Giacomini Pacurar, E., Trigano, P., Zamfirescu, C (2005).
An IMS LD Graphic Editor using the graphs representation for modifying the course structure
Journal of Interactive Media in Education, 2005(15)
[jime.open.ac.uk/2005/15].

Published
25 August 2005

ISSN: 1365-893X

An IMS LD Graphic Editor using the graphs representation for modifying the course structures

Ecaterina Giacomini Pacurar,
Philippe Trigano, Cristian Zamfirescu

Abstract:

We developed a Web portal, named netUniversité, that enables the teachers to create their courses online then to visualize, manage and participate to them. In order to create and execute the courses content, we have implemented an IMS LD player and an IMS LD web course content editor. These modules are integrated in the netUniversité. Using this application the teacher can automatically generate educational Web site structures, adding the pedagogical contents in these structures. However, the main weakness of a web editor (like ours) is the lack of a global view of course structure. These aspects prompted us to create a graphical editor that offers to the teacher a perspective on the IMS LD document with the possibility to perform some basic editing operations (adding, moving, deleting) in order to complete the web editor. In this paper we describe our approach concerning this graphical editor that is included in the web editor to the netUniversité web portal.

Keywords: pedagogical web sites, automatic generation of course structures, IMS LD graphic editor

Commentaries:

All JIME articles are published with links to a commentaries area, which includes part of the article's original review debate. Readers are invited to make use of this resource, and to add their own commentaries. The authors, reviewers, and anyone else who has 'subscribed' to this article via the website will receive e-mail copies of your postings.

1. Introduction: presentation of the context

Considering the diversity of digital material and taking into account that users can access these elements, distribute them, exchange them, update them, the designers of electronic teaching objects have underlined the importance of standards. These standards provide a "common language", which is currently used for indicating, organizing and describing the digital educational resources.

Our analysis regarding various norms and standards (IEEE, 2001; Friesen et al., 2002; ADL, 2001; IMS, 2003) enabled us to choose the standard IMS LD for representing the pedagogical content in the models of educational Web sites. In realizing these teaching models, we are interested to take into account several concepts like: theories and models of teaching and training, curricular areas, roles of actors (learning, group leaders, professors, tutors, etc), interactions between these various actors in the training environment. The use of this language facilitates the implementation of the dynamic evolution of a training course. Consequently, all elements that have been described encouraged us to apply the standard IMS LD in the creation of educational Web sites.

We have designed an online guide called CEPIAH in order to help teachers to implement pedagogical web sites and to produce on-line courses. This system is composed of three modules: Help in the Design, Help with the Evaluation and Assistance with the development of online course structures. After having integrated the first two modules into the CEPIAH guide (Trigano and Giacomini, 2004; Giacomini and Trigano, 2003), in the third module we were interested in assisting the design and development of educational Web sites structures. We developed a Web portal, named netUniversité, whose general architecture is represented in figure 1. Using this application the teacher can automatically generate educational Web site structures, adding the pedagogical contents in these structures, then visualize, manage and participate to his courses. The students can visualize and participate to the courses using the navigator integrated in netUniversité.

These structures are generated starting from answers given by the user (the teacher) by responding to two interactive questionnaires: a pedagogical questionnaire and a GUI (graphic user interface) related questionnaire (cf. figure 1).

