Research and development of software tools to support teaching and learning collaborative operations

Mario J. López
Departamento de Ingeniería Industrial
Universidad de Santiago (USACH)
Santiago, Chile

Francisco F. Moreno
Departamento de Ingeniería Industrial
Universidad de Santiago (USACH)
Santiago, Chile

Jorge E, Bravo
Departamento de Ingeniería Industrial
Universidad de Santiago (USACH)
Santiago, Chile

Víctor F. Paredes
Departamento de Ingeniería Industrial
Universidad de Santiago (USACH)
Santiago, Chile

Héctor R. Ponce
Departamento de Auditoría y Contabilidad
Universidad de Santiago (USACH)
Santiago, Chile

ABSTRACT

A recently developed educational paradigm shows that teaching and learning has shifted form teaching oriented towards learning oriented (Entwistle 2000). In this new paradigm lecturers and students have more active roles to play in the teaching and learning process. This process is made out of concrete experiences followed by observation and reflection of such experience, which leads to the formation of abstract concepts and the constructions of principles or generalisations that is followed by testing of such concepts in new situations (Kolb 1991). Thus lecturer and student interactions are of new form.

This paper describes both a research into teaching and learning centred in learning and the development of software tools to support the collaborative operations between lecturer and students and among students.

Keywords: TIC; software tools; e-learning; teaching and learning; deep knowledge.

1. INTRODUCTION

The authors' educational experiences had showed that the traditional approach to teaching made students to recall by memory with a risk of failing to remember as time passes.

The development of both educational theory and practice and information and communication technologies have empowered a research and development project, at the Department of Industrial Engineering, to develop a set of software tools to support the operations that emerge from a renewed relationship between lecturers and students.

The main conceptual learning frameworks are the "experiential learning approach" and the distinction between "deep learning" approach and "superficial learning" approach. As a result of the use of these frameworks, our own ideas of what learning meant also changed.

The paper presents the theoretical elements that contributed to build a renewed lecturer student relationship; the experiential approach to learning is described as well as the distinction between deep and surface learning.

Next, the paper describes the design of the renewed lecturer student relationship; that is contents and related activities and evaluations to match the four stages of experiential learning approach: concrete experiences; observation and reflection; formation of abstract concepts and the testing of such concepts in new situations.

The paper then moves to its kernel, the design and development of information and communications technology to support the renewed student lecturer operations. The four modules are detailed as well as their three-layer structure (database services, Web services and client). The description focuses on one of the software procedures of the module for lecturers; the one that allows lecturers to upload previously modelled and developed contents, activities and evaluations. The lecturer has only to follow a dialog box that asks for the files. What is behind these actions is then described; that is, the syntax graph designed and constructed is explained as well as its associated parser and discriminator.

Finally, some remarks are made as a way of conclusions.

2. BACKGROUND

Three topics are discussed as background or theoretical elements that contributed to build renewed lecturer student interactions: the distinction between deep and surface learning; teaching and learning centred in learning and the experiential approach to learning.

2.1 Deep and surface learning

The first theoretical source that influenced our development was the distinction between the superficial approach and the deep approach to learning. With these important concepts, we realized that some students have different ways of confronting the learning process. While some students take learning simply as a matter of memorizing concepts and reproducing knowledge; other students approach learning with interest in ideas and understanding and with a clear intention to transform such ideas based essentially on their previous experiences and knowledge.

2.2 The learning centred model

Some of the characteristics attributed to the teacher-centred or content oriented approach to teaching and learning (Entwistle, 2000) are that the lecturer is the teaching and learning centre; he decides what students must learn; that students participate executing the activities assigned by the lecturer. Thus, students are mostly passive and wait to receive knowledge from the lecturer and it falls on him/her the whole responsibility for the success or failure of student learning. This model, in hands of a good lecturer has proved to be very effective, and for many years, it has satisfied the needs of the academic community. The main difficulty with this model, however, is that it encourages a surface approach to learning (Entwistle, 2000). Under this teaching and learning model, students take learning simply as a matter of memorising concepts and reproducing knowledge.

On another hand, a learning centred model promotes the deep approach to learning among students. That is to say, it develops an environment that fosters the interest in ideas and understanding in contrast with the prevalent approach that principally focus on memorising concepts. The aim is to conceive a series of related activities that helps to stimulate and develop, in students, the ability to seek meaning, relate concepts and make sense of their experiences within and beyond the frontiers of our courses (Entwistle, 2000). In this model, the teaching and learning process changes from being centred in teaching to be centred in learning. Educators became facilitators and learners are much more active (Gibbs 1999).

