ADRIAN
E-Learning Content Production
(creating online exams)

Giovani Rubert Librelotto[*]
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
grl@di.uminho.pt

José Carlos Ramalho
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
grl@di.uminho.pt

Pedro Rangel Henriques
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
{jcr, prh}@di.uminho.pt

May 13, 2004

Abstract

Universities and other institutions related to education are investing time and resources in E-learning initiatives. This leads to an increasing number of course offers in E-learning format. There are environments, called Learning Management Systems (LMS), designed to help teachers in the management of their courses. Basically a LMS provides functionalities to manage student records, to facilitate communication between students and between students and teacher, to control accesses and produce statistics, schedules, evaluation and an open platform to help teachers make lecture content available online. However they do not dictate what kind of technology or format should be used to prepare those contents. Although this issue can be seen as an advantage in certain contexts it leads to a format anarchy and makes support for content production impossible. Here is where ADRIAN comes into the scene providing support for content production.

ADRIAN is composed by several components: one component to help producing lessons and lab guided sessions; one component for the production of tests and exams; one component to support the production of multimedia

[*]Bolsista CNPq - Brasil

presentations; and one component to generate interfaces that integrate all the material produced (content parts) by the other components or developed elsewhere by the teacher.

The whole system is being developed with XML (eXtended Markup Language) using descriptive markup for content, and related technologies like XSL (eXtended Stylesheet Language) for content transformations. This way we ensure the portability and platform independence of the system.

The last mentioned component, the integration component, is based on ontologies; the user is asked to define an ontology for his course. After that the system generates automatically the web interface that integrates all the courseware components.

We start by describing the ADRIAN architecture (a more detailed description was published at M-ISCTE conference) and then we present the Tests and Exams Production application.

The main idea behind Tests an Exams is that structure is not free. This ADRIAN's component enforces a specific structure specified in an XML Schema. This is the way to achieve normalization in the content production. However to convince teachers to use we provide all the editing and transformation tools. The user only has to use an interface to introduce content. After that the system takes care of the electronic publishing: producing paper and web versions and in this case (tests and exams) managing the interaction between students solving the online exam and the system.

In this paper we characterize the different kinds of exams and present the steps towards the XML Schema definition. Then we describe this application lifecycle and the implementation we have.

1 Introduction

Today the web provides an excellent channel to distribute information and to access it. Its application in learning environments was a question of time. Today is the reality.

The first experiences we have made of using the web for teaching purposes date back to 1994. Back then we were using mailing-lists to interact with students and web pages to make available all kinds of information.

Nowadays the demand for E-learning courses and materials is growing everyday. The possibility of being able to follow a course from the comfort of one's home is attractive to many people that can not be present at the time the lectures occur. On the side of the institution, E-learning courses have also some advantages like space economy and a lower resource occupation. Although this physical economy, E-learning demands more resources; preparing lectures and managing student records consumes time and human resources.

Here is where the so called Learning Management Systems (LMS) appeared. There is a considerable number of these systems in use throughout the world, some of them are considered as references: Blackboard, Luvit, WebCT, Lotus Learning Space, TWT,... Some of them came out of academic experiences while others were developed by the software industry. Basically a LMS provides functionalities

to manage student records, to facilitate communication between students and between students and teacher, to control accesses and produce statistics, schedules, evaluation and an open platform to help teachers make lecture content available online. However they do not dictate what kind of technology or format should be used to prepare those contents. Although this issue can be seen as an advantage in certain contexts it leads to a format anarchy and makes support for content production impossible.

ADRIAN is being created in a special context: the authors are computer science teachers, part of them belong to the campus E-learning task force and they have the responsibility of several courses that should be offered in E-learning format. Each component of ADRIAN started as an individual project. These projects gave birth to a set of prototypes that are now being further developed for real cases use. In parallel we were developing another project called METAMORPHOSIS [clei03,coopmedia03] aiming at exposing and integrating information systems on the web through the use of ontologies to specify the different knowledge views. When we joined the set of prototypes we have developed for E-learning content production with the help of METAMORPHOSIS, ADRIAN was born.