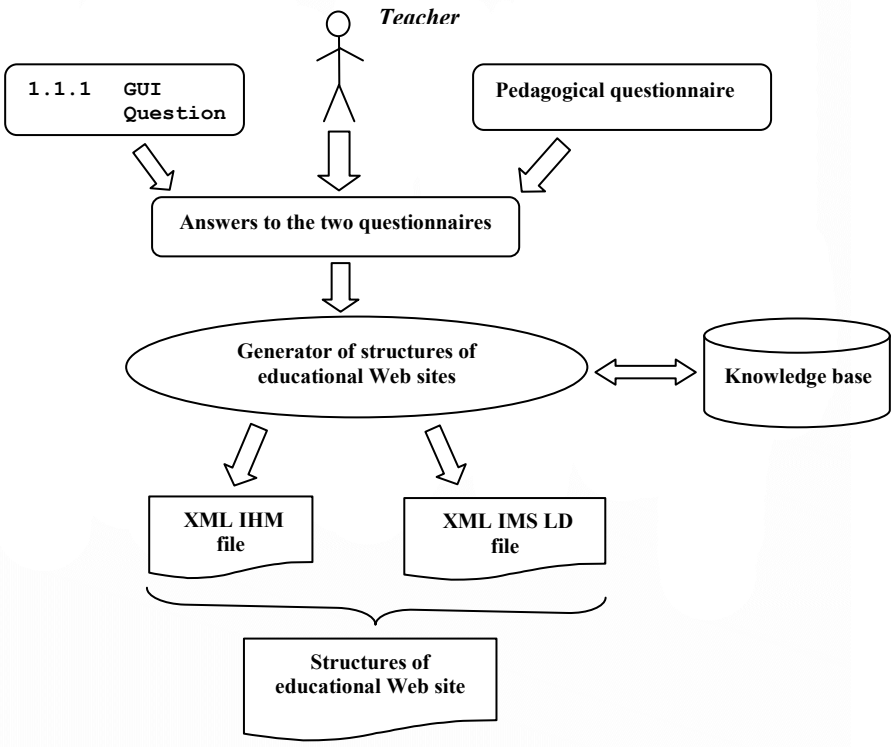

Figure 1. The general framework of automatic generation of educational Web sites

These two interactive questionnaires are represented in XML files. Each generated Web site structure is represented by two XML files, one for the graphical interface models and another for storing the pedagogical content organization in IMS LD format. The problem of the complexity of the conditions used for the automatic generation led us to a knowledge-based system. The number of entries (represented by the possibilities of response to the questionnaires) and the great number of pedagogical course structures obtained resulted in the development of a rules-based system (using JESS). In order to generate the pedagogical

web sites structures we use a list of primary pedagogical models (i.e., basic models) also named *building blocks*. These building blocks are used to build the final models (structures of course generated). The inference rules in the knowledge base specify the way in which these elementary bricks are associated. These last are components IMS Learning Design, which can be create by default at the time of the generation or conceived beforehand and recorded in files XML that respect this standard.

We don't present here a more detailed description of the principle of generating educational Web sites structures because our goal is to describes the functionalities of an IMS LD graphic editor using a representation by graphs (representation by threes) in order to modify the course structures.

After generating the course structures, the teacher has the possibility to administer and edit the course: adding participants to the course, create/edit pedagogical resources (documents like pdf, word, html), modify the course structure etc. The modifications related to the pedagogical structure and content are realized using an IMS Learning Design web editor. This editor supports the full IMS LD specification (levels A, B and C). The main weakness of a web editor (like ours) is the lack of a global view of course structure. These aspects prompted us to create a graphical editor that offers to the teacher a perspective on the IMS LD document with the possibility to perform some basic editing operations (adding, moving, deleting) in order to complete the web editor.

Therefore, the graphical editor is included in the web editor and can be accessed through a

link located on the web editor page.

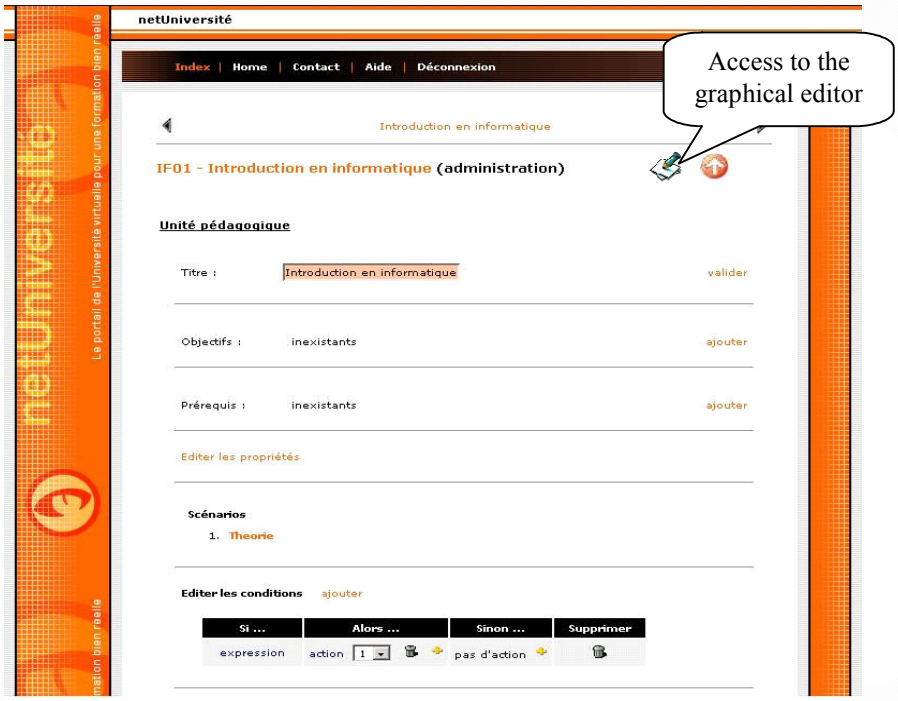

Figure 2. Accessing the graphical editor from the web editor

The graphical editor uses the IMS LD document generated after completing the questionnaires (see figure 1) and doesn't run independently of the web editor (it doesn't allow to import an IMD LD structure created outside the netUniversité system).