2.3 The experiential approach

A third theoretical element that came to contribute to our development of an e-learning environment was the experiential approach to learning (Kolb, 1984). The crucial question that this approach addresses is how students learn. Although originally formulated to address the question of adult education, it has constituted in an important contribution to understand how students learn in general. In this approach, learning is understood as a process in which "people generate from their experience the concepts, rules and principles that guide their behaviour in new situation" (Kolb, et al., 1991, p.60). The effectiveness of their behaviour depends on how they adapt their concepts, change their rules or discover new guiding principles.

The learning takes place through a continuous and recurrent sequence of actual experiences and, as experiences for themselves are insufficient, they must be accompanied by thought, observation, abstract concept construction and trying out these concepts in new experiences. Thus, the learning process is conceived as a four-stage cycle. (1) concrete experiences are followed by (2) observation and reflection of such experience, which leads to the (3) formation of abstract concepts and the constructions of principles or generalisations which follows (4) the testing of such concepts in new situations.

We realised that following this model we could incorporate new learning activities to emphasise each phase of the cycle. Thus, we thought of introducing seminars with small groups, with a view to have debates or discussions on new concepts. Assessing students through the development of a case study applied to a real world situation, which was conducted as a course project. Assigning minor research work on the Web and developing a more personalised instruction.

We also realised that information technologies, such as those integrated to Internet, GroupWare systems and CDs, adequately mixed into the educating practices, have ample possibilities to enact both the learning centred approach to learning and experiential learning. Consequently, using previous courses bank of students' projects, we thought of making them available through the Internet, preparing them as simulations. We also thought it appropriate to prepare audio and video clips with main concept explanations.

The change of paradigm can be summarised as follows. The lecturer in the traditional paradigm is a deliverer, unique assessor and decides what and how students learn. Students are dependent, individualist and receptive. On the other hand, in the new paradigm lecturers are managers, planners, designers, facilitators and guides. Students are autonomous, participating, collaborative and engaged (Cervera & Gonzalez 1997).

Information technologies also enable a better teaching and learning process, which can have a greater student

emphasis via the use of learning resources available with greater degrees of freedom: place and time.

3. LECTURER STUDENT INTERACTIONS

Next, it was necessary to design a teaching and learning process, centred in learning and using the experiential approach. All this with a view to promote deep learning,

Thus, a new lecturer student relationship was conceived; that is contents, related activities and evaluations were redefined to match the four stages of experiential learning approach: concrete experiences; observation and reflection; formation of abstract concepts and the testing of such concepts in new situations.

Mainly, the new relationship was based upon two aspects: a new course organisation and the development of a synchronous and asynchronous Web site to support course activities.

3.1 Courses

Consequently, courses were organised in a new and different manner to provide students with opportunities for experiences, thoughts, observations, abstract concept construction and probe of new experiences.

For abstract conceptualisation there was an initial conference per course unit as well as reading of articles, papers and book chapters. For experiences and probe of new ones, students developed, throughout the course, a case study. Seminars with small groups, with a view to have debates or discussions on new concepts were introduced. Assessing students through the development of a case study applied to a real world situation, which was conducted as a course project. Assigning minor research work on the e-learning environment. For thought and deliberation small group seminar were introduced.

According with their conceptual maps, contents were modelled and separated into those to be seen in conferences and those to be presented via the Web site. Conferences embodied the main contents, activities, assignments and the unit's calendar. Web contents were presented in three levels. Level 1 included an explanatory video, annotated presentation of unit concepts and links to the next level. Level 2 covered a detailed explanation of concepts in level 1 and links to level 3, which presented primary material to expand the concepts such as summaries of book chapters, final year projects, articles, papers and lecturer's notes.

Each unit included a set of activities for students experiencing with conferences and Web site contents. The activities consisted of a number of assignments, which made up the development of a case study that students conducted in small groups.

A continuos assessment was employed: Tests on the students' reading and understanding of abstract concepts. The practical work was assessed in two ways, the writing up of one report per unit and the short dissertation as introduction to the seminars. The three required University partial tests were also conducted. An equation to calculate the final mark was agreed with students.

