CDG domain ontology describes the main concepts of Traffic Light installation problematics. Concepts like Centralized, Shape, District refer to properties of a traffic light. The Applicant Type or Request type are related to a request for Traffic light installation. The Hot Point is atomic like activity. The domain entities are used for reporting of application environment events or context of events, related to Problem (Workflow Instance) or used for calculating similarity methods.

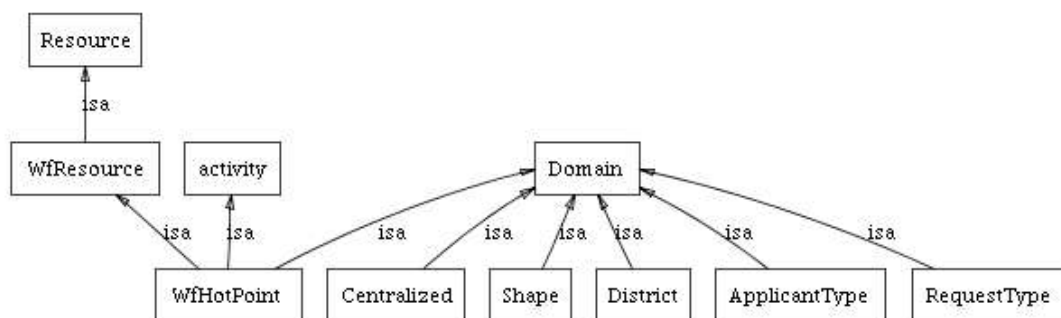


Figure 6.10: Main Domain Concepts of CDG Ontology

6.2.2.2 Project Management (MMBG) Domain Ontology

MMBG domain ontology defines mainly concepts related to different types of documents as well as the Project (Problem or Workflow Process instance) related concepts such as a theme, a type or an action line of the project. These concepts are reported in events and then processed by algorithms for resource and context updating. Moreover some of them are used for similarity calculations.

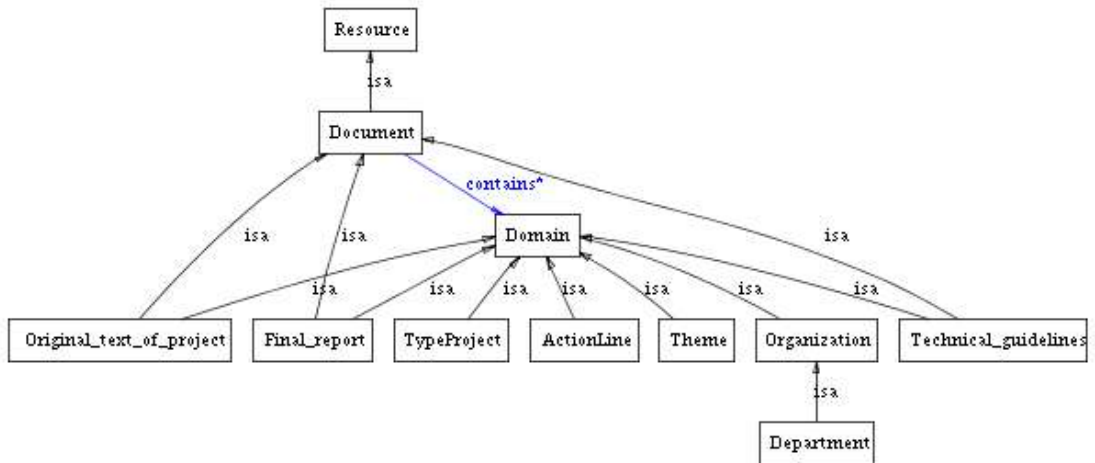


Figure 6.11: Main Domain Concepts of MMBG Ontology

6.2.2.3 Call Center (SADESI) Domain Ontology

The SADESI domain ontology was more complicated because this pilot site did not adopt similarity calculations. Returned experience for users had to be always relevant as it was used mainly by unexperienced employees. This pilot site concerned solving and processing the problem of telephone incidences. Most of the Domain ontology concepts are incidence properties such as outbuilding, a priority, a type or a reporting person. This pilot site business process was tied up with email communication between telecommunication operators and human agents of a call center. Thus events from the application environment were related not only to workflow activities but also to the received email messages.