2. The aim of the Graphical Editor module

Teachers generate courses using netUniversité. Once a course is generated, its structure cannot be modified. We would like to give the possibility to change the order of the activities, to add or delete them. As the project addresses teachers who do not have experience with computers, we have decided to set up a graphic editor based on tree-representation (or graphs representation), which should help them to modify the IMS LD course structure. We think that this kind of presentation makes their work much easier (accordingly with Paquette, Marino, De la Teja, Léonard & Lundgren Cayrol (2005) as it

provides, at the same time, an overview of the course.

Currently, there is another graphic editor based on IMS LD, called MOT+ (Paquette, De la Teja & Léonard, 2005). MOT+ is a very interesting application designed to represent several types of models, the options being: standard, flowchart, educational and ontology. However, MOT+ is a stand-alone software application while our editor is a client/server application. Paquette, Marino, De la Teja, Léonard & Lundgren Cayrol (2005) also describe their Instructional Structure Editor which allows users to build a tree-based representation of an LD method.

Our graphic editor allows the modification of the structure but not that of the content. All the required resources will be introduced in the server through our IMS LD (levels A, B and C) course content editor, which is integrated in netUniversité.

3. Description of the module

The module is implemented as a Java applet, chosen to support client/server applications. This is very important as the file that is to be modified is on the web server. The communication between the applet (client) and the server (also written in Java) is described in figure 2.

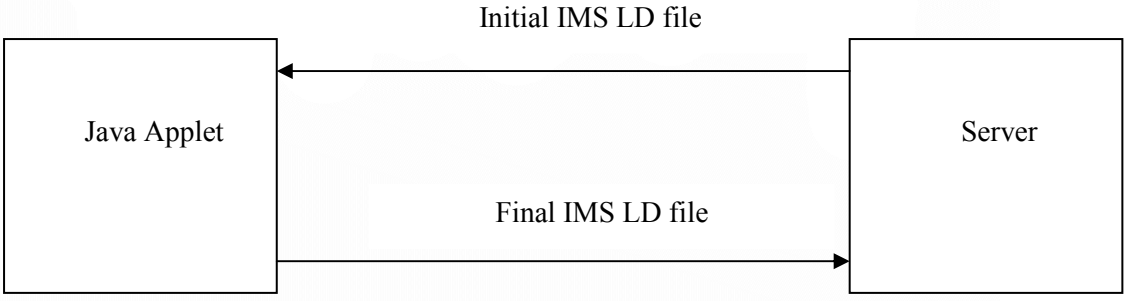

Figure 2. Communication between the applet and the server

Initially, the client application requests the file containing the course from the server application. Once received, the content is displayed in the form of a tree (see figure 3), where it can be modified, before the final version of the file is sent to the server application which saves it on the disk.

The file is sent as a URL item. We prefer this form because it may be serialized and it eases

the communication. Practically speaking, we ask it to send the item and to read it at destination.

After the teacher has modified the structure of the course, it is sent back to the server application in a similar way, the only difference being that the positions of client application and server application are reversed.

