# XEDU, a Framework for Developing XML-based Didactic Resources

F.Buendia, P.Diaz*, J.Sahuquillo, J.V. Benlloch, J.A Gil, M.Agustí
Dpto. de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
*Departamento de informática
Universidad Carlos III de Madrid
Email: fbuendia@disca.upv.es, pdp@inf.uc3m.es

## Abstract

*Recent educational software applications use Web technologies like XML to improve teaching methods in distance learning environments. Though XML has been already used to implement a high number of didactic resources, specification methodologies to develop these resources are rarely applied. As a consequence, the reuse and maintenance of those resources become a difficult task. This paper emphasises the use of hypermedia models to deal with this problem. Hypermedia models have long considered to have a great potential to represent educational applications. The current work proposes the XEDU framework that works over the Labyrinth hypermedia model, to manage and organise didactic resources. The proposed framework provides a set of abstract didactic structures and the interface to associate them either to XML-based contents and other complex didactic resources.*

## 1. Introduction

Recent educational applications use Web technologies to improve teaching methods in distance learning environments. This circumstance has determined the development of multiple courses teaching different subjects using the Web as a delivery medium. Nevertheless, most of these didactic proposals use proprietary tools or are based on rigid formats like HTML dropping their effectiveness. XML [1] and related technologies are opening new educational possibilities and, in fact, they have been applied to implement a high number of didactic resources. Campos *et al* [2] use SGML (the XML ancestor) to define teaching material patterns, e.g., questionnaires. The Nederland Open University proposes EML [3] to represent educational resources. Bourda and Hélier [4] use XML to implement IEEE learning objects. Organisations like IMS [5] propose the use of XML for describing didactic resources.

However, specification methodologies to develop these resources are rarely applied; thus, the resultant educational application is hard either to maintain and to be reused. What we propose in this paper is the use of hypermedia models to deal with this problem. Hypermedia models have long considered to have a great potential to represent educational applications**.** One of their advantages is the use of an abstract notation which supports advanced structure and navigation issues. On the other hand, XML is an adequate notation to represent different contents and publish them in a Web environment. Linking both aspects is one of the main purposes of the current work.

This paper presents XEDU, a framework to represent XML-based didactic resources and to organise them and their multiple relationships by means of a hypermedia model. In this case, we have selected Labyrinth [6] as the hypermedia model, since it provides elements to model interactive contents, virtual objects created at runtime, multimedia presentations where contents can be aligned and synchronised, personal views and user's access control mechanisms; features all of which can be used to increase the usefulness of educational hypermedia environments. XEDU provides a set of abstract didactic structures and the interface to associate them either to XML-based contents and other complex didactic resources such as simulators.

The remainder of the paper is organised as follows. Section 2 describes some works related to the application of hypermedia environments in educational applications. Section 3 shows the main concepts of the Labyrinth model and section 4 describes the XEDU framework and how it is based on the Labyrinth entities. Section 5 describes the XEDU authoring and publishing mechanisms and section 6 shows an example. Finally, section 7 presents some remarking conclusions.

## 2. Related work

While conventional data-oriented environments allow the management of information, hypermedia environments add the ability to manipulate structures among this information. Nürnberg and Tochtermann [7]

classify hypermedia architectures for educational applications in closed and open hypermedia systems.

Closed hypermedia systems are divided into two parts: the hypermedia engine and the data store engine. The hypermedia engine implements in a unique process the front-end part, the link service and the storage mapping. The data store engine may be any kind of process that serves data (e.g. a file system, a database, an HTTP daemon, or a ftp daemon). Current WWW browsers are examples of closed hypermedia systems and there are multiple educational proposals based on such systems [8][9]. They are simple to build, using HTML documents, and their access via WWW browsers is easy and very popular. However, such systems do not separate the document structure from its content and this hinders a convenient navigable structure that would help users to construct a coherent mental representation of the educational information. Moreover, closed hypermedia systems are difficult to maintain and reuse.

On the other hand, open hypermedia systems (OHS) split the hypermedia engine into several independent parts: clients that would correspond to the front-end part in closed systems, structure servers representing the link services and hyperbases that correspond to the storage mapping part. The key difference is that there can be a set of structure servers, each one with a different functionality and representing several client views of the educational information. There are few WWW-based environments for authoring and publishing educational applications using OHS concepts. Gentle [10] is the most representative environment example which is based on the Hyperwave server. Hyperwave is a sophisticated hypermedia server that provides features such as access control on a document-by-document basis and link independence from documents. It comes from an early hypermedia model called HyperG.