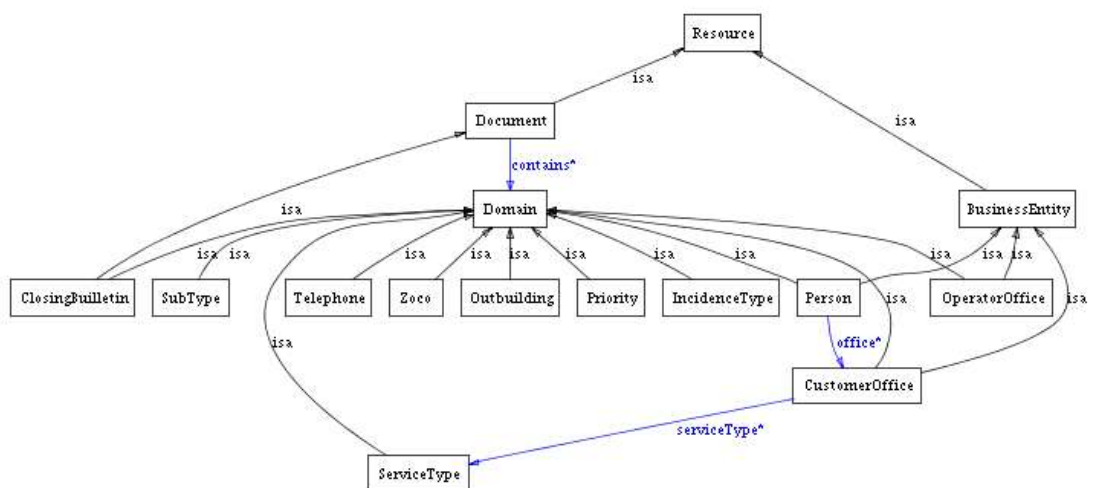


Figure 6.12: Main Domain Concepts of SADESI Ontology

6.2.3 Conclusion

The Pellucid project provided an enormous opportunity to test and evaluate the results presented in the work. We have built on experience gathered in the project to achieve better results with EMBET.

6.3 Znalosti Project

Znalosti (Research and Development of Tools for Knowledge Discovery, Maintenance and Presentation, SPVV 1025/04) is a Slovak project. Some ideas and results were used for preparation of the project proposal. The project has started in September 2004 and it is focused on discovery, maintenance and presentation of knowledge. Project objectives are not directly related to the work and cannot be used for example presented model but the project will use the same infrastructure as presented in work; namely CommonKADS methodology, Protégé ontology editor and Jena Semantic Web library. The Pilot application is Job search application, where tools are used to find, download, categorize, annotate, search and display job offers to job seekers.

6.3.1 Semantic Annotation

As stated in the title of the project, this project focuses on development of reusable tools. One of such tools is a tool for semantic annotation and its purpose is to create ontological meta data of web documents according to the application domain ontology model. This is very similar to objectives of semantic annotation used in the EMBET system. Thus the EMBET library (Anotate class) is used to annotate web documents.