3.2 The Web site

The information and communications technology designed and developed are made up of a model, e-learning tools, an e-learning engine and an e-learning administrator. It is a model of e-learning in that it supports both the experiencial and deep learning. It is an e-learning tool because it allows lecturers to upload content materials, activities and evaluations and it provides students with access to those materials and activities. It is an e-learning engine since it links up client applications with Web and data services and files in any format. It is an e-learning administrator in that it controls accesses, activates and deactivates access to asynchronous and synchronous tools and administrates student syllabus.

The sets of tools are assembled in three distinct modules (student, lecturer and administrator) plus a module shared by the other three. The spread of options available for students and lecturers is as follows.

Figure 1: Menus

The option "connection" gives students access to the synchronous activities (video broadcast, chats, and tests). The other options manage the asynchronous actions, which follow the modelling (contents, activities and assessment) and other options to ease the navigation (bibliography, resources and personal data). The menu for the lecturer is also organised following the modelling: "contents", "activities" and "assessment"; for each, the lecturer can create, alter and upload. The option "connection" allows the lecturer to set date and time for the synchronous activities as well as to supervise them.

4. SOFTWARE TOOLS

The software tools associated to the Web site were built in three layers (Lopez et al 2002). The basic layer is constructed by an object relational database (Dorsey 1999) that administrates file locations, as well as students, syllabuses, courses, activities, assessments, lecturers, hits and others. The middle layer is built by the Apache server, which provides the Web services. The logic of the tools is concentrated in this layer. Communication with the other two layers is administrated as well. This service processes client requests and responds with "html" pages or with a Java class, which includes data and methods. The third layer is made up by a browser, which generates the client requests to the Web server. The Web server, through session administration, sends the clients data and methods in classes.

The four modules (student, lecturer, administrator and shared) have a total of 260 software procedures (Dorsey, & Hodipka, 1999) of different complexity. The shared module has three software procedures, the module for student has thirty six procedures, the module for lecturers has one hundred and thirty seven software procedures and the database and webmaster administrator has eighty four procedures.

The module for the lecturer requests the definition of the list of contents via the provision of labels for chapters, sections and topics. Technically, this is done through an ordered and linked list. Next, the dialog moves to the contents for each of the topics previously defined. These are Web pages prepared by the lecturer, or his/her assistants or specially prepared by hypermedia professionals. Each Web page may have up to three levels of depth, to be adjusted to the three-layer model. For each defined topic, there opens another dialog box, which asks for the name of the parent Web page. The box has a button to examine the folder and file structure of the personal computer of the lecturer. The Web server syntactically analyses the file, identifying internal links and renaming them for definitive location in both the data server and file server. This is repeated for the three layers of the model.

Technically, and due to the necessary administration of various courses, several lectures, multiple students and manifold topics, this is done by the Web server, which sends a class to the client. The class is named ‘LecturerFile’ and, among other attributes, it includes the identification of the lecturer and that of the course. Amid the methods there are the ‘Parser’, which performs the syntactical analysis and asks for the linked files; the ‘FileSaver’, which renames all files with a number generated by a saved procedure in the database, saves names and location in the database and save the file in the web server. Many formats are administrated, among them are: ‘asp’, ‘html’, 'xml', 'cgi', ‘php’, ‘cfm’, ‘jsp’, 'xls', ‘doc’, ‘zip’, 'exe', ‘pdf’, ‘ppt’, 'pps', 'tif’, 'bmp', 'png', ‘jpg’, ‘gif’ 'wav', 'mp3', 'mov', ‘avi’, ‘asf’, ‘swf’, ‘fla’, ‘svg’. This is done recursively three times to accomplish the model.

Of the 137 software procedures available for lecturers, the one that is crucial to the new student lecturer relationship is that which allows the uploading of previously modelled contents. It also better illustrates the design of the site. In the final interface, as described, the lecturer has only to follow a dialog box that asks for the files.

What is behind these actions is now described; that is, the syntax graph designed and constructed is explained as well as its associated parser and discriminator.

With the purpose of designing friendly and easy of use interfaces, syntax or conceptual graphs were used because they express meaning in a form that is logically precise, humanly readable, and computationally tractable. With a direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they serve as a readable, but formal design and specification language. Conceptual Graphs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing (Guy et al 2003).

