eLEARNIG INITIATIVE

PRAISE:

***P**eer **R**eview Network **A**pplying **I**ntelligence to **S**ocial Work **E**ducation*

Grant agreement number: 2003 - 4724 / 001 - 001 EDU - ELEARN

Technical aspects

Version number 1.0

30.5.2005

Executive summary

Name	Technical aspects
Version	2.0
Date	30.10.2005
Status	Final
Confidentiality	
Participant Partner(s)	UHI
Author(s)	PdT
Task	5
Distribution List	
Abstract	
Keywords	
Previous Versions	
Version Notes	

Table of contents

Technical Aspects

1.1 Introduction

All the pedagogical processes developed and tested during the project are supported by a complex and innovative technological infrastructure. In particular, the technical components support and integrate three main functions:

- a complete Virtual Learning Environment (VLE), that allows creation and management of on-line courses, and delivery of such courses to groups of students;
- a repository of case studies, known as the Case Study Database (CSDB), where social workers can upload their experience in a structured way, and all students and other workers can search and browse the contents;
- a semantic indexing subsystem that links and connects the information in the two above systems.

In particular, technical innovation in the research project mainly lies in the semantic elaboration infrastructure, that allows a deep integration of didactic contents, available in the VLE, and reflection about practical experience, stored in the CSDB. Such integration is visible by the user in the form of “links” that appear while students are reading the on-line course materials. Such links lead to a list of interesting case studies, relevant to the current course contents. The main innovation is in creating an automated and integrated system to discover automatically, owing to artificial intelligence techniques, which case studies are relevant.

The choice of *automatic* and *intelligent* association yields several advantages: the author of the course material is not required to know personally all the relevant experiences at the European level, and new experiences loaded into the CSDB will be immediately connected to the relevant didactic modules. In particular, this implies an automatic updating of the course contents, where the module content is revised only when the authors need to change it, yet the overall contents presented to the users are continuously updated and always refer to the latest additions to the case study archive.

The system is extremely complex to build and to manage, since it needs to integrate highly interactive e-learning functions with sophisticated artificial intelligence algorithms based on an ontology built by an international team of experts. However, such complexity is totally hidden behind an extremely simple user interface, that for end users (students) just consists in a **Related experiences** button, while content creators (here, authors of didactic modules and of case studies) need simply select items by navigating through an annotated thematic index.