For the moment, ADRIAN is composed by four modules to assist the creation of: tests and exams; guided lessons and guided lab sessions; multimedia presentations; and an interface to glue all the others. In the following sections we start by explaining the integration of the three content production modules. Then we will describe each of the individual modules.

2 ADRIAN architecture

Figure 1 illustrates ADRIAN's operational architecture.

In the architecture illustrated by figure 1 we can distinguish four major parts:

Content Provider Applications: Currently this part comprehends three applications:

- **Xlessons:** to produce lectures: lecture notes, laboratorial guided sessions, exercise sheets, ...
- **Xexams:** to assist teachers in the production of tests and exams: multiple choice, true and false, development, progressive, ...
- **Xslides:** to assist the production of slide based presentations.

Content Repository: Normally it will be a subtree in the server filesystem. At present this structure should be frozen and known to the other applications. An ongoing project will take care of these restrictions enabling the user to dynamically change the Content Repository structure. The structure we have defined and that we are using is presented in figure 2.

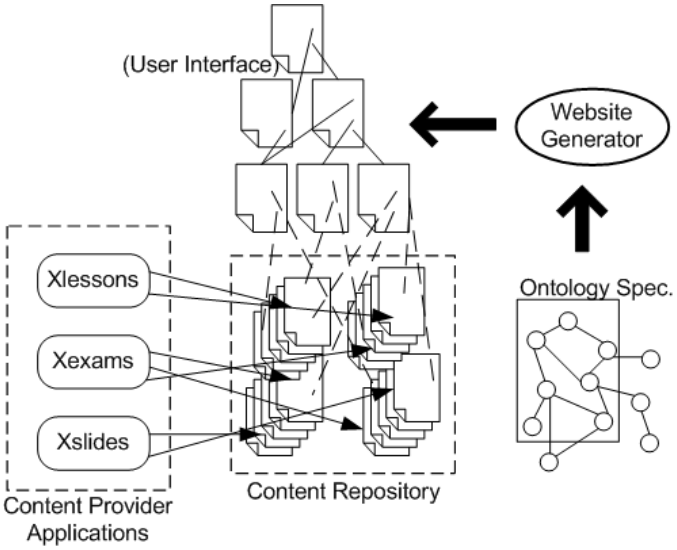

Figure 1: E-Learning platform

Ontology: This part corresponds to an abstract specification of the Content Repository structure. The content provider still has to specify this by hand. There is an ongoing project to compute most of the ontology automatically. The idea is to crawl in the Content Repository structure collecting superclass-subclass relationships and the list of all documents produced so far and to generate the corresponding ontology. At the end the user will be able to add new relationships to the ontology. This ontology represents the input to the Interface Generator, a tool that we have developed that generates a complete website to access and navigate among a set of resources described through an ontology specification.

Interface: This part corresponds to a website from where the user can access all the produced documents.

In the following sections we describe with more detail each of these components.

3 Content Provider Applications

Our content provider applications share a common philosophy: they are based on the principles of structured documentation and they use descriptive markup to structure the documents that are produced. With this statement we mean that the format of the content being produced is not free. It has a textual representation and has its structure completely described through the use of markup and formally specified with a grammar.

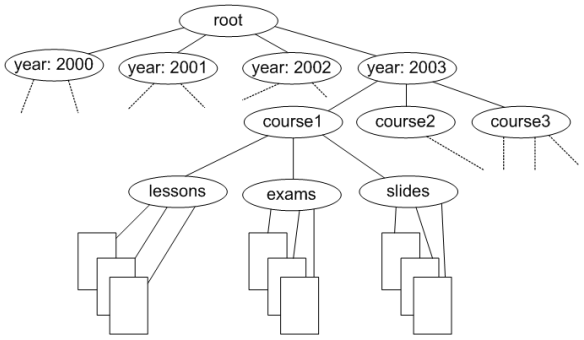

Figure 2: Content Repository structure.

