

## A Framework for the Management of Digital Educational Contents Conjugating Instructional and Technical Issues

**Félix Buendía-García**

Department of Computer Engineering  
Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain  
Tel: +34 96 387 7000 (ext. 75734)  
Fax: +34 96 387 7579  
[fbuendia@disca.upv.es](mailto:fbuendia@disca.upv.es)

**Paloma Díaz-Pérez**

Department of Computer Science  
Universidad Carlos III de Madrid, Av. Universidad, 20, 28911 Leganés, Spain  
Tel: +34 91 624 9456  
Fax: +34 91 624 9129  
[pdiaz@inf.uc3m.es](mailto:pdiaz@inf.uc3m.es)

### Abstract

Nowadays, the e-learning industry is focused on producing and managing digital learning contents. There are several examples of learning objects, content packages or metadata proposed by different organizations such as IMS, IEEE or ADL. However, little attention has been paid to the specification and management of instructional processes, teaching strategies and learning activities. These issues are addressed by instructional design approaches but there are few formal computer-based notations and methods to specify and implement them. This paper describes a framework for managing digital contents and the processes that use them in an instructional way both from an instructional point of view as well as from a software design perspective. Such framework is based on an instructional application model which provides entities such as the *User Profile* that stores relevant information concerning the learning and teaching processes; the *Learning Scenario* as the set of terms, conditions and activities that characterize the user learning in a specific context and the *Didactic Structure* that is addressed to organize educational contents in a didactic way. The proposed framework supports the translation of these instructional entities to hypermedia entities in order to support the software design of e-learning products. Thus, this design method eases the development of Web-based instructional applications that can be adapted to different learning contexts.

### Keywords

Instructional Design, Learning Contents, Educational Modeling Languages, Hypermedia Model

### Introduction

Nowadays, the e-learning industry is chiefly focused on producing and managing digital learning contents that can be reused in different educational settings, for which several representational notations have been proposed. For instance, according to the IEEE LSTC examples of learning objects include “multimedia content, instructional content, learning objectives, instructional software and software tools, and persons, organizations, or events referenced during technology supported learning” in the Learning Object Metadata (IEEE LSTC, 2003) and metadata are defined as “the attributes required to fully/adequately describe a Learning Object”. IMS Global Learning Consortium (IMS, 2002) is developing and promoting open specifications for managing learning objects, providing content package structures, metadata or sequencing schemas, as well as XML bindings. Other institutions and organizations such as Ariadne (Ariadne, 2003), DublinCore (DC, 2003) or Advanced Distributed Learning (ADL, 2003) are also providing specifications, guidelines and run-time environments for learning objects.

However, little attention has been paid to the specification and the management of instructional processes as the cornerstone that facilitates successful learning experiences, going beyond the boundaries of deploying learning objects and putting more emphasis on instructional design as a discipline to build systems that take account learning needs and goals in an effective way. There are several instructional approaches in the literature that provide guidelines to apply learning theories and models, ranging from more traditional views (including objectivist and cognitivist) such as the Dick & Carey work model (1996), Gagne conditions (1977) or the Merrill's Component Display Theory (1983), to constructivist proposals such as the Jonnasen's Constructivist

Learning Environment (Jonassen & Rohrer-Murphy, 1999). A review of these approaches and a study of their application to the design of Web-based courses can be found in Moallem (2001). But still these instructional models do not provide a computer-based formal model for specifying and managing instructional entities such as user needs, objectives, task or activity definitions, or media and content specifications and therefore, the gap between technical and instructional issues is not covered.

Recently, EML's (Educational Modeling Languages) have arisen to model the learning process and not just learning contents. The survey of existing EMLs reported in CEN/ISSS Learning Technology Workshop (CEN/ISS, 2003) led to select EML-OU (Koper, 2002) as the basis for the Learning Design specification (IMS LD, 2003) that combines EML with other IMS specifications, particularly, those concerning content packaging, metadata and simple sequencing. The main goal of Learning Design is to offer a framework of elements that can be used to describe any teaching or learning experience in a formal way characterized by its completeness, pedagogical flexibility, personalization, reproducibility, interoperability, compatibility and reusability.

All these approaches provide instructional designers with means to develop formal instructional specifications. However, they are addressed to develop final products that are used in very specific environments and, consequently, they are usually focused on translating such specifications, generally written in XML, to a very specific displaying and publishing format, such as HTML or PDF. A step further would be to map instructional products into technical specifications that are independent from implementation platforms, which is a main concern in this work. In this way, not only interoperability is supported, making possible to use the same instructional application in different hardware and software platforms, but also software design entities, representing the instructional application from a technical point view, can be automatically generated.

The framework here proposed is addressed to conjugate instructional and technical issues by processing instructional specifications from a software design perspective. The basic assumption of this proposal is that educational software development is a two-faced process: on the one hand, instructional design has to be considered as efficient learning and teaching is pursued; on the other hand, technical design has to be supported as a complex and quality software tool is being built. The framework is then based on an instructional application model, called Xedu (Buendía et al., 2002), that provides entities representing the main instructional components such as learning scenarios, contents and user information, and that will drive the instructional design process. These instructional entities are automatically translated into the entities of a hypermedia reference model, called Labyrinth (Díaz et al., 1997a; Díaz et al., 2001a), that will drive the technical design of e-learning products. The closeness between the learning material, traditionally based on document formats, and the hyperdocuments managed by hypermedia models, supports this proposal. Moreover, hypermedia models provide independence from implementation technology improving interoperability.

The remainder of the paper is organized as follows. Second section describes the Xedu model entities, focusing on *Didactic Structures* that organize contents from a didactic point of view. Third section describes the Labyrinth model and its support to model structural and navigation issues. Fourth section explains the design method that links both models. An application example is described in the fifth section. Finally, section 6 presents some remarking conclusions.

