

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF INTELLIGENT SYSTEMS

## UBIQUITOUS LEARNING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

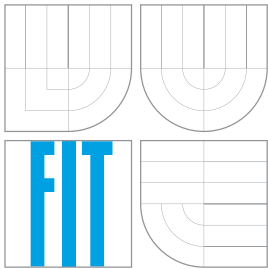
AUTHOR

MIROSLAV SOBOTKA

BRNO 2007



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**  
BRNO UNIVERSITY OF TECHNOLOGY



**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**  
**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**

FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF INTELLIGENT SYSTEMS

## **UBIQUITOUS LEARNING**

UBIQUITOUS LEARNING

**DIPLOMOVÁ PRÁCE**

MASTER'S THESIS

**AUTOR PRÁCE**

AUTHOR

**MIROSLAV SOBOTKA**

**VEDOUCÍ PRÁCE**

SUPERVISOR

**Ing. BOHUSLAV KŘENA, Ph.D.**

BRNO 2007

## Abstrakt

Tato práce se věnuje všudypřítomnému vzdělávání (u-learning) a mapování ontologií, je součástí dlouhodobého projektu MAPLE a vypracoval jsem ji během studijního pobytu na francouzské univerzitě v La Rochelle, v rámci mezinárodního výměnného programu Socrates-Erasmus. Hlavní cíle práce lze rozdělit na dvě související části. První část má za úkol seznámit čtenáře s prostředím pro všudypřítomné vzdělávání a s vhodnými dostupnými nástroji. Jedním z těchto nástrojů je i komerční program LMA, který je určený pro tvorbu mobilních prezentací a kurzů, důraz je kladen na jejich přenositelnost mezi jednotlivými mobilními platformami. Cílem je demonstrace jeho funkčnosti a vytvoření zkušební mobilní prezentace. Dalším krokem je seznámení čtenáře s virtuálním výukovým prostředím (VLE) a systémy pro správu a řízení výukových kurzů (CMS). Vybraný, volně dostupný CMS se jménem Moodle jsem nainstaloval, nakonfiguroval a vytvořením fiktivního kurzu otestoval a vyzkoušel jeho vlastnosti a funkčnost. Druhá část práce se zabývá ontologiemi, dostupnými nástroji pro práci s nimi a mapovacími algoritmy. Čtenář je seznámen s patřičnou teorií a s motivacemi pro využití ontologií v projektu MAPLE, větší prostor je věnován volně dostupnému programu Protégé, který slouží hlavně jako editor ontologií, ale díky jeho rozšiřitelné architektuře může být využit k nejrůznějším účelům. Důležitým bodem práce je analýza a realizace doporučeného mapovacího algoritmu, který jsem implementoval jako zásuvný modul pro program Protégé. Cílem projektu MAPLE je návrh a realizace pedagogického systému pro mobilní, aktivní a participativní výukové prostředí, které obohatí tradiční model prezenční formy výuky. Díky mobilním zařízením, spolupracujícím s pracovní stanicí vyučujícího, budou mít studenti možnost okamžité zpětné vazby. Dalšími záměry projektu jsou mobilní přístup k datům a výukovým kurzům, online komunikace a spolupráce zúčastněných osob, efektivní vedení a administrace kurzů pro vyučující a také spolupráce a propojení již existujících systémů v jeden funkční celek. Motivací pro využití ontologií v projektu MAPLE je správa výukových modulů a prezentací, jejich efektivní katalogizace, znázornění závislostí a vztahů mezi uloženými daty a jejich rychlé a přesné vyhledání. Mapovací algoritmus umožní spolupráci mezi moduly a ostatními databázemi, které používají rozdílné ontologie pro popis a charakterizaci uložených dat.

## Klíčová slova

Všudypřítomné vzdělávání, ontologie, mapování ontologií, mobilní vzdělávání, MAPLE projekt, Protégé, LMA, systémy pro správu kurzů, virtuální výukové prostředí

## **Abstract**

This work is devoted to the ubiquitous learning and the ontology mapping. The former part presents the environment of ubiquitous learning and available tools. The aim is to show the way to the mobile content creation, using the LMA software and the realization of the virtual learning environment – installation, testing and setting up of the Moodle open source course management system. The ontology mapping part acquaints readers with ontologies and ontology mapping, the aim is to analyze available resources, to choose suitable tools and finally to realize own implementation of ontology mapping algorithm. This work is a part of the MAPLE project, the overall objective is to enable realization of pedagogical framework for a mobile, active and participative learning environment.

## **Keywords**

Ubiquitous learning, ontology mapping, m-learning, MAPLE classroom, Moodle, Protégé, Virtual Learning Environment, Course Management System, Learning Mobile Agent

## **Citace**

Miroslav SOBOTKA: Ubiquitous Learning, diplomová práce, Brno, FIT VUT v Brně, 2007

# Ubiquitous Learning

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Bohuslava Křeny, Ph.D., další informace a pomoc mi poskytl pan Dr. Michel Eboueya. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....  
Miroslav SOBOTKA  
21st May 2007

## Poděkování

Chtěl bych poděkovat mému vedoucímu práce panu Ing. Bohuslavu Křenovi, Ph.D., za pomoc, rady a konzultace, panu Dr. Michel Eboueya, který byl mým vedoucím a zároveň zadavatelem diplomové práce během mé zahraniční stáže, a také paní Michaele Studené, která mi pomohla zahraniční výjezd realizovat.

© Miroslav SOBOTKA, 2007.

*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                                       | <b>3</b>  |
| 1.1      | Context of the work . . . . .                             | 3         |
| 1.2      | Aim of the work . . . . .                                 | 4         |
| <b>2</b> | <b>Ubiquitous learning</b>                                | <b>6</b>  |
| 2.1      | Basic definitions . . . . .                               | 6         |
| 2.1.1    | Ubiquitous learning environment . . . . .                 | 7         |
| 2.1.2    | Course Management System . . . . .                        | 8         |
| 2.1.3    | Virtual learning environment . . . . .                    | 9         |
| 2.1.4    | Mobile content . . . . .                                  | 9         |
| 2.2      | Available tools . . . . .                                 | 10        |
| 2.2.1    | Learning Mobile Agent . . . . .                           | 10        |
| 2.2.2    | Moodle Course Management System . . . . .                 | 13        |
| 2.3      | Pilot mobile presentation . . . . .                       | 16        |
| 2.3.1    | Using Learning Mobile Agent . . . . .                     | 17        |
| 2.3.2    | Possible drawbacks . . . . .                              | 17        |
| 2.4      | Moodle installation and testing . . . . .                 | 18        |
| 2.4.1    | The Microsoft Virtual PC . . . . .                        | 18        |
| 2.4.2    | Moodle on virtual server . . . . .                        | 19        |
| 2.4.3    | Moodle in practise . . . . .                              | 20        |
| <b>3</b> | <b>Ontology mapping</b>                                   | <b>22</b> |
| 3.1      | Introduction to ontologies and ontology mapping . . . . . | 22        |
| 3.1.1    | Ontologies . . . . .                                      | 22        |
| 3.1.2    | Ontology mapping . . . . .                                | 26        |
| 3.2      | Working with ontologies . . . . .                         | 27        |
| 3.2.1    | RDF . . . . .   | 28        |
| 3.2.2    | OWL . . . . .   | 29        |
| 3.2.3    | Frameworks . . . . .                                      | 30        |
| 3.2.4    | Methods and tools . . . . .                               | 31        |
| 3.2.5    | Protégé . . . . .   | 32        |
| 3.2.6    | Summary . . . . .   | 34        |
| 3.3      | Protégé & OWL . . . . .                                   | 35        |
| 3.3.1    | Protégé-OWL API . . . . .                                 | 35        |
| 3.3.2    | Protégé Plug-in Development . . . . .                     | 36        |
| 3.3.3    | Summary . . . . .   | 37        |
| 3.4      | Ontology mapping algorithm . . . . .                      | 38        |
| 3.4.1    | Proposed design . . . . .                                 | 38        |

|          |   |           |
|----------|---|-----------|
| 3.4.2    | Proposed improvements . . . . .             | 47        |
| 3.5      | Implementation of Protégé plug-in . . . . . | 48        |
| 3.5.1    | Implementation details . . . . .            | 48        |
| 3.5.2    | Plug-in interface and control . . . . .     | 49        |
| <b>4</b> | <b>Conclusion</b> . . . . .                 | <b>52</b> |
| 4.1      | Achieved results . . . . .                  | 52        |
| 4.2      | Future work . . . . .                       | 52        |

# Chapter 1

## Introduction

This diploma thesis evolved during an international scholarship (*Socrates-Erasmus* an exchange program) at the University of La Rochelle, Faculty of Science (Laboratory L3i), where I spent a winter semester and a month (total duration of my stay was 5,5 months).

The main purpose of my stay was to elaborate my final project, but I have also completed courses on computer networks and security – *Administration Systèmes et Réseaux Sécurité*, an English course – *Cours d'anglais* and a French language course *CUFLE (Cours de français pour les étudiantes étrangères)*. Studying these courses was very interesting experience and also a strong improvement for my French language skills.

This thesis was created under supervision of Dr. Michel EBOUEYA (taskmaster) and Ing. Bohuslav Křena and is a part of the MAPLE research project (*Mobile Active Participative Learning Environment*). The closest predecessor is a work of S. Niwattanakul (presented at conferences viz. [17] and [18]) and work of Jiří Špička ([22], elaborated during an internship in the summer semester 2006).

During my stay, I attended the conference *Assises du GDR CNRS I<sup>3</sup>* organized in La Rochelle (15.-17. January 2007) by research group *Laboratory L3i*. I also participated in the two-day *L3i séminaire Île de Ré* (1.-2. February 2007), organized by the same research group. This was a great opportunity to meet other researchers in the *L3i* group and to find out how this laboratory works.

### 1.1 Context of the work

The aim of the MAPLE (Mobile Active Participative Learning Environment) project is to establish lightweight effective systems for future classrooms using advanced tools and combining advantages of face-to-face teaching processes with e-learning (m-learning) educational methods. My goal is to maximize utilization of existing tools, adapt them to set up co-operation, and to create purposive systems.

The basic principle is to create an online learning community and knowledge database, using an open source Course Management System (Moodle), following the ideas of grid computing. All services are easily accessible through the WiFi network. The content of a knowledge database is designated to mobile devices and PDA's, created with LMA30 (Learning Mobile Author).

Concerned elements are classified by ontologies, using a Protégé ontology editor and knowledge-base framework, thus desired materials should be effectively traced. To allow cooperation with a different technique of data description, ontology mapping algorithm



have been implemented like a Protégé plug-in.

**The objective of the MAPLE project** is to design and implement a pedagogical framework for a mobile, active and participative learning environment. A MAPLE classroom enhances the traditional face-to-face delivery mode – all learners can provide immediate feedback to the teacher through a ubiquitous mobile device which is summarised in real-time on the teacher workstation. MAPLE can be used for immediate feedback, real-time assessment, collaborative problem-solving, building learning communities, participative simulations and peer-directed learning, and in teacher training. MAPLE is also a resource of training courses and learning materials, which can be easily traced up by advanced searching algorithms.

**A MAPLE classroom** is applicable in most educational contexts but the target group for this project is learners in higher education. A taxonomy of learner cohorts will be developed to permit holistic comparisons of the effectiveness of MAPLE which will include: minority groups (gender imbalance, races/cultures, disadvantaged, special needs); undergraduate and postgraduate levels.

## 1.2 Aim of the work

The aim of this work can be divided into two parts – *Ubiquitous Learning* and *Ontology Mapping*.

### Ubiquitous Learning

The main objective of the first part is to be familiarized with the ubiquitous learning environment and available applications and tools. The next step is the creation of a pilot mobile presentation in LMA and summarization of LMA's functionality and principles of work.

The last part of this section is work with course management system (CMS), to acquaint and to set up Moodle, and to test its configuration.

### Ontology Mapping

The objective of this section is to learn why and how to use ontologies, how to work with them and to understand this issue. The next step is to analyze available applications and environments, to choose suitable tools and to decide which techniques will be the best for mapping algorithm realization.

The final part is devoted to mapping algorithm – firstly an introduction and proposed design, followed by own improvements and suggestions and finally the realization of this mapping algorithm in the chosen environment.

### Semestral and annual projects

There is no connection between my semestral and annual projects and this diploma thesis. The reason is established practice in the Erasmus exchange program – student have to accept arranged assignment proposed by foreign university.

I have passed through the same procedure, so I did not have the possibility of continuation in my annual project. I have changed completely the course of study, from the system for automatized generation of documentation I have moved to the proposed theme – ubiquitous learning and ontology mapping.

## Chapter 2

# Ubiquitous learning

*Ubiquitous* can be defined as “existing or being everywhere at the same time”, “constantly encountered” and “widespread”. *Ubiquitous computing* (a.k.a. pervasive computing) integrates computation into the environment, rather than having computers which are distinct objects so we can say that technology is everywhere and we use it all the time.

### 2.1 Basic definitions

*Ubiquitous learning* (u-learning) can be understood like an equivalent of mobile learning, e.g. that learning environments can be accessed in various contexts and situations, but the objectives of ubiquitous learning can be characterized as the following list [25]:

**Permanency:** Learners never lose their work unless it is purposefully deleted. In addition, all the learning processes are recorded continuously everyday.

**Accessibility:** Learners have access to their documents, data, or videos from anywhere. That information is provided based on their requests. Therefore, the learning involved is self-directed.

**Immediacy:** Wherever learners are, they can get any information immediately. Thus, learners can solve problems quickly. Otherwise, the learner can record the questions and look for the answer later.

**Interactivity:** Learners can interact with experts, teachers, or peers in the form of synchronous or asynchronous communication. Hence, the experts are more reachable and the knowledge becomes more available.

**Situating of instructional activities:** The learning could be embedded in our daily life. The problems encountered as well as the knowledge required are all presented in their natural and authentic forms. This helps learners notice the features of problem situations that make particular actions relevant.

**Adaptability:** Learners can get the right information at the right place with the right way.

The learning process can be divided into two main branches – face-to-face (present) learning and distance learning (*d-learning*). Face-to-face learning represents classical presence teaching in classrooms whereas the d-learning branch stands for distance education – learners and teacher do not have to be on the same place in the same time.

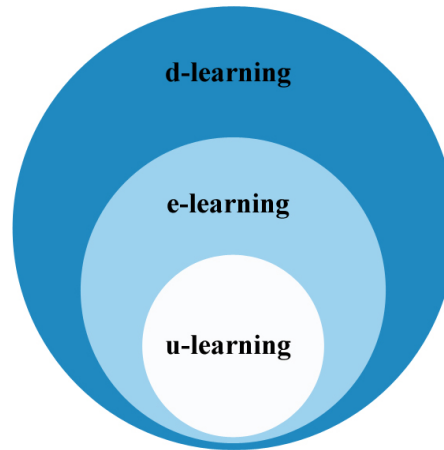


Figure 2.1: Relationship between d-learning, e-learning, and u-learning

As can be seen on the figure 2.1, the subset of d-learning is *e-learning*. E-learning is an approach to facilitate and enhance learning through both computer and communications technology (personal computers, CDROM's or television, Internet, email, discussion forums, collaborative software, etc.).

Ubiquitous learning (u-learning), also denoted like a *mobile learning* (m-learning) is the subset of e-learning. Difference between e-learning and m-learning can be seen on the table 2.1.

### 2.1.1 Ubiquitous learning environment

*Ubiquitous* = pervasive, omnipresent, ever present, everywhere

*Learning* = educational, instructive, didactic, pedagogical

*Environment* = surroundings, setting, situation, atmosphere

Ubiquitous learning environment (ULE) is an adaptive teaching system using ubiquitous technology (ubiquitous computing). The aim of ULE is to enable an easy access to learning resources and to grant ubiquitous education for all learners.

The requirements of the MAPLE project are specific and should also be slightly modified during future evolution. Practical testing and real service will affect numerous groups of people and will take a considerable amount of resources, thus the selection of an appropriate learning environment is an important part of the project.

#### Requirements of u-learning

Many students are moving around and their nature is becoming increasingly nomadic. Thus, the question presents itself as how do these students access a flexible learning environment when the methods of access are restricted to a desktop computer?