4.1 The need for design

A lecturer prepares the contents to be uploaded; these take the form of a "html" file, which has links to other documents, images or other files in other formats. For example, figure 2 shows the modelling of contents.

The modelling of contents included a first level with an "html" file describing the headlines of three concepts. The model shows the link to the second level for the first concept ("globalización"), which has a descriptive text, an image and a link to the third level.

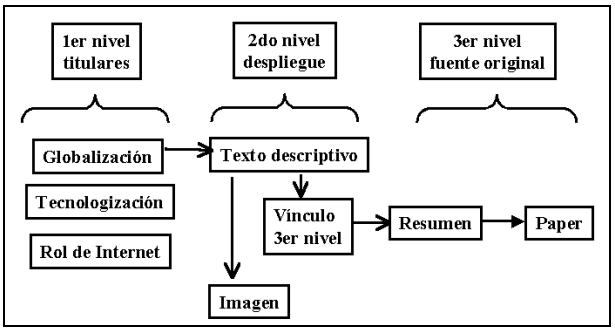

Figure 2: Model of contents

4.2 A grammar analyser

Therefore, a grammar analyser was designed to locate, in the "html" file, references to other files, images or files in other formats. As well as identifying files, the analyser must save file names and location in database and Web services respectively.

In brief, the analyser should first discriminate between valid and not valid references to tags inside the "html" code. Valid tags will be those containing references to any other file in any format or URL addresses. All other tags will be invalid for the purposes of the analysis, since they will not make reference to other files.

Formally, what the analyser should do was:

```
If symbol is aspirant to script or style tag
    locate whole script or style tag
Else
    If script tag
        find end of script tag
        If style tag
            find end of style tag
        Else
            do nothing
    Else
        analyse tag
        If valid symbol
            Read file or URL name
        Else
            do nothing
```

Figure 3 shows the syntax graph for the described analyser.