## The Xedu Model for Instructional Design

The Xedu model is oriented towards offering instructional designers a framework for the specification of any instructional application from two different perspectives:

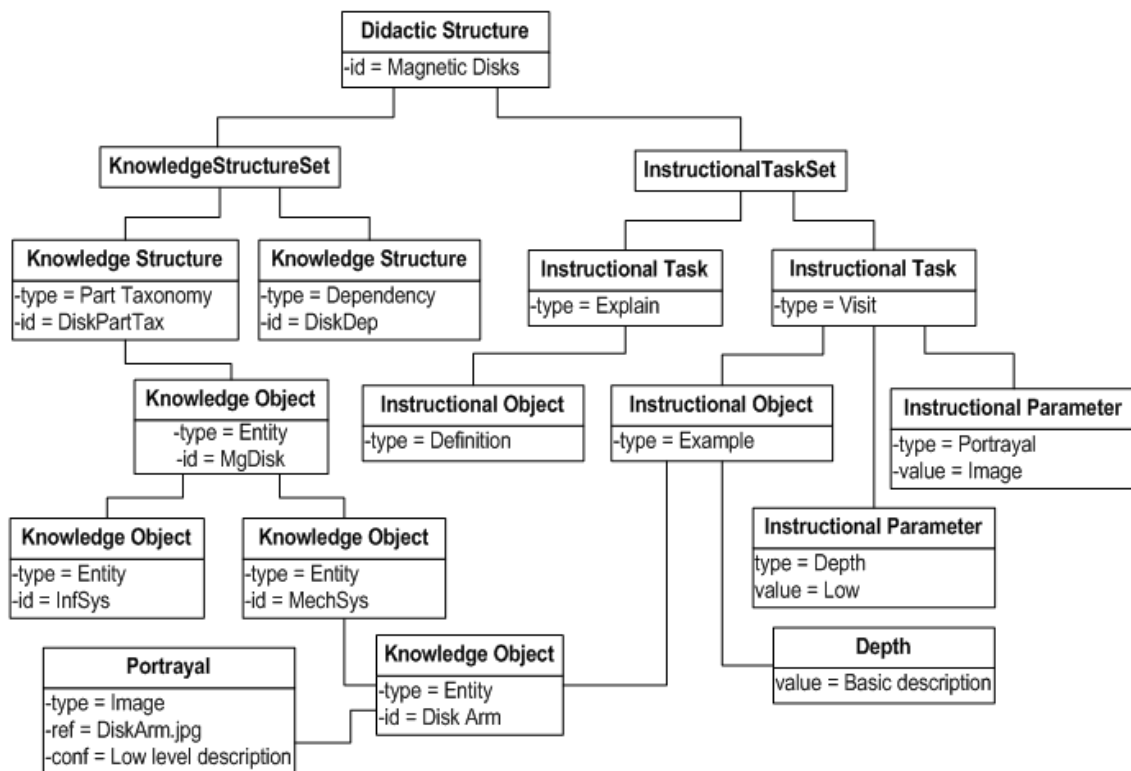
- the semantic view, where the instructional aspects are specified using an information model representing the composition of learning activities, contents and user information as well as their relationships; and,
- the operational view, where the application use and behavior are defined.

An *Instructional Application* in Xedu is composed by three main entities:

- *User Profile*: that stores relevant information concerning the learning and teaching processes.
- *Learning Scenario*: that is defined as the set of terms, conditions and activities that characterize the user learning in a specific context.
- *Didactic Structure*: that is addressed to organize educational contents in a didactic way.

Since this paper focuses on the management of contents from an instructional point of view and *Didactic Structures* are specially devised for this purpose, the remaining of this section will concentrate on this element. Detailed information on other Xedu elements can be found in (Buendía et al., 2002). To enhance readability,

descriptions are illustrated by means of an example shown in Figure 1 where a *Didactic Structure* oriented towards learning about magnetic disks in a Computer Architecture course is shown.



**Figure 1.** Example of Didactic Structure about “Magnetic Disk” concepts

A *Didactic Structure* is composed by a *KnowledgeStructureSet*, that is an aggregation of one or more *Knowledge Structures*, and an *InstructionalTaskSet* that also aggregates one or more *Instructional Tasks*. *Knowledge Structures* are based on the concept of Knowledge Structure proposed by Merrill & ID2 (1996) and they are used to organize the contents of *Didactic Structures* whose atomic units are called *Knowledge Objects*. There are several types of *Knowledge Structures*, each of which proposes a way to structure *Knowledge Objects*, such as “List”, “Part Taxonomy” or “Dependency”, described in Merrill & ID2 (1996). *Knowledge Objects* are characterized by attributes such as the identifier or the type (“Entity”, “Process” or “Action” as defined in Merrill & ID2 (1996)), and elements such as *Portrayal* or *Properties*. In the example, two *KS* types are considered:

- a “Part-Taxonomy” that is aimed at describing the components of a magnetic disk, and
- a “Dependency” that will establish a sequence of knowledge items that help to understand how a file is physically stored on a disk.

In the example only the “Part-Taxonomy” *Knowledge Structure* is presented, and not completely for readability purposes, to illustrate the Xedu components. The “Part-Taxonomy” *Knowledge Structure* is a hierarchy rooted at the element called “KSDiskPartTax” whose child concerning the disk device is decomposed into “Mechanical System” and “Information System”.