The answer is to enable our learning environment from accessible and more portable devices, such as personal digital assistants (PDA) that can be physically carried by students and is not as cumbersome as a laptop computer. This is known as ubiquitous access and is a common foundation of what is known as u-learning. The remainder of this research

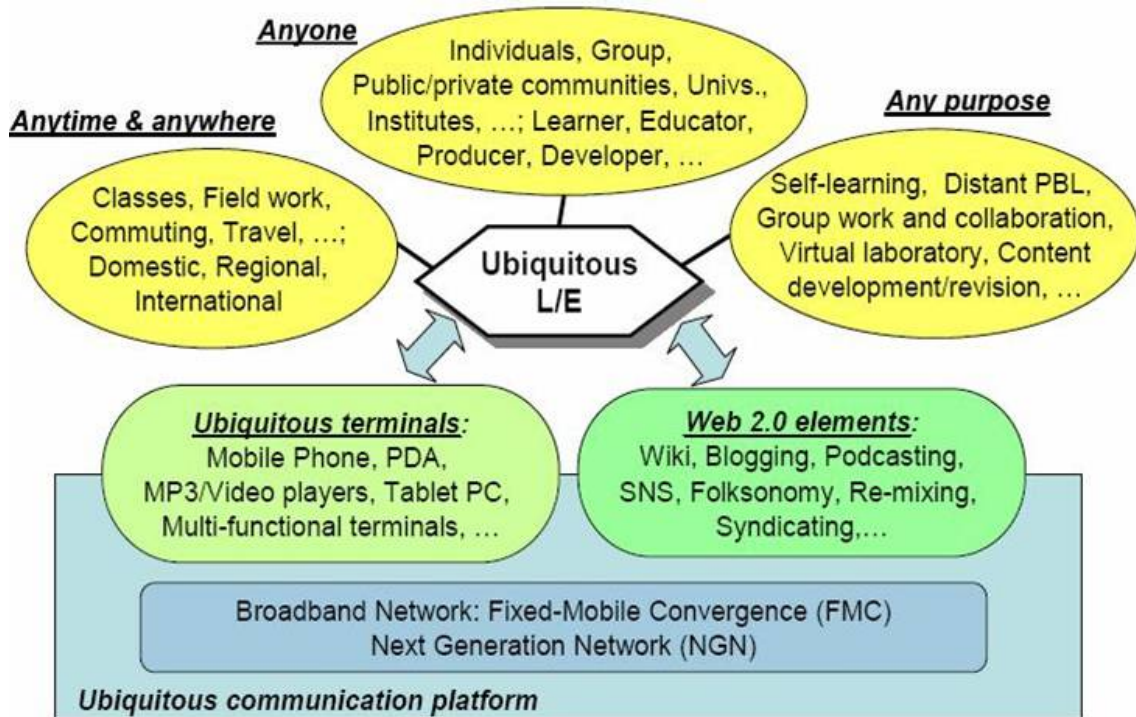


Figure 2.2: Ubiquitous Learning Environment

paper will concentrate on the implementation details of enabling ubiquitous access to the Course Management System (CMS) unit [15].

### 2.1.2 Course Management System

A Course Management System (CMS) represents a new form of educational technology. CMS is a set of teaching and learning tools designed to enhance a student's learning experience by including computers and the Internet in the learning process. Learners and teachers may work closely together while not being active at the same time and space – asynchronous learning.

CMS is a software package that provides support for managing student access to courses, monitoring student activity, administering and scoring student progress and performance tests, storing student records, managing access to student and course records, and reporting on student and course activity.

Principal components of CMS are:

- mapping of the curriculum into course topics that can be assessed and recorded
- tracking of student activity and achievement within the curriculum presented online
- support of online learning, including access to learning resources, assessment and guidance
- online tutor support
- peer group support

- general communications, including email, group discussion and web access
- links to other systems, both in-house and externally

A CMS allows active teacher management of classroom participation, to provide information of student learning progress, serve as a basis for student feedback, allow real time adjustments to instructional presentations, and allow the teacher to focus on the instructional needs of specific students. A course management system allows the teacher to implement various video, audio, graphics, text, and programs without regard to their data format which enables teachers to electronically present instructional materials from different source formats in automated classrooms. As well, a CMS provides computer mediated support for course development and revision, either through built-in functions within the software of the CMS, or through integration with other curriculum development tools [5, p.10].

Advanced CMS provides synchronous and asynchronous tools for both teacher and students such as e-mail, threaded discussions, chat rooms, white boards, video conferencing, online testing and evaluation, bulletin boards, and teleconferencing.

### 2.1.3 Virtual learning environment

The education using computers, networks and new technology is topical problem and lot of people are involved in development and usage. That is why we can find many abbreviations and terms with similar meanings or very small differences.

As mentioned in [28], *computerised learning*, mostly called *e-learning* is facilitated by e-learning systems. These e-learning systems are called *Course Management System (CMS)* or also *Learning Management System (LMS)*, *Learning Content Management System (LCMS)*, *Managed Learning Environment (MLE)*, *Learning Support System (LSS)* or *Learning Platform (LP)*; it is education via *Computer-Mediated Communication (CMC)* or *Online Education (OE)*.

A more correct term may be a *virtual environment for learning*, rather than *virtual learning environment*. This removes any ambiguities and identifies that it is the environment which is virtual and not the learning.

In the United States, CMS and LMS are the more common terms, however LMS is more frequently associated with software for managing corporate training programs rather than courses in traditional education institutions.

In the United Kingdom and many European countries the terms VLE and MLE are favoured, however it is important to realize that these are two very different things. A VLE can be considered a sub system of a MLE, whereas MLE refers to the wider infrastructure of information systems.

I do not want to confuse readers of my work, I have mentioned these abbreviations and terms, but I use only the CMS which can be considered like a tool for VLE realization or in a simplified way CMS and VLE are synonyms.

### 2.1.4 Mobile content

Mobile content accessible through CMS (courses, presentations, tests, multimedias) should be adjusted to the environment of mobile devices.

### Small screen problems

One of the biggest challenges concerning the mobile devices is to find acceptable solutions adapted to the small screen size. There is simply not enough space for all the information found on a traditional web page on the small screen. Another problem is the limited data transfer rate and processing power found in mobile devices. We might want to serve the mobile users an alternative page compared to that of which is served to our traditional clients. There are some mobile browsers that could reduce the need of developing specific pages for the mobile client. This is not standard at the time of writing.

### Possible solutions

Possible solutions presented in [22]:

- Using browsers – web browsers for mobile devices, for example Pocket Internet Explorer for the Pocket PC Mobile operation system. However, PIE is limited by both the device on which it runs and in functionality.
- Identifying target devices – the solution is to implement a facility where the requesting web browser is firstly identified and decisions regarding content and its display format are made accordingly. This is done because of differences between desktop web browsers and browsers for small devices, for example Internet Explorer and Pocket Internet Explorer.
- Cascading style sheets – Cascading Style Sheets (CSS) are used to add screen properties such as colour and text sizes. A CSS file is used to set these properties for the whole web site, avoiding the need of setting these properties for each individual HTML page.
- Themes in Moodle CMS (see the section 2.2.2) – another possible solution how to enhance the learning content readability in Moodle is changing the theme (for example applying the PDA theme called “orangewhitepda”).

| <b>FUNCTIONALITY</b>          | <b>MOBILITY</b>  |
|-------------------------------|--|
| Computers<br>Laptop computers | PDA's, Handhelds, Palmtops<br>Smartphones, Mobile phones |
| <b>E-learning</b><br>←→       | <b>M-learning</b><br>←→                                  |

Table 2.1: Displaying e-learning content on small displays ([22])

## 2.2 Available tools

### 2.2.1 Learning Mobile Agent

Mobile content accessible through CMS (courses, presentations, tests, multimedias) should be adjusted to the environment of mobile devices. One of the possible solutions is to use LMA30 (Learning Mobile Agent) from Hot Lava Software, Inc (*www.hotlavasoftware.com*).

Learning Mobile Agent (LMA) is a user friendly proprietary software, which allows you to generate, customize, design, deploy, edit, and interact mobile content – an electronic document designated for mobile devices:

- Text, HTML
- Multimedia  
Image, Video, Sound
- Interaction  
Test, Quiz

### Working with LMA

The main screen of LMA is divided into three parts (viz. figure 2.3):

- Structure window** contains a hierarchical structure tree of whole module, where each entry represents one module item. The top level item in the list is the module itself, than all pages are lined up in the order they will be shown during presentation. By a simple click of mouse on the + icon beside the page entry, the content of each page can be expanded thus we can graphically see what is inside the page. With a right click of mouse various actions can be executed (add a new item, delete or modify existing item).
- Preview window** displays view of the selected page (selected page in the *structure window*). The preview represents final appearance of the page – how the page will be displayed during the module presentation, so the content and visual aspect of that page can be continuously checked and promptly modified.
- Edit window** enables editing of selected components. When we select an editable item in the *Structure window* (an item with content which can be changed) the content of the selected item is shown in the *Edit window* and there can be edited (added, changed, modified).

**Projects in LMA** are called Modules. Every module has **Name**, **Creator** (max. 4 characters) and **Contents** (various elements).

### Content elements :

**Page** (*basic element*) – page can contain all other elements which are marked like *page element*.

**Test** (*basic element*) – the test is not a page element, its standalone item which is on the same level as *page*, test can contain many partial *quiz* elements which will be evaluated and summarized in the end of *Test*. The user will pass all *quizzes* and no matter how he responds, the result page can be shown at the end of the *Test*, depending on the administrator's decision.



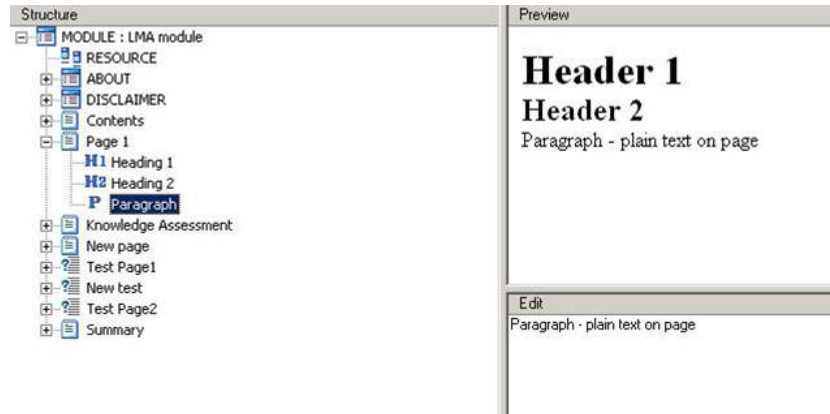


Figure 2.3: LMA main screen

**Heading1, heading2** (*page element*) – There are two types of headings, all headings will be shown in the table of content (TOC).

**Paragraph** (*page element*) – contains plain text.

**List** (*page element*) – represents the list of items.

**Image** (*page element*) – it is necessary to add the image to the *Resource list*, then it is possible to add that image like a picture to a page. The required color depth is 8 bit per pixel, a good standard for an image size is 100x75 pixels (depending on the target platform).

**Quiz** (*page element*) – represents a set of questions, during the presentation, users cannot pass *Quiz* until the correct answer is selected. If the wrong answers is chosen, an extra attempt to select right ones will be given.

**TOC** (Table Of Content) (*page element*) – summary of the whole module (Heading1 and Heading2), each item in the TOC works in reference to the item in the content

**Link** (*page element*) link has 3 parameters – **Link text**, **Link type** (external or internal) and **Link to** (reference to desired page, depends on link type). Link type:

- External – reference external website (URL).
- Internal – reference to any page in module which has header (*Heading1* or *Heading2*).

### LMA output

**-Palm** – exports the module to an executable file format under the Palm platform.

**-PocketPC** – exports the module to an executable file format under the PocketPC platform.

**-HTML** – generates the HTML version of presentation to a local repository. HTML does not have the capability for the right function of tests and quizzes, displaying only questions and correct answers.

-**Web / WAP** – 2 possibilities:

- Direct upload to server – modules can be exported into desired format and directly deployed to Hot Lava server <sup>1</sup>, login, password and a correct address of the server is required.
- Export to a local repository – modules can be manually uploaded to the Hot Lava server (using web portal).

-**Windows (PC)** – exports the module to the *.exe* file format. This module is bound to LMA viewer (a small application which interprets module functionality to Windows OS). This output possibility is dependent on a current license policy, e.g. in the trial *beta* version of LMA option was not accessible.

### 2.2.2 Moodle Course Management System

Purchased proprietary course management systems come with features that are necessary for the implementation of constructivism <sup>2</sup> in the classroom, depending on conditions. Generally all upgrades and updates, extra features, and each user will incur extra costs, so the purchase of software is not the only expense.

For our purposes, if we found that existing Open Source CMS was spread world wide, with a wide community of users and developers which ensures needful updates and future development, it should answer our needs better than closed and expensive proprietary tools. We have decided to use open source course management system called Moodle.

**MOODLE** – **M**odular **O**bject **O**riented **D**ynamic **L**earning **E**nvironment is a free, open source course management system for online learning. Moodle is an active and evolving open source course management system which fulfills the majority of our requirements.

Moodle promotes a social constructionist pedagogy – it promotes collaboration activities and critical reflection within a group. It is suitable for 100% online classes as well as supplementing face-to-face learning. Moodle installs a simple, lightweight, efficient, compatible, low-tech browser interface that is easy to use and intuitive to the student's needs. Moodle is easy to install on almost any platform that supports PHP, moreover it requires only one database.

The course listing shows descriptions for every course on the server, including accessibility to guests. Courses can be categorized and searched – one Moodle site can support thousands of courses. There is an emphasis on strong security throughout; forms are all checked, data validated, and cookies are encrypted. Also, text entry areas (resources, forum postings, journal entries etc.) can be edited using an embedded WYSIWYG HTML editor as long as the browser that is used can support it.

Moodle has more than 150,000 registered users and wide community of developers. Moodle is modular and can be enriched by numerous additional modules and plugins.

---

<sup>1</sup>Clients of Hot Lava Software have the possibility to create an account on the Hot Lava server, where modules can be stored and presented.

<sup>2</sup>Constructivism is a set of assumptions about the nature of human learning that guide constructivist learning theories and teaching methods of education. Constructivism values developmentally appropriate teacher-supported learning that is initiated and directed by the student [26].

## Requirements

Moodle installation is well documented on the official web site and the process is more or less user-friendly. In case of any trouble, help or additional information can be found easily thanks to the wide user community.

The idea of this section is not to copy instructions from Moodle documentation and web site, I only want to give a brief resume which will explain the issue. The majority of information originates from the official Moodle website [16].

The first rule by Moodle is *Firstly don't panic!*

**Hardware requirements** depends on the mode of application – the total number of users and size of stored data. Minimal configuration is 160MB of free space on hard disk and 256MB of RAM.

### Estimation of users number

*Browsing users* is the maximum number of users able to browse your Moodle site (number of computers in organization or on course).

*Concurrent database users* is the maximum number of concurrent database users (number of users who will be using Moodle at the same time).

The general approximate rule based on experience is:

$$\text{Max concurrent users} = \text{RAM(in GB)} * 50$$

$$\text{Max browsing users} = \text{Max concurrent users} * 5$$

### Software requirements

- **Web server software** – most common is use of Apache, but Moodle should work fine under any web server that supports PHP, such as IIS on Windows platforms.
- **PHP scripting language** – for Moodle version 1.6 or later, the minimum version of PHP is 4.3.0 (or 5.1.0).
- **Working database server** – MySQL or PostgreSQL are completely supported and recommended for use with any version of Moodle (support for Microsoft SQL Server and Oracle has been added in Moodle 1.7). For larger deployment, PostgreSQL is more suitable than MySQL, although MySQL is very popular. Moodle 1.6 or later requires MySQL 4.1.16 and the minimum version of PostgreSQL is 7.4.

**Download** of Moodle can be realized in two ways, like a compressed package or via CVS.

- From the official web site of Moodle, two types of compressed packages can be downloaded:
  - Standard distribution (only Moodle files) – latest stable release (1.7.1) has about 10MB (zip archive).
  - Complete install (also programs to operate Moodle in a web environment) – latest release of Moodle packages for Windows, built using XAMPP, has about 53MB (zip archive). Binaries for Mac OS X have 87MB.
- From *Moodle Sourceforge CVS repository*.

## Installation

Depends on number of users on desired Moodle server

**Smaller sites installation** (less than 30 users) simple installation, using complete install packages, can be done.

Because it is not easy to install an Apache web server and to add MySQL, PHP and Perl, all prerequisite software can be installed using XAMPP. XAMPP is an easy to install Apache distribution containing MySQL, PHP and also Perl.

The process of complete package installation is user friendly and fast and more information can be easily found on the web site.

**XAMPP security notice** for safety's sake :

*XAMPP is not meant for production use but only for developers in a development environment. The way XAMPP is configured is to be open as possible and to allow the developer anything he/she wants. For development environments, this is great. However in a production environment, it could be fatal. A list of missing security in XAMPP:*

- The MySQL administrator (root) has no password.
- The MySQL daemon is accessible via network.
- phpMyAdmin is accessible via network.
- Examples are accessible via network.

**Medium to large installations** – manual installation is recommended, all following steps have to be done.

1. **Plan your system capacity** – depends on the number of users in your organisation.
2. **Install your database server** – MySQL (recommended), Microsoft SQL Server 2005 (Moodle 1.7 or later) or Oracle.
3. **Install PHP** – instructions in *How to install PHP 5.x on Windows Server 2003 with IIS 6*.
4. **Install your web server** – Apache 2 (recommended), IIS or other web servers (Lighttpd).
5. **Install Moodle** – detailed information on the Moodle website.
6. **Setup backups** – full site backup, course backup or system state backup.
7. **Check your server security and performance** – RTM.
8. **Set-up Active Directory authentication** – LDAP authentication or integrated NTLM authentication .