The advantages of descriptive markup are well documented in the bibliography published so far [MA96] [Meg98] but we can enumerate the most important ones:

Portability: since the markup is descriptive it has no operational meaning; it is possible to move documents between different systems and application without any changes.

Flexible formatting: descriptive markup leads to a complete separation between content and form; in order to visualize a document one has to associate it with visual form specification; later on if one wants to change the look of it information, only needs to change this specification the content remains unchangeable.

Longevity: since there are no ties with operational systems and software platforms it is easy to use and reuse this information in tomorrow systems.

In order to clarify this technological choice, the following subsection explains some inherent to the descriptive markup paradigm.

3.1 Descriptive Markup

The idea of using descriptive markup in an electronic publishing environment dates back to the 1960's. However it was only in 1986 that SGML [GM02] [Gol90] has emerged as an ISO standard (ISO 8879). SGML is a meta-language with which is possible to define specific markup languages. Although it has some advantages its complexity led to a small community of users. To overcome this problem XML [HM01] [Meg98] was presented to the public in 1998. XML is a lot lighter, easy to process and new tools and environments appear everyday.

An XML document is a logic structure, a hierarchy of components. Each component can be differentiated from the others through the use of markup that is added to the document. According to this perspective a document has two types of information: data and markup. The following example shows a piece of a document that specifies a lab guided session.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AulaPrática>
  <meta>
    <disciplina>Processamento Estruturado de Documentos</disciplina>
    <data>2003.10.01</data>
    <objectivos>
      <para>O objectivo principal desta ficha é familiarizar o aluno com o XPath.</para>
      <para>Para ...</para>
    </objectivos>
    <recursos>
      ...
    </recursos>
  </meta>
  <corpo>
    <introdução>
      <para>Para, ...
      </para>
      ...
    </intridução>
    ...
  </corpo>
</AulaPrática>
```

This example only presents two components of an XML document. An XML document has three possible components:

XML declaration: All XML documents must begin with this declaration:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

DTD (Document Type Definition) or Schema: A grammar that specifies what markup is required, what is optional, where it is required and where is optional. This grammar defines the markup language, in other words, a DTD or Schema defines an XML dialect.

The document: This component corresponds to the document itself. It is composed by text, markup, and optionally a reference to a DTD or Schema.

An XML document that follows a DTD or Schema is said to be a valid document according to that DTD or Schema. An XML document that does not follow any DTD or Schema is said to be a well formed document.

Figure 3 gives an idea of XML documents lifecycle. This figure illustrates the methodology followed to develop the three applications being discussed in this section.

Figure 3 illustrates the structured documents life cycle with 5 stages: analisys, edition, validation, storage and formatting. The analisys stage corresponds to the study of a kind of documents. This stage is expected to produce a DTD or a Schema [MA96] [KJO+01] that completely defines the structure for a certain class of documents. After there is a small cycle between two stages: the user edits a document and asks the editor to test its conformance with the DTD; if there are reported errors the user will correct them until the document passes the validity

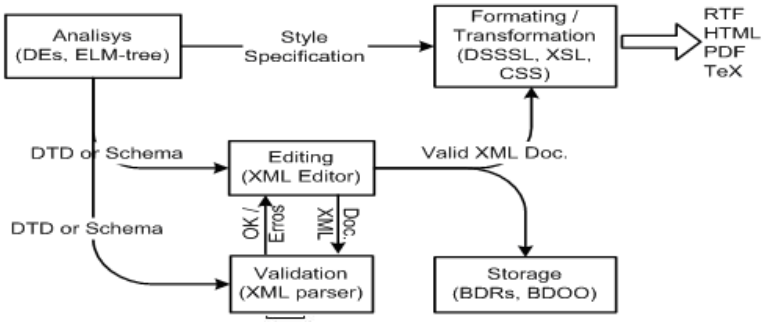

Figure 3: XML documents lifecycle.

check. This stage (2 stages: edition+validation) will produce valid XML documents. With an XML valid document one can store it [Wil00] or process it in order to obtain a specific output. There are many solutions for storage that will not be discussed here. The transformation process is normally specified in an XSL stylesheet [Tid01] (XSL is an XML syntax that is used to specify transformation processes in a declarative/functional way).

Let's see how these concepts were applied to the three content provider applications.

3.2 Tests and Exams

This is probably the most complex type of document in the ADRIAN framework. In an E-Learning context we normally refer to online exams. There are two species of online exams: static (the exam is just displayed inside an ordinary web page (this situation is common to most of other types of documents) and dynamic (the exam is displayed but students are expected to interact with it, they can solve the questions and have immediate feedback from the system). ADRIAN supports dynamic exams and that is the kind of exams that is presented in the following.

Dynamic exams can be divided in the following types:

Multiple choice: each question has a set of answers; the student must pick the right one;

True and False: each question has a set of answers; for each answer the student must give a true or false answer.

Development: the student has a blank area to develop his answer to the question.

The ADRIAN tests and exams application is being developed to support all these three kinds of exams.

Further more, functionally we can classify exams in one of the following types:

One attempt: the student has only one attempt to solve the exam (this is the traditional approach);

Time limit: the student has to solve the exam inside a specific time interval;

Progressive: the exam is presented by levels; to access the following level the student has to reach a minimum in the current level;

Random: the system scrambles the questions; the order in which questions are presented is always different.

An exam does not need to strictly belong to one of those types. It can be a mixed of all or some of those types. In order to achieve this we deal individually with each question. Questions are the building blocks of our exams. To give an idea of the structure defined for tests and exams some parts of the DTD [SdCVMR03] are shown below:

```
<!ELEMENT question (description, choices?)>
<!ATTLIST question
  numberQ ID #REQUIRED>

<!ELEMENT choices (choice+)>

<!ELEMENT description (para+)>
<!ATTLIST description
  tbox (small | medium | big) #IMPLIED>

<!ELEMENT choice (para+)>
<!ATTLIST choice
  answer (true | false) #REQUIRED>
...
```

A document instance would look like:

```
...
  <body>
    <questions>
      <question numberQ="Q1">
        <description>
          <para>No Tratado de Roma assinado em 1957 pelos seis
          Estados-membros de então, ficava expresa a intenção em realizar:</para>
        </description>
        <choices>
          <choice answer="false">
            <para>uma união aduaneira;</para>
          </choice>
          <choice answer="false">
            <para>uma união aduaneira, um mercado comum e uma
            união económica;</para>
          </choice>
          <choice answer="true">
            <para>uma união aduaneira e um mercado interno;</para>
          </choice>
          <choice answer="false">
            <para>um mercado comum, uma união económica e uma união
            económica e monetária.</para>
          </choice>
        </choices>
      </question>
      ...
```

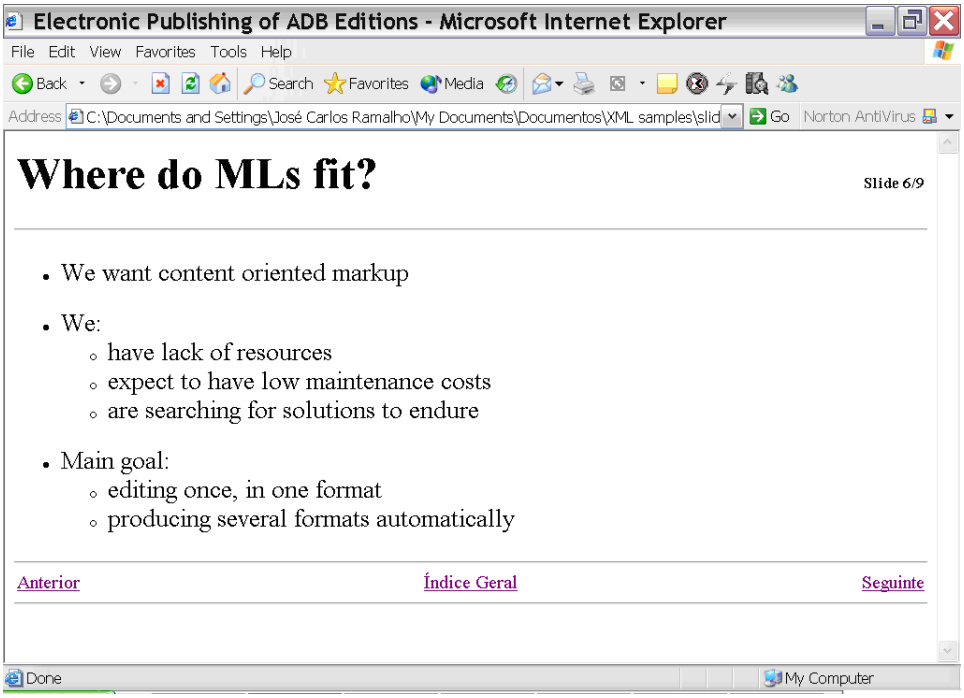

Figure 4: One slide screenshot.