All these *Knowledge Objects* include information about the concepts they deal with, but it is still required to specify how these items can be used to support learning in an efficient way. *Knowledge Structures* are devoted to gathering the structure of learning contents, but learning material can be used in different ways depending on the learning goals and needs. To cope with this requirement, the model introduces *Instructional Tasks* that, extending the concept of “transaction shell” (Merrill & ID2, 1996), are devoted to supporting the access to the contents of a *Didactic Structure* in a pedagogical way. With this purpose, they use *Instructional Objects* and *Instructional Parameters*. *Instructional objects* are used to deal with *Knowledge Objects* from an educational point of view while *Instructional Parameters* values represent the conditions in which these *Knowledge Objects* are accessed, such as the abstraction level or the portrayal configuration. The didactic knowledge taxonomy proposed by Leidig (2001) is used to represent *Instructional Objects* (including types like “Definition”,

“Example” or “Explanation”). In the example, there are two *Instructional Tasks* attached to this *Didactic Structure*:

- “Visit”: aimed at accessing descriptive data about the *Knowledge Structure* components using different *Instructional Objects* and *Instructional Parameters*. In Figure 1, the “Example” *Instructional Object* is used as a learning activity where images of the “Disk Arm” component are shown (see the value of the *Instructional Parameter* “Portrayal”) with a low level of depth (see the value of the *Instructional Parameter* “Depth”).
- “Explain”: is a more complex *Instructional Task* that involves *Instructional Objects* of type “Explanation” aimed at providing a deeper knowledge on a particular topic. This kind of *Instructional Objects* are usually attached to other types of *Instructional Objects* such as “Activity” or “Question” that allow the user to interact with *Knowledge Objects* to assess if the content was understood. The values returned by this *Instructional Task* can be used to determine how to navigate through the *Knowledge Structure* taking into account the student’s progress.

Compared to other EMLs, Xedu offers several features that are useful for instructional purposes:

- The detachment between the learning scenarios and activities (represented by *Learning Scenarios*) and the contents used by them (organized by means of *Didactic Structures*), allows the instructor to design generic learning scenarios that can be linked with specific contents at runtime. In proposals such as EML-OU, learning activities and contents are statically linked, so little support is given for adaptive systems.
- The use of a variety of *Knowledge Structures* to provide different perspectives about a topic or a subject in order to improve the learner’s understanding of complex aspects that are part of such topic or subject in contrast with the rigid hierarchical structure proposed in EML-OU and Learning Design.
- The separation between instructional information (represented by *Instructional Objects*) and knowledge information (represented by *Knowledge Objects*) that allows the instructor to teach the same contents using different approaches, e.g. changing the abstraction or depth level at runtime. In EML-OU, there are several kinds of learning objects but the instructional information is embedded into these objects.

As it can be seen, Xedu provides instructional designers with elements to specify advanced educational issues. These elements can be easily and automatically translated into XML, so that educational software can be produced. However, a Xedu specification has no use for software engineers and programmers. If the product under development is a complex software system, an engineering perspective is also required to improve reusability, maintainability and quality. For that purpose, a technical model is required. In our case, Labyrinth a hypermedia reference model was chosen, since it could also contribute with the benefits of hypermedia as a learning tool.

## The Labyrinth Model for Hypermedia Specification

The Labyrinth model (Díaz et al., 1997a; Díaz et al., 2001a) provides formal elements to describe the static structure and dynamic behaviour of this kind of non-linear, multimedia and interactive applications. This is a technical model which provides software engineers and programmers with a complete and consistent specification of the components, structures, behaviors and features of the system to be developed. Since most instructional applications are being deployed as web applications, which can be considered as a special case of hypermedia, the use of this kind of model to specify technical entities seems quite appropriate. Moreover, hypermedia has been proved as a useful learning tool in many experiences and, therefore, a hypermedia model could contribute with its associative structure.

Labyrinth represents a hypermedia application or hyperdocument by means of a *Basic Hyperdocument* which includes the elements that can be accessed by users. The access to this hyperdocument is controlled by means of a security mechanism aimed at safeguarding the information confidentiality by means of negative access control lists, where users or groups who cannot access a specific item are included. Moreover information integrity is preserved by means of context-dependent user abilities (Díaz et al., 2000). For example, a teacher can be allowed to modify her courses but just to browse the courses of her colleagues. In addition, each user or group of users can define and manage their *Personalised Hyperdocuments* in which the components of the *Basic Hyperdocument* are modified, deleted or created to fulfil the user needs and preferences. For example, an educational hyperdocument can be adapted to the student's knowledge in order to support an individualised learning process. This personalization can also be defined at the group level to cope with students sharing a same learning style. In fact, the security mechanism underlying the Labyrinth model has been used to personalize hypermedia learning systems (Aedo et al., 2002).

A Labyrinth hyperdocument (see Figure 2) is composed by the following elements: *users*, *nodes*, *contents*, *anchors*, *links*, *attributes* and *events*. The *user* set includes the potential users of the system. Composition mechanism for users is provided where users can be aggregated in teams (e.g. courses, students profiles) or generalized into roles (e.g. the learner, the teacher, the tutor).

A *node* is an information container which structures the set of resources and a *content* represents a piece of information. *Contents* can be placed in the *nodes* using a *Location Function* which specifies when and where the content has to be displayed, but they are maintained as separated entities. This separation between structure and content makes possible to share 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 generalization defines a composite element whose components inherit all its properties.

Other key concepts in a hypermedia are *anchors* and *links*. An *anchor* in Labyrinth determines a reference locus into a *node* or a *content*. For example, a hotword in a textual *content* or an area of a *node* can be anchors and, therefore, act as the source or the target of a *link*. In turn, a *link* is a uni or bi-directional labeled 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 associative 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 topic a *node* deals with or its educational goal, and the level of difficulty of a question. Educational metadata, such those proposed by the Dublin Core initiative (DC, 2003) or LOM (IEEE LSTC, 2003) are treated as Labyrinth *attributes*.