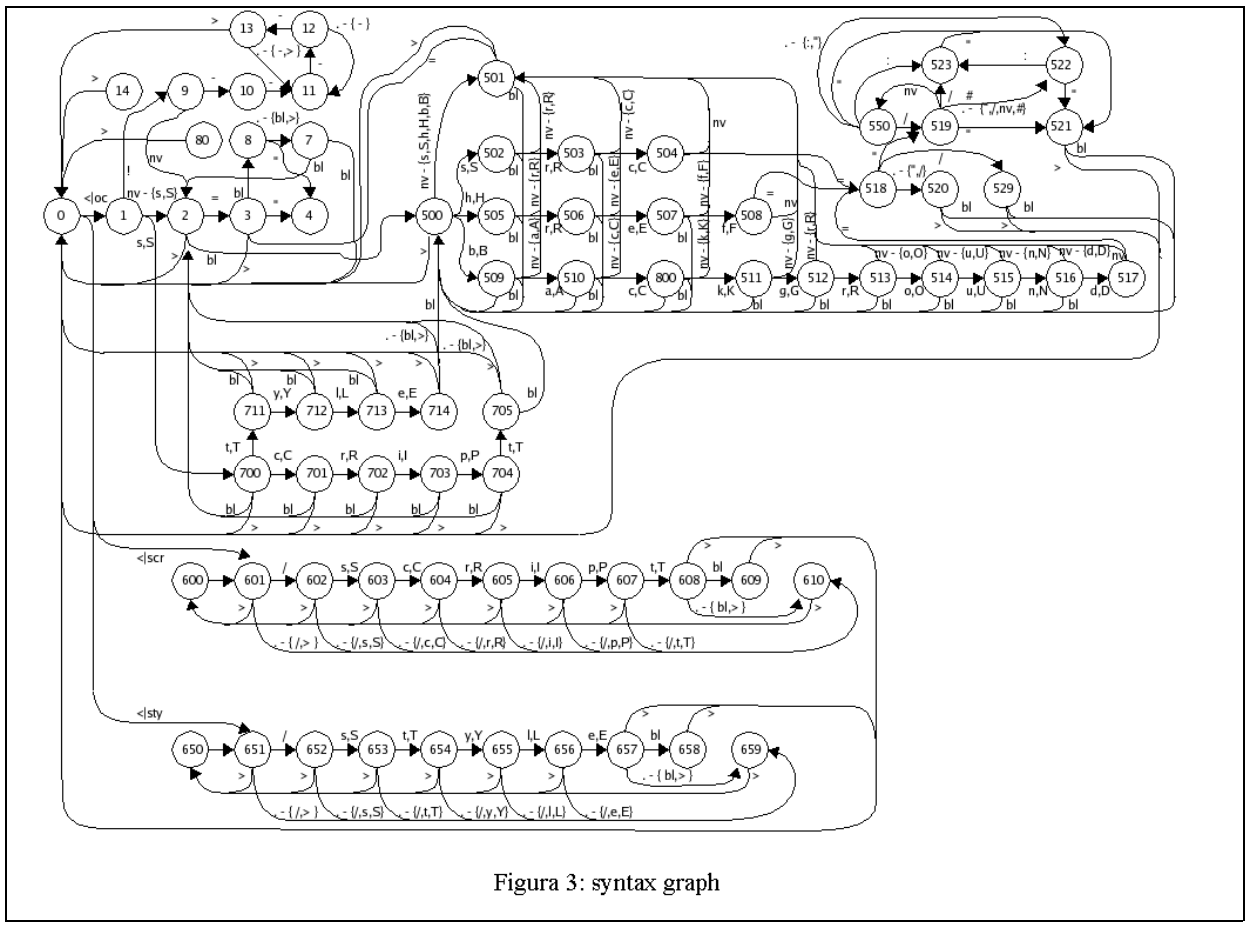

Figura 3: syntax graph

The main actions of the analyser are fulfilled by five distinct blocks, which were numbered for convenience only:

Blocks of states 0 to 14 and 80: The opening of tags is analysed if tag is valid, it considers both cases: fulfilment and unfulfilment of html w3c standards (World Wide Web Consortium).

Blocks 500 to 550: Possible file names are analysed for tags beginning with "script", "href" or "background". URLs and references are also considered.

Blocks 600 to 610: Once the sequence "script" is read, it waits for the "/script" to return to normal processing.

Blocks 650 to 659: Once the sequence "style" is read, it waits for the "/style" to return to normal processing.

Blocks 700 to 714: After the sequence "<s" or "<S", it is discriminated if it corresponds to "script" or "style" and variables src or sty are activated. Otherwise processing returns to state 2.

4.3 Syntax graph description

The analyser defines a state variable as an integer with 0 as initial value. For each "html" code element a new value for the variable is obtained; this is shown in the syntax graphs over the arrows between states.

Some of the conventions in the syntax graphs are:

bl = blank space, tabulation or return.
nv = any upper or lower case character, arithmetic or grammatical signs.
<!--...--> = comments.
<script> </script> = embedded code java script, VB script and other scripts.
<style> </style> = styles defined within the document.

The different states in figure 3 are now explained:

State 0: It waits for the tag opening. In a normal process it waits for the "<" character. Once read it passes to state 1. If it is waiting for an end of script it derives control to state 600. If it waits for an end of a style tag, it passes control to state 650.

State 1: Once the tag is opened and if the first character read is the beginning of an element's name, the control passes to state 2. If the character read is "/" (tag closing), then a file is no longer waited and control goes to state 14. If the symbol read is "!", then what follows is a comment and control goes to state 9.

State 2: If the symbol read is "=", then an element of a tag is waited and control goes to state 3. The symbol ">" closes the tag and control goes back to state 0 to begin reading again. If the character read is a blank, tabulation or end of line, control goes to state 7.

State 3: After the "=" symbol, a second member of the tag is waited. If inverted commas are read, the control passes to state 8. If a blank is read, tabulation or end of line is read, control goes to state 8. A ">" symbol goes control back to state 0, otherwise control goes to state 4.

State 4: It waits for inverted commas to close the second member of the tag, then control passes to state 521.

State 7: The reading of a second member without inverted commas began. A blank symbol ends this state and closes the tag, passing control to state 521. If a">" is found, the tag is closed and control goes back to state 1.

State 8: It begins the reading of a second member of a tag, which was opened with inverted commas. A second set of inverted commas are waited to close the element, control passes to state 4.

States 9 to 13: These states read the symbols between "!-", "any set of symbols" and "-->" which correspond to a commentary.

State 500: It waits for an "script", "href" o "background". An "s" or "S" control goes to state 502; and an "h" or "H" to state 505 and a "b" or "B" to sate 509. If it is another symbol that can initiate a tag, control goes to state 501.