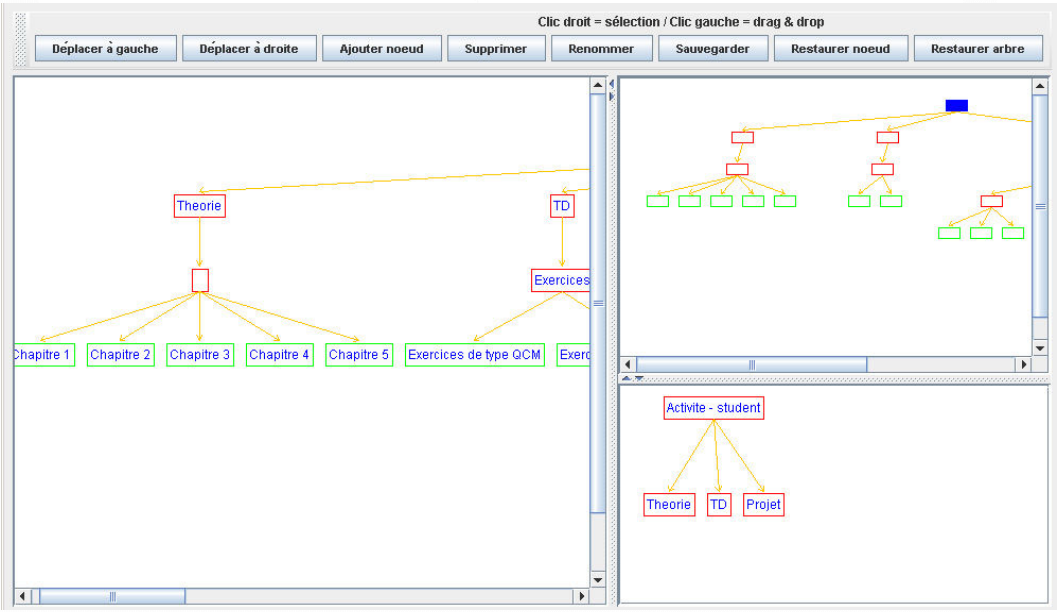

Figure 3. An example of the graphical editor

3.1 Loading and saving course structure

The file containing the course structure resides on the server. It is necessary to load it, make changes and then save on the same location. The security restrictions for applets are very severe: they are not allowed to write files on any disk (local or remote), through network they can connect only to the server from which they have been downloaded. We can now see a first problem: how do we save the file back on server? The solution used was to write another module, a server developed in java that will run on the web server. This will be used for 2 things: obtaining the file location and saving it in the modified version.

First step is to read the file. For this we need the exact location in the form of an URL. Due to security reasons we preferred not to send it as an applet parameter in the file from which it's being loaded. The solution adopted is to receive an ID which will be sent to the server. Once the ID received, the server will check in the database and return the necessary information.

3.2 Building the tree

For each role we have a tree but we shall merely discuss how to make one of them. The principle is the same, the difference being that we search for activities belonging to another role. We now describe the file structure used (figure 4). This figure shows the tag <learning-design> containing more tags and attributes.

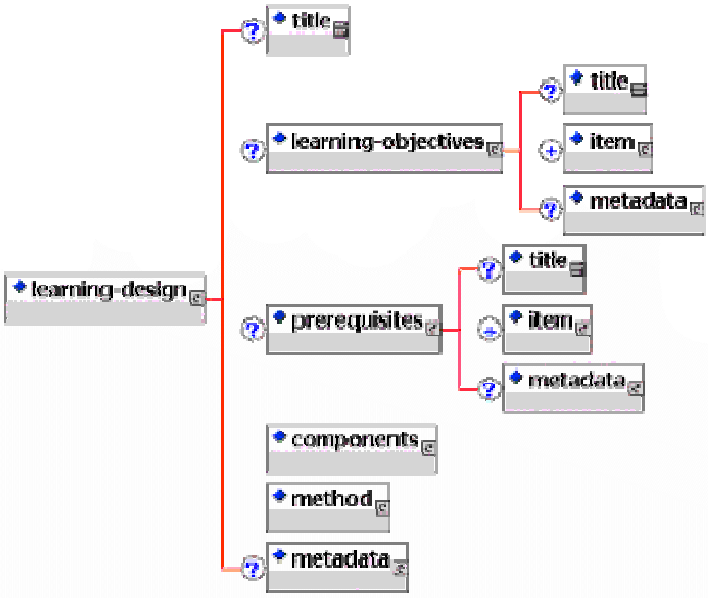

Figure 4. The LD used by graphic editor

We are interested only in <components> and <method>. In <components> we have useful

information on the roles and activities. In <roles> the roles are presented in this course and in <activities> all activities are defined as we see in figure 5.

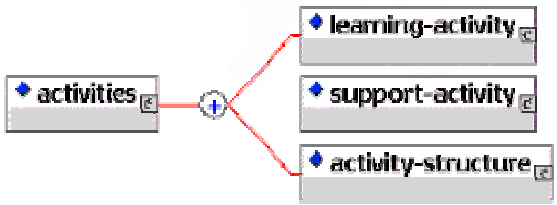

Figure 5. Activities used by the graphic editor

An activity structure can contain references to an undetermined number of activities or activity structure. In <method> which is defined in <learning-design> we have the way activities are passed.

Each <play> is a pedagogical scenario. Plays are divided one or more acts, each act being made of one or more role-parts in which are the references to activities. At the role-part level the role is defined for which that activity is a part of.

The root node is generic, being used only to show that role's activity. Its children are of Play type. Starting with level 3 all nodes are of Activity Structure, Learning Activity or Support Activity type. The firsts 2 levels are easy to fill, the first is static and for the second we look in <method> for plays containing activities belonging to that node. The rest is filled starting from the role-parts level. If we have reference to an activity then we shall add it, as child, in the corresponding node with information of play type. If it is an activity structure we shall go through each reference in the order in which they have been defined. For each activity structure we are doing the same thing.

3.3 Drawing the tree

We describe the drawing process only for the view with full tree since it is very similar in the others. For the view with empty tree, the difference is that we don't care about text size, all nodes being identical. In the detailed tree view we shall have the same sizes as in the full

tree view but we show only a small part of it.

First of all we have to compute the necessary width in pixels for drawing the tree. We shall make a recursive cross of the tree starting with root. The width of each node is determined by the maximum between its width and the sum of its children's widths plus the necessary space between them. As a result we start computing from the lowest level and advance to the root. The method guarantees there are no errors in computing as the width of the children is always computed before the parent's.