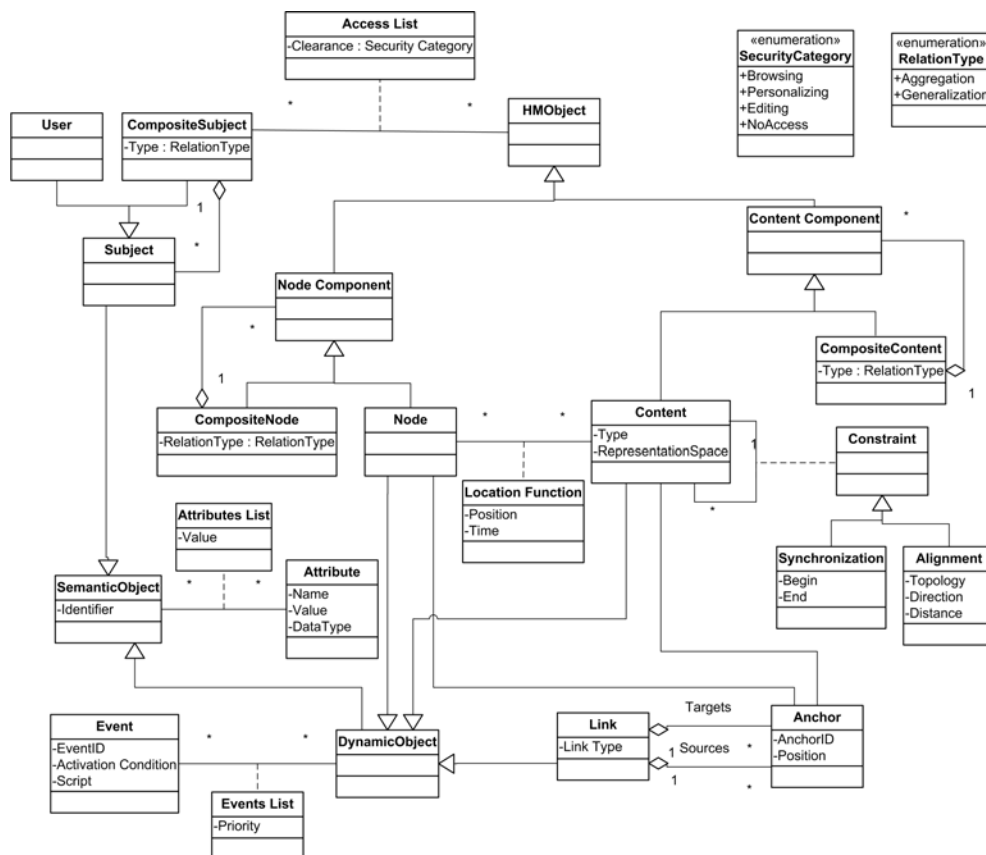


Figure 2. UML representation of the Labyrinth Hyperdocument

Finally, any Labyrinth element can be associated with a given action which is performed when such element is accessed and a certain condition is fulfilled. This dynamic behavior is modeled by means of *events* which can be used to define interaction mechanisms as in Díaz et al. (2001a) where a crossword exercise in the Now-Graduado educational application was implemented using *events*. 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 (Brusilovsky, 1998), can be modeled by means of an *event* tied to the *node* where the *link* has to be embedded.

The Labyrinth model was selected to design Xedu applications from a software point of view, for several reasons. Firstly, this model, that has been successfully applied in some educational examples (Díaz et al., 1997b; López-Rey et al., 1999) makes a clear distinction between structure and content that does not appear in some reference hypermedia models; a separation which is quite useful in learning environments to reuse contents as well as structures. For example, the same structure can be accessed with different contents to deal with students with different backgrounds. On the other hand, there are several features of this model which can help to create useful educational environments, including:

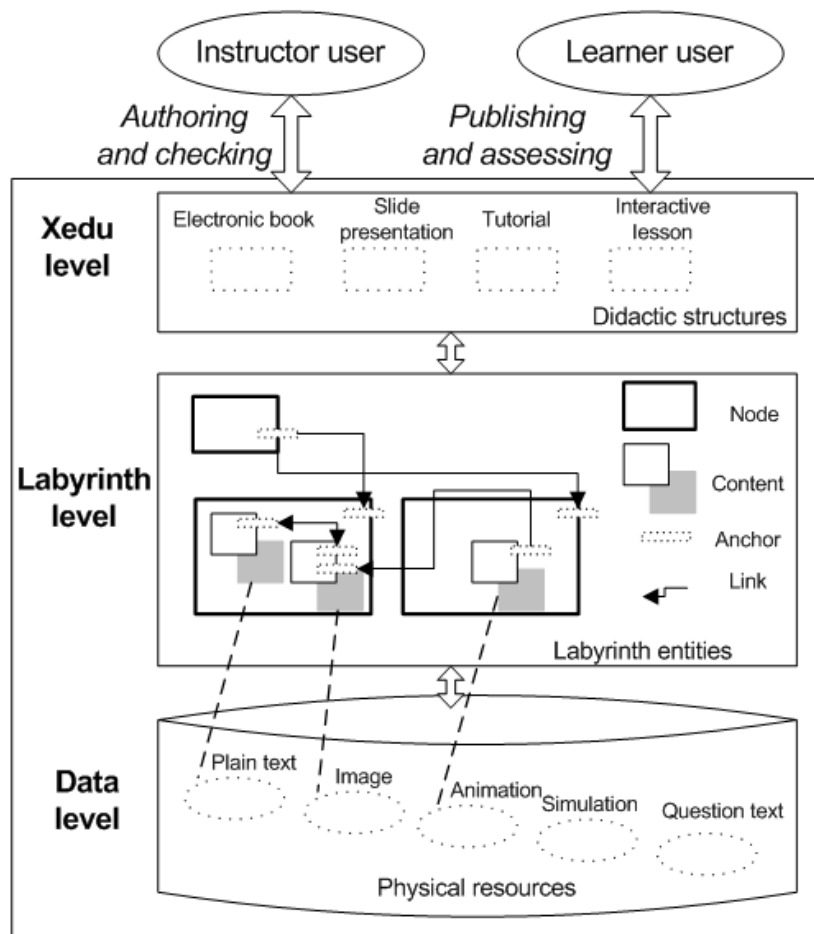
1. The use of personal views to support individual and cooperative work spaces.
2. The separation between contents and links, which makes the hyperdocument easier to maintain (Mendes and Hall, 1999).
3. The possibility of creating links anchored in any kind of content.
4. The inclusion of mechanisms to create multimedia presentations in which elements can be organised and harmonised in the bi-dimensional space (the screen) as well as in the 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 an access policy where the rules to access the hyperdocument are defined, whether for security purposes or to support adaptive environments.