Many other hypermedia models have been developed but few of them have been implemented on WWW educational environments which are close to the OHS features [11][12]. WebCosm [13] intends to integrate link services into the document delivery service but they are not specific to educational topics. Helic [14] uses concepts as hypermedia composites to define different *classes* of educational applications. Each class has the predefined navigational structure and the visualisation that best match the application requirements, although the data contents are also stored as HTML documents. Another approach for applying OHS for educational purposes is based on the use of adaptive hypermedia systems [15] as a way to adapt the content and the links of hypermedia pages to the user requirements.

The current work aims at using hypermedia models which enable OHS features such as separation between structure and contents, and open link services. Next section describes Labyrinth as the hypermedia model we have selected in this paper.

## 3. The Labyrinth model

The Labyrinth model [16][17] provides formal elements to describe the static structure and dynamic behaviour of this kind of non-linear, multimedia and interactive applications. Labyrinth represents a hypermedia application or hyperdocument by means of a *Basic Hyperdocument* which includes the elements that can be accessed by all users. The access to this hyperdocument is controlled by means of a security mechanism aimed at safeguarding the information confidentiality, by means of access control lists, as well as integrity by means of context-dependent user abilities [18]. For example, a student can maintain private data within the hyperdocument by denying access to the rest of users. Moreover, users are granted access capabilities for the hyperdocument elements in such a way that the same instructor can be allowed to modify only his lessons. In addition, each user or group can have a *Personalised Hyperdocument* in which the components of the Basic Hyperdocument are modified, deleted or created to fulfil the user requirements. For example, an educational hyperdocument can be adapted to the student's knowledge in order to support an individualised learning process using personalised hyperdocuments whether for individual users or groups representing a specific learning style.

A Labyrinth hyperdocument is composed by the following elements: *users*, *nodes*, *contents*, *anchors*, *links*, *attributes* and *events*. The *user* set includes the possible users (e.g. instructors and students in an educational application), as well as the groups that can be identified (e.g. courses, students profiles). A *node* is defined as an information container which structures the set of didactic resources and a *content* is the entity that represents a piece of information. Contents can be placed in the nodes but they are maintained as separated entities. This separation between structure and content permits sharing contents among nodes teaching different subjects as well as having nodes related to the same subject but which differ in the number or the depth of their contents. Labyrinth also provides composition mechanisms to model complex structures. For example, aggregation allows different elements to be referred to by means of a single composite element and generalisation defines a composite element whose components inherit all its properties. Other key concepts in a hypermedia system are anchors and links. An *anchor* in Labyrinth determines a reference locus into a node or a content. A *link* is a uni or bi-directional labelled connection between two sets of anchors, sources and targets. Four types of links are defined in Labyrinth: referential, aggregation, generalisation and version. The referential type is used to represent arbitrary connections

between elements (nodes or contents) while the other ones are used to create composite objects from the primitive nodes or contents (e.g. to aggregate related nodes such as those that describe the problems of a certain topic or to group the temporal versions of a program content). Version links can be very useful in educational applications, particularly to support co-operative work. Labyrinth *attributes* are properties that are associated to users, nodes, contents and links in order to increase their semantics. There are no restrictions in the number of attributes for each element of the model, although some of them have mandatory attributes. For example, useful attributes in an educational hyperdocument can be the subject a node deals with or its educational category, and the level of difficulty of a question. Finally, any Labyrinth element can be associated with a given action when such element is accessed and certain conditions occur. This dynamic behaviour is modelled by means of *events* which can be used to define interaction mechanisms as in [17], where a crossword exercise in the Now-Graduado educational application was implemented using Labyrinth. Moreover, events set the basis for the definition of virtual objects created at runtime. For example, adaptive links which are presented only when the student has acquired a certain level of knowledge as in Interbook or elm-art [15], can be modelled by means of an event tied to the node where the link has to be embedded.

The Labyrinth model was selected to develop XEDU for several reasons. On the one hand, this model, that has been successfully applied in some educational examples [19][20], makes a clear distinction between structure and content that does not appear in reference hypermedia models like Dexter [21]; a separation which, underlying the development of XML, is easy to implement using this markup language. On the other hand, there are several features of this model which can help to create useful educational environments, including:

1. The use of personal views to support individual and cooperative work spaces.
2. The separation between contents and links, which makes the hyperdocument easier to maintain.
3. The possibility of creating links anchored in any kind of content.
4. The inclusion of mechanisms to create multimedia presentations in which elements can be organised in the space and time.
5. The possibility of including properties to categorise the different elements.
6. The use of events to define interactive behaviours and virtual objects.
7. The ability to establish a security policy where the rules to access the hyperdocument are defined.