```
27      </questions>
28    </corpo>
29  </exame>
```

In this example we are looking at a multiple choice question. The teacher creates this document where together with the question he also provides the answers. This way it is possible to write a processor for automatically evaluate student's answers. Currently four types of outputs are generated: a web version of the exam enabling students to solve it being located anywhere, a PDF version of the exam so teachers and students can obtain a print copy of the exam, a web version with student answers (only for the teacher and with all possible automatic evaluations) and a PDF version of the exam in the same context. The automatic evaluation process will not be detailed here but its domain is composed by all the questions with answers provided by the teacher (excluding development questions).

3.3 Slide based Presentations

This application follows the same approach as the previous one. An XML Schema was defined for this kind of documents together with two transformation specifications: one to generate a website that drives the slide presentation (figure 4) and the other to generate a PDF version of the slides.

Figure 5 presents a structure diagram of the markup language defined for this application.

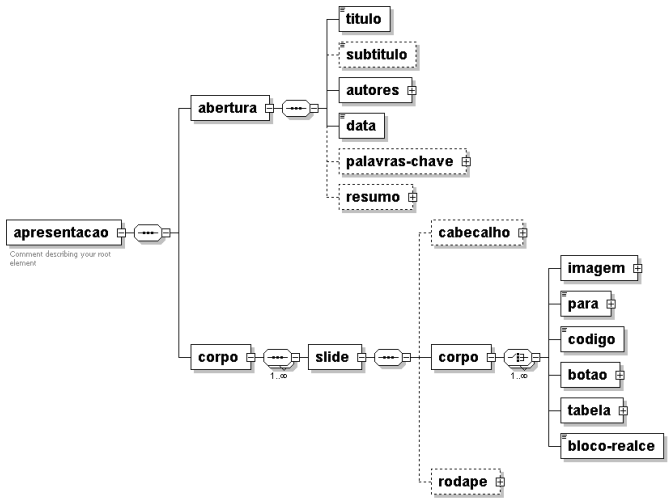

Figure 5: Slide schema diagram.

3.4 Lessons and Laboratory Guided Sessions

This application is being used in three different courses to support practical lessons and laboratory guided sessions. For those courses the paper circuit was eliminated being replaced by web pages accessible from anywhere (what raises another problem called copyright). The same approach was followed to develop the application thus we present a small document instance and the corresponding web page to give you a glimpse of this application.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AulaPrática>
  <meta>
    <disciplina>Processamento Estruturado de Documentos</disciplina>
    <data>2003.10.01</data>
    <objectivos>
      <para>O objectivo principal desta ficha é familiarizar o aluno com o XPath.</para>
      <para>Para atingir esse fim, o aluno irá utilizar o XPath para realizar queries
            sobre alguns documentos XML.</para>
    </objectivos>
    <recursos>
      <documento url="http://www.di.uminho.pt/~jcr/XML/didac/xmldocs/poema.xml">
         Poema: "Soneto Já Antigo", de Álvaro de Campos
      </documento>
      ...
    </recursos>
  </meta>
  ...
    <exercício>
      <título>Queries sobre o poema</título>
      <enunciado>
        <para>Faça o download do poema.</para>
        <para>Crie um documento baseado no DTD acima com
expressões XPath que respondam às seguintes alíneas:</para>
        <alíneas>
          <alínea>Seleccione o terceiro verso da última quadra.</alínea>
```