Even though this model has been used to implement some educational hypermedia applications, it does not offer any kind of instructional support. Consequently, to be really useful in e-learning developments it should be coated with an instructional design layer.

## A Framework Reconciling Technical and Instructional Design

An e-learning environment has an instructional goal, for which instructional design methods are required, but it is also a computer-based system, for which software engineering methods are needed. The main goal of the framework proposed in this paper is to guide the design of e-learning products that manage digital contents conjugating these two perspectives. Thus, the framework will made possible to have instructional design and software design for a same product but maintaining them as two different and complementary views that are useful for different kinds of developers such as instructional developers and software engineers. Educational digital content management is performed at three different levels that are shown in Figure 3:

- The **Xedu level** represents the higher abstraction level and it is based on Xedu entities such as *Didactic Structures*. Figure 3 shows some examples of *Didactic Structures* such as an “Electronic Book” or a “Tutorial” (see boxes with the dashed line) which identify specific ways to organize educational contents. These entities are close to the user point of view and their implementation allows instructors and learners to interact with them.
- The **Labyrinth level** specifies the hypermedia entities that model the *Didactic Structures*, using the Labyrinth notation. It addresses a software design perspective, independent from any specific computer-based technology, but useful for developing the implementation of the e-learning products that manage the *Didactic Structures*.
- The **Data level** represents the physical resources storing the educational contents such as text items, images, animations or simulation tools.



**Figure 3.** Content management framework

Users interact with educational contents by means of Xedu entities. Instructors have different rights to access to the contents organized in the *Didactic Structures* that will determine whether they can browse, edit or personalize these contents. Learners access these *Didactic Structures* through *Learning Scenarios* that enable and assess the use of *Didactic Structures*. The only way for a learner to access to these entities is using *Instructional Task* operations that are called from the *Learning Scenario*. This feature provides a strict instruction control that makes possible to check the learning effectiveness.

Working with *Instructional Tasks* involves several entities such as *Instructional Objects*, *Knowledge Structures* and *Knowledge Objects* that are described in the Xedu model section. These entities are close to the instructor point of view but they are distant from a software designer perspective. For that reason the translation of Xedu entities to Labyrinth notations is proposed.

Labyrinth *nodes* are used to represent *Didactic Structures*. They are containers that store information about the *Didactic Structure* components and which corresponding implementation units are visualisation areas (i.e. windows or frames). Additionally, information such as the component type or location can be represented using Labyrinth *attributes*. The structural relationships between these components are based on *aggregation* and *generalization links*. Other kinds of relationships such as the connection between *Instructional Tasks* and *Knowledge Structures*, or between *Instructional Objects* and *Knowledge Objects*, are specified by means of *referential links*. These Labyrinth entities can be used to build a navigation representation, as the one shown in Figure 4. This diagram shows the steps required to access to the list of *Instructional Tasks* for a given *Didactic Structure* called “Magnetic Disk”, and to select and execute one of them. The selected *Instructional Task* belongs to a “Visit” type and it has a reference value (“VisitDisk”) that locates its implementation in the *Didactic Structure* context. The “VisitDisk” *Instructional Task* has semantic relationships with an “Example” *Instructional Object*, a “Portrayal” *Instructional Parameter* that selects the media resource used to display the “Example” information, and a “PartTaxonomy” *Knowledge Structure* that specifies the *Knowledge Object* to be “visited”. The “Example” *Instructional Object* is also related with the *Knowledge Objects* whose type is

“Entity”, in order to select the physical resources that are stored on the Data level and meet the “Portrayal” features. This graphical representation of instructional issues that is based on hypermedia components can help instructors to assess their specifications.

Translation of these instructional entities to hypermedia components of a reference model has two main advantages. On the one hand, a more technical specification can be provided to software engineers to assist them during the development process. On the other hand, since the hypermedia model is platform-independent it can be transformed into different implementation platforms including mark-up languages (e.g. HTML, XML or Smil) as well as authoring tools.