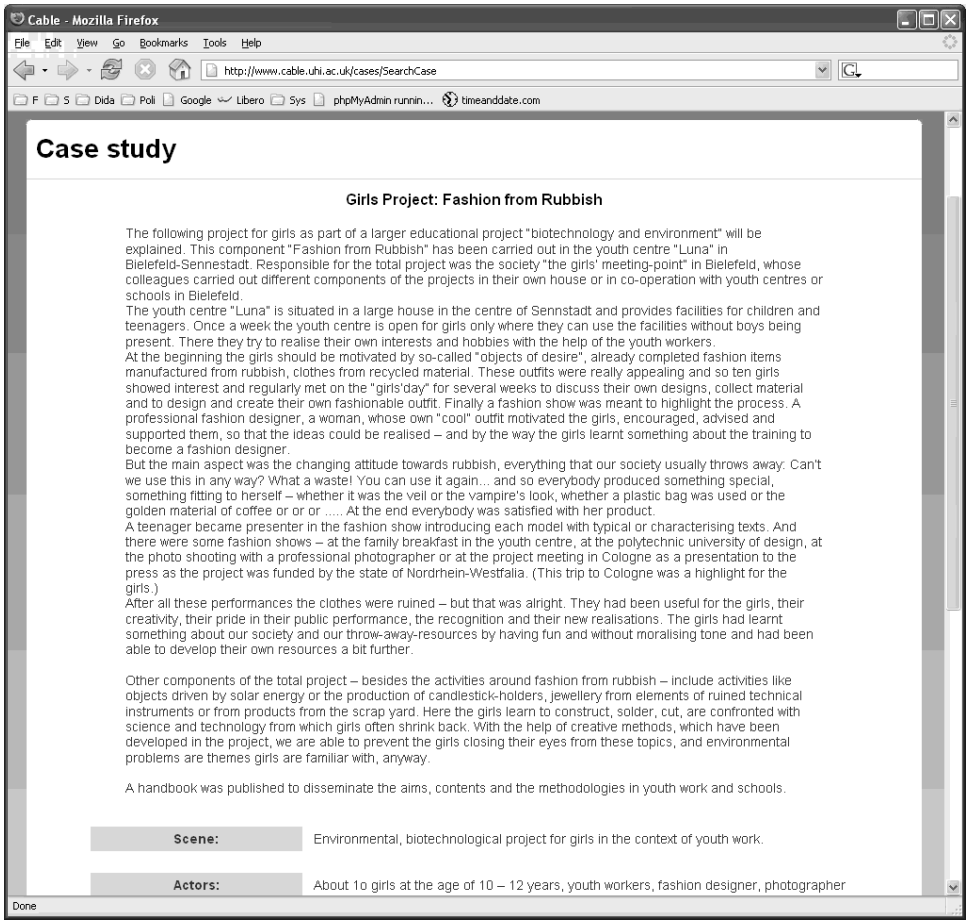

Figure 0.1: Example of a case study

1.2 Case Study Database Subsystem

The CSDB is a content management system built mainly by YHSV (Yrkeshögskolan Sydväst), with the aim to provide an effective and easy to use application for the publication of case studies in different languages.

A content management system (CMS) is an environment aimed at retrieval, management and publication of contents on one or more communication media. Typically, in the retrieval phase, content is generated through authoring new documents, or retrieval of contents from other sources of data. In the management phase, contents are processed and stored for further elaboration, and are tagged with metadata, i.e., information describing the content and its state. The content then is modified following a well defined work-flow with manual or automatic procedures. The final phase is publication where various components are assembled together with a suitable form and presentation, ready to be delivered to the end user. Not all CMS have the same functions or can manage the same type of data, because they are aimed at very different kinds of publications, different organizations and different costs.

1.2.1 Case Study architecture

Here, a case study is a document describing a particular situation and is composed of the following content blocks:

- narration;
- scene;
- actors;
- actions;
- reactions from the environment;
- time and space;
- aim;
- context.

The users of the system are administered through LDAP[1]. With LDAP the users have the same username and password as in the VLE (Virtual Learning Environment).

The possible operations are typical for a CMS, both for the administration of the system and the management of contents, like:

- adding, editing or deleting a coupling to semantic network;
- adding, editing or deleting a case study;
- translating a case study;
- searching a case studies.

The multi-lingual aspect is managed, marking every document with a language identification.

The content creator must, before starting to write a case study, choose the language in which it is written (or the default language associated with the user will be associated with the content).

There is also a modified editing environment aimed at facilitating the translation from a language into another. When a translation is done the fields of the case study in the original language are displayed in the web browser window side by side with the corresponding field in the case study in the new language.

Case studies in different languages are also tracked together, for an easier control from administrators and content creators.

[1] Lightweight Directory Access Protocol. See http://www.kingsmountain.com/LDAPRoadmap/index.shtml,

http://www.redbooks.ibm.com/abstracts/sg244986.html or http://www.cru.fr/ldap/ (in french)

1.3 Virtual Learning Environment Subsystem

A Virtual Learning Environment (VLE) is a Content Management System (CMS) aimed to provide an environment where, starting from chunks of content stored in a central repository, didactic content could be delivered to the user with a computer system, and the users' achievement could be checked and memorised. Typically a VLE has an interface that allow users to register and take courses, with a consistent interface for the duration of the course and an authoring interface that allows teachers to write new courses, manage the students, create drills, check the drill results, personalise the aspect of the courses, etc.