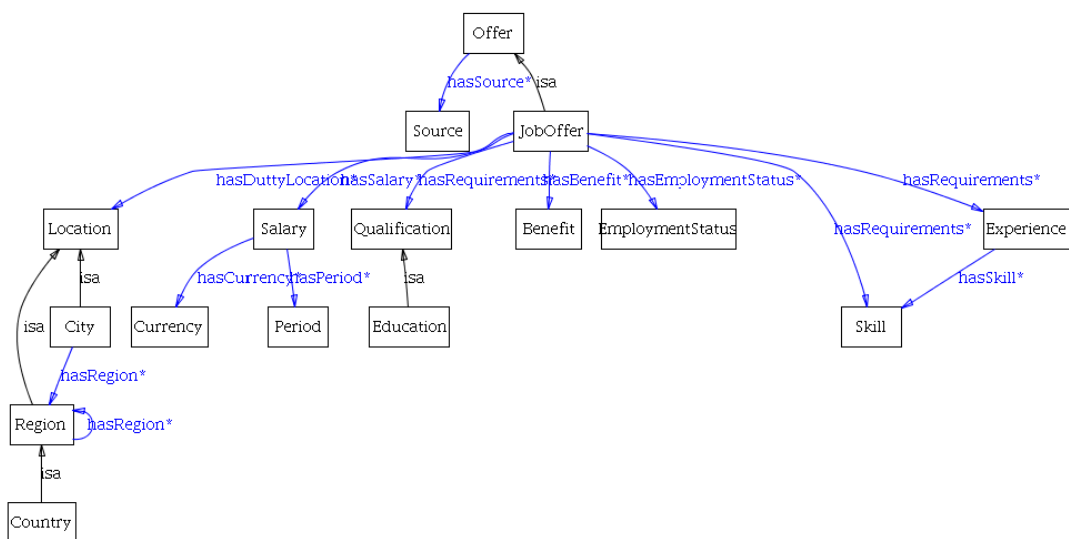


Figure 6.13: Job Offer Ontology

On Figure 6.13 main components of Job Offer ontology is displayed. Important fragments on ontology are Location (same as in the K-Wf Grid project Figure 6.6) or Skills individuals which can be then detected by annotation.

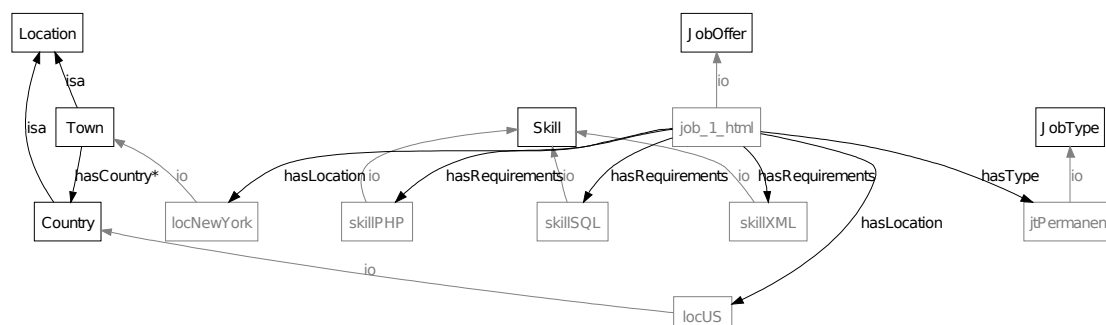


Figure 6.14: Job Offer Individual Created by EMBET Annotation

On Figure 6.14 individual of Job Offer is created based on semantic annotation of a Job Offer document (Figure 6.16), using simple regular expression patterns as showed on Figure 6.15 where main individuals can be detected by title property such as sillSQL or skillPHP individuals. Other specialized patterns such as pattFullTime are used to detect concrete job offer properties – jtPermanent individual, which represents a permanent job position. In this example the job offer location - New York and USA are identified by a regular expression „([A-Za-z]+)“ a „([-A-Za-z0-9]+ []+[-A-Za-z0-9]+)“, because individual locNY has the property title „New York“, locUS has the property title „USA“. Similarly other ontology elements are detected. Some regular expressions search for ontology individuals, other ontology classes and others such as pattFullTime annotate a job offer by a concrete individual jtPermanent if expression “(Full[-]Time)” is found. Systems detect ontology elements based on domain ontology. In this example it is ontology of job offers.

