Intelligent use and Reuse of Computer-based Learning Material for CSCL

Nelson BALOIAN*1, Nelson VIDAL*1, Alfredo DIAZ*1, Eduardo S. VERA*1
Ittsuki YASUYOSHI*2, Mitsuji MATSUMOTO*2, Toshihiko WAKAHARA*2

*1 FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS, UNIVERSIDAD DE CHILE
Av. Blanco Encalada 2120, Casilla 2777, Santiago Chile
E-mail: mbaloian@dcc.uchile.cl

*2 Global Information and Telecommunication Institute, Waseda University
1-3-10 Nishi-Waseda, Shinjuku-ku, Tokyo 169-0051 Japan
E-mail: matsumoto@giti.waseda.ac.jp

Abstract Developing quality may be the most difficult and costly task of developing a computer based courseware. Despite this, little has been done to support the reuse of multimedia learning material in a coherent and systematic way. In this paper, we propose a particular way of structuring a courseware to conform Lessons, Units and Courses in order to facilitate the use of the same material in different situations as well as the use of the material in other lessons. This is based on structuring a lesson's computer-based learning material as a "Didactic Network". A Didactic Network is a semantic network which represents the lesson. Although this approach was not developed exclusively having the wireless and mobile scenario in mind, it is a good example of developing courseware which should be used in different situations, including the mobile scenario.

Keyword Courseware, CSCL, Computer-based Learning
1- Introduction

It is easy to imagine, that one of the most important challenge when developing electronic learning material for courses which will be supported by computer technology is the production of the material itself. This is especially true if we want to create a high quality course whose content takes the maximum advantages of today’s computer capabilities. Because of this, we would like to have some mechanisms that help us to reuse the sometimes expensive developed material. There have been made some efforts at the level of the software components, in order to reuse them, using some of the principles of the object-orientation strategy, as it has been done with development of hardware. However, in the field of courseware development we do not see an effort to develop a similar approach. Although there are some systems which may use a “component-based” courseware structure in which a www-page or a certain type of multimedia document is the minimum component which may be used in different lesson units they do not allow to group a set of documents and regard them as a higher level component (which would be similar to new objects which are composed from a set of other objects) and they make little or no use of the information about the learning context in which these components could be reused (which would be similar to the interface specification of an object).

Today there are many commercial and non-commercial system which help the author of a courseware to create, maintain and administrate a courseware, like WebCt, Luvit, TopClass, Learning Spaces, Authorware and CalSurf. The different systems have they strengths and shortcomings in different areas which is a reflection of the learning model the creators of the systems had in mind while developing the system. Some products have their strengths in the support and management of the communications between students and teacher as well as among students themselves like Learning Spaces, WebCt and TopClass. Others are better in the management and delivering of the courseware content, as well as monitoring the student’s activity, informing the teacher what has the student already seen and where the results of the tests of the different units (Authorware and calsurf).

From the point of view of the lesson’s structure and the intelligent use of the information about the lesson’s structure and the learning context in which the educational material should be used we can mention the CalSurf. This system makes some intelligent analysis of the lesson’s structure interpreting the requirements and goals of each lesson unit and it creates a learning path, but fails to give a strong support in the reuse of parts or the whole lesson in other contexts. Many systems implement the lesson structure as a strictly linear succession of “pages” (like www-pages or other kind of multimedia documents). The most developed lesson structure we may see is a tree. But all these lack powerful edition tools which may help the author to shape better the structure of a lesson.

We can imagine at least two ways in which a courseware can be reused in a more elaborated and effective way than:

a) By using the same lesson in different situations. This means that the lesson’s structure should have the sufficient flexibility to allow the use of the material in one order or another, to use all or only a part of the material, but always maintaining a coherence on the whole lesson. According to the situation and the user’s preferred learning strategy, the structure’s information can be used to recommend a certain learning path, selecting which learning material should be visited and in which order.

b) By using part of a lesson to develop another one. This should work like a copy-and-paste facility, just like one author would use a sentence or a paragraph of one document as part of a new one. For example, in the case when the lesson is represented by a tree we would like to copy a branch and paste is to another lesson being developed.