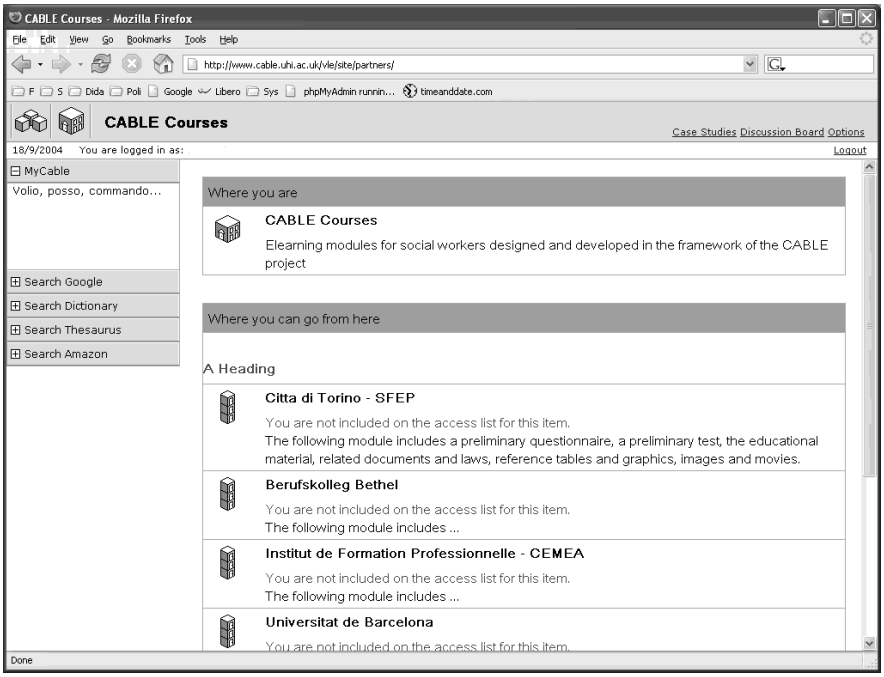

Figure 0.2: Example of a Bodington VLE session

1.3.1 Bodington System

In the context of this project, the selected VLE is the Bodington system[2].

Originally developed by the University of Leeds to create its virtual learning environment, Bodington is open source software that the worldwide academic community can use for free and can contribute to its development. Bodington powers virtual learning environments in the UK and worldwide.

[2] See http://www.bodington.org/index.html

The Bodington System is distributed with an open source License: the software was produced within the University of Leeds with the intention of benefiting students and teaching staff and not for commercial exploitation.

Bodington is being used by a range of learning institutions, including UHI Millenium Institute, University of Leeds, University of Oxford, University of Manchester, Yorkshire Coast College and Eton College.

The Bodington System is a software tool that can be used, in conjunction with a database product, to implement an interactive web site. Its original purpose was to implement a virtual learning environment for the University of Leeds - Bodington Common - but it could be used as the basis for other kinds of interactive sites. It would more correctly be described as an application server since modules can be plugged into it to implement different functionality.

The Bodington system has some features like a multi-user operating system but one that is accessed via a world wide web interface. The Bodington System avoids complex configuration pages with this object-oriented approach. You enter the item you wish to work with and within that item the management pages list only options relevant to that type of item.

The resources and pages in a web site powered with the Bodington System exist in a tree shaped structure just like folders and files on a disk filing system but to make navigation easy some of the "folders" use the metaphor of a building, a floor or a room. That means you can tell someone where to find a resource in plain English instead of with its web address. Unlike some virtual learning environments, tools such as communication rooms, questionnaires etc. exist as objects within this tree structure so when you enter a questionnaire, for example you get simple menus of commands and options that relate specifically to the questionnaire, uncluttered by text that relates to other tools in the vicinity. A clear navigational bar allows you to jump out of the current tool easily.

VLE software products often dictate the structure of a course area, and could lack the features needed for the personalisation for a particular course. With the Bodington System there are ways to change everything about the way it appears and the way it works. Changing icons and illustrations can be done simply by replacing the graphics files in the templates folder. Changing instructional text on the forms and pages is similarly easy since they exist as ordinary HTML files in the templates folder. Fragments of text that are inserted into the templates interactively are generated by software at run-time so changing these involves some work with the software itself but all the source code is provided and the licence permits to modify it in any way could be deemed useful.

Increasingly, legislation around the world, including in the USA, the UK, Sweden, Netherlands, etc., requires that all students, regardless of any disability they may have, must be given an equal standard of educational provision. That means that when accessing a VLE is an integral part of your course, then the VLE must be accessible to everyone. The Bodington System tries as far as possible to adhere to the W3C’s recommendations. These recommendations are mostly aimed at avoiding bad use of HTML that might defeat assistive technologies such as audible web browsers. However, this does not sufficiently make the web pages accessible. To

help dyslexic and visually impaired users in particular, the Bodington System can provide users with a list of style sheets. The system records each user's selected style sheet and will automatically apply it when that user logs in.