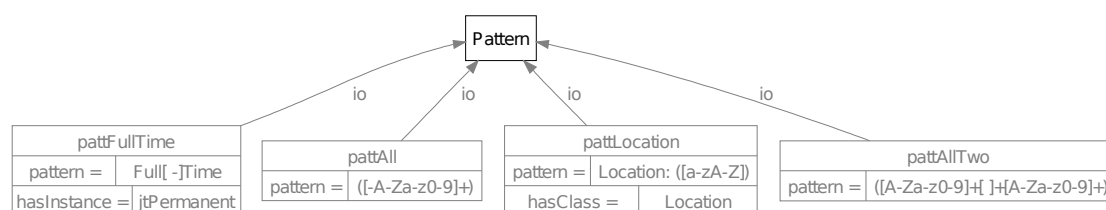


Figure 6.15: Fragment of Pattern Individuals

Web Developer	
! Get Notifications + Add job	
Web Developer	
Certify Your PHP Expertise. It's easy...	
Description:	Insight Out Of Chaos (IOOC) has a fulltime position available for an energetic, motivated individual seeking a position in a small, dynamic database marketing company. We manage data for large companies and need additional web programmers to present this data on the web. Our speciality niche is loyalty marketing -- and we take the concept into areas where others fear to tread. A knowledge of PHP is essential. We interact with MS SQL Server but this position does not require an extensive knowledge of databasing as all data is done with Stored Procedures BUT if you know databases, that is a benefit. You should be well-versed in DHTML, CSS and other basic web-interface technologies. You get points for knowledge of Browser DOM, XML and MS SQL Server. You do not have to be the best nor the greatest -- we are willing to train the right person -- but you should have a working knowledge of all phases of web development processes. And you must be willing to work in a demanding environment where we expect pixels to be aligned and accounts to balance out to the penny. You WILL be challenged. This position is available now. Please contact seth at iooc.com with your resume, an introductory letter and any other items that you feel will raise our eyebrows.
PHP experience:	1-2 years
Other skills:	DHTML, CSS, DOM
Worksite:	Onsite
Job Type:	Permanent , Full-time
Annual Salary:	From \$48000 To \$60000
Company info:	Insight Out Of Chaos http://www.iooc.com Contact name: Seth J Hersh Contact email: seth@iooc.com 220 East 23rd St Suite 600 New York, New York 10010 United States Phone: 212 935 0044 Fax: 646 742 1245
Entered:	April 25, 2005

Figure 6.16: Example of Job Offer

6.3.2 Conclusion and Future work in Znalosti Project

Project Znalosti gives us a good opportunity to evaluate and evolve a semantic annotation solution. It is important to evolve this solution also within the entire EMBET system because it is feasible, when percentage of relevant annotation results is increasing. We will work further in this area to find non-directly related ontology elements, create non-existing individuals or properties and evaluate relevance of different annotation approaches.

6.4 Conclusion

We believe that this chapter shows a wide scope of use of the EMBET system. Results have been tested and evaluated in several applications and the work will continue with the K-Wf Grid and Znalosti project.

7 Conclusion

The world economy is moving towards the so called knowledge economy, where knowledge is what is valued more than any other resource. Companies value their knowledge and try to rebuild themselves towards a knowledge enterprise. This effort includes creation and maintenance of knowledge or experience management systems, which strengthened motivation for creation of the EMBET System.

People like to enter notes or memos. This is a natural way of notifications for themselves or others to remind them of problems, activities or solutions. Therefore we think that such solution can be successfully applied and commercialized with good results.

The approach has been successfully used and evaluated in several R&D projects, mainly in the projects K-Wf Grid [KWFGGRIDWEB] (Knowledge-based Workflow System for Grid Applications EU RTD IST FP6-511385) and Pellucid project [PELL02] (A Platform for Organisationally Mobile Public Employees EU 5FP IST-2001-34519). Such solution may be applied in many further areas where the user problem can be detected. Usually this is in any business process where actions are performed via a computer, e.g. workflow processes, intranets, document management, supply chain management or dynamic business processes where email communication is in use.

7.1 Summary of work

The work describes the EMBET system based on ontology knowledge model and text notes representing experience. A model, modeling methodology as well as several knowledge manipulation algorithm were developed. It was proved that such system is implementable and can be used in different knowledge management applications.

7.2 Results

Generic Ontology Knowledge model for Experience Management – a model, which can be used for development of an experience management system was defined. Such a model is easily manageable with the Protégé ontology editor. Such a knowledge model can be extended and used to model many experience management applications with their environments —this has been demonstrated in several projects and applications.