In the rest of this paper we will present following issues: Point 2 describes a way to structure the courseware syllabus called “Didactic Network”. Point 3 describes the authoring tool for Didactic Networks and how to exploit the lesson’s structure in order to use it as part of another lesson in a coherent way. Point 4 describes how to use the same courseware in different situations.
2- Didactic networks for structuring computer based learning material

In the traditional intelligent tutoring systems (ITS) the curriculum structure was used to represent the knowledge the student has to acquire and in which order. It serves as the knowledge representation the system has about the domain being taught and it is used also as a basis for modeling the student’s knowledge. The curriculum structure was typically represented as a graph in order to achieve flexibility in the order concepts could be learnt by the students but maintaining the coherency about which concepts should be learnt first [McCalla, 1992]. In these graphs each node represents a concept or a subject and the link represents how these concepts are related to each other. Graphs of this kind have been also been called concept graphs [Novak, 1990]. The relationship often imposes a partial order among the nodes defining the sequence in which concepts can be learnt. The different ways the graph can be traversed represent different learning paths. A certain learning path may suit better one student than another. Systems based on a graph-like curriculum structure can use the information contained in the nodes and links to assist the learner with some intelligence.

In the context of designing curricula for adaptive and intelligent computer tutors, Halff defines the curriculum as the selection and sequencing of learning material [Halff, 1989]. McCalla redefines this as the selection and sequencing of knowledge, which are adequate for the learner in order to meet certain learning goals according to his background and learning possibilities [McCalla, 1992]. McCalla also distinguishes two phases in the development of the curriculum. The first one is the so called generic curriculum, which consists of the selection of material and learning activities. This is done by specialists in the learning domain with a prototype student in mind. The second phase consists of tailoring the curriculum designed in the first phase to fit the learning needs and other requirements, such as a preferred learning or teaching strategy, or a particular learning context (background, learning possibilities, etc.) for a specific student group. The second phase is performed by the teacher in the classroom. If a curriculum is represented by a graph, it is easy to draw a parallel between the first phase of the curriculum development with the construction of a graph and the second part of the curriculum development with the use of this graph as a basis for performing a lesson in the electronic classroom with computer based learning material.

In this work, a model of a curriculum representation for courseware based on a graph is introduced. This has been called a “didactic network” of a lesson or simply the lesson graph, and the path followed to perform a lesson is called the “learning path”. In the next two sections the characteristics of the nodes and links of a didactic network are described.

2.1 The nodes

In a didactic network, a node represents an abstraction of a subject that is to be taught and learnt. Nodes are labeled with a name which is given by the author of the lesson graph (the curriculum designer). This name should be a mnemonic for describing the subject represented by that node. Nodes are also typified, this means, a node can be only of a type chosen from a pre-defined types set. The type of the node represents the learning activity which should take place when visiting the node. For performing the learning activity, teacher and students use (also) the computerized learning material which is associated to the node. The node types are the following:

<table>
<thead>
<tr>
<th>node type name</th>
<th>activity involved while visiting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphic presentation</td>
<td>for presentation of mostly graphic information to the students</td>
</tr>
<tr>
<td>Animation presentation</td>
<td>for presentation of animated &amp; interactive simulation programs</td>
</tr>
<tr>
<td>Audio Presentation</td>
<td>for presentation of mostly audio type information to the students</td>
</tr>
<tr>
<td>Video presentation</td>
<td>for presentation of video clip to the students</td>
</tr>
<tr>
<td>Test presentation</td>
<td>for presentation of mostly textual information to the students</td>
</tr>
<tr>
<td>Discussion</td>
<td>for a group discussion</td>
</tr>
<tr>
<td>Individual work</td>
<td>the main activity is the individual work of the students</td>
</tr>
<tr>
<td>Group work</td>
<td>for a structured collaborative work (e.g. collaborative problem solving)</td>
</tr>
</tbody>
</table>

By typing the node according to a learning activity, the author of the lesson graph must first think about the subject and the concrete activity through which the subject will be taught/learnt and afterwards about the computerized learning material required. Moreover, the computer material can be associated to the node long after its creation. This also helps the author to clarify his ideas about the dynamics of a lesson based
on this graph. These types of nodes may also give some hint about the time needed to teach the subject. For example, it is quite likely a video clip or an audio presentation will take less time than a discussion, individual work or group work. Because in this paper we will present only a model of the didactic network, we will not discuss about the association of learning material to the nodes. This practical implementation of didactic networks for the electronic classroom are described in the following papers [Baloian, Hoppe & Kling ; 1995], [Baloian, Gassner & Hoppe ; 1997], [Baloian, et.al ; 2000].