The width of each node is made by the width of the text in the window where it will be drawn to which we add a space between the text and rectangle's border that is surrounding it. The height is computed in the same way. Consider W the width of the node. Between the nodes on same level there is a separation space marked with S, the width of a node is given by the following formula:

$$W = MAX(W + 2 \cdot S, \sum_{i=0}^{nr.fii} W_i \cdot (nr.fii + 1) \cdot S)$$

W_i is the width of child i.

Space S exists between the nodes and also between them and the drawing surface border (or the nodes on same level, depending on situation).

The necessary width is computed and we can now move on to drawing it. This will be also done recursively starting from the root. In any moment we know the distance to the left border (of drawing surface). Starting from that point we draw one by one the nodes that have a common parent. Each time we go to the next one we add a node's width and space between them to the distance from left border. The only exception from the rule is root which is going to be centered in the space allocated for the whole tree.

3.4 Adding new nodes

There are 2 categories of nodes that can be added: play type and activity type. Those of activity type can be a simple activity or an activity structure. The nodes of play type can be added only to the root node, the others on any level equal or higher than 3.

When a new play is made we must also specify the position where it is inserted. We are not able to change in the future this position because a play can contain activities from different roles. If we wish to change the order only for one of them it would be necessary to split the

play in 2 new plays, situation that is unlikely to be wanted by user and also can't be controlled. In the dialog used for adding a new play we must specify the relative position to the existing ones: before or after one of them. Depending on this, an empty play will be inserted in <method>.

It is possible to add nodes with type Activity Structure (AS), Learning Activity or Support Activity only in nodes of AS type. This restriction is from IMSLD standard. The operation is done in 2 steps: make a new activity in the <activities> tag, add in corresponding AS on last position a reference to it. Positioning will be done later using the toolbar buttons or drag & drop.

Another aspect that must be catered for is that we have more roles. At this moment the navigator doesn't allow an activity to be part of more roles but the editor is already made with possibility to manipulate this kind of structures. This is important for us, the SA in which we insert can be in more roles. In this case we must make the change (on screen) for all of them by searching in all trees and where is found make the insertion.

3.5 More options

Renaming activities is made by simply changing the corresponding <title> tag. It is allowed for all nodes except root.

4 The user interface

4.1 General presentation

The figure 6 is a screenshot that represents the graphical editor module. In the upper part we have a toolbar, providing support for all operations. The buttons from left to right are: *Deplacer a gauche* - moves selected node one position to the left; *Deplacer a droit* - moves selected node one position to the right; *Ajouter noeud* - adds a new node; *Supprimer* - marks the selected node for deleting; *Renommer* - renames the selected node; *Sauvegarder* - save the current configuration; *Restaurer noeud* - undelete the selected node; *Restaurer xarbre* - undelete the selected node and all its sub-trees.

In the lower part there are three views:

- To the left is the main view, here the whole tree is represented. For each role we have a tree. We shall explain what the tree represents in the next paragraph.
- In the upper right part is the empty tree view. The difference compared with the

previous one is that all nodes are empty and we display only one tree.

- In the lower right part is a view with only a node and his child. When we select one in any of the previous views it will appear here as a root and all his children will be shown.