# 4. The Xedu framework

The XEDU framework is based on an OHS model whose internal structure is shown in Figure 1. The three main parts of this model are: *Client Interface*, *Structure Server* and *Data Mapping*.

## 4.1 Didactic structures

The *Client Interface* level allows users to access the XEDU services and it offers two kinds of services: authoring and publishing didactic structures. Didactic structures are hypermedia composites used to gather educational resources. They represent the higher abstraction level in the design of an educational application and they can be assigned to different navigational modes (sequential, hierarchical, relational, or adaptive [15]), as well as distinct presentation styles.

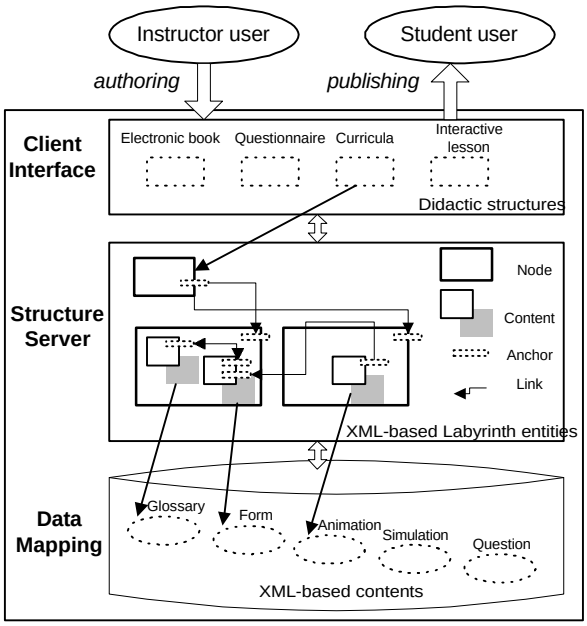


**Figure 1.- XEDU internal structure.**

Didactic structures can be used by instructors and teachers to organise their educational material and to develop new resources. They mainly differ from the entities proposed by Helic [14] in the higher abstraction level hiding the underlying hypermedia model, and the fact that XML is used either to implement the own didactic structures and the final contents, instead of HTML documents. These XML-based contents are managed in the lower XEDU level (*Data Mapping*). The relationship between didactic structures and XML-based contents is defined in the *Structure Server* level.

The *Client Interface* level includes several examples of didactic structures which can be divided into two categories. "Standard" didactic structures such as *electronic books* and *questionnaires* represent the first category and they are based on proposals coming from

institutions or consortiums [5], [22]. The second category is represented by "ad hoc" didactic structures which can be developed by the own instructor for a particular purpose. For example, the *curricula* organisation on a given topic or an interactive *lesson* to check the student knowledge about this topic.

### 4.2. Implementing didactic structures

Didactic structures are implemented in the XEDU *Structure Server* level using a XML-based version of the Labyrinth hypermedia model. A didactic structure is represented by means of a Labyrinth *node*. The aggregation node property is used to define the full hierarchy of this structure. The navigational mode assigned to the structure is based on Labyrinth entities such as *anchors*, *links* and *events*. As an example, Figure 2 shows the structure of a *curricula* organisation defined in the *Client Interface* level as a *Curricula* node [23].

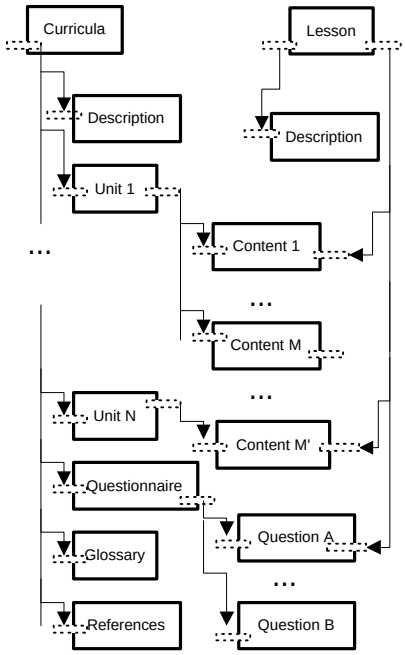


**Figure 2.- Didactic structure example**

In this example, the *Curricula* node aggregates the *Description*, *Unit (1..N)*, *Questionnaire*, *Glossary* and *References* nodes which can also contain other nodes making a complex hierarchy. For example, a *Unit* can aggregate *Content* nodes. Down arrows represent these aggregation links which provide the structural view of the didactic resource. A node can be also generalised to define a composite element whose components will inherit all its properties. For example, several Units can be instantiated from a common Unit template.