## “Real world” Moodle installation

Proposed “real world” Moodle installation can be realized on the basis of the figure 2.4. Evidently this model is more hardware resource demanding, but the apparently complicated connection enables higher security of the system.

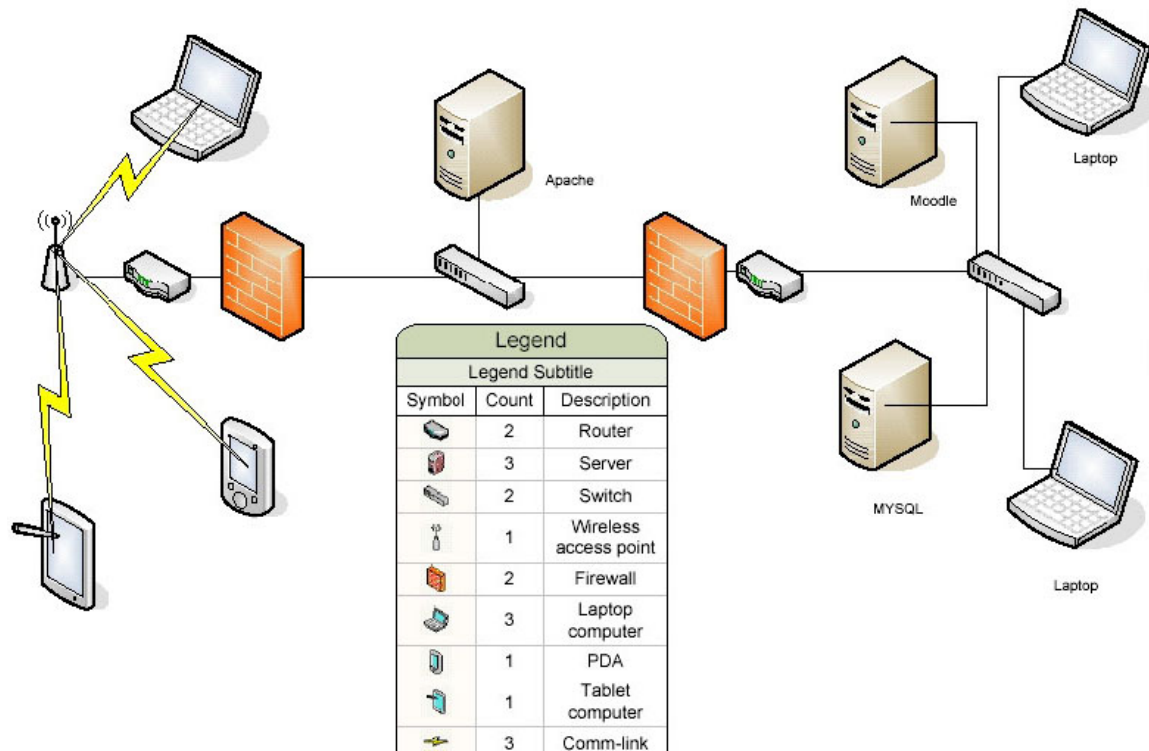


Figure 2.4: Proposed Moodle environment [22]

This environment contains 2 firewalls for security reasons, separating external wireless network (wireless access point) from local area network. The MySQL database, a part of the Moodle system, is presented by standalone database server.

Network schema recommendations [22]:

1. Install at least one firewall between internal Moodle/MySQL server and the wireless access point. With two firewalls, an Apache server can be placed between them in the perimeter network and achieve greater security.
2. At the external firewall (or router acting as a firewall) allow only traffic destined for the Apache server at TCP port 80.
3. At the internal firewall (or router acting as a firewall) allow ingress traffic from the Apache server to the Moodle/MySQL server only.
4. Be sure to mirror the traffic on the firewalls (or routers acting as firewalls).
5. Recommendation: install only one major service on a server. This will require more hardware, but will be easier to configure for security and backup purposes.

### 2.3 Pilot mobile presentation

One of aims of my work is to test discussed tools (LMA) and to create a pilot mobile presentation called “What Is Grid Computing?”



Figure 2.5: The LMA module “What is Grid Computing?” in the HTML format

### 2.3.1 Using Learning Mobile Agent

I have described main characteristics and features of LMA in the section 2.2.1. I have used successfully all available module elements and functions, thus I can claim the LMA software is working satisfactory.

Unfortunately I have not received promised PDA from my French director, so my possibilities of module testing were limited.

I have exported my module to all mentioned formats successfully, but the only format accessible for functionality testing was the HTML and the Web/WAP version. As mentioned in 2.2.1, in HTML version quiz and tests are not working properly (because of missing logic behind the HTML code), questions are displayed with highlighted right answers. The Web/WAP format was well working, I have utilized the Hot Lava server with mobile delivery and tracking system (MDTS), I have uploaded the module (using LMA and manually too) and I have tested successfully the functionality of uploaded modules.

Another problem came up with license policy of LMA software. University of La Rochelle bought the software, but the time of use was limited and my director economized it, so I was working with the trial version of LMA and some features were restricted. I came together with modules exported for desktop platform (Windows XP) like an .exe files, but my version of LMA was not able to make this conversion, so as I mentioned above, the only format accessible for me was the HTML and the Web/WAP. The representation (by Mozilla Firefox) of my module in HTML version can be seen on the figure 2.5.

### 2.3.2 Possible drawbacks

During the use of LMA, I ran into a few difficulties caused by non-logic control of some LMA functions, especially if someone is used to work with mainstream applications and their unwritten conventions. For instance, different responses during actions on elements with equivalent priority (deleting list item/paragraph), lack of “undo” and “redo” functions

– after deleting a paragraph it is necessary to rewrite it (unless there’s a backup copy). Another possible disadvantage is the license policy.

Finding suitable images or adapting the existing ones to avoid horizontal scrolling is also time consuming, but it is not the drawback of this software but a disadvantage in creating mobile presentations.

On the other side, the LMA seemed to be intuitive and familiar to use. The creation of modules was straightforward and the majority of requirements were satisfied.

## 2.4 Moodle installation and testing

For my testing purposes, I have decided to use a complete package installation on the Windows platform. Because of security reasons and for simulation of real server application, I have placed the Moodle server to the Microsoft Virtual PC environment (running on Windows XP) and I have established a virtual network connection between Host PC and a virtual Moodle server, as can be seen on the figure 2.6.

Reasons why I choose to realize this virtual model instead of the installation of a “real” Moodle server are the following:

- Security – in school laboratory conditions, it was a problem to integrate my server into the existing computer network – network administrators were not looking with favor on my attempt. Also as mentioned in the section 2.2.2, the XAMPP package is not configured securely, thus, these experiments with settings should be applied in a separate environment than in an existing network.
- Portability and practicality – especially for my purposes and manner of work, the portability of a realized solution is a very important attribute – the virtual Moodle server should be presented and discussed within range of my laptop – thus anywhere.

The installation of Moodle is briefly described in the section 2.2.2, I have followed instructions and I succeeded in the whole process of installation. As written above, for my purpose I chose the *complete package installation* using XAMPP, thus the whole process was pleasantly brisk.

### 2.4.1 The Microsoft Virtual PC

The installation of Microsoft Virtual PC is similar to the installation of a standard desktop computer, I have to set up a desired operation system and all necessary components. In my case, I have only installed the Windows XP operation system and set the virtual network between Host and Virtual PC, then the virtual machine was ready for the Moodle server installation.

To set up the virtual network, it is necessary to configure the Host PC:

- *Install Microsoft Loopback* adapter – a detailed guide is available on the Microsoft web site.
- Allow and set up a virtual network – in *Network Connections*, properties of Microsoft Loopback Adapter, TCP/IP settings, set the netmask (255.255.255.0) and IP address value – the best option is to use one from a range of non-routable TCP/IP addresses – for example an address of the form 192.168.x.y, where  $x \in \langle 0, 255 \rangle$  and  $y \in \langle 1, 254 \rangle$

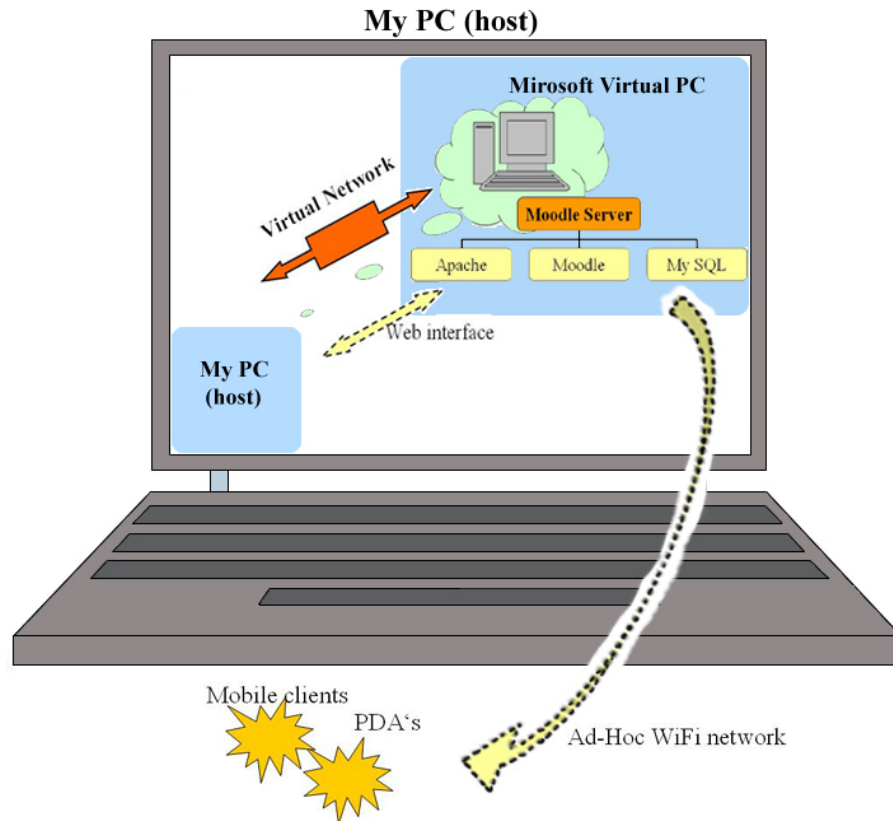


Figure 2.6: Testing configuration

(e.g., 192.168.0.2) . The value of  $x$  must be the same on the Host operating system and each guest operating system that is to be part of the virtual network.

and also configure the Virtual PC:

- allow and set up a virtual network – in networking options of Virtual PC configuration, the Microsoft Loopback adapter has to be enabled in the network adapter. Then the same settings for the IP address as mentioned above (same format of IP address – 192.168. $x$ . $y$  where  $x$  is the same but the  $y$  value must be different from the Host PC configuration, for example 192.168.0.1).

**Configuration** of Moodle and related software is described in the documentation, I will emphasize only few points – to avoid problems it is good to set up :

```
ServerName 192.168.0.1:80 in /apache/conf/httpd.conf
```

and check whether the value of `$CFG->wwwroot` corresponds with the IP address of Virtual PC in the virtual network :

```
$CFG->wwwroot = 'http://192.168.0.1'; in /moodle/config.php
```

#### 2.4.2 Moodle on virtual server

If the virtual network between Host PC and Virtual PC works fine (it can be tested with *ping* command) and if the Moodle server was installed successfully, the Moodle site can



be accessed from the Host PC by typing the IP address of Virtual PC (192.168.0.1 in my example) into the web browser.

So, after successful installation and configuration, we have a working model with the Moodle server on the Virtual PC and a client who is connecting through a virtual network from the Host PC (through HTTP and HTTPS) or from other PCs which can be connected to the Host PC through wireless ad-hoc network.

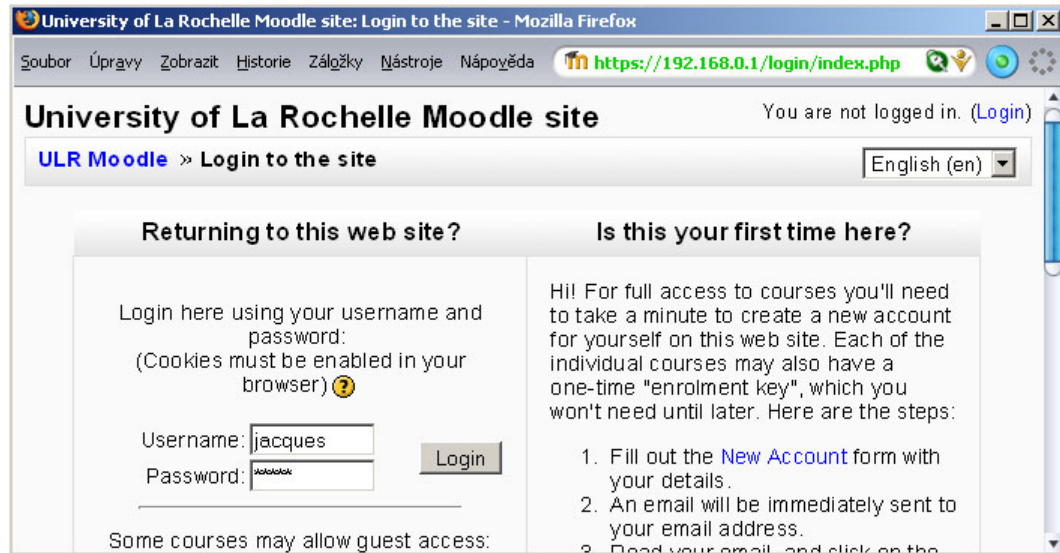


Figure 2.7: Login (HTTPS) from Host PC to Moodle site, server runs on Virtual PC

Now the Moodle functionality can be easily tested and the model corresponds with a “real life” application – network traffic goes through a firewall. The security of the web and SQL server can thus be tested without risk.







### 2.4.3 Moodle in practise

To test Moodle features, I have created a pilot course with a small database of uploaded study materials (LMA modules, see the figure 2.8) and I have done administrative tasks over this course and whole site. I must note that all proclaimed functions and features were working satisfactory.

Administrative (not only) functions can be extended by various plug-ins, so Moodle can suit different needs and requirements.

**First test course with modules** You are logged in as [Jacques Premier](#) ([Logout](#))

[ULR Moodle](#) » [CF101](#) » [Files](#) » [Math](#)

| Name  | Size    | Modified              | Action  |
|---|---------|-----------------------|---|
|  Parent folder  |         |                       |   |
| <input type="checkbox"/>  1stGrade_Math_PRACTICETEST.zip | 218KB   | 27 Oct 2006, 09:35 AM | <a href="#">Unzip</a> <a href="#">List</a> <a href="#">Restore</a> <a href="#">Rename</a> |
| <input type="checkbox"/>  2ndGrade_Math_PRACTICETEST.zip | 264.4KB | 27 Oct 2006, 09:36 AM | <a href="#">Unzip</a> <a href="#">List</a> <a href="#">Restore</a> <a href="#">Rename</a> |
| <input type="checkbox"/>  4thGrade_Math_PRACTICETEST.zip | 205.3KB | 27 Oct 2006, 09:36 AM | <a href="#">Unzip</a> <a href="#">List</a> <a href="#">Restore</a> <a href="#">Rename</a> |
| <input type="checkbox"/>  5thGrade_Math_TEST.zip         | 59.6KB  | 27 Oct 2006, 09:36 AM | <a href="#">Unzip</a> <a href="#">List</a> <a href="#">Restore</a> <a href="#">Rename</a> |
| <input type="checkbox"/>  6thGrade_Science_TEST.zip      | 6KB     | 27 Oct 2006, 09:36 AM | <a href="#">Unzip</a> <a href="#">List</a> <a href="#">Restore</a> <a href="#">Rename</a> |

With chosen files...

Figure 2.8: Uploaded LMA modules – study materials in demonstrative course

## Chapter 3

# Ontology mapping

An ontology defines a common vocabulary for concerned persons who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them [19]. The use of ontologies enables the knowledge sharing and the explicit data description. Some of the reasons for using ontology:

- To share common understanding of the structure of information among people or software agents.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from the operational knowledge.
- To analyze domain knowledge.

**In the MAPLE project,** the motivation for ontology use is to enable an effective data retrieval and management of learning modules. In case of a huge database of learning courses and materials, it is hardly possible to find desired data only using a primitive methods for searching and data description, so I will use an advanced method.

Ontologies allow to establish a structural network which will represent dependencies and bindings between stored materials and enables an exact description of each database item. The mapping algorithm will enable cooperation between stored data described by different ontologies, or with an another database of learning modules using different ontology for data characterization and description.

### 3.1 Introduction to ontologies and ontology mapping

#### 3.1.1 Ontologies

The term “ontology” has its origin in philosophy, where it is the name of one fundamental branch of metaphysics, concerned with analyzing various types or modes of existence, often with special attention to the relations between particulars and universals, between intrinsic and extrinsic properties, and between essence and existence, it is the study of being or existence. It seeks to describe the basic categories and relationships of being or existence to define entities and types of entities within its framework.