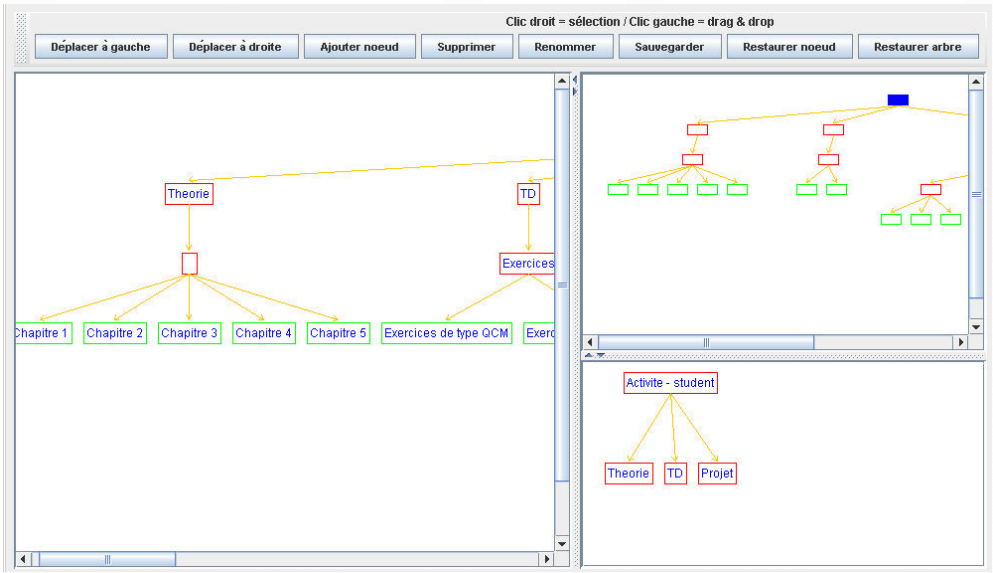

Figure 6. The graphical editor module

These trees represent a course structure. The root has a generic name; it's the activity for that role. To understand how a student will go through the course we read the tree from left to right. The second level in the tree is made by plays. They are a way to organize the course presentation. In the next part we provide more details about the buttons on toolbar:

- *Deplacer a gauche* and *Deplacer a droit* moves the selected node one position to the left or right.
- *Ajouter noeud* adds to the selected node a child. If the selected node is the root one we can add only a new play. If it's a play or activity structure the options are "Learning activity", "Support activity" and "Activity structure".

- *Supprimer* marks a tree and it's sub-trees as deleted like in the next figure (see figure 7).

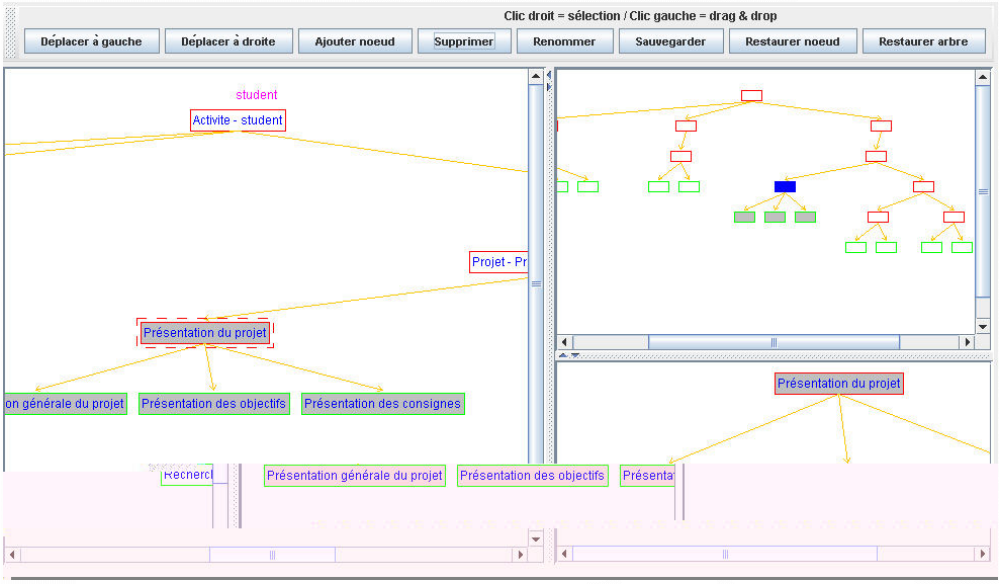

Figure 7. Deleted trees and sub-trees

When we save the file, all information marked as deleted will not appear in the file. In case that we change our mind about the deleted nodes there are 2 buttons that can help: *Restaurer noeud* and *Restaurer arbre*. The first one will mark as undeleted the current node and all it's parents, the second will do the same as the previous, but will also mark as undeleted all his children. *Renommer* renames the selected node and *Sauvegarder* saves the configuration.

Another way to change the course is by drag&drop operation (see figure 8).

While moving the node we compute in each moment the distance to all other nodes. The destination node is the closest to the one we are moving. The distance is based on the next formula which computes the length of the segment with end points at coordinates (x1, y1) and (x2, y2):

$$d=\sqrt{(x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)}$$

We can select a node with the left button and while keeping it pressed move it to the new desired location. In every moment we can see the position where it will be placed. In the next figure (figure 8) we can see how the node is being moved.