State 501: If an end of tag is found (">"), control goes back to state 0. A blank symbol goes to state 500 and "=" takes control to state 3.

States 505 to 508: The beginning of a tag was read that makes reference to a file with the structure "href". If it is not the "href" sequence, control goes to state 500. Otherwise the next symbol is read. Control goes back to state 0 if the symbol found is ">".

States 502 to 504: These states present a behaviour similar to the states 500 to 505, but for the element "script".

States 509, 510, 800, 511and 517: These are similar to the 500 to 508 states, but the element "background".

State 518: Inverted commas initiate the reading of the second member of a tag, which may correspond to a file name and control is given to state 519. If the name begins with "/", control goes to state 529. Otherwise control goes to state 520 to register the file name.

State 519: A file name is waited. If the read symbol is inverted commas, control goes to state 521. If the symbol is "/", it means the file name has a path (file located in other folder) and control passes to state 523. If a "#" is read, then the file name lecture is complete and control moves to state 522. If "f" or "F" are read, then it may be a reference with protocol and the potential file name is read. Otherwise control goes to state 550 to save the file name.

State 520: If the blank symbol is read, then control goes to state 500. If ">" is read, then it is the end of the tag and control goes to state 0. If an "f" or "F" is read control goes to state 530 and the potential file name is registered.

State 521: Symbols blank, tabulation or end of line mean a new tag element will be analysed, therefore control goes back to state 500. If the symbol is ">" the tag is closed and control goes back to state 0.

State 522: A section name of file is waited, which was initiated with a "#". If inverted commas are read, it is the end of the reference, and control goes to state 521 to process the file name. If ":" is read, then what has been read is a protocol (http, ftp, smb or another) and control passes to state 523. The kept file name is disregarded. If "#" is read, then an inverted comma is waited and the file name obtained is added to the list.

State 523: An end of a second member is waited. Inverted commas confirm that end and control goes to state 521 and the file name is added to the list.

States 524 to 527: It is waited for completing the sequence "file:", if it is so, it is processed as if it were a protocol and control passes to state 521. Otherwise, control goes back to state 522.

State 529: It waits for a blank character to close the tag and pass control to state 500. With symbol ">" the tag is closed and control passed to state 0. In both cases the name of the file is added to the list. Otherwise, it keeps reading symbols.

States 530 to 533: These are similar to states 524 to 527, but it focuses in elements without the inverted commas.

States 600 to 610: These states process the “</script>” tag. If the sequence is different, then state 609 is maintained until the closing of the tag, it then returns to state 0.

States 650 to 659: Similar to the above, but they process the “</style>” tag. State 658 is alike 609.

States 700 to 705: Here, it is discriminated if the reading is over “<scrip " (the blank is part of the string) or “<script>”. State 705 passes the control to state 500 if the last symbol read was “ “ or to state 0 if that symbol was “>”. Otherwise control goes to state 2.

States 700, 711 to 714: Similar to the above, but they process the “<style ” (the blank is part of the string) or “<style>”.

4.4 The operations as a result of the syntax graph

The syntax graph described allowed the construction of simple and easy of use human computer interfaces. A first procedure permits the upload of previously prepared files with previously modelled contents. The software tools ask for the first file to be uploaded and then, after processing the links in that first file, the whole list of referenced files is asked. Figure 4 shows the dialog panel for the first file to be uploaded.

Figure 4: Upload of first file

This first file is processed as already explained and the list of references files is presented for uploading. Figure 5 illustrate how the interface asks for the list of referenced files.

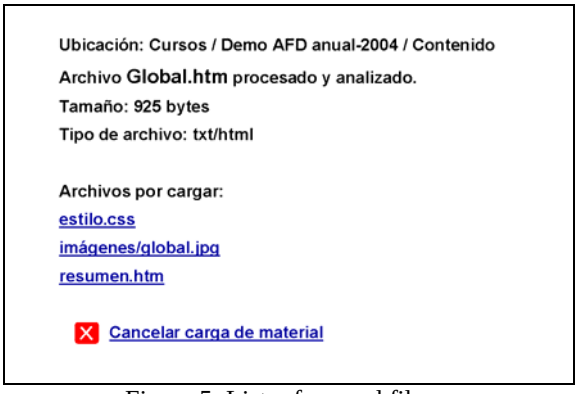

Figure 5: List referenced files

4.5 Example of file uploading

Be a chapter "Generalidades", a section "Introducción" and a theme "Globilización". To this theme it will be uploaded a set of files. The first file is "Global.htm", which makes reference to a style file (estilo.css), an image (global.jpg) and another html file (resumen.htm). The first fourteen states will be as shown in figure 6.