2.2 The links: The links in a didactic graph are directed and labeled. The label establishes a relationship between the subjects represented by the nodes. The labels are also typified in order to make an intelligent guidance to the users possible, especially during the lesson development. The typing also allows the user of one of these graphs during the lesson to unequivocally understand what the author had in mind.

The link types to be provided depend on the intended system support, i.e., the guidance functions the system will offer to its users. Intelligent systems which use a graph for knowledge representation tend to use semantic links [Chaffin & Douglas, 1984]. Rhetoric links [Trigg, 1983] on the other hand, describe the strategy or discourse used by the author to present the information represented by or contained in the nodes. Because the aim of our system is to support the authoring and presentation of learning information and not to give help based on the knowledge acquired by the learning group, it seems to be natural to use rhetorical links for constructing didactic networks.

The set of rhetoric links presented by Trigg will be modified and complemented in order to provide the curriculum designer a set of relations for constructing the lesson graph according to a preferred presentation strategy. The following table represents the relationship established by a link which points node Y from a node X.

<table>
<thead>
<tr>
<th>link type name</th>
<th>represented relationship</th>
<th>intended usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>X introduces to Y</td>
<td>X introduces the subject Y</td>
<td>recommended for the beginning of a lesson and introduction of new topics</td>
</tr>
<tr>
<td>X refined by Y</td>
<td>the subject Y is a part of the subject X</td>
<td>may be used to split a topic in various subtopics.</td>
</tr>
<tr>
<td>X explained by Y</td>
<td>the subject X is explained more deeply by the subject Y</td>
<td>may be used to justify or support the idea of the predecessor.</td>
</tr>
<tr>
<td>X exemplified by Y</td>
<td>Y is an example of the subject X</td>
<td>may be used to illustrate the idea of the predecessor's subject</td>
</tr>
<tr>
<td>X summarized by Y</td>
<td>Y is a summary for all the nodes linked by this type of link, having Y as successor</td>
<td>there should be normally more than one predecessor linked with this type of link.</td>
</tr>
</tbody>
</table>

Fig 1: An example of a didactic network for a lesson on Data Structures and Algorithms

2.3 Formal description of a didactic network

Definition: a didactic network is a directed graph G=(V,E) in which:
- $V$ is a non-empty set of nodes $\{v_0, \ldots, v_n\}$ where $|V| = n$.
- $v_i$ is called the root of the graph and it is the only one which has no predecessor.
- $E$ is a set of labeled links. The type of label must be of a pre-defined set. The link which starts at
  the node $v_i$ and ends at the node $v_j$ is noted as $e_{ij}$.
- There is a function $D$, which transforms the label of a link in a numeric value. This value is
called the distance of the link and is noted as $D(e_{ij}) = d_{ij}$.

This is of course an abstraction of a didactic network since many aspects are not considered here like, for
example, the contents of the nodes. The given definition is not enough to assure every node $v_i$, $i=0..n-1$ can
be reached by traversing the graph starting from $v_0$, which is a very important condition in these graphs (see
Fig. 2). It would be desirable to find a condition assuring this and, at the same time, be easy and fast to
determine during the construction of the graph. The following condition can be proved to be necessary and
sufficient:

$B1$: For any partitioning of $V$ into two subsets $V'$ and $V''$ with $V' \cup V'' = V$, $V' \cap V'' = \emptyset$ and $v_0 \in$ $V''$ there is at least one link $e_{ij}$, where $v_i \in V''$ and $v_j \in V'$.

This condition can simply be checked during construction of the graph by first creating the starting node.
Then, the authoring system for the graph should check any just created node is linked to at least one node
of the reachable set of nodes.

We have already defined the structure that a lesson will have. Now we are going to explain how can this
structure help to reusing the learning material. For this we will have to talk about the systems which
implement these ideas. We can easily see that there should be two different systems: The lesson authoring
system and the lesson navigation system.

3. Using the graph structure to support lesson authoring and reusing parts of a lesson in another

In order to take advantage of the structure of a lesson we have to include this goal in the design of the
authoring tool which will allow authors to create Didactic Networks which represent a lesson, a learning
unit or a whole course. It is indeed in this activity when authors can reuse parts of already created networks
to create new ones. The best way we have imagined how the construction of a Didactic Network should
look like in order to achieve our goals was permitting an author to "paint" the graph in a specific separate
working area (window) for each Didactic Network. The drawing should be based on direct manipulation of
objects on the window which permit the creation. Nodes and links by just selecting their type and
positioning them in the working area. Contents should be added and modified by selecting functions from a
pop-up menu which every node has. We think that the construction of a Didactic Network in this way has
many advantages:

- Authors can always have a global view of the structure of the lesson, unit, or course they are creating.
  If the structure is too big, reduced and partial views can be provided.
- Copy-and-paste operations involving different graphs can be easily performed and implemented.
- We separate the problem of deciding how to structure and sequence the goals of a lecture from the
  problem of deciding which learning material we will use for that.