There is another didactic structure (*Lesson*) involved in the example. The *Lesson* node forms an aggregation with a single component (*Description*). It also has a set of referential links (left arrows) which represents the navigational mode. The mode selected in the example is based on navigating through certain Content and Question nodes in a sequential way. Such navigation is addressed to evaluate a given competence about a topic. It means that there is a precedence relationship between the component nodes in such a way a node cannot be accessed before the previous one. If a different navigational mode is required, for example a free access to these nodes, a new referential link path has to be configured.

### 4.3 Managing contents

The interface between the XEDU didactic structures and the XEDU contents is based on the Labyrinth content concept. Labyrinth permits representing several types of multimedia contents such as text, images, or audio, as well as the possibility of defining as many representation spaces as needed for each content. This feature allows us to obtain different views for the same educational content. For example, if the content of a unit section is organised in paragraphs, the first paragraph can be used to prepare a slide presentation.

XEDU integrates content types which are not considered in the original Labyrinth model. That is the case of XML-based contents and other complex didactic resources such as simulators or animations. Content management in Labyrinth is based on defining anchors using predefined units depending on the content type (e.g. seconds and frames for a video, or characters and sentences for a text). Supporting these new content types requires other ways to specify anchors in a content. For example, a *question* content which is a component of one or more *questionnaire* didactic structures. It can be implemented in a XML document with a certain type definition. This definition can include several versions of the same question depending on the difficulty level. Publishing such question requires locating the appropriate version according to the desired difficulty.

The Labyrinth unit concept is not enough to locate anchors in content types which have their own inherent structure. The XEDU alternative is to adapt the Labyrinth model to allow the management of these contents, preserving their own format. The Labyrinth XML-based version permits to define anchors on every content element even for those which are a component of a XML-based content. Anchors can also be defined as program components representing data structures and functions visible to other contents. This feature is useful to specify active contents in XEDU such as simulations or animations [24].

There is also a dynamic view in the *Structure Server* level, related to the event management which is also based on the Labyrinth model. Presentation, navigation issues and interactive behaviours can be event-driven in Labyrinth. Events are typically used in XEDU to select

the paths that a user can choose to navigate through a didactic structure or to allow this user to answer a question content. The actions triggered by a given event, are operations which are performed on XEDU entities, e.g. accessing to a certain anchor or "playing" a certain content. They can be implemented by assigning values to the entity attributes or by invoking processing functions which act on these attributes.

The lower level is the *Data Mapp*ing which manages the XEDU contents. They are based on XML documents that can be stored on conventional databases and managed via XML Query procedures [25].

## 5. Authoring and publishing Xedu entities

Section 4 introduces the framework that underlies the services provided to the XEDU users. Authoring services are divided into two categories: those addressed to manage XEDU didactic structures and those services oriented towards the management of educational contents. Didactic structures are the basic entities for reusing educational resources. Therefore, a first authoring step consists in the generation of the fundamental didactic structures which can be used to create new structures. For example, a *curricula* structure can represent a hierarchy of theoretical notions, practices and questions which form the knowledge pool on a given topic. Using this pool, the instructor can define structures such as an interactive *lesson* described before, an *exploratory tutorial* which navigates the unit nodes in a free way or a *slide presentation* to be used in a "traditional" classroom. Once the structure aspects have been modelled, the next steps will configure the dynamic issues such as the user's interaction, the process definition, or the event control.

On the other hand, the management of educational contents assumes that these contents have been implemented and they only have to be adapted to the XEDU requirements. This adaptation process is straightforward when using XML-based contents, since they have a given structure. This structure definition can be used to locate anchors and define links in high-level entities such as the didactic structures. Several XML-based languages such as MathML [26], SVG [27] or SMIL [28] allow to implement different types of contents. Moreover, standard educational resources such as questionnaires [5][29] have predefined DTD's. Otherwise, XML formats can be developed for specific purposes. XEDU also considers other educational resources such as simulators which can be implemented using design methodologies like object-oriented method [30]. In this case, the adaptation process is straightforward too.