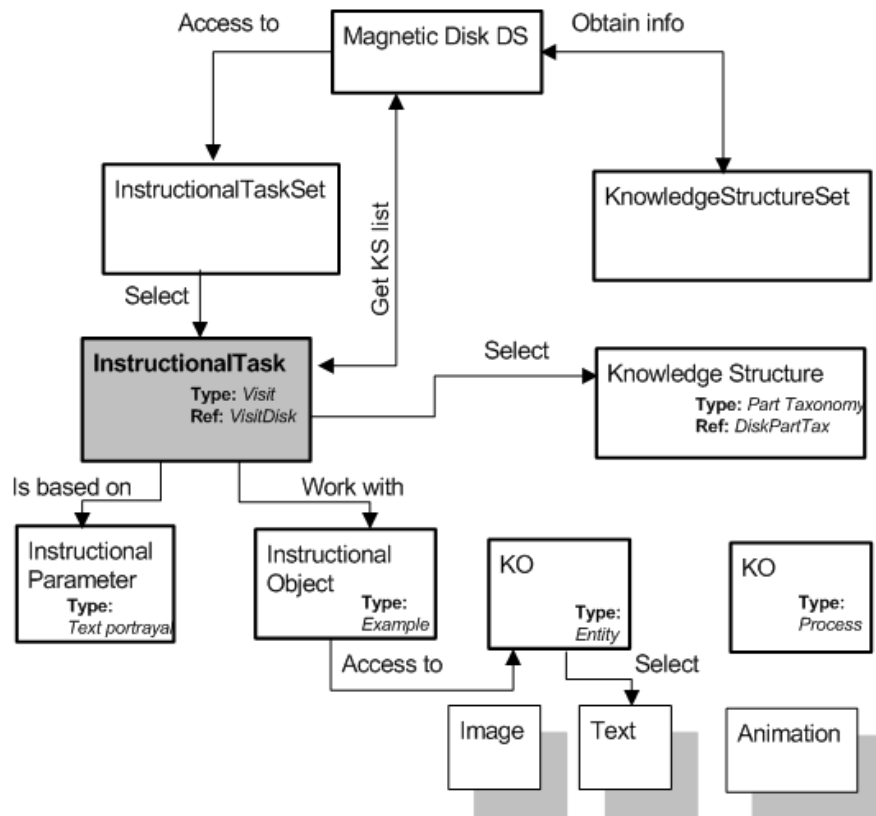


Figure 4. Navigation through the “Magnetic Disk” Didactic Structure

## Application Example

This section describes an example of instructional application whose development is based on the proposed design framework. This application is called *Tutorial Tool* and it is addressed to display the contents organized in a *Didactic Structure*. Figure 5 shows a screenshot of the teacher’s view of an animation about the component assembly of a magnetic fixed disk. The *Tutorial Tool* runs on a Web environment and it divides the application window into several frames:

- The *Main Area* is located at the upper frame and it displays a form that shows the components of the *InstructionalTaskSet* node (grouped by means of aggregation links) and the *Knowledge Structure* types (grouped by means of generalization links). The user can select one of these components and types by means of the “Selection” button or obtain their description using the “Help” button. The event assigned to the “Selection” button enables the action represented by the “Select” referential link that targets the chosen *Instructional Task* node (see Figure 4).
- The *Control Area* is located at the left frame and it is addressed to controlling the components of the selected *Instructional Task* node. There is a form to select of *Instructional Object* and its configuration parameters. It also includes a list of the *Knowledge Structure* nodes that belongs to the *Knowledge Structure* category chosen in the Main Area. The “Display” button enables the action represented by the “Select” referential link that targets the selected *Knowledge Structure* node (see Figure 4).

- The *Display Area* is located at the right frame and it displays the results of linking an *Instructional Task* and a *Knowledge Structure*. Figure 5 shows a form identified as “Event control” that permits the selection of a *Knowledge Structure* component. Below, part of the animation content is displayed, including a form that controls its display through two buttons, “Next” and “Previous” that control the access to the image sequence.

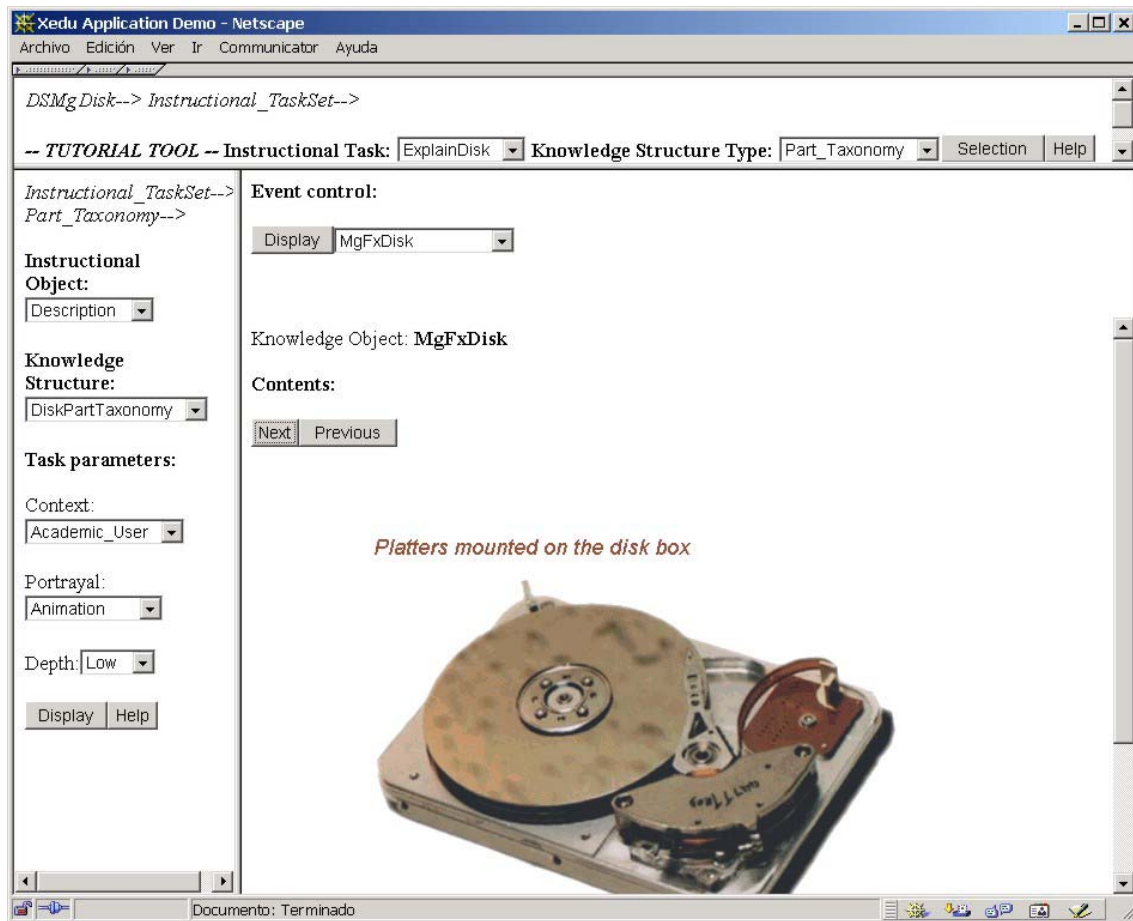
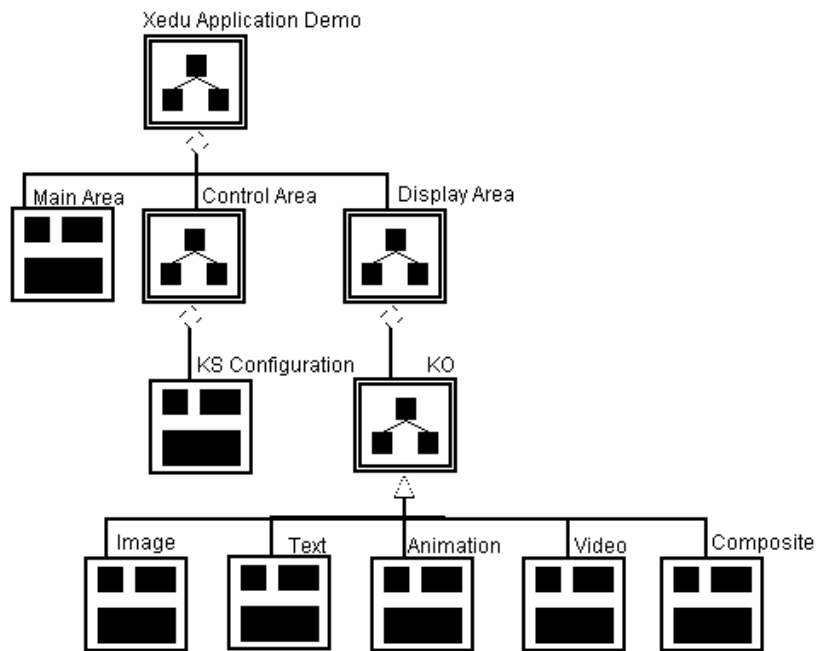