Ontology can be said to study conceptions of reality, ontology investigates both the question of what actually exist and the question of the meaning of the word existence. Information in this section are mainly borrowed from [27].

**Ontology** has one basic question: “What is there?” Different philosophers provide different answers to this question. One common approach is to divide the extant entities into groups called “categories”. However, these lists of categories are also quite different from one another. It is in this latter sense that ontology is applied to such fields as theology, information science and artificial intelligence.

**Ontology in computer science and in philosophy** has in common the representation of entities, ideas, and events, along with their properties and relations, according to a system of categories. Differences between the two are largely matters of focus. Philosophers are less concerned with establishing fixed, controlled vocabularies than are researchers in computer science, while computer scientists are less involved in discussions of first principles (such as debating whether there are such things as fixed essences, or whether entities must be ontologically more primary than processes).

### Metaphysics

Metaphysics is very related to ontology, it investigates the most general principles about the nature of reality. The term originates from the work of Aristotle called “Metaphysics”, which was divided into three parts, which are now regarded as the proper branches of traditional Western metaphysics:

- *Ontology* – the study of Being and existence; includes the definition and classification of entities, physical or mental, the nature of their properties, and the nature of change.
- *Theology* – the study of God; involves many topics, including among others the nature of religion and the world, existence of the divine, questions about Creation, and the numerous religious or spiritual issues that concern humankind in general.
- *Universal science* – the study of first principles, which Aristotle believed to be the foundation of all other inquiries. An example of such a principle is the law of non-contradiction and the status it holds in non-paraconsistent logics.

The work of Aristotle related to ontologies was described as “the science of being qua<sup>1</sup> being”. According to this theory, then, ontology is the science of being inasmuch as it is being, or the study of beings insofar as they exist.

The metaphysician also attempts to clarify the notions by which people understand the world, including existence, objecthood, property, space, time, causality, and possibility. More recently, the term “metaphysics” has also been used more to refer to “subjects that are beyond the physical world”.

Ontology concerns determining what categories of being are fundamental and asks whether, and in what sense, the items in those categories can be said to “be”.

---

<sup>1</sup>The word “qua” means “in the capacity of”.

### Ontologies in informatics

In the context of computer science, an ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts – “a description of the concepts and relationships that can exist for an agent or a community of agents”. It is used to reason about the objects within that domain. Ontology is generally written, “as a set of definitions of formal vocabulary” [27, 2].

Ontology consists of (see the figure 3.1):

- **Individuals** – the basic objects.
- **Classes** (Concepts) – sets, collections or types of objects.
- **Attributes** – properties, features, characteristics or parameters that objects can have and share.
- **Relations** (Relationships) – objects can be related mutually.

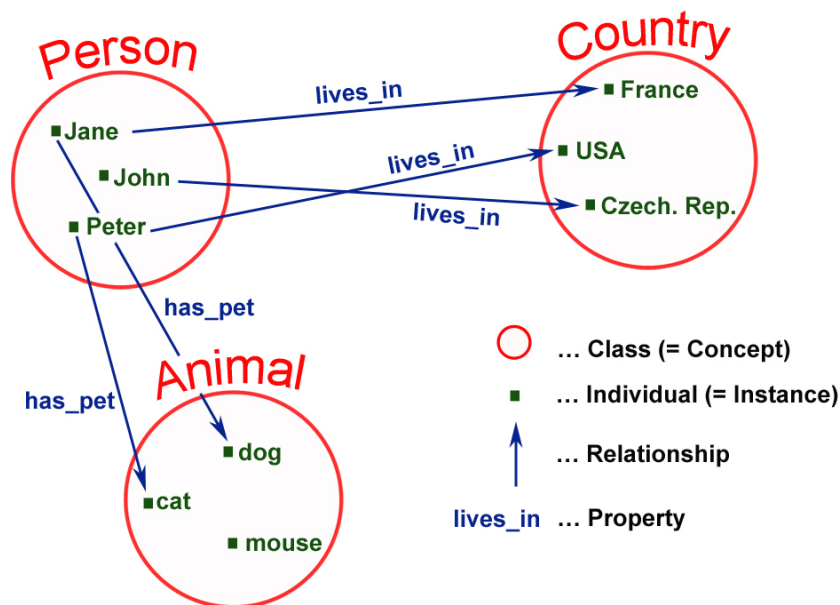


Figure 3.1: Ontology schema

**Individuals** or also “instances”, are the basic elements (objects) of ontologies. Individuals may represent concrete objects (e.g. people, animals etc. – depends on the ontology domain) or also abstract items (numbers, words). There are not any restrictions that the ontology has to include some individuals, but one of the main purpose of ontology use is to classify individuals even if those individuals are not explicitly a part of the ontology. Individuals are members of classes (individual can be a member of multiple classes).

**Classes (concepts)** are abstract groups, sets, or collections of objects. The membership of a class is dependent on its logical description, not on its name. Classes may contain individuals, other classes, or a combination of both. A class can *subsume* or be *subsumed* by other classes – if a class contains other classes, these can be denoted like *sub-classes* or *children*, a superordinate class should be denoted like a *super class* or a *parent*.

For example, Vehicle subsumes Car, since (necessarily) anything that is a member of the latter class is a member of the former. The *subsumption* relation is used to create a hierarchy of classes, typically with a maximally general class like *Thing* at the top, and very specific classes at the bottom (see the figure 3.2).

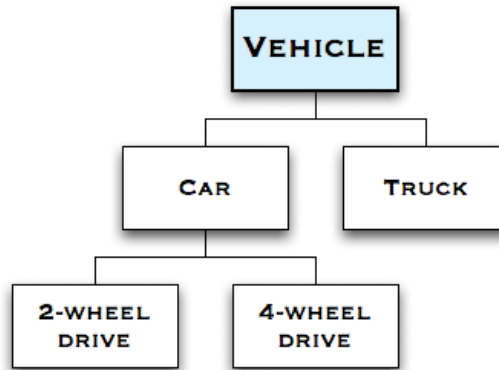


Figure 3.2: Hierarchy of classes in the partial ontology diagram [27]

A *partition* is a set of related classes and associated rules that allow objects to be placed into the appropriate class. In the scope of the figure 3.2 there is a partition of the Car class into the classes 2-Wheel Drive and 4-Wheel Drive. The partition rule determines in which class the particular car is placed.

If the partition rule(s) guarantee that a single Car object cannot be in both classes, then the partition is called a *Disjoint Partition*. If the partition rules ensure that every concrete object in the super-class is an instance of at least one of the partition classes, then the partition is called an *Exhaustive Partition*.

**Attributes** – individuals in the ontology can be described by assigned attributes. Each attribute has a name and a value (furthermore can be specified by e.g. *datatype*, *maximal/minimal value*, ...) and is used to store specific information which is attached to the individual. Each attribute has at least a name and a value.

An ontology with concepts which does not have defined attributes is not considered to be the true ontology, is described as a *taxonomy* (if hyponym relationships exist between concepts) or a *controlled vocabulary*.

**Relations** represents relationships between objects in the ontology. Typically the relation is realized by an attribute, whose value is an another (related) object in the ontology. Much of the power of ontologies comes from the ability to describe these relations. Together, the set of relations describes the semantics of the domain.

The most important type of relations is the *subsumption* relation which defines membership of classes (subclass, superclass, ...).

Another type of relation is the *meronymy* relation (means *part-of*) which describes the combination of objects, how are combined to form composite objects.

### 3.1.2 Ontology mapping

*Ontology Mapping* is the process whereby two ontologies are semantically related at a conceptual level, and the source ontology instances are transformed into the target ontology entities according to those semantic relations[20].

Thus, ontology mapping allows cooperation between different ontologies which are classifying (describing) objects in the same (similar) domain. The aim is to find corresponding concepts and objects between the source ontology and the second one, which is needed to be mapped, if it is possible. See the example on the figure 3.4.

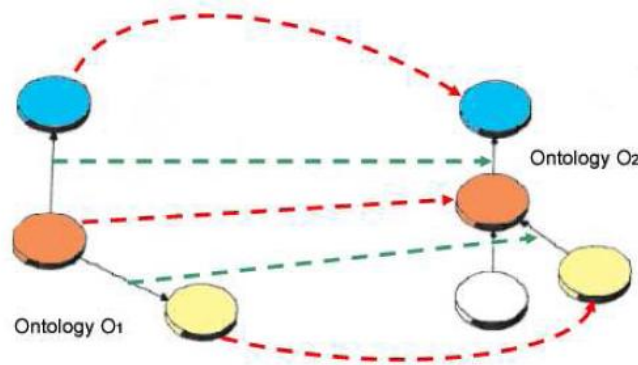


Figure 3.3: Ontology mapping example no.1

#### Algebraic definition of ontology and ontology mapping [8]

represents ontologies as logical theories. An ontology is a pair  $O = (S, A)$  where  $S$  is the (*ontological*) *signature* describing the vocabulary and  $A$  is a set of (*ontological*) *axioms* specifying the intended interpretation of the vocabulary in some domain of discourse.

Typically, an ontological signature will be modeled by some mathematical structure, it could consist of a hierarchy of concept or class symbols modeled as a partial ordered set (poset), together with a set of relations symbols whose arguments are defined over the concepts of the concept hierarchy. The relations themselves might also be structured into a poset.

*Ontological signature morphisms.* We understand ontology mapping as the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected. Structure-preserving mappings between mathematical structures are called morphisms; for instance, a function  $f$  between two posets that preserves the partial order ( $a \leq b$  implies  $f(a) \leq f(b)$ ) is a morphism of posets. Hence, we shall characterise ontology mappings as morphisms of ontological signatures as follows.

A *total ontology mapping* from  $O_1 = (S_1, A_1)$  to  $O_2 = (S_2, A_2)$  is a morphism  $f : S_1 \rightarrow S_2$  of ontological signatures, such that,  $A_2 \vdash f(A_1)$ , i.e., all interpretations that satisfy

$O_2$ 's axioms also satisfy  $O_1$ 's translated axioms. This makes an ontology mapping a *theory morphism* as it is usually defined in the field of algebraic specification.

In order to accommodate a weaker notion of ontology mapping, we will say that there is a *partial ontology mapping* from  $O_1 = (S_1, A_1)$  to  $O_2 = (S_2, A_2)$  if there exists a sub-ontology  $O'_1 = (S'_1, A'_1)$  ( $S'_1 \subseteq S_1$  and  $A'_1 \subseteq A_1$ ) such that there is a total mapping from  $O'_1$  to  $O_2$ .

### Types of ontology mapping

There are many various methods and algorithms for ontology mapping, this subject area is very wide and many engaged groups have developed sophisticated techniques.

But also, as written in [8] the interested practitioner in ontology mapping, is often faced with a knotty problem: there is an enormous amount of diverse work originating from different communities who claim some sort of relevance to ontology mapping. For example, terms and works encountered in the literature which claimed to be relevant, include: *alignment*, *merging*, *articulation*, *fusion*, *integration*, *morphism*, and so on. Given this diversity, it is difficult to identify problem areas and comprehend solutions provided.

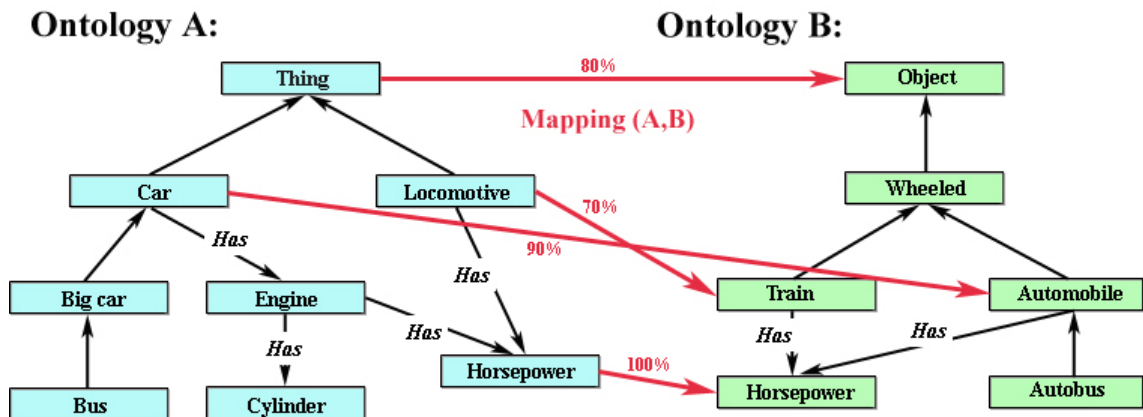


Figure 3.4: Ontology mapping example

## 3.2 Working with ontologies

Working with ontologies is a particularly topical problem, so not only a considerable quantity of various mapping methods (as mentioned above), but also lots of related software is available. My interests were concentrated on an open source tools because advantages resulting from this license were important for my following work.

This section contains an overview of available tools, frameworks and methods concerning the ontology mapping problem and ontology problematics. The aim is to put my work into the context of existing tools and to discuss the possibility of cooperation and utilization, eventually to find proper tool/framework/method suitable for my purposes.



**Open source software** for an ontology development, management and visualization is easily accessible from the World Wide Web. Each product has advantages and also disadvantages, viz. the comparative study [3].

| Name              | Type&description          | Author         | Updated | Lang. |
|-------------------|---------------------------|----------------|---------|-------|
| Protégé           | ontology editor&framework | Stanford Univ. | 2006/11 | Java  |
| Eclipse           | open development platform | Eclipse team   | 2006/11 | Java  |
| - IBM IODT        | Eclipse plug-in           |                |         | Java  |
| - Ontology Editor | Eclipse Ontology Editor   | S. Deshpande   | 2003/05 | Java  |
| - CEV             | CoBrA Eclipse Viewer      | Harry Chen     | 2004/07 | Java  |
| Ontolingua        | www app www environment   | Stanford Univ. | 2003/07 | Java  |
| WebOnto           | applet online editor      | John Domingue  | 1999/01 | Java  |
| Ontosaurus        | www app Loom web browser  | Ramesh Patil   | 2005/09 | Java  |

Table 3.1: Software for ontology development, management and visualization

### 3.2.1 RDF

The **Resource Description Framework (RDF)** is a W3C specification of general-purpose language for representing information in the Web. Originally designed as a metadata model, bur has come to be used as a general method of modeling information, through a variety of syntax formats.

**RDF Schema (RDF-S)** is a standard which describes how to use RDF to describe RDF vocabularies on the Web (example of use: – major component in W3C’s Semantic Web activity, – DAML Ontology Library which organizes hundreds of ontologies in a variety of different ways (keyword, organization, submission date, etc)).

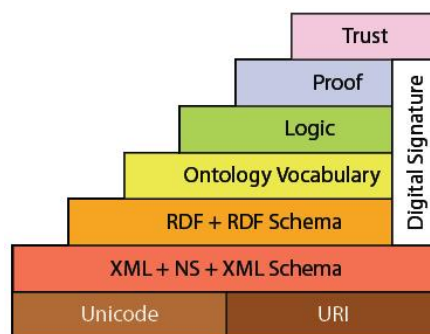


Figure 3.5: Layer cake

**The RDF metadata model** is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, called triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, one

way to represent the notion “The sky has the color blue” in RDF is as a triple of specially formatted strings: a subject denoting “the sky”, a predicate denoting “has the color”, and an object denoting “blue”.

**RDF and ontologies:** A collection of RDF statements intrinsically represents a labeled, directed pseudo-graph. As such, an RDF-based data model is more naturally suited to certain kinds of knowledge representation than the relational model and other ontological models traditionally used in computing today. However, in practice, RDF data is often stored in relational database representations sometimes also called triple stores. As RDFS and OWL demonstrate, additional ontology languages can be built upon RDF.

### 3.2.2 OWL

The Web Ontology Language (OWL) is a language for defining and instantiating Web ontologies. An OWL ontology may include descriptions of classes, along with their related properties and instances. OWL is designed for use by an applications that needs to process the content of information instead of presenting information to humans. It facilitates greater machine interpretability of the Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics (= OWL semantically extends RDF-S). OWL is based on earlier languages OIL and DAML+OIL, and is now a W3C recommendation (i.e., standard) [27].

**The OWL language** provides three increasingly expressive sub languages designed for a use by specific communities of implementers and users.

**OWL Lite** supports those users primarily needing a classification hierarchy and simple constraint features. For example, while OWL Lite supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and provide a quick migration path for thesauri and other taxonomies.

**OWL DL** supports those users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. OWL DL includes all OWL language constructs with restrictions such as type separation (a class can not also be an individual or property, a property can not also be an individual or class). OWL DL is so named due to its correspondence with description logics [Description Logics], a field of research that has studied a particular decidable fragment of first order logic. OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for reasoning systems.

**OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the predefined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature of OWL Full.

### 3.2.3 Frameworks

Frameworks are mostly a combination of tools, provide a methodological approach to mapping, and some of them are also based on theoretical work [8]. Undermentioned brief description will acquaint readers with available frameworks.