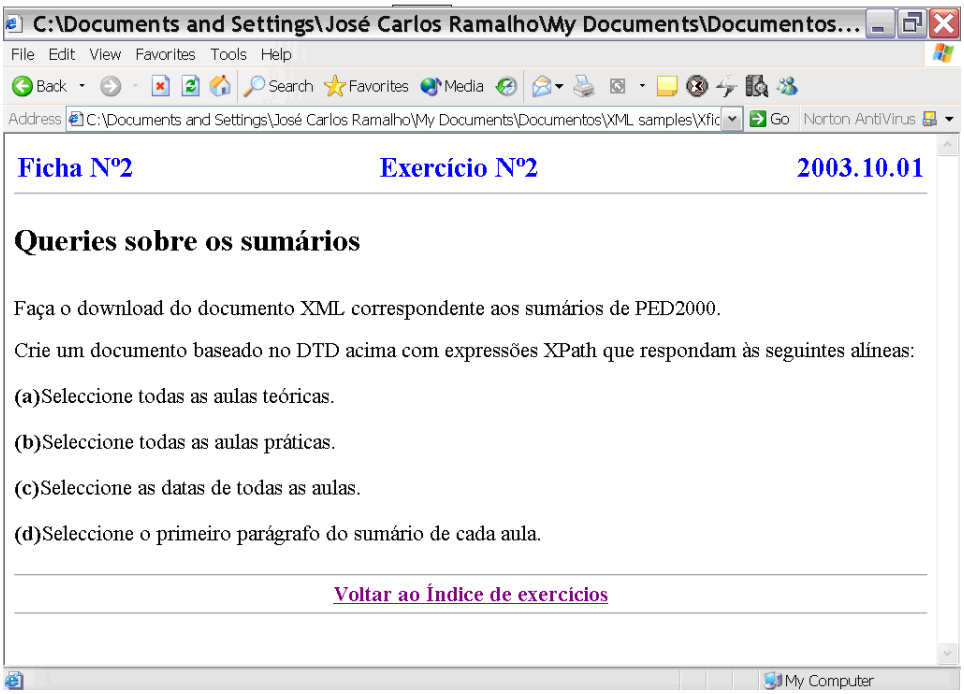

Figure 6: Practical lesson screenshot.

```
          <alínea>Seleccione todos os nomes.</alínea>
          <alínea>Seleccione os versos do segundo terceto.</alínea>
          <alínea>Seleccione os nomes ou lugares começados pela letra L.</alínea>
        </alíneas>
      </enunciado>
    </exercício>
...
```

4 Content Repository / Content Management System

All the documents being produced have to be stored somewhere. Besides that for each XML document being created there are pipeline transformations that have to be executed in order to have the web versions or the PDF versions. The ideal solution would be a Content Management System. The project that aims the development of that system is just starting so only a small idea of it can be given: in this starting project we will use the COCOON framework from Apache; with COCOON a web interface will be created enabling the user to create any of the three kinds of documents described above; the back-end will automatically generate all the versions: web pages, PDF, abstracts, ... The whole process of transformation and deployment of XML documents will be transparent to the user.

For the moment documents are stored in a particular filesystem subtree as the one presented in figure 2. The user must be aware of this structure and sometimes

he will to do some management ”by hand”.

5 Ontology

In another project, called *Metamorphosis* [LRH03a] [LRH03b], ontologies are being used to integrate and to expose heterogeneous information systems on the web. The main idea is to make it simple and fast. To achieve this information systems remain untouched, *Metamorphosis* only extracts a small subset of metadata from the information system in question. This metadata is used to build a semantic network with concepts and relations. These concepts and relations represent knowledge and will be used later to navigate semantically among information resources.