Modeling Methodology for extending Generic model for Experience Management is a methodology, which adapted features of CommonKADS and Protégé as a methodology tool. It is usable for extending such model for different application domains.

The *EMBET System* is an experience management solution based on ontology and text notes; This covers examples of a use in applications as well as detail description of all system components.

Algorithms supporting actor's resources or context updating or semantic annotation according to the used knowledge model.

Presentation Methods based on existing commercial solutions for presenting of ontology based knowledge.

Concrete Examples of Use – use and evaluation of results in pilot operations.

7.3 Scientific Contribution

Presented work can effect several areas in computer sciences.

- Introduction of *Generic Ontology model for Experience Management* (Chapter 4.1). Using of such model leads developers to follow needed modeling extensions. The model clearly defines the application and the actor of environment and speeds up design and customization of Experience Management solutions for different applications.
- Introduction of *Modeling Methodology* (Chapter 4.2). Its contribution lies in adoption of existing methodologies and tools for experience management model which helps with applying such model.
- Introduction of *EMBET System* (Chapter 5). Presents design and implementation of the experience management solution based on ontology and text notes; Using of EMBET library enables the knowledge management community to exploit the RDF/OWL based models for the experience management, use developed semantic annotation algorithm for creating ontological metadata for concrete ontology model and show reusable presentation solution of RDF/OWL based ontological data.
- Presented Model, Methodology, EMBET System and Library can accelerate movement from scientific results to commercial application in the field of Experience Management.

7.4 Future Work

Presented work will be further developed and extended. Such process has already begun and we are trying to apply the developed model, methodology and infrastructure to solve different problems.

We will strive to focus on developing better algorithms for context matching and user context updating, to develop mechanisms for easier integration of the experience management solution with existing systems and support easier knowledge maintenance in the system. The future work will continue with currently running projects K-Wf Grid [KWFGRIDWEB] (Knowledge-based Workflow System for Grid Applications EU RTD IST FP6-511385), Raport (Research and development of a knowledge based system to support workflow management in organizations with administrative processes APVT-51-024604) and Znalosti (Tools for acquisition, organization and maintenance of knowledge in an environment of heterogeneous information resources SPVV 1025/04).

Bibliography

- [ANOTEJA]: SWAD, Annotea Project, 2001, <http://www.w3.org/2001/Annotea/>
- [BAL03A]: Zoltan Balogh, Michal Laclavik, Ladislav Hluchy, Use of Ontology in Virtual Organizations for Environment Risk Management, 11th ERCIM Environmental Modeling Group Workshop 2003, Smolenice, Slovakia, 2003
- [BAL03B]: Zoltan Balogh, Michal Laclavik, Ladislav Hluchy, Pellucid Agent Architecture for Knowledge Management in Public Organizations, Informatika 2003, Bratislava, Slovakia, 2003
- [BAL04A]: BALOGH Z., LACLAVIK M., HLUCHY L., BUDINSKA I., KRAWCZYK K., REMARK - Reusable Agent-Based Experience Management and Recommender Framework, Springer Werlach in ICCS'04/ Agent Day 2004; Proc. of International Conference on Computational Science, 2004
- [BERG02]: Ralph Bergmann, Experience Management: Foundations, Development Methodology, and Internet-Based Applications (Lecture Notes in Artificial Intelligence), ISBN:3540441913, 2002
- [BSWEB]: Berners-Lee, Building the Semantic Web, 2001, <http://www.xml.com/pub/a/2001/03/07/buildingsw.html>
- [CARTOO]: Cartoo Technologies, Knowledge Presentation by Graphs, 2004, <http://www.kartoo.net/>
- [CKADS]: August Schreiber et al., Knowledge Engineering and Management: the CommonKADS methodology, ISBN:0-262-19300-0, 2000
- [DAML]: DARPA, DAML Website, 2002, <http://www.daml.org/>
- [DAVEN00]: Thomas H. Davenport, Laurence Prusak, Working Knowledge, ISBN:1578513014, May, 2000
- [DENNY02]: Denny M., Ontology Building: A Survey of Editing Tool, 2002, <http://www.xml.com/pub/a/2002/11/06/ontologies.html>
- [DLBOOK]: Franz Baader, Deborah McGuinness, Daniele Nardi, The Description Logic Handbook, ISBN:0521781760, January 9, 2003
- [EXPEXT]: Softeco, Lino Ferrentino, Simona Stringa, Experience Management Extract – Internal Pellucid Document, October 2003

