Building Learning Management Systems Using IMS Standards: Architecture of a Manifest Driven Approach[1]

José Luis Sierra[*], Pablo Moreno-Ger[#], Iván Martínez-Ortiz[#],
Javier López-Moratalla[#], Baltasar Fernández-Manjón[*]

[*] Dpto. Sistemas Informáticos y Programación. Fac. Informática. Universidad Complutense.
28040, Madrid. Spain.
{jlsierra,balta}@sip.ucm.es
[#] Centro de Estudios Superiores Felipe II. Aranjuez. Spain
{pmoreno,imartinez, jlmoratalla}@cesfelipesegundo.com

Abstract. Among the existing web-based *Learning Management Systems* (LMSs), there is an exponentially increasing need of content interoperability. This has caused the apparition of different standardization initiatives. In this paper we describe our approach to the design of <e-Aula>, a new LMS which adheres closely to IMS standards in an attempt to evaluate the practical viability of those standards. The architecture of our system, focused on the IMS *manifest*, has yielded a powerful and modular system that goes beyond the initial intention of evaluating the proposed standard and can be used as a robust production system in a real environment. We describe our IMS driven approach, as well as an architecture based on this approach that has been implemented using well-known and robust Java based web technologies.

1 Introduction

The IMS proposals [10] are a comprehensive collection of specifications covering the needs of e-learning systems that allow a high durability, reusability and portability of the educational contents. For the last two years, the efforts of our group have been centered on the experimentation with these standardization proposals suggested by the IMS Global Consortium. In this way, we have implemented <e-Aula> [1,7,15], an IMS compliant *Learning Management System* (LMS) supporting several e-learning specifications: IMS CP (for packaging contents), LOM (for expressing metadata) [8], IMS QTI (for tests and assessments) and IMS LIP (for storing information about the learners). In <e-Aula> we use what we have called a *manifest driven approach* to the construction of an IMS based LMS, which is described in this paper.

The structure of the paper is as follows. Section 2 describes the details relative to IMS needed to understand the rest of the paper. Section 3 describes the manifest based approach itself. Section 4 describes the software architecture of <e-Aula>, which is based on this approach. Section 5 compares our approach with other related

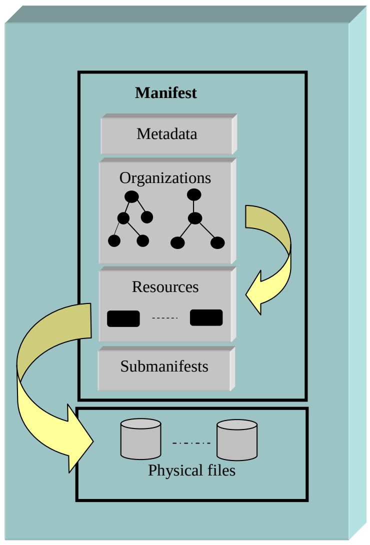
[1] The Spanish Committee of Science and Technology (projects TIC2001-1462, TIC2002-04067-C03-02 and TIN2004-08367-C02-02) has partially supported this work.

works. Finally, section 6 gives some conclusions and outlines some lines of future work.

2 The IMS Content Packaging and the Concept of Manifest

The IMS Content Packaging specification (IMS CP) defines how to aggregate the educational contents into *packages* in order to let different heterogeneous systems interchange these contents. This specification is available in the IMS web site together with the rest of IMS specifications [10]. The structure of these packages is depicted in Fig. 1a. According to this, a package is formed by a set of physical archives with the contents and a *manifest*. This manifest is a XML document that reflects global information about the package, the structure of the contents, their types and their possible organizations. More precisely, the manifest contains:

(a)



(b)

```
<manifest identifier="UML">
  <metadata> (…) </metadata>
  <organizations default="ORG1">
    <organization id="ORG1">
      <title>Organization 1</title> (…)
      <item id="ORG1_3">
        <title>Class Diagrams</title>
        <item id="ORG1_3_0" id-ref="rec3.0">
          <title>Introducion</title>
        </item>
        <item id="ORG1_3_1" id-ref="rec3.1">
          <title>Associations</title>
        </item>
        <item id="ORG1_3_2" id-ref="rec3.2">
          <title>Classes</title>
        </item>
      </item>(…)
    </organization>
  </organizations>
  <resources> (…)
    <resource id="res3.0" type="eaula">
      <file href="3.0.xml"/>
    </resource>
    <resource id="rec3.1" type="eaula">
      <file href="3.1.xml"/>
      <file href="fig3.1.1.gif"/>
      <file href="fig3.1.2.gif"/>
    </resource> (…)
  </resources>
</manifest>
```

Fig. 1. (a) Structure of an IMS package, (b) fragment of an IMS manifest extracted from a course deployed on <e-Aula>. It has been simplified for presentational purposes and for the sake of clarity.