Figure 5. Example of Tutorial Tool Web application

That is an example of application that shows the advantages of the Xedu model to manage learning contents, a management that is not bounded to display these contents. Adaptive features, such as the employment of user information (e.g. objectives or preferences), can be introduced to filter the content information. For example, the *Instructional Task* parameter called “Context” in Figure 5 allows the learner to choose the access method to the *Knowledge Structure* components (e.g. direct or sequential access). As it can be seen, this tool provides the instructor an easy and visual access to instructional specifications, so that it can be helpful with checking purposes (e.g. to verify how many types of *Instructional Tasks* are defined or which *Knowledge Objects* can be accessed from a specific *Instructional Operation*).

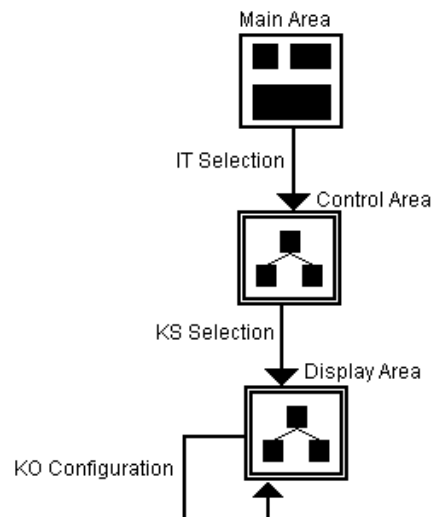
The *Tutorial Tool* application is based on XML documents that specify the Xedu entities and they are processed by means of XSL scripts that interpret the Labyrinth conversion. For example, the node components or attributes can be represented using selection buttons in an HTML form. A Javascript function can be invoked to implement the selection of a referential link. These systematic procedures help to develop the instructional application in a hypermedia environment like the Web.

These elements can be transformed into hypermedia components providing a software engineering specification. As an example, Figure 6 shows a Structural Diagram where the nodes making up the application and their relationships are represented, and Figure 7 shows the Navigation Diagram where the browsing paths are depicted. Both diagrams are products proposed by the Ariadne Development Method (ADM) (Díaz et al., 2001b) which taking as core components the elements of Labyrinth offers a number of activities, steps and products to develop hypermedia in a systematic way.



**Figure 6.** Structural Diagram for the Xedu Application

As it can be seen in Figures 6 and 7 this representation is a quite useful one for programmers, who are not concerned with learning issues but with technical ones. For example, while Figure 4 shows a navigation structure representing relationships among instructional entities that is useful for instructional designers, Figure 7 uses technical entities to describe how to browse the software application which is useful for software developers.



**Figure 7.** Navigation Diagram for the Xedu Application

## Conclusions

This paper has introduced a framework for managing digital contents and the processes that use them, from an instructional as well as a software design perspective. Such framework is based on an instructional application model, called Xedu that provides entities representing the main instructional components such as learning scenarios, contents and user information. The Xedu model supports the specification of instructional issues in a formal and systematic way. The main model contributions related to the content management are: (1) the possibility of establishing dynamic links between *Learning Scenarios* and *Didactic Structures* so that the best

*Didactic Structure* can be chosen at runtime according to some adaptation rules (e.g. user preferences, learning style or features of the environment); (2) the representation of multiple *Knowledge Structures* that organize contents beyond the traditional hierarchical structures, the learner to obtain several perspectives about a topic, easing his learning; (3) the use of *Instructional Objects* that add didactic information to the *Knowledge Objects*, so that the same contents can be presented using different approaches, e.g. changing the abstraction or depth level at runtime.