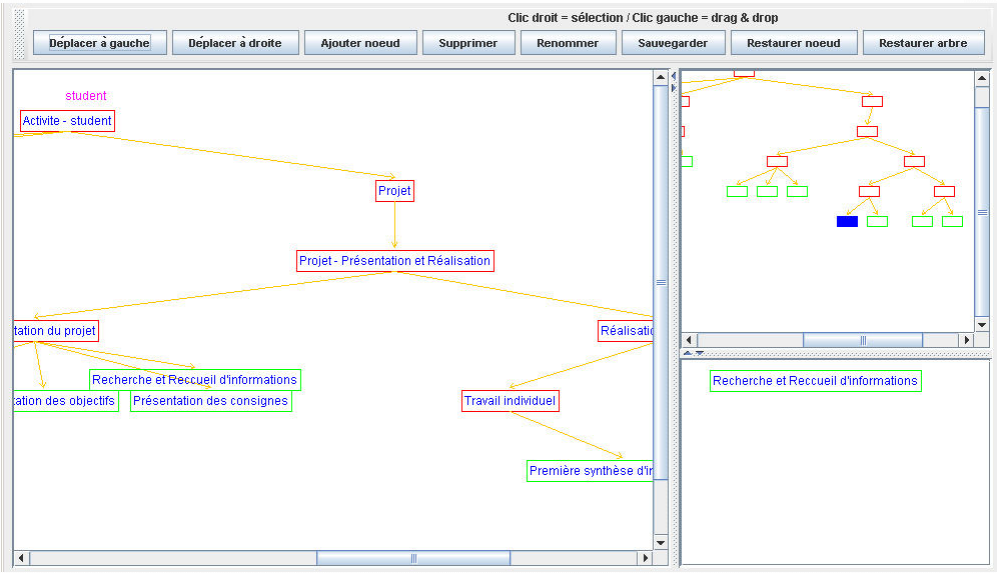

Figure 8. The possibility of moving a node

4.2 An example illustrating the modification of a course structure

As we already specified at the beginning of sections 2 and 3, the teacher can edit the contents of the generated course structure by using an IMS LD editor. Nevertheless, once that the course is created, the teacher can consider that certain modifications are necessary. Figure 9 illustrates an example of a course structure generated by the system.

This structure is built from two "Theory" plays (composed of activity structures called Chapter 1, Chapter 2) and a "Problem solving" play (or TD, composed of activity structures called TD 1, TD 2). Using the graphical editor the teacher can add for example, at the same

level in the tree structure, a new play "Project". In order to effectuate this operation he must choose the option "Add node" (Ajouter nœud). Therefore, using the dialog window (see figure 9) the teacher has the possibility to edit the name of this new element and also to choose its position within the list of existing plays.

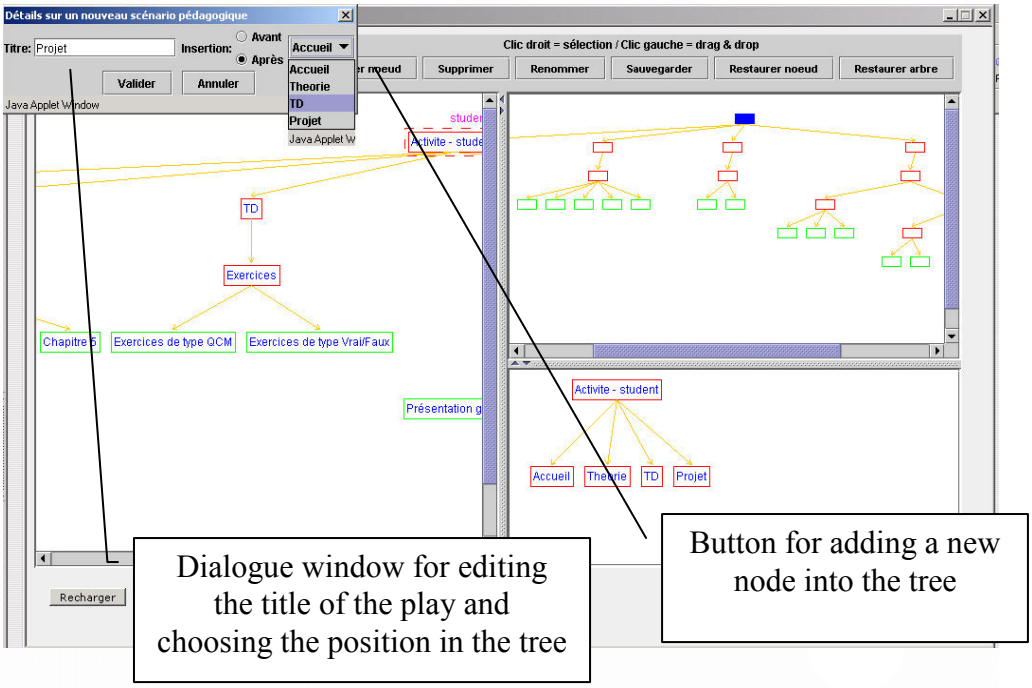

Figure 9. Example of modification of a course structure by adding a new play

In this structure the teacher can also add activity structures and learning activities. For example, in figure 10 we can observe that a new activity-structure node named "Chapter 3" is added. It is composed of an activity called "Definitions and explanations" as well as an activity called "Examples". By following the same principle of the course structure construction, the teacher can integrate learning activities composing this structure.