This means that the web site can be displayed in high contrast colours, low contrast colours, big text, small text etc. as appropriate for the user.

The Bodington System is an application server. That means that it provides services for storing and querying data in a database, systems for managing users, systems for controlling access to resources etc. and a programmer can take advantage of this underlying functionality to "plug in" additional code that can achieve a lot of functionality with the minimum of programming.

A single installation of the Bodington System can be divided into a number of administrative zones. The administrators of an administrative zone can create user names and passwords, manage group membership etc. in isolation from the other administrative zones. If they want, managers of one administrative zone can give access to specific resources to specific users from other administrative zones and this enables collaborative use of the web site.

Most VLE products have at their heart an entity that they call a course and within that there are a number of tools that you can switch on or off. By trying to encapsulate the concept of a *course* within a software product they often lock you into the way of teaching they had in mind when they wrote the software. The Bodington System is different: it gives buildings, floors, and rooms to allow to structure the material the way that most fits the structure of specific courses and it gives tools such as discussion rooms, questionnaires, web documents, multiple choice papers, etc., which could be arranged freely.

1.4 Semantic indexing service subsystem

This section will explain the principles behind the semantic subsystem, that connects the virtual courses subsystem with the case studies subsystem. The semantic system is mainly an indexing service and works "behind the scenes"; it is not directly seen by the users, but is indeed a key and central part of the system.

This subsystem is not a classic text retrieval system because it does not have words and phrases as a basic search unit, but uses a semantic approach. In this approach the basic search unit is the concept.

1.4.1 Text retrieval systems and their drawbacks

The first idea to associate automatically lessons and case studies (that will be labeled as documents hereafter) is to add some keywords to the description of all the documents inserted in the system, using a classical text retrieval system to index the keywords or make a full text indexing on the body of documents. It should then be possible to query the system for relevant documents, given an appropriate search string. Text retrieval systems are used in the information technology field since the '70s and a lot of libraries and programs are available both free and proprietary, and the literature on the subject is extensive.

Unfortunately, this solution has some drawbacks that make it not suitable for this application.

The first problem is that two or more words could be homographs, i.e., they have the same orthography, but different derivations and meanings. For example the term *chair* means a seat for one person, the position of professor, the officer who presides at the meetings of an organization or an iron block used on railways to support the rails. The term *lounge* means an upholstered seat for more than one person or a public room with seating where people can wait. If somebody would like to search information about lounge furniture, and uses a search engine relying on keywords, such as Google, with the search term *lounge* the first results page would be about record labels and broadcasters. Of course it is possible to add search terms to refine the query, but some interesting results could be thrown away.

There is also the opposite problem, the synonyms. For instance, *sofa* has the same meaning of *lounge*, but Google does not capture this, so it is necessary to repeat the query using *sofa*, still having record label websites populating the search results. Yet another problem arises when the search is about a general term, e.g. *furniture*, and there is also interest on more specific terms, like *chair*, *table* and *cabinet*.

Another problem arises when a document in different languages is used; here, a search engine is not able to relate a term in one language to a term in another without an additional mapping. It is very difficult to link a document in one language to the same document translated in another language, but these documents are about the same *things* even though they use different *labels* to describe them.

1.4.2 The semantic approach

The first step to solve this problem is to assign only one conventional meaning to each keyword: using the terms above we can choose to define, in our application context, a *chair* as a seat for one person and a *lounge* as upholstered seat for more persons. We can also define synonyms of the terms, implying that more than one word may represent the same concept.

Furthermore, we may define concepts that do not correspond to any single word in our language, but nevertheless have a well-defined and unambiguous meaning. In our example we could define a concept of **sittable**[3] as **a thing suitable for sitting**.

Assigning only one meaning to a keyword, or a group of keywords corresponds to defining a new concept. A concept, in the semantic subsystem is defined by:

- an unique identifier (id), that is used internally to the system;
- a small description (sofa);
- a definition (A long seat, usually with a cushioned bottom, back, and ends, much used as a comfortable piece of furniture.

[3] NB: **sittable** is not word found in English but rather a term constructed for argument's sake.

Having defined the concepts, is possible to link them with relations. A concept could cover a more precise significance respect a more broad one, or represent a quality or a part of another concept. Using relations permits an easier method to catalog the documents, but more importantly permits to make more powerful searches, permitting to relate document that are about similar concepts.