The design of authoring tool, which we are developing, was taken from the SEPIA authoring tool system.
SEPIA is a collaborative authoring tool for multimedia documents. The structure of the multimedia
documents in SEPIA is also a graph and the authoring tool requires the authors to first define a node of the
document before they can assign the multimedia content to it. The nodes can be placed anywhere in the
working area so the author can choose the arrangement which makes the graph look clearer. We are
including all these features in the Didactic Network Authoring tool, which make the tool very user friendly
and easy to use. It is important that the authoring tool permits the user to have more than one window
corresponding to the working area of different Didactic Networks because only in this context that
operations like copy-and-paste from one network to another can be performed.

We have defined three levels of grouping and composing graphs:

- The lesson: the first level is called the Lesson and is the grouping of nodes and arcs in a Didactic Network describing the goals and containing the learning material for a single lesson, which is intended to be used in one session. The Lesson has a name and learning goals defined.
- The Unit: the second level is the Unit, which is a grouping of two or more Lessons which are related in their content. These Lessons may also be related to each other as a graph, being the each Lesson a node of the Unit. Every Unit has a name and learning goals defined.
- The Course: the third and higher level of gouging is Course which groups several Units which are intended to form part of a regular course. A Course has a name and learning goals defined.

These grouping does not define another structure. In fact, authors must not group the nodes of a Didactic Network to form Units and Courses if they don't want to, but by doing it, they can take more advantage of the reusing possibilities of the graph. The authoring tool permits the reuse of the material at various levels: On a basic level, we can copy one node of one lesson (including its content) to the Lesson graph of another lesson. We can also copy a group of nodes with the links defined among them and paste them to form part of another Lesson graph. On a higher level, we can copy an entire lesson or a group of Lessons. Finally, at the highest level, we can copy and paste Units from one lesson to another.

4. Using the graph for supporting different learning strategies and situations with the lesson navigation system

After creating a Course, it must be published. To take advantages of the structure of the graphs defined as Lessons, units and Courses we must implement a special navigation tool, because the HTML links are not labeled. Therefore we must implement a new navigation tool which is aware of the labels on the arcs and can take advantage of this. To explain this we must note that the graph has important information about the Lesson, Unit or Course being modeled. It is possible to use it as a basis for supporting the teaching/learning process. There should be a mechanism to guide users traverse a didactic network during a lesson. This mechanism should be flexible enough to coach users to choose the best path according to their needs and the dynamic nature of the lesson. Therefore, the guidance should come as an answer to a user's requirement. There are two basic functions which may give a good advice in many possible situations when guidance is required: a) displaying the graph in a way that highlights the possible lecturing threads b) answering to the question „which is the best next node to visit ?“ Both functions should take into account the user's profile.

The tree of a lesson graph

It is desirable to provide teachers with the possibility of visualizing the lesson graph at any time during the lesson development. The user interface must allow this in a quick and easy way, for instance, by clicking over a button. The view of the lesson graph must make visited nodes, not visited nodes, and the current node clearly distinguishable from each other. It should be also possible to jump to any node just by clicking over it in the graph representation. But the drawing and presentation of the whole graph is not an easy task. The already mentioned requirement to highlight the possible lecturing paths according to the user's profile makes the task even more complex.

A good alternative to display the whole graph is to show a spanning tree of it. It has many advantages over the graph:

- A tree is easier to draw than a graph with a general algorithm.
- The learning material is presented in an „extended“ form, where paths and endpoints of the lesson are easy to distinguish.
- It is even possible to show all the incoming and outgoing links for the current node in a separate window in order to have a better assessment of the situation.
- We can expect the structure of a didactic network representing a lesson or learning unit will have a structure similar to a tree. In fact, most printed learning material comes in the form of a tree with title, subtitles, chapters etc. and people tend to adopt this structure while organizing digitized information. Cross references from one part to another are added afterwards [Wong & Chan; 1997].