Publishing XEDU entities is a service that allows students or teachers the access to didactic resources. The service implementation is only a function of the delivery media. The platform independence property is one of the advantages of using hypermedia structures and XML-based contents. Figure 3 shows an implementation based on a Web medium. It implements authoring and publishing services using Java "servlets". These servlets are based on libraries specialised in processing XML documents [31] and they run on a standard Web server like Apache [32]. The inputXEDU servlet receives XEDU documents which are sent by instructors or course managers using a Web browser. On the other hand, the XEDU rendering can be based on specialised servlets (viewXEDU) or standard XML-processing servlets.

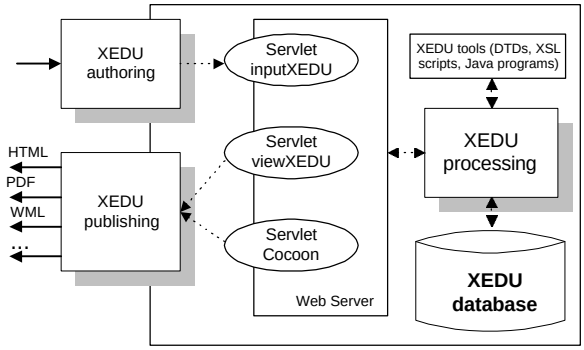


**Figure 3.- XEDU management system.**

Servlets are connected to the XEDU processing section. Figure 3 shows that this section processes either the input and the output XEDU documents. The input XEDU documents can be validated using DTD elements or stored in the XEDU database. The output XEDU documents can be formatted using XSL scripts or through Java programs which process the XEDU entities using XML parsers.

## 6. Application example.

In this section we present an example based on a common topic in computer teaching: the main memory. Among the different points of view this topic can be described (such as hierarchy, physical access, or operating system) we have selected the operating system. Figure 4 shows a block diagram of an *interactive lesson* structure associated to the example. The lesson includes text definitions, images, animations, input forms, simulation programs, questions, glossary, and references, but only some of these contents are shown. They are grouped into node structures using location functions which are represented by means of dotted lines and right arrows. For example, the *Introduction* node has contents such as *Basic Concepts*, or *Memory definitions*. Aggregation links (down arrows) are used to set up the node hierarchy and referential links (left arrows) represent concept relationships which can be navigated through. All these hypermedia elements have been implemented by means of XML documents in the XEDU framework as explained below.

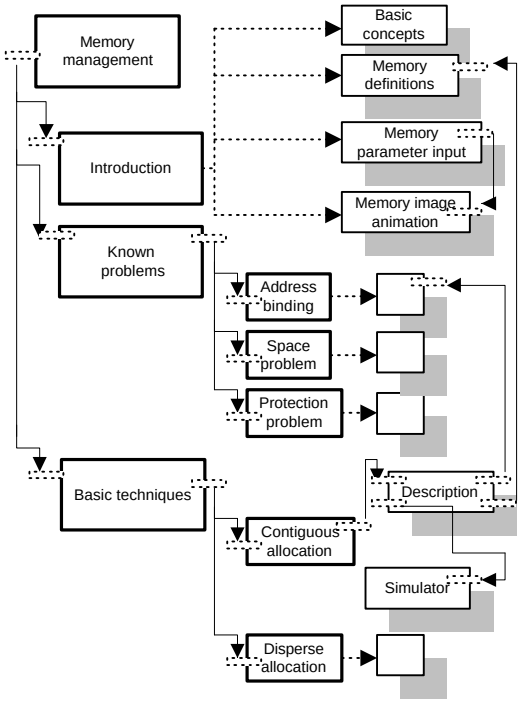


**Figure 4.- XEDU Memory Management specification.**

### 6.1. Implementing the didactic structure

The `Memory Management` node is the hub node in the specification. It aggregates a set of nodes describing basic issues about the memory management topic such as Introduction, Known Problems or Basic Techniques.

```
<Node> id="GM-intro" category="browsing"
    <label>Memory Introduction</Label>
</Node>
```

a) Node XML document

```
<Content>
id="GM-BasCon"
category="browsing"
type="text"
units="paragraphs"
target="Introduction/MemoryBasicConcepts">
    <label> Memory basic concepts</Label>
    <author>Félix Buendía García</Author>
</Content>
```

b) Content XML document

```
<Location Node="GM-Intro">
<content target="GM-BasCon">
    <position coordinate ="…" />
    <time start="…" duration="…" />
</content>
<content target="GM-DefMem"
    <position coordinate ="…" />
    <time start="…" duration="…" />
</content>
<content target="GM-MemPar"
    <position coordinate ="…" />
    <time start="…" duration="…" />
</content>
</Location>
```

c) Location function XML document

**Figure 5.- Didactic structure specification.**

Each one of these nodes can aggregate other nodes or they can have multiple contents. Figure 5a shows an example of XML document associated to the `Introduction` node. It groups a set of contents to illustrate the model implementation. The node XML file has two attribute values: `GM-intro` that is the node identifier (`id`) and a `browsing` category that represents how the node is accessed. An additional label element is used to supply a short description.