Using a semantic approach using concepts also assists in solving the multiple languages problem.

Concepts, by definition, are not tied to a single language, so it is possible with this approach to search for similar arguments between documents in several languages. In fact is possible to have different small descriptions, definitions and terms in more languages, all tied to a single concept. A document in Italian that talks about *sedia* could be related to a document in English that talks about *chair*.

With a given set of concepts and appropriate relations between them, it is possible to create a conceptual model that describes the knowledge domain in which the application works. Such a model is known as an ontology. A simplified ontology may contain only a hierarchical classification and only one type of relation: and it is more precisely called a taxonomy and could be represented as a tree, with nodes representing concepts, and arcs representing the relations between them. An ontology, on the other hand, could be represented as a graph, where relations are represented as labeled arcs and without a hierarchical structure.

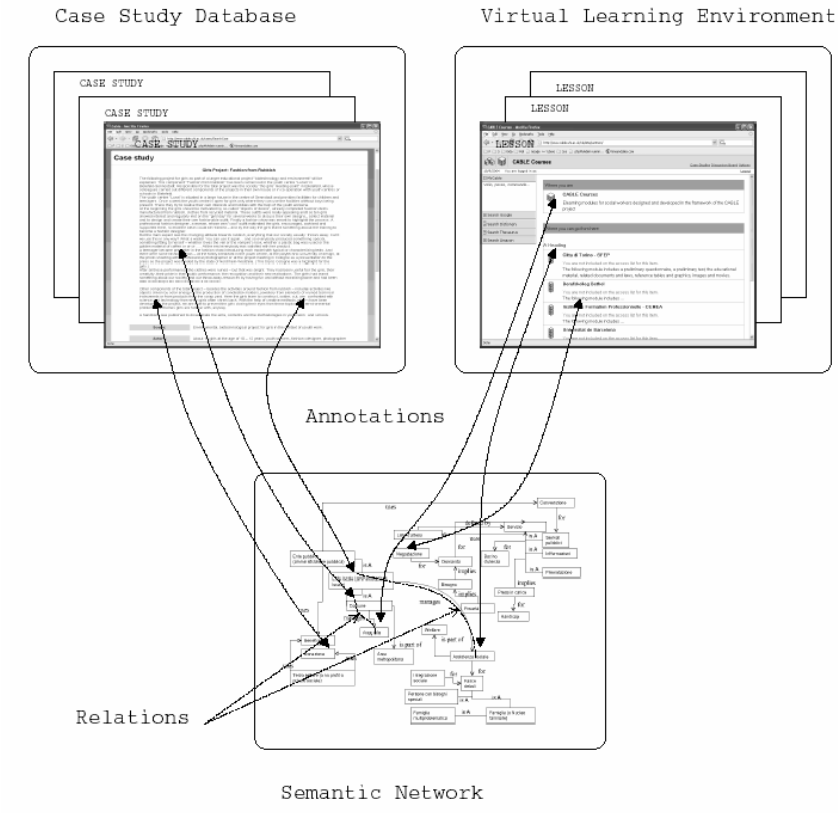

Figure 0.3: Putting case studies and VLE together, using the semantic engine

1.4.3 H-Dose

The system used for the semantic indexing is H-DOSE. The joint adoption of Web Services and Multi Agent Systems enables the provision of easy to access semantic functionality, with a particular focus on semantic indexing, searching and deep-searching functionality.

H-DOSE, being developed as an university research project by the Politecnico di Torino, is open source: the project is distributed under a free license and uses free technologies to run: an Apache Tomcat servlet container, PostgreSQL relational database management system, the Apache Axis SOAP framework, Hewelett-Packard's Jena API for ontology access and management, and TILAB's[4] Jade framework for agent deployment.

H-DOSE also has a distributed architecture, i.e., it is possible to spread various subsystem on different computers, communicating each other my means of SOAP5 messages, giving more flexibility in the layout of the system and improving overall performances.

The platform has the ability to natively support resources in different formats and languages, although its current focus is on textual resources. H-DOSE aims at providing semantic functionality for web applications through an easy to access interface, allowing rapid integration of provided services into the existing development work flow and trying to maximise the benefit/cost ratio of such processes. The platform services are focused on semantic indexing and retrieval tasks with in the context of a specific knowledge domain defined by the application that uses the platform.