**MAFRA** framework for distributed ontologies in the Semantic Web is part of a multi-ontology system, and it aims to automatically detect similarities of entities contained in two different department ontologies.

MAFRA framework also defines *semantic bridge*, it is a module that establishes correspondences between entities from the source and target ontology based on similarities found between them. All the information regarding the mapping process is accumulated, and populate an ontology of mapping constructs, the *Semantic Bridge Ontology* (SBO).

**OntoMapO** framework for accessing and integrating upper level ontologies, provides also a service that allows a user to import linguistic ontologies onto a Web server, which will then be mapped onto other ontologies.

Apart from the OntoMapO primitives and design style, also a set of primitives that OntoMapO offers for mapping was elaborated. Two sets are defined – *InterOntologyRel* and *IntraOntologyRel*, each of them has a number of relations that aim to capture the correspondence of concepts originating from different ontologies (i.e., equivalent, more-specific, meta-concept), a typology of these relations is given in the form of a hierarchy.

**IFF** framework for ontological structures to support ontology sharing, it is based on the Barwise-Seligman theory of information flow [1].

The author argues that IFF represents the dynamism and stability of knowledge. The former refers to instance collections, their classification relations, and links between ontologies specified by ontological extension and synonymy (type equivalence); it is formalized with Barwise-Seligman’s *local logics* and their structure-preserving transformations-logic infomorphisms. Stability refers to concept/relation symbols and to constraints specified within ontologies; it is formalized with Barwise-Seligman’s *regular theories* and their structure-preserving transformations-theory interpretations. IFF represents ontologies as logics; and ontology sharing as a specifiable ontology extension hierarchy.

generic ontologies are also consensual agreements but across communities.

**Madhavan and colleagues** framework and language for ontology mapping enables mapping between models in different representation languages without first translating the models into a common language.

The framework uses a *helper model* when it is not possible to map directly between a pair of models, and it also enables representing mappings that are either incomplete or involve loose information. The models represented in their framework are representations of a domain in a formal language, and the mapping between models consists of a set of relationships between expressions over the given models.

Also a typology of mapping properties is defined: query answerability, mapping inference, and mapping composition.

**Fernández-Breis and Marínez-Béjar’s** cooperative framework for ontology integration, they present a system that could serve as a framework for cooperatively built, integration-derived (i.e., global) ontologies.

This system is aimed towards ontology integration and is intended for use by normal and expert users. The former are seeking information and provide specific information with regard to their concepts, whereas the latter are “integration-derived” ontology constructors. As the normal users enter information regarding the concepts’ attributes, taxonomic relation, and associated terms in the the system, the expert users process this information, and the system helps them to derive the integrated ontology.

### 3.2.4 Methods and tools

This section contains overview of tools, either stand-alone or embedded in ontology development environments, and methods used in ontology mapping.

**FCA-Merge** method for ontology merging is based on Ganter and Wille’s work on Formal Concept Analysis ([6]) and lattice exploration.

Natural language techniques are incorporated in FCA-Merge, to derive a lattice of concepts which is then explored manually by a knowledge engineer who builds the merged ontology with semi-automatic guidance from FCA-Merge.

**IF-Map** automatic method for ontology mapping, based on Barwise-Seligman theory of information flow [1].

The method provides a systematic and mechanized way for deploying it on a distributed environment to perform ontology mapping among a variety of different ontologies. The process consists four major steps: ontology harvesting, translation, infomorphism generation, and display of results.

**Ontology harvesting** (acquisition)  $\Rightarrow$  **Translation**  $\Rightarrow$  **IF-Map** (infomorphism generation)  $\Rightarrow$  **Display results** (project mappings)

**SMART, PROMPT and PROMPTDIFF** tools for the Protégé ontology development environment, using linguistic similarity matches between concepts for initiating the merging or alignment process. Then use the underlying ontological structures of the Protégé-2000 environment (classes, slots, facets) to inform a set of heuristics for identifying further matches between the ontologies.

The notions of merging and alignment are distinguished, where merging is the creation of a single coherent ontology and alignment is establishing links between ontologies and allowing the aligned ontologies to reuse information from one another.

The **SMART** tool is an algorithm that goes beyond class name matches and looks for linguistically similar class names, studies the structure of relations in the vicinity of recently merged concepts, and matches slot names and slot value types.

**PROMPT** is a (semi-)automatic tool and provides guidance for the engineer throughout the steps performed during merging or alignment, where an automatic decision is not possible, the algorithm guides the user to the places in the ontology where his intervention is necessary, suggests possible actions, and determines the conflicts in the ontology and proposes solutions for these conflicts.

**PROMPTDIFF** is an algorithm which integrates different heuristic matchers for comparing ontology versions. These matchers are combined in a fixed-point manner, using the results of one matcher as input for others until the matcher produces no more changes. PROMPTDIFF addresses structure-based comparison of ontologies as its comparisons are based on the ontology structure and not their text serialization.

**GLUE** system employs machine learning techniques to find mappings. Given two ontologies, for each concept in one ontology, GLUE finds the most similar concept in the other ontology using probabilistic definitions of several practical similarity measures. GLUE also uses multiple learning strategies, each of which exploits a different type of information either in the data instances or in the taxonomic structure of the ontologies.

GLUE uses a multi-learning strategy, it can exploit the frequencies of words in the text value of instances, the instance names, the value formats, or the characteristics of value distributions. Two types of learners were developed, a content learner which uses a text classification method called Naive Bayes learning and a name learner which uses the full name of the instance instead of hits content.

**CAIMAN** system uses machine-learning for ontology mapping, elaborated on a scenario where members of a community would like to keep their own perspective on a community repository, each member in a community of interest organizes her documents according to her own categorization scheme (ontology). This is extended by the use of a user's bookmark folder as a "personal" ontology. The mapping task is then to align this ontology with the directory structure of *CiteSeer*<sup>2</sup> (also known as *ResearchIndex*).

**ONION** (ONtology compositIOn) system for resolving heterogeneity in ontologies, which provides an articulation generator for resolving heterogeneity in different ontologies. The semantic heterogeneity can be resolved by using articulation rules which express the relationship between two (or more) concepts belonging to the ontologies, but these relationships are limited to subclass of, part of, attribute of, instance of, and value of.

**ConceptTool** adopts a description logic approach to formalize a class-centered, enhanced entity relationship model. ConceptTool is an interactive analysis tool that guides the analyst in aligning two ontologies, represented as enhanced entity-relationship models augmented with a description logic reasoner. Linguistic and heuristic inferences to compare attributes of concepts in both models are also used, and the analyst is prompted with relevant information to resolve conflicts between overlapping concepts.

### 3.2.5 Protégé

Protégé is a free, open source Java-based ontology editor and knowledge-base framework, is flexible and modular. Protégé implements a rich set of knowledge-modeling structures and actions that support creation, visualization, and manipulation of ontologies in various formats. Protégé can be extended by way of a plug-in architecture and a Java-based API for building knowledge-based tools and applications – easy use of plug-ins (87 existing plug-ins [2006-11-07]) [23].

---

<sup>2</sup>Accessible at [citeseer.nj.nec.com](http://citeseer.nj.nec.com).

Ontologies can be exported into different formats (RDF(S), OWL, XML). Supported by the strong community of developers, academics and users (60 000). Protégé enables two different ways of ontology modeling, each with it's own user interface and editor:

1. **Protégé-Frames** editor enables to build and populate frame-based ontologies – in accordance with the OKBC protocol (Open Knowledge Base Connectivity). The frame-based ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes – individual exemplars of the concepts that hold specific values for their properties.
2. **Protégé-OWL** editor enables to build OWL (see the section 3.3) ontologies for the Semantic Web. “An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms [21]”.

### Potentially helpful plug-ins

List of potentially helpful plug-ins which can facilitate my endeavor:

| Name               | Type    | Description   | Author        | Updated | Lang. |
|--------------------|---------|---------------|---------------|---------|-------|
| Prompt             | plug-in |               | Natasha Noy   | 2006/8  | Java  |
| -Jambalaya         | plug-in |               |               |         | Java  |
| -FOAM              | plug-in |               | F. dos Santos | 2006/07 | Java  |
| -FOAM framework    | frame   |               | Marc Ehrig    | 2005/12 | Java  |
| Search API         | plug-in | API search    | Eric Xu       | 2006/09 | Java  |
| *String Search Tab | plug-in | string search | K. Ahsan      | 2005/6  | Java  |
| Protege Server     | plug-in | CORBA server  | M. Chisholm   | 2006/11 | Java  |
| ProtegeWebBrowser  | plug-in | web browser   | Kamran Ahsan  | 2006/4  | Java  |
| XML Backend        | plug-in | ontology2XML  | Eric Xu       | 2004/12 | Java  |
| XMLTab             | plug-in | XML in/out    | M. Sintek     | 2005/03 | Java  |
| protegeDocgen      | plug-in | documentation | B. Gregoire   | 2005/6  | Java  |
| Jambalaya          | plug-in | visualization | CHISEL        | 2006/11 | Java  |
| OWLviz             | plug-in | visualization | M. Horridge   | 2005/03 | Java  |
| PromptViz          | plug-in | visualization | D. Perrin     | 2005/4  | Java  |
| TGVizTab           | plug-in | visualization | H. Alani      | 2005/11 | Java  |

Table 3.2: Potentially helpful plug-ins of Protégé

### Prompt plug-in

allows to manage multiple ontologies in Protégé and also:

- **Compare** ontology's versions, **Map** one ontology to another
- **Move** frames between included and including project, **Merge** two ontologies into one

- **Extract** a part of an ontology, **Mapping mode** creates and executes mappings
- **Plug-in architecture** supports plug-in framework (like Protégé), you can add in your own merge/mapping algorithms or user interface components. Prompt plug-ins installed by default:

**Jambalaya** – style visualizations for mappings

**FOAM** – algorithm for mapping

**Synonyms** – enables use of synonym lexical mappings

## FOAM

FOAM is the **F**ramework for **O**ntology **A**lignment and **M**apping tool, which allows fully or semi-automatical alignment of two or more ontologies. The FOAM algorithm is based on heuristics (similarity) of the individual entities – concepts, relations, and instances.

### 3.2.6 Summary

Comparison is difficult cause each tool and application has different qualities and advantages and better suits for various purposes. The problem was that in this point of development I could not exactly determined which attributes will be the most important in my following quest. Due to these circumstances, I have designated following priorities:

- Availability of documentation and add-ons
- Difficulty of learning
- Modularity – plug-in architecture
- Network support
- Platform independent
- Up-to-date
- User-friendly interface
- Wide users community

Regarding these requirements, I have decided to use an open source ontology editor and a framework – *Protégé* (and *Eclipse IDE* for plug-in development).

Thus a logic question appears: why I want to contribute to the “ocean” of mapping methods with my “drop”, rather than to use an existing one?

As mentioned above, the motivation for ontology use in the MAPLE project is clear, my own realization of mapping algorithm should be the best way how to enhance the whole project, to enable knowledge sharing and cooperation with different ontologies, because I can flexibly react on project’s requirements and modify the code of my program easily to suit and satisfy the needs. Not only from my own experiences I can say that modification of somebody else’s wide code should cause several problems.

Another main reason was to deeper understand the issue and to gain experiences with development of my own application, to work with ontologies on a lower level and to push my thesis from the theory into the practice.

### 3.3 Protégé & OWL

As mentioned in the section 3.2.5, Protégé supports two different types of ontologies – Frame-based and OWL ontologies. Web Ontology Language (OWL) is convenient for my purpose, thus part of the following text will devote to OWL. The primary source of information for this section were the official Protégé site – *Protégé-owl api programmer's guide* [9] and W3C's *OWL Web Ontology Language Guide* [21], partially also *Wikipedia, the free encyclopedia*.

#### 3.3.1 Protégé-OWL API

Protégé-OWL API is an open-source Java library for the OWL and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic engines. Furthermore, the API is optimized for the implementation of graphical user interfaces.

The API is designed to be used in two contexts:

- For the development of components that are executed inside of the Protégé-OWL editor's user interface
- For the development of stand-alone applications (e.g., Swing applications, Servlets, or Eclipse plug-ins)

**Protégé-OWL API** extends the Protégé core API to provide access to OWL ontologies. The Protégé API can be used directly by external applications to access Protégé knowledge bases and make use of Protégé forms without running the Protégé application.

**Protégé** is a flexible, configurable platform for the development of arbitrary model-driven applications and components. Protégé has an open architecture that allows programmers to integrate plug-ins, which can appear as separate tabs, specific user interface components (widgets), or perform any other task on the current model. The Protégé-OWL editor provides many editing and browsing facilities for OWL models, and therefore can serve as an attractive starting point for a rapid application development. Components can be wrapped initially into a Protégé tab widget and later extracted, to be distributed as part of a stand-alone application.

#### Working with OWL Models

The Protégé-OWL API is centered around the collection of Java interfaces from the model package. These interfaces provide access to the OWL model and its elements like classes, properties, and individuals. The implementation of these interfaces should not be accessed directly, but only operated on the interfaces. Thanks to these interfaces, internal details of how Protégé stores ontologies are hidden, so these internals should not be considered. Everything is abstracted into interfaces and a new code should not make any assumptions about the specific implementation.

#### Working with Jena Models

Protégé-OWL uses the popular Jena API (<http://jena.sourceforge.net/>) for various tasks, in particular for parsing of OWL/RDF files. Furthermore, the Protégé-OWL API can be used



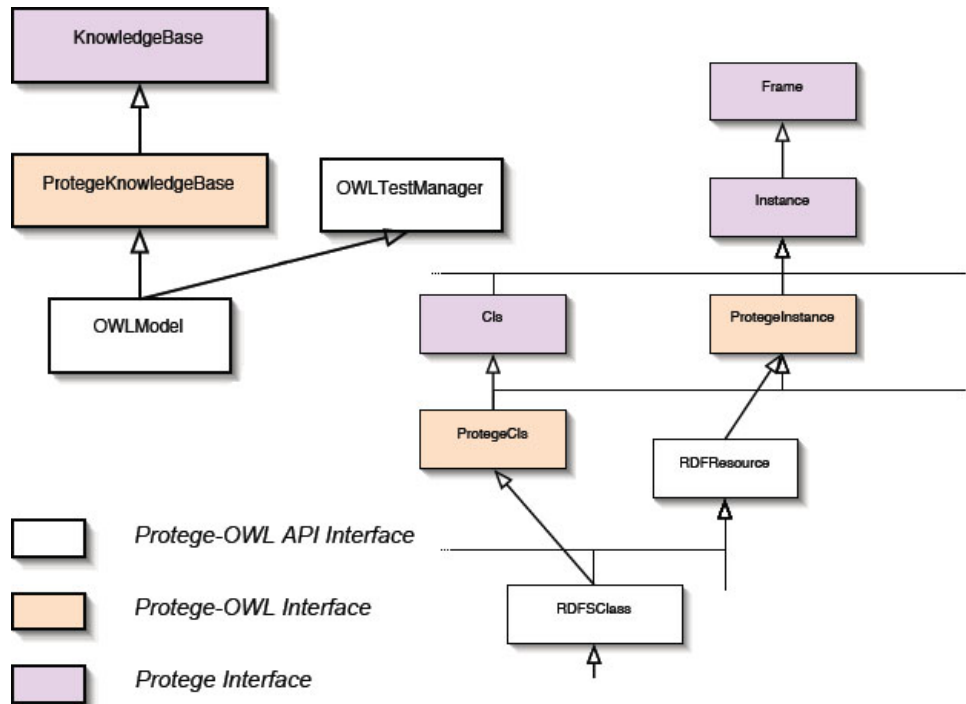


Figure 3.6: Part of Protégé API diagram [7]

to generate a Jena *OntModel* at any time. Both *JenaOWLModel* and *OWLDatabaseModel* implement the *OntModelProvider* interface, which has a *getOntModel()* method to create an *OntModel* from the current ontology.