From the various possible formalisms Topic Maps [BBN99] [PH03] [PM01] were chosen due to their abstraction level. They are abstract enough to represent almost everything and strict enough in order to make possible the creations of processors and navigation tools.

The next subsection explain the Topic Maps approach and give an idea of the navigation tools developed so far.

5.1 Topic Maps

Topic Maps is a formalism to represent knowledge about the structure of an information resource and to organize it in ”topics”. These topics have occurrences and associations that represent and define relationships between them. Information about the topics can be inferred by examining the associations and occurrences linked to the topic. A collection of these topics and associations is called a topic map.

Topic Maps can be seen as a description of what is about a certain domain, by formally declaring topics, and by linking the relevant parts of the information set to the appropriate topics [Rat03].

A topic map expresses someone’s opinion about what the topics are, and which parts of the information set are relevant to which topics. Charles Goldfarb [GP01] usually compares Topic Maps to a GPS (Global Positioning System) applied to the information universe. Talking about Topic Maps is talking about knowledge structures. Topic Maps are the basis for knowledge representation and knowledge management.

Enabling to create a ”virtual map” of information, the information resources stay in its original form and so they are not changed. Then, the same information resource can be used in different ways, for different topic maps. As it is possible and easy to change the map itself, information reuse is achieved.

Topic Map architecture was also designed to allow merging between topic maps without requiring the merged topic maps to be copied or modified.

XML Topic Maps (XTM) is basically an XML document (or set of documents) in which different element types, derived from a basic set of architectural forms, are

used to represent topics, occurrences of topics, and relationships (or associations) between topics.

As example we present a small part of a Topic Map that deals with the concepts: person, student, teacher, game, squash, player, grl, jcr, prh.

```
...
<topic id="Person">
    <baseName>
        <baseNameString>Person</baseNameString>
    </baseName>
</topic>
<topic id="jcr">
    <instanceOf>
        <topicRef xlink:href="#Person"/>
    </instanceOf>
    <baseName>
        <baseNameString>José Carlos Ramalho</baseNameString>
    </baseName>
    <occurrence>
        <scope>
            <topicRef xlink:href="#email"/>
        </scope>
        <resourceData>jcr@di.uminho.pt</resourceData>
    </occurrence>
</topic>
...
<association id="jcr-squash-association">
    <instanceOf>
        <topicRef xlink:href="#squash-game"/>
    </instanceOf>
    <member>
        <roleSpec>
            <topicRef xlink:href="#Player"/>
        </roleSpec>
        <topicRef xlink:href="#jcr"/>
    </member>
    <member>
        <roleSpec>
            <topicRef xlink:href="#Game"/>
        </roleSpec>
        <topicRef xlink:href="#Squash"/>
    </member>
</association>
...
```

5.2 Web Interface

ADRIAN has a web interface generator that takes a Topic Map as input and generates a website that enables the navigation in the Topic Map and the access to the resources pointed by the Topic Map occurrences.

Figure 7 gives one of the views of the example Topic Map presented above.

6 Conclusion

As stated along the paper ADRIAN is not completed, it is a prototype. However some parts of it are already being used in E-Learning contexts.

Figure 7: Web Interface snapshop.

There are several small teams developing components for ADRIAN:

Tests and Exams: a second version is being developed; it will support more types of questions and more interaction with students.

Practical lessons: until now this is the component with more users; however they all come from the computer science area; a second version is being developed to cope with lessons from other scientific areas.

Content Management System: a complete Content Management System is being developed in order to make it easier for teachers.

SVG Interface: The current web interface is text based. A graphical one is under development using SVG (standard vector graphics).

The daily use of some of the available components let us conclude that ADRIAN can augment content production productivity.