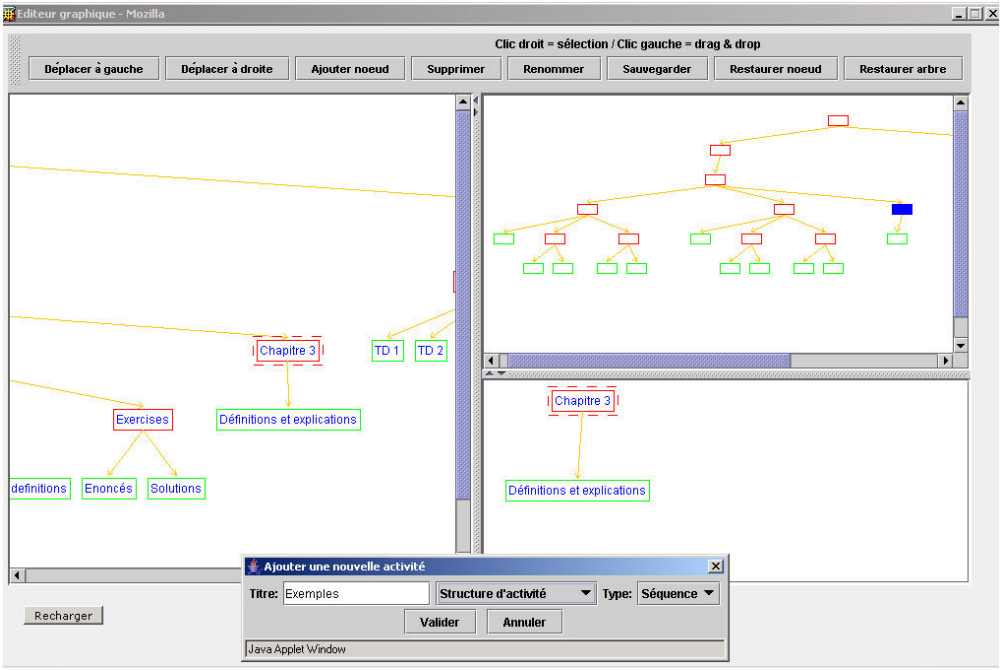

Figure 10. Example of insertion of a new « activity-structure » node

5. Conclusions

In this article, we presented our research works concerning the design and the development of the netUniversity Graphic Editor. The perspective of this application is to allow the import to the IMD LD structure created outside the netUniversité system.

We continued the development of this first version of the prototype by adding the functionalities for the integration of the teaching resources (interactive exercises, tools of communication personalized according to types of activities suggested in the generated courses). These new functionalities will allow us to generate structures of educational Web sites based on even more interactive scenarios. We think that this interactivity as well as the

addition of the communication tools in the on line courses could motivate more the students in the completion of their training tasks.

Acknowledgements: This thesis benefits from the financial support of the Pole of research STEF (Systems and Technology for Education and Training) of the Picardy Region, within the framework of the State/Region plan.

6. References

ADL (2001). The SCORM Overview : Sharable Content Object Reference Model Version 1.2. Advanced Distributed Learning Technical Team.

Friesen, N., Roberts, A. & Fisher, S. (2002). Cancore: Learning Object Metadata, s.l., Canadian Core Learning Resource Metadata Application Profile, 17 p.

Giacomini E. & Trigano P. (2003). *Flexible navigation for the pedagogical hypermedia design and evaluation improvement.* In proceedings World Conference on E-Learning in Corp., Govt., Health., & Higher Education, 206-212.

IMS (2003). IMS Learning Design Best Practice and Implementation Guide. IMS Global Learning Consortium, Inc.

IEEE (2001). (IEEE P1484.3 GLOSSARY WORKING GROUP). IEEE LTSC Glossary, draft standard, copyright 2001 by the Institute of Electrical and Electronics Engineers, Inc. IEEE Learning Technologies Standards

Paquette, G., Marino, O., De la Teja, I., Léonard, M. & Lundgren Cayrol, K. (2005). Delivery of Learning Design: the Explor@ System's Case, Chapter 20, in *Learning Design: A handbook on Modelling and Delivering Networked Education and Training*. Rob Koper & Colin Tattersall (Eds.), ISBN 3-540-22814-4 Berlin Heidelberg New York: Springer.

Paquette, G., De la Teja, I., Léonard, M. (2005). An Instructional Engineering Method and Tool for the Design of Units of Learning, Chapter 9, in *Learning Design: A handbook on Modelling and Delivering Networked Education and Training*. Rob Koper & Colin Tattersall (Eds.), ISBN 3-540-22814-4 Berlin Heidelberg New York: Springer.

Trigano P. & Giacomini E. (2004). Toward a Web based environment for Evaluation and Design of Pedagogical Hypermedia. In journal of *Educational Technology & Society, IEEE Learning Technology Task Force*, Vol. 7, n° 3.