An example of content specification is represented in Figure 5b. It is identified as `GM-BasCon` and its category is defined as `browsing`. The type attribute represents the kind of information the content stores. The example uses a `text` format that is organised into `paragraph` units. The target attribute locates the content data using a XPath expression [33], providing a unique location for any XML-based didactic resource. Additional elements are the `label` that provides a short description and the content `author`.

Nodes and contents are related using the Labyrinth location function. Figure 5c shows the location function associated to the `Introduction` node. For each content, there is a target attribute that identifies it. The `position` and `time` elements define its space and temporal co-ordinates inside the node.

### 6.2 Implementing the navigational structure

As stated above, the anchor specification depend on the content type. In the case of a text content such as the one specified in Figure 5a, the anchor is located using the paragraph unit or whatever other unit implicit to this kind of content. If the content has an explicit XML structure, the anchor location is based on Xpointer mechanisms[33]. This is the case shown in Figure 6a, which represents the data of an input form content identified as `GM-MemPar` (see Figure 5c). It has two elements: an input field identified as `i1` and a control item identified as `c1`.

```
<Form>
    <input id="i1" type="text" name="Memory size">
    <control id="c1" type="button" name="Update">
</Form>
```

a) Form content XML document

```
<Anchor id="Fm-i1" >
    <content target="GM-MemPar">
    <position value= "i1">
</Anchor>
```

b) Anchor definition XML document

```
<Link
id="Tx-Fm"
type="referential">
    <source id="Tx-p1">
    <target id="Fm-i1">
</Link>
```

c) Link definition XML document

**Figure 6.- Navigation structure specification.**

The `i1` element has a text type that enables it to enter text data and a `Memory size` name that describes the

input content. Analogously, the `c1` element has a button type that enables it to receive *click* events and a `update` name that is associated to this button. This input field is used to assign a memory parameter such as its size, using different measure units (e.g. bytes, Kbytes, Mbytes) and the control item adds the possibility to submit the introduced value to other content (e.g. the `Memory image animation` content). This form content can also be adapted to other memory parameters, by defining a general content which can be instantiated through the Labyrinth generalisation property.

Figure 6b shows a document in which the `Fm-i1` anchor points to the input field identified as `i1` in Figure 5a. The attribute target points to the `GM-MemPar` content. The exact anchor position within this content is given by the value attribute.

Links are used to connect nodes, contents or both of them. Figure 6c presents an example of link specification. In this example, a link is set up between the `Tx-p1` paragraph located in the `Memory definition` content as source anchor and the `Fm-i1` text input box defined as target anchor. The `type` attribute identifies the link category which corresponds to a referential type in the example. It refers a bi-directional link associated to the memory size concept. It means that such link is triggered if any source or target anchors is accessed. The link purpose is to relate a theoretical notion with a more practical view of this notion, or vice versa.

Another link example is shown in Figure 7. In this case, an uni-directional referential link is used to represent the relationship between the `Memory parameter input` and the `Memory image animation` contents. This relationship is addressed to display a memory area image whose size is a function of the `Memory size` input field value. It also determines the physical address range. This dynamic interaction is specified using Labyrinth events. When the `Update` button is pushed, a click event triggers the link behaviour causing the access to the `set_size` anchor defined in the `Memory image animation` content. As a consequence, the `set_size()` function is executed.

The `set_size()` function is an interface component associated to the data pointed by the content. These content data are implemented as a SVG [27] document that is written in XML. Using SVG, graphics can be coded directly into an XML document. SVG works by assigning attributes to SVG elements. For example, the element identified as `Range` has a Data attribute and its value is assigned using the `Memory size` input parameter (see Figure 7).