### 3.3.2 Protégé Plug-in Development

- T - Tab widget plug-ins** (Core Protégé feature) appears as one of the main tabs on the screen, in order to activate this plug-in in the user interface, a user has to select the Project||Configure... menu item.
- S - Slot widget plug-ins** (Core Protégé feature) can display and edit a property value on a form. You can create your own slot widgets and add them to the forms using the Forms tab.
- P - Project plug-ins** (Core Protégé feature) allow programmers to execute arbitrary code when a project is created, loaded, or closed, they can be used to add menus or toolbar buttons or also to attach arbitrary listeners to a knowledge base (agents).
- F - Resource action plug-ins** can appear in the right-click menu of a selected class, property, or individual, or in the lower left corner of a form.
- I - Resource display plug-ins** can be used to add arbitrary components to the lower right corner of each form.
- O - Ontology test plug-ins** are plug-ins that will be executed when the user presses the test ontology buttons, the tests can return a test result object, which is then used to

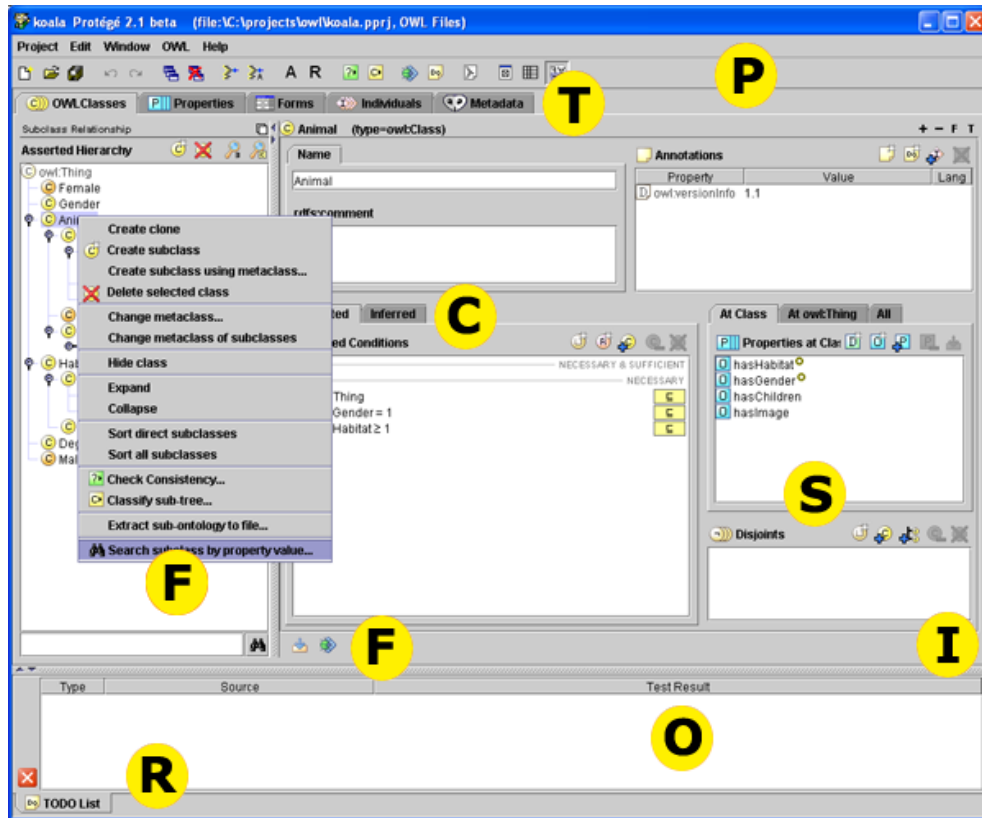


Figure 3.7: Protégé Plug-in Development [9]

display results to the user.

**R - Result panel plug-ins** are arbitrary components that can be displayed as a tab at the bottom of the screen, they can be added or removed from result panels as a result of some action.

**C - Conditions widget extension plug-ins** can be installed by a project plug-in to add additional tabs to the conditions widget [9].

### 3.3.3 Summary

My initial intention was to implement a mapping plug-in into *Prompt* (see the section 3.2.5) plug-in architecture – because of the similar functionality (management of multiple ontologies, ontology mapping). Based on the figure 3.7, *Prompt* can be classified like a “*tab widget plug-in*”, so my intention was to create the plug-in bounded on another tab widget plug-in.

But after the careful consideration I have changed my opinion, not to create a dependent tool on another plug-in bounded with its framework, I have decided to create a Core Protégé tab widget plug-in bounded directly with Protégé and Protégé (OWL) API, so I can take the advantage of all Protégé features, but also (as written in the section 3.3.1) I can easily extract my component and create a stand-alone application.

Tab widget plug-in was as well recommended by authors of Protégé as a good way of starting with Protégé plug-in development.

### 3.4 Ontology mapping algorithm

The ground of the algorithm is from Kong Choi Yu’s diploma thesis <sup>3</sup> [10], on my supervisor’s ( Dr. Michel Eboueya) recommendation I continued in his work. I tried to contact Kong Choi Yu, because of some inconsistencies and also for the demand of authorization, but I was not successful.

I took his proposed design, I have corrected errors and I have proposed on my opinion better solutions in partial comparisons and computations. Unfortunately, thanks to wrong references between formulas and missing facts in the Kong Choi Yu’s diploma thesis, the process of understanding and realization was extensively time-consuming.

| <b>Notation</b> | <b>Meaning</b>  |
|-----------------|---|
| $O_1$           | The source ontology which is primary loaded in the Protégé project. |
| $O_2$           | The second ontology which we’re trying to map.                      |
| $U_n$           | The set of instances of ontology $O_n$ .                            |
| $I_n$           | An instance of ontology $O_n$ .                                     |
| $C_n$           | A concept presents in ontology $O_n$ .                              |
| $w_i$           | Weight.   |
| $Sim(A, B)$     | The similarity between element A and element B.                     |

Table 3.3: Notations

#### 3.4.1 Proposed design

As mentioned above, the proposed mapping algorithm is taken from Kong Choi Yu’s diploma thesis, the initial impulse was the strong recommendation of my French supervisor, so I have analyzed this algorithm and after necessary modifications I have implemented the proposed design.

I want to distinguish the description of the algorithm from work of Kong Choi Yu and my changes and commentaries, so in this “Proposed design” section all my comments are written in *italics*. Note that I made changes in references between equations and in some formulas which are not mentioned, because I wanted to make the text more transparent without doubled sections (his version with mistake + corrected version).

#### Mapping process overview

Three procedures have been designed for the mapping process: pre-fetching; similarity calculation and recording results.

Pre-fetching phase is designed to minimize the number of concepts which will be mapped between two ontologies (so the online ontology mapping can be efficient), especially important for large ontologies.

<sup>3</sup>The original document I got was a study *Ontology Mapping in Pervasive Computing Environment* written by C. Y. Kong, C. L. Wang and F. C. M. Lau [11], the diploma thesis I refer above is extension of this study (which I found on the web) and which enlarges the theory of mapping algorithm, even though there are some inconsistencies.

Pre-fetching estimates the similarity between two concepts by comparing their names (concept name is unique key in OWL ontology) and filters concepts which are highly possibly unrelated to save mapping efforts. The philosophy is that concepts with totally different meanings in their names are unlikely to refer to the same thing.

*This idea is right, the problem is that the semantical comparison was not in the scope of Kong Choi Yu's work, thus the similarities between concept names are calculated only on the lexical level, the semantic meaning is not considered. This limitation is a big weakness, because the mapping process is done only with concepts which are not filtrated in the pre-fetch phase, it means that concepts with completely different names are excluded from following computations even though they can be possible candidates – results of semantical and lexical comparison are completely different.*

A concept definition in an ontology contains a concept's name/identifier, its properties and relationships with other concepts. The design makes use of them in the similarity calculation process. K-nearest neighbour is used to calculate the semantic distance between concepts  $C_1$  and  $C_2$  by converting relationships in  $C_1$  into vector space. Each relationship in  $C_1$  is a dimension in the vector space and  $C_1$  is located at the coordinate  $(1, 1, \dots, 1)$  where there are  $n$  "1"s if there are  $n$  relationship dimensions. Recall that the dynamic nature, the large number of dimensions and scaling for each dimension are the problems of using k-nearest neighbour in pervasive computing environment. This have been overcome by limiting the number of dimensions and having a dimension scale from 0 to 1. The number of dimensions are fixed to be the number of relationships in concept  $C_1$ . K-nearest neighbour locates  $C_2$  in the vector lattice by calculating the similarity between each dimension and each relationship in  $C_2$ . The similarity formula uses the names of the relationships and their related concepts for calculations and outputs a value between 0 and 1 which limits the scale of the dimension. The smaller the distance between the concepts  $C_1$  and concept  $C_2$  in the vector space, the larger their relationships similarity is.

There are different types of properties: constraints (or restrictions) and datatype properties. Constraints are the restrictions on the concepts such as the datatype range of a property or the maximum appearance of a concept in an instance. For example, a concept "person" should have maximum one "gender" property. Datatype properties describe the property types of the data content. For instance, string type or integer type. In traditional ontology mapping tools, all data contents are converted into strings and the similarity between the properties equal to the similarity between the strings. String "123456" is similar to string "12345". On the other hand, there is big difference between the values 123456 and 12345. The design, therefore, recognizes the datatypes of the properties and handles them differently. For string datatype properties, the Naïve Bayes rule is used to calculate the similarities. For integer datatype properties, mathematical calculations such as addition and subtractions are used. The similarity formula is automatically chosen by the proposed mechanism at execution time based on the datatype of the properties.

*These computations on the theoretical level make sense, but when I have tried to put this into practice, using proposed formulas, I ran up against problems. Iterative comparison of all involved values and parameters is extremely resource-demanding (Naïve Bayes rule for each string value), proposed formula for numeric value comparison gives absolutely nonsensical results and finally all computed values are multiplied by small weight variable, so the whole effect of this demanding computations on the concept similarity result is insignificant. One question appears: why is it important to compare values of properties at all cost, when the important fact during ontology mapping is, whether two concepts from different ontologies represent the corresponding "type of object" not exactly the same value?*

The overall similarity calculation process is based on the Jaccard coefficient. Jaccard coefficient determines the ratio of the overlapping data between two sets of data. Jaccard coefficient is defined as:

$$\frac{P(C_1 \cap C_2)}{P(C_1 \cup C_2)} = \frac{P(C_1, C_2)}{P(C_1, C_2) + P(\sim C_1, C_2) + P(C_1, \sim C_2)}$$

Jaccard coefficient is a commonly used similarity measure which is based on the joint probability distribution. It is adopted in the design because data instances should be taken into consideration during mapping because the research goal is to provide a service and resource instance to the requester. Jaccard coefficient outputs similarity measure between zero and one. A “1” means the two concepts are identical and a “0” means the two concepts are totally different.  $P(C_1, C_2)$ ,  $P(C_1, \sim C_2)$  and  $P(\sim C_1, C_2)$  in the formula are computed based on the definition of the concepts (i.e. the name, relationships and properties of the concept) and their instances.

### Pre-fetch phase

To increase the efficiency of mapping a concept  $C_1$  in  $O_1$  to a concept  $C_2$  in  $O_2$ , the highly unrelated concepts are filtered out by examining the names of  $C_1$  and  $C_2$  and generate a set of possible candidates of  $C_n$  from  $O_2$ . Many mechanisms are proposed to compare similarity between two strings, for example, longest common substring, longest common subsequence and hamming distance. These mechanisms are based on the syntactic meaning (i.e. the spellings) of the two strings. But for example, concept “SendTo” and concept “Recipient” are not similar as their syntactic meanings are greatly difference. However, “SendTo” and “Recipient” have similar meanings in English that reflects that semantic meanings of the concept names are more important than their syntactic meanings.

The term *Semantic Distance* is introduced as the smallest number of intermediate concepts to connect the meanings of two concepts. For example, “Send” is related to “Sendee” and the hyponym of “Recipient” is “Sendee”. As a result, “SendTo” and “Recipient” are semantically related and their Semantic Distance ( $SendTo, Recipient$ ) = 1 as there is one intermediate concept “sendee” that connects between “SendTo” and “Recipient”.

To filter un-related concepts, the semantic distance of their concept names, which is smaller than or equal to the semantic distance threshold for each pair of concepts between  $C$  defined in  $O_1$  and  $C'$  defined in  $O_2$  is retrieved.

$$\text{Semantic\_Distance}(C, C') \leq \text{Threshold}$$

For common similarity measures, similarity is ranged from 0 to 1. A “0” means the compared items are totally different and a “1” means they are identical. Our design follows this rule by normalizing each semantic distance that is smaller than the threshold as:

$$\text{Sim}(C_{name}, C'_{name}) = \frac{\text{Threshold} - \text{Semantic\_Distance}(C, C')}{\text{Threshold}}$$

In the first paper [11] presented by Kong Choi Yu the comparison between concept names was calculated using the following formula:

$$\text{Sim}(C_{name}, C'_{name}) = \frac{N(\text{longest substring})}{N(C_{name}) + N(C'_{name})} \quad (3.1)$$

Where the function  $N(C_{name})$  returns length of  $C_{name}$  and function  $N(\text{longest substring})$  denotes, I suppose, the length of the longest common substring (this fact was not mentioned). But even if we compare two similar names, the similarity will be at most 0,5, so I changed the formula to the following form (when we compare similar values the result will be 1):

$$Sim(C_{name}, C'_{name}) = \frac{2 * N(\text{longest common substring})}{N(C_{name}) + N(C'_{name})} \quad (3.2)$$

This name similarity computation is in comparison with semantic distance computation considerably simplified, but the calculation of semantic distance is only proposed and realization will follow (see the section 3.4.2).

For the set of the first  $k$  concepts with the highest similarity degree (i.e. the  $k$ -highest  $Sim(C_{name}, C'_{name})$  denoted by  $S_{k-high}$ , the possible candidates set is formed:

**Possible candidates set**

$$\begin{aligned} &= S_{k-high} \cup \forall C_i \in S_{k-high} \text{ concepts that has relationship with } C_i \\ &\cup \forall C_i \in S_{k-high} \text{ merged concepts of } C_i \text{ with each of its neighbor} \\ &\cup \quad \quad \quad \forall C_i \in S_{k-high} \text{ parent (super class) of } C_i \\ &\cup \quad \quad \quad \forall C_i \in S_{k-high} \text{ children (sub - class) of } C_i \end{aligned} \quad (3.3)$$

In ontology mapping, a concept in  $O_1$  may be split into two concepts. For instance, a concept “name” in ontology  $O_1$  may be split into two concepts, “first name” and “last name” in ontology  $O_2$ . To handle the splitting problem, the proposed mechanism merges concepts with their neighborhood concepts, parent concepts and children concepts. Merging concepts  $C'_1$  and  $C'_2$  of the same ontology is done by merging their concept names, attributes and relationships. To resolve naming conflict of attributes and relationships, attributes and relationships are renamed as  $C'_1.attributename$  and  $C'_2.attributename$  and  $C'_1.relationshipname$  and  $C'_2.relationshipname$  respectively. Duplicated relationships are removed during merging. A relationship between  $C'_1$  and  $C'_2$  is converted as attribute with the name of the relationship as the attribute name.

### Similarity calculation

The overall similarity calculation process is based on the Jaccard coefficient which determines the ratio of the overlapping data between two sets of data.

Jaccard coefficient:

$$\frac{P(C_1 \cap C_2)}{P(C_1 \cup C_2)} = \frac{P(C_1, C_2)}{P(C_1, C_2) + P(\sim C_1, C_2) + P(C_1, \sim C_2)} \quad (3.4)$$

$P(C_1, C_2)$  is defined as the equation 3.5 where  $U_1$  and  $U_2$  are the instance sets of  $O_1$  and  $O_2$  respectively.  $N(U_1^{C_1, C_2})$  is the number of instances of  $O_1$  that contain concept  $C_1$  and also concept  $C_2$ .  $N(U_2^{C_1, C_2})$  is the number of instances of  $O_2$  that contain concept  $C_1$  and concept  $C_2$ ,  $N(U_1)$  and  $N(U_2)$  are the number of instances of  $O_1$  and the number of instances of  $O_2$  respectively.

$$P(C_1, C_2) = \frac{N(U_1^{C_1, C_2}) + N(U_2^{C_1, C_2})}{N(U_1) + N(U_2)} \quad (3.5)$$

The formula  $P(\sim C_1, C_2)$  and  $P(C_1, \sim C_2)$  is not explained, so I have supplemented the work with the equation 3.6 and the equation 3.7.  $N(U_n^{C_1, \sim C_2})$  is the number of instances of  $O_n$  that contain concept  $C_1$  and do not contain concept  $C_2$ ,  $N(U_n^{\sim C_1, C_2})$  is the number of instances of  $O_n$  that do not contain concept  $C_1$  and contain concept  $C_2$ .  $N(U_n^{C_1})$  is the number of instances of  $O_n$  that contain concept  $C_1$ ,  $N(U_n^{\sim C_1})$  represents the number of instances of  $O_n$  that do not contain concept  $C_1$ .

$$P(\sim C_1, C_2) = \frac{N(U_1^{\sim C_1, C_2}) + N(U_2^{\sim C_1, C_2})}{N(U_1) + N(U_2)} \quad (3.6)$$

$$P(C_1, \sim C_2) = \frac{N(U_1^{C_1, \sim C_2}) + N(U_2^{C_1, \sim C_2})}{N(U_1) + N(U_2)} \quad (3.7)$$

These formulas are adopted when computing similarity between concepts  $C_1$  and  $C_2$  so that instances are considered when mappings are performed. To calculate  $P(C_1, C_2)$ , we should have two instance sets  $U_1$  and  $U_2$ . As discussed in the above section, it is difficult to locate the instances of partial user ontologies and to store the instances. As a result, history records to determine the instance sets  $U_1$  and  $U_2$  are used.