Xedu entities have been specified using XML documents that enable their automatic processing. However, an Xedu specification has no use for software engineers and programmers. If the product under development is a complex software system, an engineering perspective is also required to improve reusability, maintainability and quality. Such software engineering perspective has been based on the hypermedia model called Labyrinth. This model provides some features that are interesting for designing Xedu applications from a software point of view: (1) the separation between structures and contents that allows for the representation of entities such as *Learning Scenarios*, *Didactic Structures* or *Knowledge Structures* to organise educational information in multiple ways; (2) the possibility of creating *links* anchored in any kind of *content* or *node* that can be created at runtime, enabling the dynamic connection between entities such as *Learning Scenarios* and *Didactic Structures* or *Instructional* and *Knowledge Objects*; (3) the inclusion of mechanisms to create multimedia presentations in which elements can be organised and harmonised in the bi-dimensional space as well as in the time, so that it supports the display and navigation of *Knowledge Object* information using different media formats and timing requirements.

The proposed framework supports the translation of the Xedu instructional entities to Labyrinth hypermedia entities that drives the software design of e-learning products. An application example has been implemented to demonstrate the framework possibilities. It is called *Tutorial Tool* and is addressed to display the contents organized by a *Didactic Structure*. This application is not bounded to display these contents in a fixed way and it supports adaptive features, such as the employment of user information (e.g. objectives or preferences) to filter or adjust the content information.

Further works include the development of tools and methodologies to support the full design process of this kind of instructional applications. This endeavour will imply to provide guides about how to use the Xedu components to design useful instructional applications. A less ambitious project will consist on building a visual tool to generate Xedu specifications and link it with the software tool actually supporting the Ariadne method (called AriadneTool), in such a way that the framework here proposed would be fully automated.

## References

- ADL (2003) Advanced Distributed Learning (ADL), retrieved May 30, 2003 from <http://www.adlnet.org>
- Aedo, I., Montero, S. & Díaz, P. (2002). Supporting personalization in a web-based course through the definition of role-based access policies. *Interactive Educational Multimedia*, 4, 40-52.
- Ariadne (2003). Curriculum Description Format (CDF), retrieved May 30, 2003 from [http://www.ariadne-eu.org/en/publications/references/a-cdf\\_descriptor.html](http://www.ariadne-eu.org/en/publications/references/a-cdf_descriptor.html)
- Brusilovsky, P. (1998). Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies. Presented at the Workshop WWW-Based Tutoring at 4th International Conference on Intelligent Tutoring Systems (ITS'98), San Antonio, TX.
- Buendía, F., Díaz, P. & Benlloch, J.(2002). A Framework for the Instructional Design of Multi-Structured Educational Applications. *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications, ED-MEDIA 2002*, Denver (USA), 719-724.
- CEN/ISSS(2003). CEN/ISSS Information Society Standardization System, Learning Technologies Workshop. Educational modelling languages, retrieved May 30, 2003 from <http://www.cenorm.be/iss/Workshop/lt/>
- DC (2003). Dublin Core Education, retrieved May 30, 2003 from <http://dublincore.org/groups/education/>
- Díaz, P., Aedo, I., Panetsos, F. (1997a). Labyrinth, an abstract model for hypermedia applications. Description of its static components. *Information Systems*, 22 (8), 447-464.

- Díaz, P., Aedo, I., Panetsos, F. (1997b). Design of an Educational Hypermedia Application using the Labyrinth formal model. Presented at the EDMEDIA/EDTELECOM 97 Conference, Calgary, Canada.
- Díaz, P., Aedo, I., Panetsos, F. (2000). 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, 85-98.
- Díaz, P., Aedo, I., Panetsos, F. (2001a). Modelling the dynamic behaviour of hypermedia applications. *IEEE Transactions on Software Engineering*, 27 (6), 550-572.
- Díaz, P., Aedo, S., Montero, S. (2001b). Ariadne, a development method for hypermedia. *DEXA 2001*. LNCS 2113, 764-774.
- Dick, W. & Carey, L. (1996). *The Systematic Design of Instruction*, (4th ed.). New York: Harper Collins College Publishers.
- Gagné, R. M. (1977). *The conditions of learning*. (4th ed.). New York: Holt, Rinehart & Winston, Inc.
- IEEE LSTC (2003). Learning Technology Standards Committee, retrieved May 30, 2003 from <http://ltsc.ieee.org/>
- IMS Global Learning Consortium. (2003), retrieved May 30, 2003 from <http://www.imsglobal.org/specifications.cfm>
- IMS LD (2003) Learning Design Specification, retrieved May 30, 2003 from <http://www.imsproject.org/learningdesign/index.cfm>
- Jonassen, D.H. and Rohrer-Murphy, L. (1999). Activity theory as a framework for designing constructivist learning environments. *Educational Technology Research and Development*, 47(1), 61-79.
- Koper, R. (2002). Modeling Units of Study from a Pedagogical Perspective: the pedagogical meta-model behind EML, retrieved Jun 30, 2002 from <http://eml.ou.nl/introduction/docs/ped-metamodel.pdf>
- Leidig, T. (2001) L3' towards an open learning environment. *ACM Journal of Educational Resources in Computing (JERIC)*, 1 (1): 7.
- López-Rey, A., Panetsos, F. Castro, M. & Peire, J. (1999) Utilización del modelo Labyrinth en el diseño de la aplicación Now-meta. Presented at the Congreso Nacional de Informática Educativa (CONIED'99), Puertollano, Spain.
- Mendes, E. and Hall, W. (1999). Hyper-Authoring for Education: an Empirical Study. *Computers & Education*, 32, 51-64.
- Merrill, M.D. (1983). Component Display Theory. In C. Reigeluth (ed.), *Instructional Design Theories and Models*. Hillsdale, NJ: Erlbaum Associates.
- Merrill, M. D., & ID2 Research Team. (1996). Instructional Transaction Theory: Instructional Design based on Knowledge Objects. *Educational Technology*, 36(3), 30-37.
- Moallem M. (2001). Applying Constructivist and Objectivist Learning Theories in the Design of a Web-Based Course: Implications for Practice. *Educational Technology & Society* 4 (3), [http://ifets.ieee.org/periodical/vol\\_3\\_2001/moallem.html](http://ifets.ieee.org/periodical/vol_3_2001/moallem.html)