More precisely, H-DOSE offers means for classifying textual resources by associating them with concepts of an ontology. It also provides functionality for searching associations between resources and concepts, allowing the retrieval of resources that are conceptually relevant with respect to application queries. Retrieval exploits the ontological structure of the knowledge base and takes into account not explicit relationships between concepts, using a simple navigation engine.

H-DOSE has been designed trying to minimise the impact and the effort required for the integration of the platform services into nowadays applications. In such a sense, the design choices reflect this goal by keeping separate, as much as possible, the domain conceptual model and the resources and by defining associations between resources and concepts that are based on Web standards. With the same "minimum impact" philosophy, the approach chosen for addressing multiple languages consists

[4] Formerly CSELT

[5] Simple Object Access Protocol: a minimal set of conventions for invoking code using XML over HTTP, in a decentralised, distributed environment. See http://www.w3.org/TR/soap/

of a single-ontology approach based on the idea that different languages share much more commonalities at the conceptual level than differences.

The concepts are seen as language independent entities lying at a specification level that abstracts the ontology definition from language specific issues. Language dependent information is addressed by associating to each of such language independent concepts a set of different names, definitions and keywords, one per language used in the application. Concept name, definition and keywords are named synset as the keywords are usually nearly synonymous words that humans adopt to detect the semantics of documents like text resources. The structure of the ontology, and the synset are stored as XML files, making easy to modify the relation either with ad-hoc programs, like Protégé or general purpose editors.

The associations between resources and ontology concepts are named annotations. Annotations can be defined either manually, by selecting concepts suitable for a given document or part of it and sending them together with the document to be indexed, or automatically, using information extraction algorithms from the text to be indexed.

In the first case annotations possess a greater degree of trustworthiness since they are defined by *experts*; however, they are relatively few in number as they are expensive to create and they could be applied also to non-textual data, say movies, sounds or images. On the other hand, machine extracted annotations would be fairly numerous but it is likely that they will be less precise than human generated ones.

In both cases, annotations possess a weight property that allows taking into account those different degrees of trustworthiness and reliability, supporting the definition of different association strengths between resources and ontology concepts. In other words, since a resource will generally span arguments broader than a single concept, it will be pointed by a considerable amount of weighted annotations each relating the resource with a given ontology concept.

Annotations make the platform ready to deal with the intrinsic fuzziness of resources, by exploiting the annotation weights; however, ontology constraints are still valid, and the domain model of concepts and relationships is still taken into account for providing semantics rich services.

Annotations keep track of the semantic correlation between a resource and a concept, including information about the strength of correlation. The data about concepts, reference to documents and annotations group are stored in tables into a relational database, representing at a low level the arguments that are found within a document.

Actually, H-Dose needs a URI[6] to reference the content, but is sufficient to have a procedure that associate a document to a URI, and this procedure could be external

[6] A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource. A Uniform Resource Locator (URL) is a kind of URI that also provide a means of locating the resource by describing its primary access mechanism. See http://www.w3.org/Addressing/

to existing applications. Of course is possible to store annotations into the document being indexed and this is useful when annotations are generated by hand, but this is not a requirement. With this approach the data is stored outside the document, making it possible to use the system without having to modify the document structures and contents, hence H-DOSE could be used without modifying the existing application.

As indicated above, H-DOSE uses as a DBMS system, PostgreSQL, but is would be simple to adapt the system to use other database systems, free or proprietary, using a different JDBC class and adapting the syntax of the SQL queries to the different target.

External applications could interact directly with SOAP messages, but is also available an XML-RPC to SOAP gateway to give more flexibility writing the code needed to hook the HDOSE services to existing systems.

Existing applications could use H-DOSE to function like a text retrieval system, provided that calls for the indexation of new documents, deletion of old documents and document search are modified to call the appropriate H-DOSE methods instead of those found within the text retrieval system.

In conclusion, H-DOSE supports a semantic approach for document searches, with a minimum effort for adapting existing software. With its distributed approach is possible to setup a system with great flexibility, and because it is free software it is easy to adapt to different environments and requests.

Acronyms and Abbreviations

Acronym	*Description*