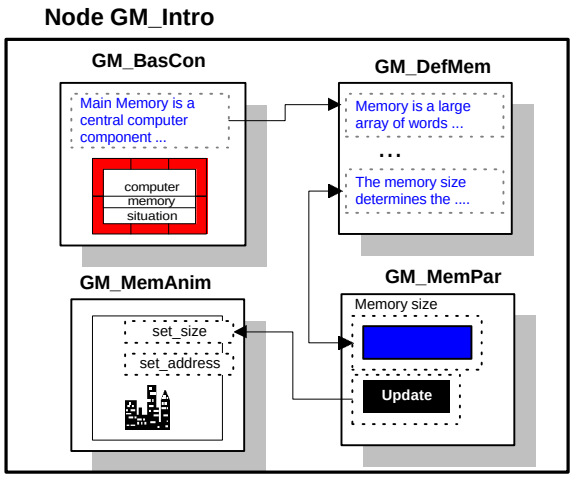


**Figure 7.- GM view.**

Figure 8 shows the Javascript code of the `set_size()` function. It acceses DOM (Document Object Model) elements that are part of the published HTML document. Details about the publishing process are described below. The DOM elements in the example are `Memory` and `Transf`. `Memory` element represents the SVG document whose `Range` component will be updated using the `setData` function. The update input value comes from the `Transf` element.

```
function set_size()
{
    var svgobj;
    var svgdoc = document.memory.getSVGDocument();
    svgobj = svgdoc.getElementById('Range');
    svgobj = svgobj.getFirstChild();
    number = document.transf.size.value;
    svgobj.setData (number-1);
    set_init_address ();
}
```

**Figure 8.- Document access using a Javascript function.**

### 6.3 Publishing the application example

Once the XEDU didactic resources have been coded as XML documents, the next step is publishing them. This process is explained taking the `Introduction` node as reference and using a XSL template that allows the access to the content components.

The system invokes the `xlaby2html-simple-content` template that works as a function with an input parameter that represents the target XML document. Other parameters such as position and time are included to define the content presentation. The template is responsible for processing the content data depending on its type. In such a way, a XEDU text content can be converted to an HTML document. For example, the content `label` and `Author` elements are formatted as HTML headings while the content data are displayed as HTML paragraphs.

## 7. Conclusions

The paper has presented XEDU as a framework based on a three level OHS architecture, addressed to develop XML-based didactic resources. An important aspect is the use of a hypermedia model like Labyrinth as a specification notation that underlies the XEDU framework. This feature is currently applied to define abstract didactic structures helping the instructor in authoring tasks. The main advantage is the possibility to reuse these structures into new ones. Another contribution of the paper is the revision of the Labyrinth model in order to make possible the access to XML-based contents. This allows the instructor to take advantage of multiple didactic resources already implemented.

The paper also proposes a system for authoring and publishing XEDU resources based on standard Web servers like Apache and Java servlets. The XEDU framework is being applied in several university projects. The Theiere project involves 80 European University institutions aiming at harmonising the curricula in EIE (Electrical and Information Engineering) throughout Europe. The experience in this project will evaluate the benefits of using XEDU-based didactic resources. Further works are addressed to analyse Web performance issues related to the XEDU application in education environments, as well as, to check sophisticated navigational modes like adaptive schemes.

## 8. References


[1] XML Extensible Markup Language, http://www.w3.org/XML/

[2] M. Campos, J.Benedito, and R. Pontin. "Tools for authoring and presenting structured teaching material in the WWW," Proc. of the Webnet98, Orlando, Nov. 1998 http://www.icmc.sc.usp.br/~mgp/webnet98/

[3] EML (Educational Modelling Language), http://eml.ou.nl/introduction/

[4] Y. Bourda, and M. Hélier, "Applying IEEE Learning Object Metadata to Publishing Teaching Programs," Proc. of the World Conference on Educational Multimedia and Hypermedia ED-MEDIA 99, http://wwwsi.supelec.fr/yb/publis/edmedia99.html

[5] IMS Global Learning Consortium, http://www.imsproject.org/

[6] P.Díaz, I. Aedo, and F. Panetsos, "Labyrinth, an abstract model for hypermedia applications. Description of its static components," Information Systems. Vol.22 , No.8, pp. 447-464, 1997.

[7] P. J., Nürnberg, and K.A Tochtermann, "Comparison of hypermedia architectures for educational applications," Proc. of the World Conference on Educational Multimedia and Hypermedia EDMEDIA 98, Freiburg, Germany, Jun. 1998.

[8] C. Steed. Web-based training. Ed. Hampshire, 1999.