The number of concepts appearing in each mapping instance is counted. For example, an ontology  $O_A$  contains concepts  $C_a$ ,  $C_b$ ,  $C_c$  and  $C_d$  and a request instance contains concepts  $C_a$  and  $C_b$ . After mapping, the total number of instances of  $O_A$  and the numbers of instances that contain  $C_a$  and  $C_b$  are incremented by 1 while the numbers of instances that contain  $C_c$  and  $C_d$  remains unchanged.  $N(U_1)$  and  $N(U_2)$ , therefore, are recorded. To get  $N(U_1^{C_1, C_2})$ , it is necessary to estimate the number of instances of  $O_1$  that contain concept  $C_1$  and concept  $C_2$ . The number of instances of that contain concept  $C_1$  in  $U_1$  can be found from the history records. The number of concepts that contain both concept  $C_1$  and concept  $C_2$  in  $U_1$  is estimated by calculating the similarity degree of the properties and relationships between concept  $C_1$  and concept  $C_2$ . If the properties and relationships of the concepts are similar, it is likely that the instance of concept  $C_1$  is an instance of concept  $C_2$ . For numerical property such as memory size, it is important that a mapping is found to another concept whose property contains also a numerical value, weights are added when calculating property similarity.

The proposed mechanism also matches instances content when comparing two concepts. When a concept of the request instance matches a concept of a resource instance or a function instance, it is likely that these concepts are matched. All the present instances of  $O_1$  and  $O_2$  are used to compare with the request instance.

Before looking at the details of our ontology mapping mechanism, the *Property Similarity* and the *Relationship Similarity* are discussed. Property similarity calculates how similar of the properties of the two concepts are and relationship similarity calculates how similar of the relationships of the two concepts are. Property similarity and relationship similarity compares all the properties and relationships exist in concept  $C_1$  and  $C_2$ .

### Property Similarity

There are three elements compose a property in ontology: *name*, *datatype* and *cardinality*. *Name* is the name of the property such as ‘‘colour’’ and ‘‘size’’. *Datatype* is the type of

the content data. For example, property “colour” is string type. “Size” is integer type. *Cardinality* refers to the restrictions of the properties such as the range of the datatype, the maximum value of the datatype and the number of appearances of the property in a concept. For example, the minimum cardinality of property “Size” = 0 means “Size” must be a positive integer. As there are many different types, they are handled separately.

For each pair of property/attribute in  $C_1$  (denoted by  $P_{C_1}$ ) and property/attribute in  $C_2$  (denoted by  $P_{C_2}$ ), compute their property similarity. The similarity of the property names are calculated using their meanings which is similar to computing the concept name similarity.

$$\begin{aligned} \text{property similarity, } ps & & (3.8) \\ &= \frac{\sum \text{frequency of property } i \text{ in } C_2 * \text{Equation}[3.10]}{\sum \text{frequency of property } i \text{ in } C_2} \\ &\text{for } i = 1 \text{ to number of property in } C_2 \end{aligned}$$

$$\begin{aligned} \text{Sim}(P_{C_1}, P_{C_2}) &= w1 * \text{Sim}(P_{C_1}\text{name}, P_{C_2}\text{name}) + & (3.9) \\ &w2 * \text{Sim}(P_{C_1}\text{cardinality}, P_{C_2}\text{cardinality})^4 + \\ &w3 * \text{Sim}(P_{C_1}\text{data type}, P_{C_2}\text{data type}) + \\ &w4 * \text{property instance similarity} \end{aligned}$$

*Property instance similarity* is calculated by counting the number of instances of  $C_2$  whose property has similar content as the corresponding property of the instances of  $C_1$ .

In the first paper [11] presented by Kong Choi Yu the comparison between property instances was calculated only when these values were in the text format, the formula was same like the equation 3.1, so after an extension of the output value (between 0 and 1) I got the following formula:

$$\begin{aligned} &\text{Sim}(\text{instance of } p_1, \text{instance of } p_2) \\ &= \frac{2 * N(\text{longest common substring})}{N(\text{value of instance } p_1) + N(\text{value of instance } p_2)} \end{aligned} \quad (3.10)$$

In the later work [10] the property instance similarity computation was extended also to compare numerical values:

$$\text{Sim}(\text{instance of } p_1, \text{instance of } p_2) = \frac{\text{value of instance } p_1}{\text{value of instance } p_2} \quad (3.11)$$

But this equation has meaningful output only when the value of instance  $p_1$  is smaller (or equal) than the value of instance  $p_2$ . And the function also does not work with zero values.

For text (string datatype) values Kong Choi Yu in his later work proposed the comparison based on Naïve Bayes rule. For similarity computation of the data content of property instances  $p_1$  and  $p_2$ , the content is divided into words as  $\{w_1, w_2, \dots, w_n\}$ . When we assume

<sup>4</sup>In the OWL ontology representation methods connected with cardinalities are now deprecated, so  $\text{Sim}(P_{C_1}\text{cardinality}, P_{C_2}\text{cardinality})$  cannot be realized and is leaved out.



that each world is independent, the Naïve Bayes rule can be written like following equation (where  $P(p_1)$  is the usage frequency of the property  $p_1$  in the data instance set):

$$P(p_1|w_1, w_2, \dots, w_n) = P(p_1) \times \prod_{i=1}^n P(w_i|n) \quad (3.12)$$

The question is whether this complicatedness of computation brings the coveted result, whether the simple string comparison or the computation of semantic distance are not better solutions.

### Relationship Similarity

*Relationship similarity* is calculated using k-nearest neighbours. For each relationship between  $C_1$  and  $C_2$ , similarity between relationship  $R_{C_1}$  and  $R_{C_2}$  is computed as below. Equation(3.14) is used to locate the concept  $C_2$  in the concept lattice of  $C_1$  in dimension  $R_{C_1}$ . Similarity of relationship names are calculated using their meanings which is similar to compute concept name similarity.

$$\begin{aligned} Sim(R_{C_1}, R_{C_2}) = & w1 * Sim(R_{C_1name}, R_{C_2name}) + \\ & w2 * Sim(R_{C_1cardinality}, R_{C_2cardinality}) + \\ & w3 * Sim(R_{C_1data\ type}, R_{C_2data\ type}) + \end{aligned} \quad (3.13)$$

The relationship similarity is calculated as the distance between the origin and the concept  $C_2$  in the concept lattice. The longer the distance from the origin means the closer to the concept  $C_1$ . It is the distance between the origin instead of the distance between the concept  $C_1$  because to make the formula consistent with the similarity calculations with less similar having smaller value and a equal totally identical having value “1”.

$$\begin{aligned} & \text{relationship similarity degree} \\ & = \sqrt{\frac{\sum \text{similarity of relationship } i \text{ in concept } C_1 \text{ and } C_2}{3}} \\ & \text{for } i = 1 \text{ to number of relationships in } C_2 \end{aligned} \quad (3.14)$$

### Calculation of $P(C_1, C_2)$

After knowing the property similarity and relationship similarity, we can use of them to compute the  $P(C_1, C_2)$  defined in the equation (3.5). The following shows the procedures.

1.  $U_1$  is partitioned into two sets. One set contains concept  $C_1$  (denoted as  $U_1^{C_1}$ ) while the other set does not contain concept  $C_1$  (denoted as  $U_1^{\sim C_1}$ ) based on the history records.
2.  $U_2$  is partitioned into two sets. One set contains concept  $C_2$  (denoted as  $U_2^{C_2}$ ) while the other set does not contain concept  $C_2$  (denoted as  $U_2^{\sim C_2}$ ) based on the history record.

3. Estimate the similarity between  $O_1$  and  $O_2$  with the equation 3 where the denominators  $N(O_1)$  and  $N(O_2)$  are the total numbers of concepts in  $O_1$  and  $O_2$  respectively. Total number of similar concepts can be computed at pre-fetching when calculating maximum concept name similarity between each concept in  $O_1$  and in  $O_2$ . If the maximum concept name similarity is larger than the threshold, the total number of similar concepts is incremented.

$$Sim(O_1, O_2) = \frac{\text{total number of similar concepts} \times 2}{N(O_1) + N(O_2)}$$

4.  $N(U_1^{C_1, C_2})$  is found.

$$\begin{aligned} & \text{Number of instance in } U_1 \text{ that contains concept } C_2 \\ = & \text{Property similarity} \times N(U_2^{C_2}) \\ = & \text{Equation(3.9)} \times N(U_2^{C_2}) \end{aligned}$$

5. Similarly, calculate  $N(U_2^{C_1, C_2})$ ,  $N(U_2^{\sim C_1, C_2})$  and  $N(U_2^{C_1, \sim C_2})$  in  $P(C_1, C_2)$ .

$$\begin{aligned} N(U_2^{C_1, C_2}) &= N(U_1^{C_1}) * ps \\ N(U_2^{C_1, C_2}) &= N(U_1^{C_1}) * \text{Equation[3.9]} \\ N(U_2^{C_1, \sim C_2}) &= N(U_1^{C_1}) - N(U_1^{C_1, C_2}) \\ N(U_2^{\sim C_1, C_2})^5 &= N(U_1^{\sim C_1}) - N(U_1^{C_1, C_2}) \end{aligned}$$

6.  $P(C_1, C_2)$ ,  $P(\sim C_1, C_2)$  and  $P(C_1, \sim C_2)$  are computed using equations 3.5, 3.6 and 3.6.
7. The similarity degree using the equation 3.4 is computed. This similarity degree is called **instance similarity degree** as we use instances to calculate.
8. The relationship similarity degree for  $C_1$  and  $C_2$  is computed using the equation 3.14.
9. The similarity between concept  $C_1$  and concept  $C_2$  is finally computed using the below equation.

$$Sim(C_1, C_2) = w_1 * \text{instance similarity degree} + w_2 * \text{relationship similarity}$$

### Comparison between ontologies $O_1$ and $O_2$

The detailed formula for calculating the similarities are introduced. The overall working mechanism is discussed in this section. Below is the methodology to compare two ontologies; *OntologyMapping()* is the mapping function and *NewMapping()* is a procedure call that is invoked when the mapping is performed from scratch.

<sup>5</sup>This formula was presented with wrong reference to another equation, so the result was always 0. I have changed the equation .

|  |
|--|
| <pre> NewMapping(<math>C_i</math>) {   Pre-fetch the candidate concepts for <math>C_i</math>.   For each candidate <math>C_k</math> found,     Computer the Jaccard coefficient <math>\frac{P(C_1 \cap C_2)}{P(C_1 \cup C_2)}</math> for <math>C_i</math> and <math>C_k</math>   If the highest similarity degree &gt; threshold,     Mapping is found.   Else     Mapping is failed. } </pre> |
|--|

Table 3.4: *NewMapping* method

|  |
|--|
| <pre> Ontology Mapping (Ontology <math>O_1</math>, Ontology <math>O_2</math>) {   Search history mapping record.   If <math>O_1</math> and <math>O_2</math> have been mapped,     If <math>O_1</math> and <math>O_2</math> have the same last modified date (i.e. the same       version number) as the history record,        For each concept <math>C_i</math> in the request instance,         If <math>C_i</math> is mapped to a concept in <math>O_2</math> in the record,           Mapping is found.         Else           Invoke NewMapping(<math>C_i</math>).     Else       For each concept <math>C_i</math> in the request instance,         If <math>C_i</math> is mapping to a concept, <math>C_p</math> in <math>O_2</math> in the record,            Compute : <math>\frac{P(C_i \cap C_p)}{P(C_i \cup C_p)}</math> for <math>C_i</math> and <math>C_p</math>            If similarity degree &gt; threshold,             Existing mapping is reused.           Else Invoke NewMapping(<math>C_i</math>).           Else Invoke NewMapping(<math>C_i</math>).     Else       For each concept <math>C_i</math> in the request instance,         Invoke NewMapping(<math>C_i</math>).   Update number of instances and concepts encountered.   For each new mapping found, add in history record:     &lt; concept in <math>O_1</math>, concept in <math>O_2</math>, similarity degree, instance count &gt; } </pre> |
|--|

Table 3.5: *Ontology Mapping* method

### 3.4.2 Proposed improvements

As I wrote in the section 3.4.1, the weakness of this design <sup>6</sup> is in the lexical comparison of concept names in the pre-fetch phase.

If the lexical comparison in the pre-fetch phase is used, the highly possible scenario of the computation is, that the corresponding counterpart from the second ontology will be eliminated in the first step of computation, because the name of that concept can be lexically different, but with the same significance. Than the whole process loses sense, because the mapping cannot be found.

The full-functional semantic comparator is the outstanding problem and the solution for my purpose should be use of an external web service.

#### Semantic mapping extension for “(1 : $n$ mapping)”

For a partially specific case I have proposed the modification which eliminates strong disadvantage of the lexical comparison and allows the algorithm to be more precise, but on the other hand a manual interference during ontology creation or loading is required.

The particular scenario is when we are using one (or only few) source ( $O_1$ ) ontologies which is wanted to be mapped to unbounded  $n$  number of other ontologies – 1 :  $n$  mapping. (For example an university uses one ontology for a subject description and want’s to cooperate with other universities, thus a precise mapping is needed.)

The main idea is that for each concept name in the principal ontology a set of synonyms (more precisely a set of words with the same meaning for the particular case) will be created manually. This step eliminates the necessity of use of the external tool for semantic string comparison, the another advantage is that the “meaning” of each concept can be denoted exactly by the set of words with the same signification, because for each particular case a mechanically created set of synonyms can contain words which does not suit the concrete situation, thus a manual creation should be much more precise.

As the manual creation (of the set of related words) is a big advantage, at the same time it is also a great disadvantage of this design, because it requires a manual intervention of an authorized person.

#### Semantic mapping using *WordNet*

is a semantic lexicon for the English language. It groups English words into sets of synonyms called *synsets*, provides short, general definitions, and records the various semantic relations between these synonym sets. The purpose is twofold: to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications. The database and software tools have been released under a BSD style license and can be downloaded and used freely. The database can also be browsed online.

The hypernym/hyponym relationships among the noun synsets can be interpreted as specialization relations between conceptual categories. In other words, WordNet can be interpreted and used as a lexical ontology.

A prominent example of using WordNet,as an ontology is to determine the similarity between words. Various algorithms have been proposed, and these include considering the distance between the conceptual categories of these words, as well as considering the

---

<sup>6</sup>Originally the lexical comparison was proposed in the study (see the section 3.4), in the thesis I refer the design was changed to proposed semantic comparison, I’ll devote to this issue later.

hierarchical structure of the WordNet ontology. A number of these WordNet-based word similarity algorithms are implemented in a Perl package called `WordNet::Similarity` [27].

**Querying WordNet** can be done by two ways – offline and online query to RDF/OWL WordNet.

- - offline – download of the appropriate WordNet version is necessary and load it into local processing software, then query languages (SPARQL, Prolog) can be used to query the data.
- - online – query the on-line version by doing an HTTP GET on WordNet URI (such as <http://www.w3.org/2006/03/wn/wn20/instances/wordsense-bank-noun-1>). HTTP GET request returns the Concise Bounded Description of the requested URI, which is an RDF graph that includes all statements in the whole WordNet RDF/OWL which have that URI as its subject. This is a far less flexible approach because it is not possible to pose queries (e.g. a query for all synsets which contain the word “bank”). However, it does give a sensible set of triples to answer the question “tell me about this resource” if the user has no prior knowledge of this resource [24].

## 3.5 Implementation of Protégé plug-in

The algorithm was successfully implemented like a Protégé tab-widget plug-in, using the Java programming language, the Protégé (OWL) API and the Eclipse IDE for development. The source code is well documented (using JavaDoc) so it is easily readable by others and thanks to OO programming the program can be modified easily. All together it has about 4000 of lines, (involving the JavaDoc documentation).

### 3.5.1 Implementation details

The understanding of various API’s and the cooperation with existing software, in combination with wrong references between formulas in the proposed algorithm and some missing facts and errors, proved to be an extensively time-consuming. The whole process of plug-in realization exceeded my timing estimate severalfold.

#### Implemented classes

To achieve structured and transparent source code, I have separated classes with completely different functionality into standalone modules, brief description of involved modules follows.

**OntoPlugin** – extends `AbstractTabWidget` and represents the main class which creates Protege tab widget plugin and enables the communication between plugin and Protege application. In `Ontology Mapping Algorithm` class `OntoPlugin` represents top level class (*70 lines of code*).

**MyLogging** – contains methods and resources for login creation and also stores information which are displayed in GUI of mapping algorithm in the text area (*400 lines of code*).

Subclasses: `MyInfo`.

**MyOWL** – contains methods for loading ontologies from file into JenaOWLModel (*300 lines of code*).

**MyPanel** – represents graphical user interface of my mapping algorithm. Part of the code was generated using JFormDesigner and the rest written manually (*1000 lines of code*).

Subclasses: MyFilter.

**OntologyMappingAlgorithm** – implements algorithm for ontology mapping. Algorithm is realized on Protege-OWL API Interface and together with other involved classes it forms Protege tab widget plugin (*2300 lines of code*).

Subclasses: Compare, ComparisonOfConcepts, Concept, ConceptsDatabase, Config, HistoryRecords, InstanceSet, OntoModel, PreFetch, Property, PropertyComputation.