The question about how to automatically generate the spanning tree of the lesson graph arises because there
are various ways to construct it. The best known algorithm for constructing spanning trees of a graph are the „Kruskal“ and „Prim“ algorithms. We consider at least two meaningful criteria for constructing the spanning tree in this case:

A) „Good looking“ tree: Our aim is to give a good overlook of the entire structure in this case. A good looking tree is a well balanced tree with a relatively short depth. If we follow the assumptions made by Wong & Chan about the structure of a lesson, we conclude a spanning tree with the shortest depth will be the best for overlooking the entire lesson structure, in order to avoid cycles introduced by cross references. This spanning tree can easily be generated by a breadth-first traversal of the graph, based on Dijkstra’s algorithm to find the shortest path from one node to all others.

B) „Most related concepts“: Our objective now is to have a better close-up view of the current situation. For the implementation of this approach, the distances defined over the label-types of the links are crucial because they define how closely related are two concepts to each other. Accordingly to this, a link-type which links to closely related concepts must be given a shorter distance. Prim’s algorithm for constructing a spanning tree is the most suitable here because it starts from a given point, which will be the starting point v₀ of the lesson’s graph.

It is not the aim of this paper to define a final function for the distance but to analyze the consequences of applying different functions and which functions perform well in representing and supporting a certain learning/teaching strategy. The following is an example how can values be given and what are the consequences:

<table>
<thead>
<tr>
<th>Link type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>introduces to</td>
<td>5</td>
</tr>
<tr>
<td>explained by</td>
<td>1</td>
</tr>
<tr>
<td>exemplified by</td>
<td>2</td>
</tr>
<tr>
<td>refined by</td>
<td>3</td>
</tr>
<tr>
<td>summarized by</td>
<td>4*</td>
</tr>
</tbody>
</table>

* this link is considered only if all the predecessors of the pointed node were already visited.

With this distance function, the closest node contains an explanation of the subject represented by the current node. If there are no more unvisited neighboring nodes which are an explanation to the current one, nodes containing an example of the current subject will be visited. After that, the nodes representing a sub-division of the current subject will be considered (with refined by). A node containing a summary of the current subject can only be considered after all other predecessors to the node have been visited. The nodes linked with introduced to have the lowest priority because these links can point to unrelated subjects.

**Implementation of the NEXT-NODE help function**

**A simple help**: A very easy to implement but powerful function to suggest navigation routes through the lesson graph is the automatic positioning of a „current node“ pointer at a node which has at least one non-visited son. The teacher may continue the lesson by choosing any of the non-visited sons or backtracking on the tree until eventually reaching the root, if desired. This is very helpful in the case of visiting a leaf or reaching a node whose sons were already visited by following another path. This can spare the need of pressing a „go back“ button while navigating through the network. It is easy to note that this function will leave no node unvisited, unless the teacher explicitly wants it.

![Fig 4: The graph to the left/right shows the pointer to the current node before/after applying the function. (Visited nodes are shown in black, unvisited in white.)](image)
Introducing teaching/learning preferences: With a combination of ideas of the most related concepts spanning tree and the positioning of the current node pointer at a node with at least one non-visited son it is possible to implement some interesting algorithms which support the traversing of the graph under different students' and/or teachers' profiles, or different requirements which arise from the teaching/learning context. The approach is the following: a spanning tree of the lesson graph is generated after the principle of the "most related concepts" but the table with the weight-function is given as a parameter, reflecting the teacher's and/or students' preferences. By calling the NEXT-NODE help function, the following algorithm is performed: if the current node has non-visited sons then select and return the one connected with the link which has the shortest distance. Else, go up to the father of this node and recursively apply the algorithm. This will indeed find the node with the most related subject to the current one, which has not been visited yet. This algorithm can be easily extended to construct a list with the possible next-nodes ordered by closeness according to the distances. Of course, if the function which assigns a distance to a link-type changes, the result of the help function also changes. Not only because the selection at the time of performing the help function will have different values to compare but also because the minimal spanning tree is generated in another way. Let us see now how we can implement the support to different learning strategies by changing the distance function.

a) Inductive Learning: Under inductive learning we understand the learning process follows an inductive strategy: this means individual examples and facts will induce the understanding of general rules and laws. This is why in such a learning mode the examples have higher priority than the explanations. This can be implemented by assigning the links labeled with exemplified-by a small distance. Table 1 is an example of implementation of this learning mode.

b) Deductive Learning: In a deductive learning mode the rules and laws are explained first and then individual facts and examples are deduced from them. This can be implemented by assigning the links labeled with explained-by a short distance. Table 2 is an example of this.

c) A "short version" of the lesson: In this learning mode only some of the nodes are visited in order to have an overview of the whole lesson without getting into details. In order to let some nodes out of the traversing path we can mark some links with an infinite distance. This means these links will never be selected during the construction of the minimal spanning tree. The nodes we can leave out of the lesson for a short version are the one linked with an explained-by and/or exemplified-by. Table 3 is an example of this.