Looking at the state of the art we can split content production in two different stages: metadata and the content itself. Until now the community has been concerned with the metadata about what is designated by *Learning Object*. Several standards were worked out and some of them are in use by the biggest groups of interest in this area: IMS EML [EML03] and LOM 1484 [LOM03]. These metadata standards were developed for the overall community so they are quite big what can give rise to a poor application. In our Content Provider Applications we have included a small amount of metadata and we have centralized our markup languages

in content. Now we are starting a new project aiming at the integration of our applications with those metadata standards. The main idea is to keep it as simple as possible for the end-user, in this case teachers. There are other projects like EUME Onto [aMLaESaXV03], that are integrating this metadata standards with other content standards like Docbook [WM99]. In our opinion initiatives like these will give birth to systems too complex to use since there are no effort in trying to simplify the final markup language.

References

[aMLaESaXV03] Ricardo Amorim Manuel Lama Eduardo Sanchez Xosé Vila. An educational ontology based on metadata standards. In *2nd European Conference on e-Learning*, 2003.

[BBN99] Michel Biezunsky, Martin Bryan, and Steve Newcomb. ISO/IEC 13250 - Topic Maps. ISO/IEC JTC 1/SC34, December, 1999. http://www.y12.doe.gov/sgml/sc34/document/0129.pdf.

[EML03] http://eml.ou.nl/eml-ou-nl.htm, 2003.

[GM02] Lars Marius Garshol and Graham Moore. The Standard Application Model for Topic Maps. In *ISO/IEC JTC 1/SC34 N0356*. http://www.y12.doe.gov/sgml/sc34/document/0356.htm, December, 2002.

[Gol90] Charles Goldfarb. *The SGML Handbook*. Clarendon Press, Oxford, 1990.

[GP01] Charles F. Goldfarb and Paul Prescod. *XML Handbook*. Prentice Hall, 4th edition, 2001.

[HM01] Elliote Rusty Harold and W. Scott Means. *XML in a Nutshell*. O'Reilly & Associates, 2001.

[KJO+01] K.Cagle, J.Duckett, O.Griffin, S.Mohr, F.Norton, N.Ozu, I.Rees, J.Tennison, and K.Williams. *Professional XML Schemas*. Wrox Press, 2001.

[LOM03] http://ltsc.ieee.org/wg12/, 2003.

[LRH03a] Giovani Rubert Librelotto, José Carlos Ramalho, and Pedro Rangel Henriques. Geração automática de interfaces web para sistemas de informação: Metamorphosis. In *COOPMedia 2003*, ISEP, Porto, Portugal, 10.08 2003.

[LRH03b] Giovani Rubert Librelotto, José Carlos Ramalho, and Pedro Rangel Henriques. Tm-builder: Um construtor de ontologias baseado em topic maps. In *CLEI'03*, La Paz, Bolívia, 09.29 2003.

[MA96] Eve Maler and Jeanne Andaloussi. *Developing SGML DTDs: From Text to Model to Markup*. Prentice-Hall, 1996.

[Meg98] David Megginson. *Structuring XML Documents*. Prentice-Hall, 1998.

[PH03] J. Park and S. Hunting. *XML Topic Maps*, volume ISBN 0-201-74960-2. Addison Wesley, 2003.

[PM01] Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification, August, 2001. http://www.topicmaps.org/xtm/1.0/.

[Rat03] H. Holger Rath. White Paper: The Topic Maps Handbook. Empolis, 2003. `http://www.empolis.com/download/docs/whitepapers/empolistopicmapswhitep%aper\_eng.pdf`.

[SdCVMR03] Daniel Edgar Pinto Soares, Maria da Conceição Vieira Mota, and José Carlos Ramalho. Xexam: uma linguagem de suporte para exames online (e-learning). In *XATA2003 - XML: Aplicações e Tecnologias Associadas*, Vila Verde - Portugal, 02.13-14 2003.

[Tid01] Doug Tidwell. *XSLT*. O'Reilly, Agosto 2001.

[Wil00] Kevin Williams. *Professional XML Databases*. Wrox Press, 2000.

[WM99] Norman Walsh and Leonard Muellner. *Docbook: The Definitive Guide*. O'Reilly, 1999.