### 3.5.2 Plug-in interface and control

In the algorithm, there are thresholds which affect the whole computation, some of them are set in the source code, but two of them can be regulated through the plug-in interface. But these thresholds are not the only one options, the interface on the first look should be little bit confusing, so I will shortly describe main controls and options.

**Load Ontologies** area enables the choice of ontologies which will be mapped. Working with Protégé normally begins with opening of an existing project (= ontology) or creation of a new one, thus with the option

- **Use project ontology** can be decided whether the first ontology for mapping will be taken from the actual project, or if another ontology should be loaded.
- **First ontology** field represents the first ontology for the ontology mapping process, depends on the option above whether this ontology corresponds with the ontology in the actual project.
- **Second ontology** field represents the second ontology for the ontology mapping process.

**Settings** area enables detailed adjustment of logs and thresholds.

- **Enable logging** option determine whether the logging information will be stored into the external file. Note that the logging process should slow down whole computation markedly (especially when the log level is set to a lower priority).
- **Use history records** option determines whether the plug-in will try to load results from previous mapping (only if ontologies and options are unchanged). This feature is not yet implemented.
- **Select log level** choice influences the quantity of logging messages which will be stored.
- **Current log file** field informs which file will contain log messages.

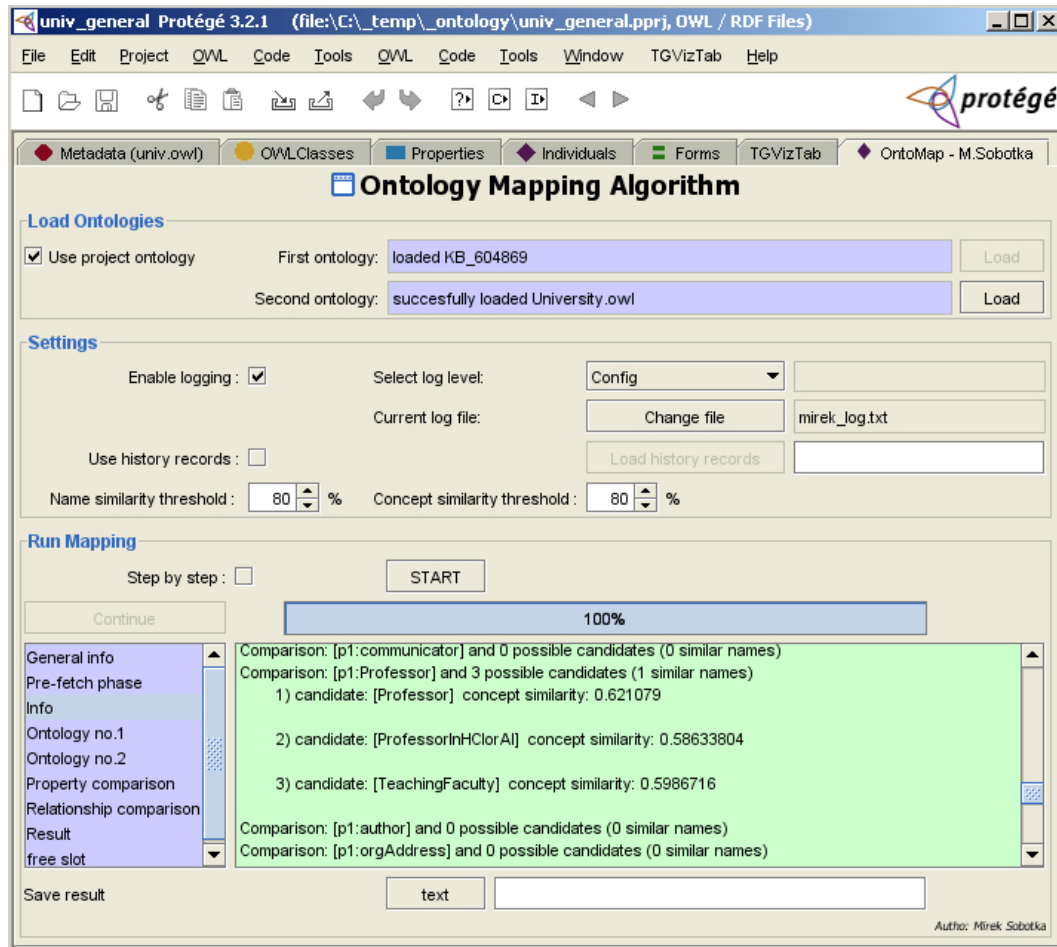


Figure 3.8: Realized tab widget plug-in – Ontology Mapping Algorithm

- **Name similarity records** spinner denotes the required similarity between concept names in the pre-fetch phase. Higher the threshold is, less concepts are taken into the set of possible candidates for each concept from first ontology. Problem is if the threshold is too small, the set of possible candidates will be very large and the whole computation becomes extremely time-consuming.

This part is a big weakness, because as mentioned in the section 3.4.2 the comparison between concept names is only lexical, so the corresponding counterpart from the second ontology can be eliminated in the first step of computation.

- **Concept similarity threshold** spinner denotes the minimal similarity (after the computation) between two concepts, which determine whether the matching counterpart was or was not found. It means this value determines the minimal required similarity for the mapping acceptance.

**Run Mapping** area starts up the computation, informs users about the progress and displays achieved results and debug information. Displayed information are intentionally detailed and the content can be easily changed in the source code, this possibility is important for the debugging of mapping algorithm.

- **Step by step** option stops the computation after every important step in the algorithm (for example after the pre-fetch phase). To continue the computation it is necessary to press the button “Continue”.
- **Start** button starts the computation.
- **Info** displays achieved results and debug information, these information are divided into following categories: *General info, Pre-fetch phase, Info, Ontology no.1, Ontology no.2, Property comparison, Relationship comparison, Result*.
- **Save result** button saves achieved results into the specified file. This feature is not yet implemented.

I must admit that the functionality of implemented Protégé plug-in is limited and that the results of realized mapping algorithm are not satisfactory.

Despite these facts I claim that primary aims have been fulfilled – my program was successfully attached to Protégé application like a plug-in, I have implemented all necessary methods for working with ontologies and all individual elements, all comparisons and computations proposed in the algorithm were also implemented, everything is written using principles of object oriented programming, thus all formulas and computations can be easily changed or modified during future development and algorithm debugging.



# Chapter 4

## Conclusion

### 4.1 Achieved results

In the beginning I had divided the objective of my work into two sections so in the final conclusion I will follow this distribution.

#### **Ubiquitous Learning**

I was familiarized with the ubiquitous learning environment and with the recommended commercial tool LMA. I had created a pilot mobile presentation and brief guide about the handling of this application and I summarized its functionality, advantages and disadvantages.

Then, I acquainted myself with Course Management Systems and Virtual Learning Environments. The Moodle CMS was chosen (based on conclusion from work [22] of my predecessor in the MAPLE project) from a variety of tools as the most suitable. I created a virtual Moodle server, set up the system and filled the pilot course database with sample learning modules. The features of Moodle were successfully tested and the whole process of testing and configuration described.

#### **Ontology Mapping**

The domain of ontologies is nowadays discussed and a particularly topical problem. It was necessary to make an introduction to this issue and to explain how we can use and work with ontologies.

The next step was also completed with success by analyzing the available tools and to choose the appropriate and suitable tool or manner which will satisfy our needs. The result of my analysis was to implement the Protégé plug-in, using Protégé (OWL) API and Java programming language. All used resources are properly presented and their use substantiated.

The final part of this section is devoted to ontology mapping algorithm, description of proposed design, implementation details and demonstration of achieved result.

### 4.2 Future work

As I wrote in 3.4.2, the extension of semantic comparison in the mapping algorithm is highly suitable, so one part of the future work is to implement semantic comparison (online/offline

Wordnet, manual specification).

The next step should be cooperation with a visualization tool to represent the achieved mapping result. From possible alternatives, I found most suitable the TGVizTab Protégé plug-in, but the cooperation means understanding and modifying the source code of this plug-in. Originally it is designated for the representation of one ontology tree and for my purpose I need to represent two ontology trees connected with the appropriate relations.

After elaborating these extensions, it is obvious to make the evaluation of this algorithm and comparison with rivalrous algorithms and mapping methods. It is then possible to decide whether this mapping technique is suitable for our purposes.

The individual components were successfully tested and the next step is to make the working system and to put it into practice. Concretely it means to establish an independent Moodle server for a wide group of students, fill up the database with learning objects, create ontology (or use existing one) for learning objects and course description. Finally, thanks to the ontology mapping algorithm, cooperation is allowed between other institutions and systems.

# Bibliography

- [1] Barwise, J., Seligman, J. Information flow: The logic of distributed systems, 1997.
- [2] Drummond, Nick, Horridge, Matthew, Rogers, Jeremy. *A Practical Introduction to Ontologies & OWL*. Ontology tutorial, The University of Manchester, 6 2005.
- [3] Duineveld, A. J., Stoter, R. *WonderTools - A comparative study of ontological engineering tools* [online]. Last revision 2005-01-01 [cite 2006-11-10].
- [4] Eboueya, M., Lillis, D., Jo, J., Cranitch, G., Martin, Ph. *Mobile Active Participative Learning Environments for the 21st Century Classroom : THE MAPLE PROJECT*. European conference on european models of synergy between teaching and research in higher education, University of La Rochelle, 2006.
- [5] Entremont, Corey. *VLE: Using Online CMS to Implement Constructivism in Learning at the Secondary Level* [online]. Last revision 2004-06-31 [cite 2006-11-10]. <[http://moodle.org/other/dEntremont\\_Final\\_Paper.pdf](http://moodle.org/other/dEntremont_Final_Paper.pdf)>.
- [6] Ganter, B., Wille, R. Formal concept analysis: Mathematical foundations, 1999.
- [7] Horridge, Matthew. *An overview of the available interfaces - API diagram* [online]. Last revision 2007-02-11 [cite 2007-02-10]. <<http://protege.stanford.edu/plugins/owl/api/ProtegeOWLModel.pdf>>.
- [8] Kalfoglou, Yannis, Schorlemmer, Marco. Ontology mapping: The state of the art. In Kalfoglou, Y., Schorlemmer, M., Sheth, A., Staab, S., Uschold, M., editors, *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005. <<http://drops.dagstuhl.de/opus/volltexte/2005/40>> [date of citation: 2005-01-01].
- [9] Knublauch, Holger. *Protégé-owl api programmer's guide* [online]. Last revision 2006-08-21 [cite 2007-02-10]. <<http://protege.stanford.edu/web/guide.html>>.
- [10] Kong, C. Y. *Effective Partial Ontology Mapping in a Pervasive Computing Environment*. Diploma thesis, The University of Hong Kong, cykong@cs.hku.hk, 11 2004.
- [11] Kong, C. Y., Wang, Lau, F. C. M. *Ontology Mapping in Pervasive Computing Environment*. Technical report, The University of Hong Kong, Department of Computer Science, cykong@cs.hku.hk, 11 2003.

- [12] Martin, Ph., Eboueya, M. *Toward a Cooperatively Built Ontology of Knowledge Engineering*. Conference on computer engineering and applications, University of La Rochelle, 2007.
- [13] Martin, Ph., Eboueya, M., Blumenstein, M., Deer, P. *A Network of Semantically Structured Wikipedia to Bind Information*. Aace conference on e-learning in corporate, University of La Rochelle, 2006.
- [14] Martin, Ph., Eboueya, M., J., J. Jo, Uden, L. *Between too informal and too formal*. International conference on knowledge management in organizations, University of La Rochelle, 2006.
- [15] Mifsud Trent, Dr., Casey Des, Dr. *E-learning to u-learning, adapting learning environments to mobile devices*. report, Monash University, Faculty of Information Technology, 2005.
- [16] Moodle, Team. *Moodle - A Free, Open Source CMS for Online Learning* [online]. Last revision 2007-02-01 [cite 2007-02-01]. <<http://moodle.org/>>.
- [17] Niwattanakul, S., Eboueya, M., Lillis, D. *Describing and Researching of Learning Resources with Ontology Model*. John vincent atanasoff international symposium on modern computing, University of La Rochelle, 2006.
- [18] Niwattanakul, S., Eboueya, M., Lillis, D. *Research and Description of Learning Resources on Ontology Model*. The first international conference on knowledge, University of La Rochelle, 2006.
- [19] Noy, Natalya F., McGuinness, Deborah L. *A Guide to Creating Your First Ontology* [online]. Last revision 2005-01 [cite 2006-11-10]. <[http://protege.stanford.edu/publications/ontology\\_development/ontology101-noy-mcguinness.html](http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html)>.
- [20] Sinir, Siyamed Seyhmus. *Ontology Mapping Survey* [online]. Last revision 2007-02-10 [cite 2007-02-10]. <[www.srdc.metu.edu.tr/webpage/seminars/Ontology/Ontology%20Mapping%20Survey.ppt](http://www.srdc.metu.edu.tr/webpage/seminars/Ontology/Ontology%20Mapping%20Survey.ppt)>.
- [21] Smith, Michael K., Welty, Chris. *OWL Web Ontology Language Guide* [online]. Last revision 2007-01-01 [cite 2007-02-10]. <<http://www.w3.org/TR/owl-guide/>>.
- [22] Spicka, Jiri. *Mobile Active Participative Learning Environment*. Diploma thesis, Brno University of Technology, jirispicka@seznam.cz, 6 2006.
- [23] Stanford, University. *Protégé* [online]. Last revision 2006-11-10 [cite 2006-11-10].
- [24] van Assem, Mark, Gangemi, Aldo, Schreiber, Guus. *RDF OWL Representation of WordNet* [online]. Last revision 2007-02-11 [cite 2007-02-11]. <<http://www.w3.org/2001/sw/BestPractices/WNET/wn-conversion.html>>.
- [25] Wiki, Edutech. *Edutech Wiki: Ubiquitous learning* [online]. Last revision 2007-03-01 [cite 2007-03-01]. <<http://edutechwiki.unige.ch/en/Ubiquitous.learning>>.
- [26] Wikipedia, the free encyclopedia. *Constructivism (learning theory)* [online]. Last revision 2007-04-27 [cite 2007-04-29]. <<http://en.wikipedia.org/wiki/Constructivism>>.

- [27] Wikipedia, the free encyclopedia. *Ontology* [online]. Last revision 2007-02-10 [cite 2007-2-10]. <<http://en.wikipedia.org/wiki/>>.
- [28] Wikipedia, the free encyclopedia. *Virtual learning environment* [online]. Last revision 2007-02-10 [cite 2007-2-10]. <[http://en.wikipedia.org/wiki/Course\\_management\\_system/](http://en.wikipedia.org/wiki/Course_management_system/)>.

## Glossary

|         |  |
|---------|--|
| API     | Application programming interface                    |
| CMS     | Course Management System                             |
| CVS     | Concurrent Versions System                           |
| DAML    | DARPA Agent Markup Language                          |
| DL      | Description Logic                                    |
| DTD     | Document Type Definition                             |
| IDE     | Integrated development environment                   |
| OIL     | Ontology Integration Language                        |
| FOAM    | Framework for Ontology Alignment and Mapping         |
| HTML    | HyperText Markup Language                            |
| HTTP    | HyperText Transfer Protocol                          |
| HTTPS   | HyperText Transfer Protocol Security                 |
| L3i     | Laboratory Information-Interaction-Intelligence      |
| LAN     | Local Area Network                                   |
| LDAP    | Lightweight Directory Access Protocol                |
| LMA     | Learning Mobile Agent                                |
| LMS     | Learning Management System                           |
| MDTS    | Mobile Delivery and Tracking System                  |
| MOODLE  | Modular Object Oriented Dynamic Learning Environment |
| NTLM    | New Technology LAN Manager                           |
| OISs    | Ontology Integration Systems                         |
| OO      | Object-Oriented                                      |
| OKBC    | Open Knowledge Base Connectivity                     |
| OWL     | Web Ontology Language                                |
| PC      | Personal Computer                                    |
| PDA     | Personal Digital Assistant                           |
| PHP     | Hypertext Pre-processor                              |
| RDF     | Resource Description Framework                       |
| RDF-S   | Resource Description Framework Schema                |
| RTM     | Read The Manual                                      |
| SBO     | Semantic Bridge Ontology                             |
| SPARQL  | Simple Protocol And RDF Query Language               |
| SQL     | Structured Query Language                            |
| SSL     | Secure Sockets Layer                                 |
| ULE     | Ubiquitous Learning Environment                      |
| URI     | Uniform Resource Identifier                          |
| URL     | Uniform Resource Locator                             |
| VLE     | Virtual Learning Environment                         |
| TCP/IP  | Transmission Control Protocol/ Internet Protocol     |
| TOC     | Table Of Content                                     |
| WAP     | Wireless Application Protocol                        |
| WWW     | World Wide Web                                       |
| WYSIWYG | What You See Is What You Get                         |
| XAMPP   | Apache & MySQL & PHP & Perl                          |
| XML     | eXtensible Markup Language                           |