Entry	State	Next state
<	0	1
h	1	2
t	2	2
m	2	2
l	2	2
>	2	0
\n	0	0
<	0	1
h	1	2
e	2	2
a	2	2
d	2	2
>	2	0
\n	0	0
...	...	...

Figure 6: Table of states

The associated actions are to assign a code name, save the original file name and path of the names for the files "Global.htm", "estilo.css", "global.jpg" and "resumen.htm". At the end of the process values are assigned as shown in the following table.

File name	Code name	Path	State
Global.htm	0308051239371.html		Received
estilo.css	0308051239372.css		Expected
global.jpg	0308051239373.jpg	imagenes	Expected
resumen.htm	0308051239374.html		Expected

Figure 7: Process

The code names are made up with the full date, full time and sequence of upload; thus, code name 0308051239373 was upload on year 2003, month 08, day 05 at 12 hour, 39 minutes, 37 seconds and was the third to be loaded.

5. CONCLUDING REMARKS

The followings remarks are made as a manner of conclusions.

There exists the experiential approach to teaching and learning which is centred in learning and fosters deep learning. This approach redefines the relationship between lecturer and student through the inclusion of concept construction; carrying out activities to experience with those and new concepts; and providing opportunities for thought and reflection.

Information and communications technologies were developed to support the new lecturer student operations. The asynchronous modules allowed uploading contents, activities and evaluations. All this permitted to use previous lecturer timetable to conduct seminars to put into practice the opportunities for thought and reflection.

The developed ICT took the form of software tools for the lecturer, student and administrator, totalling 260 software procedures. These procedures were thoroughly and carefully designed with a view to develop friendly and easy of use human computer interfaces.

One of the important procedures developed was the one that allows the lecturer to upload contents, activities and evaluations. This allowed the reorganisation of activities to conduct seminars for thought and reflection. For the design of this particular interface, a concept graph was used, which was detailed explained in this paper.

The uses of these graphs not only guarantied constructing friendly interfaces, but also because they express meaning in a form that is logically precise, humanly readable, and computationally tractable and serve as an intermediate language for translating computer-oriented formalisms to and from natural languages.

Finally, a robust ICT was developed to support students' learning and students have found it not only useful but also said that they would like to see more course in a similar format (Lopez & Ponce 2003). Additionally, and perhaps more importantly, students have recognised that they had gained a deeper knowledge and that they would very much appreciate more courses (Lopez & Ponce 2004).

6. REFERENCES

Cervera, M. & Gonzalez, A. (1997) El docente y los entornos virtuales de enseñanza aprendizaje. Edutec 97 (Universidad Rovira i Virgili).

Dorsey, P. & Hodipka, J. (1999). Oracle 8. Database design with UML. McGraw-Hill.

Entwistle, N. (2000) Promoting deep learning through teaching and Assessment: conceptual frameworks and educational contexts. TLRP Conference, Leicester, November.

Gibbs, G. (1999) Teaching in Higher Education: theory and practice, how students differ as learners, Milton Keynes: Open University.

Guy W. Mineau, G, Moulin, B & Sowa J (2003). "Conceptual Graphs for Knowledge Representation". Canada: Springer-Verlag.

Kolb, D. (1984) Experiencial Learning: experience as the source of learning and development, New Jersey: Prentice-Hall.

Kolb, D., Rubin, I. and Osland, J. (1991) Organizational Behaviour: an experiential approach, New Jersey: Prentice-Hall.

Lopez M & Ponce H (2004). e-learning promotes students' deep learning: a case study. Proceedings of the Communication Systems, Networks and Digital Signal Processing (CSNDSP' 2004). Newcastle upon Tyne, England, 20 – 22 Julio.

Lopez, M, Bascuñan, C, Bravo, J. & Paredes, V. (2002) Research and development of e-infrastructure to support students' learning, in Carrasco, R (ed). Proceedings of the Communication Systems, Networks and Digital Signal Processing (CSNDSP' 2002). Stafford, England, 15 – 17 Julio.

Page-Jones, M. (2000). Fundamentals of Object Oriented Design in UML. Addison Wesley.