- [FIPASL]: FIPA, FIPA SL Content Language Specification, 2000
- [HORR04]: Ian Horrocks, Reasoning with Expressive Description Logics: Logical Foundations for the Semantic Web, Keynote talk at ICIP, Beijing, China, October 2004
- [JENAWEB]: HP Labs and Open Source Community, Jena Semantic Web Library, 2004, <http://www.sf.net/>
- [KIFWEB]: American National Standard, KIF specification, 1998, <http://logic.stanford.edu/kif/dpans.html>
- [KMG04]: KITOWSKI J., ... HLUCHÝ L., BALOGH Z., LACLAVÍK M., ... , Model of Experience for Public Organisations with Staff Mobility, Proc. of IFIP Conference (KMGov 2004), Wimmer, M.A. (Ed.), LNCS 3035, Springer-Verlag, ISSN 0302-9743, Krems, Austria, 2004
- [KWFGRIDWEB]: K –Wf Grid Consortium, K –Wf Grid IST Project Website, 2005, <http://www.kwfgrid.net/>
- [LAC03A]: M. Laclavik, Z. Balogh, L. Hluchy, G. T. Nguyen, I. Budinska, T. T. Dang, Pellucid Agent Architecture for Administration Based Processes, IAWTIC 2003, Vienna ,2003
- [LAC03C]: Michal Laclavik, Zoltan Balogh, Ladislav Hluchy, Renata Slota, Krzysztof Krawczyk, Mariusz Dziewierz, Distributed Knowledge Management based on Software Agents and Ontology, PPAM 2003, Czestochowa, Poland, 2003
- [LAC04A]: LACLAVÍK M., BALOGH Z., HLUCHÝ L., KRAWCZYK K., DZIEWIERZ M., KITOWSKI J., MAJEWSKA M, Knowledge Management for Administration Processes, Proceedings of Znalosti 2004, February 2004
- [LAC05A]: Michal Laclavik, AgentOWL: OWL Agent memory model, 2005, <http://jade.tilab.com/community-3rdpartysw.htm#AgentOWL>
- [LAC05B]: LACLAVÍK M., BALOGH Z., NGUYEN G.T., GATIAL E., HLUCHÝ L., Methods for Presenting Ontological Knowledge to the User, Proceedings of Znalosti 2005, Vysoke Tatry, Slovakia, 2005
- [LAC05C]: Michal LACLAVIK, Emil GATIAL, Zoltan BALOGH, Ondrej HABALA, Giang NGUYEN, Ladislav HLUCHY , Experience Management Based on Text Notes (EMBET), Challenges Conference, 19 - 21 October 2005, Ljubljana, Slovenia ,2005
- [LAC05D]: Michal Laclavík, Zoltán Balogh, Emil Gatial, Ondrej Habala, Giang Nguyen, Ladislav Hluchý, e-Collaboration and Knowledge Sharing based on Text Notes, Informatika 2005, Bratislava, Slovakia, 2005
- [OILED]: , Ontology Editor: OilEd, 2004, <http://oiled.man.ac.uk/>
- [OMGWEB]: OMG Group, OMG Group Website, 2005, <http://www.omg.org/>
- [ONTOVIZ]: Michal Sintek, OntoViz Tab: Visualizing Protégé Ontologies, 2004, <http://protege.stanford.edu/plugins/ontoviz/ontoviz.html>
- [OWLWEB]: W3C, Web Ontology Language OWL, 2004,