[9] T. Ebner, and C. Magele, "Design and Implementation of Interactive, Web Based Courses," Proc. of the WebNet99, AACE, Charlottesville, VA, USA, 1999.

[10] H. Maurer, Th. Dietinger, "How Modern WWW Systems Support Teaching and Learning," Proc of the ICCE 97 (Ed. Z. Halim, T. Ottmann, Z. Razak), Kuching, Sarawak Malaysia, December, 1997, pp 37–51.

[11] H. Davis, G. Hutchings, and W. Hall, "Microcosm: A Hypermedia Platform for the Delivery of Learning Materials," http://www.bib.ecs.soton.ac.uk/data/1332/html/html

[12] E. Duval, H. Olivié, and N. Scherbakov, "Contained hypermedia," The Journal of Universal Computer Science, Vol.1, no. 10, pp. 687-705.

[13] L. Carr, D. De Roure, and G. Hill, Ongoing Development of an Open Link Service for the World-Wide Web, http://www.ecs.soton.ac.uk/~lac/tr1/tr1.html

[14] D. Helic, "Authoring and Maintaining of Educational Applications on the Web," Proc. of the World Conference on Educational Multimedia and Hypermedia ED-MEDIA 99.

[15] P. Brusilovsky, "Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies," Proceedings of Workshop WWW-Based Tutoring at 4th International Conference on Intelligent Tutoring Systems (ITS'98), San Antonio, TX, August, 1998.

[16] P.Díaz, I. Aedo and F. Panetsos, Labyrinth, an abstract model for hypermedia applications.Description of its static components. Information Systems. Vol.22, No.8, 1997, pp. 447-464.

[17] P.Díaz, I. Aedo and F. Panetsos, "Modelling the dynamic behaviour of hypermedia applications," IEEE Transactions on Software Engineering , vol 27 (6), June 2001, pp. 550-572.

[18] P.Díaz, I. Aedo and F. Panetsos, Definition of integrity policies for hypermedia systems. "Integrity and Internal Control in Information Systems Strategic Views on the Need for Control". Eds. Margaret E. van Biene-Hershey y Leon A.M. Strous. Kluwer Academic Publishers. 2000..85-98.

[19] P.Díaz, I. Aedo and F. Panetsos, "Design of an Educational Hypermedia Application using the Labyrinth formal model," Proc. of EDMEDIA/EDTELECOM 97, 1997.

[20] A. López-Rey, F.Panetsos, M. Castro, and J. Peire. Utilización del modelo Labyrinth en el diseño de la aplicación Now-meta. Congreso Nacional de Informática Educativa (CONIED'99)",, Puertollano, Spain, Nov. 1999

[21] F. G. ,Halasz, and M. Schwartz, "The Dexter Hypertext Reference Model," Proc. of World Conference of Hypertext, 1990, pp. 95-133.

[22] DocBook, http://www.oasis-open.org/docbook/

[23] F. Buendía, J.V. Benlloch, J.A. Gil, M. Agustí. XEDU, a XML-based framework for developing didactic resources. EAEEIE' 01 Annual Conference on Education in Electrical and Information Engineering. May 2001 Nancy.

[24] F. Buendía, J.V. Benlloch, and J.M. Soriano. Development of didactic resources for distance learning based on simulation. Computers and education: towards an interconnected society. Editors M.Ortega and J. Bravo, Kluwer Academic Publishers 2001 (in press).

[25] XML Query, http://www.w3.org/XML/Query

[26] MathML Mathematical Markup Language, http://www.w3.org/Math/

[27] SVG Scalable Vector Graphics, http://www.w3.org/Graphics/SVG/Overview.htm8

[28] SMIL Synchronized Multimedia Integration Language, http://www.w3.org/AudioVideo/

[29] T. Ferrandez. Development and testing of a standardized format for Distributed learning assessment and evaluation using XML. Phd Thesis http://cran.mit.edu/~vernier/teresa/Thesis.html

[30] F. Buendía, M. López, I. Blesa, and J.V. Benlloch. Using Hypermedia Techniques for Developing Object-Oriented CourseWare about Computer Systems. EAEEIE' 97 Annual Conference on Education in Electrical and Information Engineering. Jun. 1997 Edinburgh.

[31] H. Maruyama, K. Tamura, and N. Uramoto. XML and Java. Developing Web Applications. Ed. Addison Wesley, 2000.

[32] Apache XML Project, http://xml.apache.org/

[33] XML Pointer, XML Base and XML Linking, http://www.w3.org/XML/Linking