<table>
<thead>
<tr>
<th>Type of link</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>introduces to</td>
<td>5</td>
</tr>
<tr>
<td>explained by</td>
<td>3</td>
</tr>
<tr>
<td>exemplified by</td>
<td>1</td>
</tr>
<tr>
<td>refined by</td>
<td>4</td>
</tr>
<tr>
<td>summarized by</td>
<td>2*</td>
</tr>
</tbody>
</table>

Table 1

The three examples we showed here about different learning modes that can be implemented are in no way the best nor the only possible implementation of learning modes. Their presentation here serves only for a better understanding of the underlying idea about the key role of the distance function for supporting different learning paths.

The node types as information for learning path support

Until now we have only used the information from the links for supporting different learning paths, but the node-type information can also be used. We have already said didactic networks should also adapt themselves to different learning situations. For example, if the lesson graph is currently being used by a
single student, he will be certainly not want to perform a discussion or a group work session, unless this is carried on asynchronously. In the situation of the material being used by a teacher in a classroom, she/he may want to avoid the performing of a student’s individual work node during his lecture. Another important issue is that in some mobile scenarios when the material is being consulted, heavy weight multimedia material should be avoided, like high quality videos or sound. In general, the system should be able to support profiles which exclude all nodes of a certain type from the path of the lesson. The backbone of the navigation support during the lesson is the spanning tree of the lesson. Therefore, the exclusion of the nodes do not fit the given profile. At first glance, this may seem a trivial task because it consists of just bypassing the nodes which should be avoided during the computing of the NEXT-NODE help-function. However, the new link which bypasses the node consists of the concatenation of two links, having two different labels. Which label should then have the bypassing link? The solution we propose here is to assign to the new link the label of the first link, because it can be regarded as the original relation of the starting node with the subtree which had the bypassed node as its root. If the bypassed node has more than one son, these sons will all be connected to the father (see Fig 5).

![Diagram](image.png)

**Fig 5:** To the left it can be seen the bypassing of a node when it has only one son and to the right when it has more than one. (Note the link labels before and after.)

### 5. Conclusions

This paper presented a model for structuring a curriculum for a lesson or course with computer based learning material. By structuring the computer based learning material in a curriculum as a graph, we wanted to achieve the following goals: The most important goal was to help authors of courseware to reuse the material they have already developed. The most important issue we tried to address is to put this material in a structure according to learning goals and learning activities. This information will be used for helping the author of a Lesson, unit or Course to copy part of the material from another lesson and insert it in a new one. Also the user (learner or teacher) will benefit of this information by better selecting the material she/he want to use. With this we give not only learners but also teachers the possibility to use the computer material of a lesson in an order according to their preferred teaching/learning strategies, the needs of the learning group, and circumstances of the learning context, by making possible the traversing of the didactic network of a lesson through different, but conceptually coherent learning paths. We think we also achieved other secondary goals:

- Make the curriculum designer aware of his decision of using a certain computer based material and/or programs during a lesson, and how the learning activity which will be supported by computers fits into the overall curriculum of a lesson or course.
- Help the curriculum designer to develop a discourse strategy to display the planned lesson and perform the learning planned activities.
- Help the curriculum designer to structure a lesson with independence of the learning material which will be used. This independence is achieved by first defining the concepts or skills which are to be taught and/or learnt, along with the activity with which the concept is to taught (nodes). These concepts will be organized in a coherent and consequent learning unit (links). The computer material which will be used to support the teaching/learning activity can be associated afterwards.

A preliminary version of the didactic network model was implemented at the University of Duisburg by the
COLLIDE group in the context of a system for supporting authoring and presentation of computer supported collaborative lectures for the in-classroom face-to-face situation. [Baloian & Hoppe, 1995] [Hoppe & Tewissen, 1994]. Now, a more efficient version is being developed which is more

References


- 28 -