- <http://www.w3.org/TR/owl-features/>
- [PELL02]: Pellucid Consortium, Pellucid IST Project Website, 2002, <http://www.sadiel.es/Europa/pellucid/>
 - [PELL03]: Pellucid Consortium, Knowledge management for organisationally mobile public employees, KMGov 2003, Rhodes Island, Greece, 2003
 - [PELLD4]: Pellucid Consortium, Deliverable D4 Environment and Requirements, , 2002
 - [PELLTA]: Pellucid Consortium, Pellucid Project Technical Annex, 2001
 - [PROTKADS]: Guus Schreiber, Monica Crubezy, Mark Musen, A Case Study in Using Protege-2000 as a Tool for CommonKADS, , 2001
 - [PROTWEB]: Stanford University, Protégé Ontology Editor, 2004, <http://protege.stanford.edu/>
 - [RDFANNOT]: The Institute for Learning and Research Technology, RDF Annotations, 2001, <http://ilrt.org/discovery/2001/04/annotations/>
 - [RDFDBWEB]: , rdfDB Query Language, 2004, <http://guha.com/rdfdb/query.html>
 - [RDFW3C]: W3C, Resource Description Framework RDF, 2004, <http://www.w3.org/RDF/>
 - [RUBY]: W3C, Ruby Annotation, 2001, <http://www.w3.org/TR/ruby/>
 - [SQISHQL]: Miller, Libby; Seaborne, Andy; Reggiori, Alberto, Three Implementations of SquishQL, a Simple RDF Query Language, , 2002
 - [TSWEB]: Edited by Davies J., Fensel D., van Harmelen F., John Wiley & Sons, TOWARDS THE SEMANTIC WEB Ontology-driven Knowledge Management, ISBN:, 2003
 - [UML0MG]: OMG Group, The Unified Modeling Language – UML, 2005, <http://www.uml.org/>
 - [UMLWEB]: UML.ORG, Unified Modeling Language, 2004, <http://www.uml.org/>
 - [WANG01]: Paul P. Wang (Editor), Computing with Words,, ISBN:ISBN: 0-471-35374-4, 2001
 - [WOOL02]: Michael Wooldridge, Introduction to MultiAgent Systems, ISBN:047149691X, 2002
 - [XMLRPC]: xml-rpc.com, XML-RPC Home Page, 2005, <http://www.xml-rpc.com/>
 - [XMLW3C]: W3C, eXtensible Markup Language XML, 2004, <http://www.w3.org/XML/>
 - [XSLW3C]: W3C, XSLT, 2004, <http://www.w3.org/Style/XSL/>

Resumé

V dnešnom svete majú znalosti čoraz väčšiu hodnotu vo všetkých oblastiach hospodárstva ako aj v každodennom živote. Práca opisuje riešenie pre manažment skúseností s použitím znalostných ontológií.

Prezentované riešenie pre manažment skúseností je založené na textových poznámkach napísaných bežným užívateľom. Hlavná myšlienka spočíva v tom, že užívateľ vloží takéto poznámky v určitej situácii (kontexte), ktorá môže byť zistená v počítači. Poznámky môžu byť v budúcnosti zobrazené iným užívateľom v podobnej situácii (kontexte). Kontext užívateľa sa zistí z počítačových úkonov užívateľa. Nie celý zistený kontext je relevantný pre danú poznámku a preto systém za pomoci sémantickej anotácie, ontologického znalostného modelu a s pomocou užívateľa musí zistiť čo je relevantné. Aplikačná doména je popísaná tiež pomocou ontologického modelu.

Navrhnuté riešenie bolo úspešne použité a vyhodnotené v niekoľkých výskumno- vývojových projektoch, v projekte Pellucid a K-Wf Grid a je ďalej rozvíjané v rámci projektov K-Wf Grid a Znalosti.

Na modelovanie znalostí bola použitá CommonKADS metodológia a Protégé ontologický editor. OWL ontologický jazyk bol použitý na reprezentovanie znalostí. Sémantická knižnica Jena bola použitá na prácu so znalostným modelom a Java programovací jazyk bol použitý na implementáciu systému.