- A section of *metadata* summarizing global (meta)information about the package. This metainformation follows the *Learning Object Metadata* (LOM) specification defined by the IEEE LTSC. LOM is also used to convey the metadata associated with the other elements in the manifest (resources, organizations, and submanifests).
- The description of the package's *resources*. In its simplest form a resource is associated with a physical archive with learning content. It is also possible to describe resources associated with a main file and a set of secondary files. This makes it possible to bundle content sets like a main HTML file and the images

related with this. Also, the resource can include metadata about itself and, most importantly, it defines the type of the content within it. Finally, each resource must have a unique identifier.

- The *organizations* of the resources. Each organization represents a tree structure whose nodes can refer to resources. The nodes in this tree are called *items* and they contain a reference to their corresponding resources using the unique identifiers of the resources. Therefore, an organization provides a tree based structuration of the resources of the package (and thus, of its physical files). It is also to be noted that a manifest can include several organizations, each one providing an alternative way to organize the contents, and therefore a different view of the package.
- The *submanifests*. A manifest can contain other simpler manifests that in turn exhibit the same structure outlined here.

In Fig. 1b a manifest taken from a course deployed on <e-Aula> is depicted. This example shows a tree based organization linked to different resources by means of the usual XML *id-idref* mechanism. In their turn, the resources contain URLs pointing at the actual files.

As a final remark, it is important to note that IMS does not impose any restrictions on the format or type of the content files. Usually LMSs support the most common formats for contents such as HTML and PDF files. In <e-Aula> we also support directly XML files created according to descriptive markup languages specific to each project. This adds all the benefits of the content structuring power of descriptive markup [5].

3 The Manifest Driven Approach

The previous section has presented the IMS manifest as a mechanism that allows the structuration of the contents in an IMS package with interoperability purposes. An IMS compliant LMS can *import* IMS packages and recover the educational contents using this manifest and it can *export* contents by packaging them according with the IMS CP specification. Nevertheless, IMS specifications do not dictate how this LMS must behave out of the scope of the aforementioned interoperability processes. As it has already been mentioned, in <e-Aula> we propose what we have called a *manifest driven approach*. In this approach *the manifest is used as the key element for driving the design and architecture of the entire LMS beyond interoperability processes*.

The manifest driven approach encourages that the LMS maintains continuously a representation of the manifest for each course. In this way, the manifest is used to structure contents not only when interoperating with other systems but also when these contents are managed inside the LMS. This way, for every operation executed on a course there will be a corresponding operation executed on its manifest. Consequently, for each course within the system, its manifest will be the fundamental reference for performing the different tasks related to the course: *presentation*, *edition*, *importation* and *exportation*. Next subsections analyze how the manifest driven approach facilitates these tasks.

3.1 Course Presentation

Course presentation can be naturally addressed by providing suitable browsing and presentation semantics to the organizational information included in its manifest. Indeed:

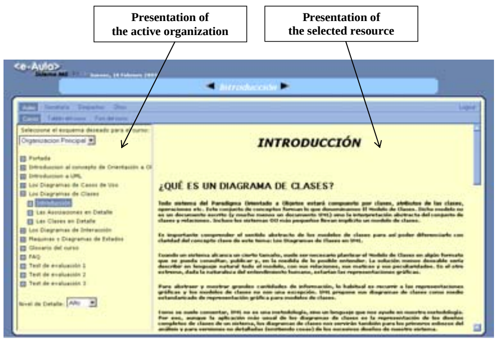

Presentation of the active organization
Presentation of the selected resource
<e-Aula>
Introducción
Seleccione el esquema deseado para el curso:
Organizacion Principal
Portada
Introduccion al concepto de Orientación a O
Introduccion a UML
Los Diagramas de Casos de Uso
Los Diagramas de Clases
Introducción
Las Asociaciones en Detalle
Las Clases en Detalle
Los Diagramas de Interacción
Maquinas y Diagramas de Estados
Glosario del curso
FAQ
Test de evaluación 1
Test de evaluación 2
Test de evaluación 3
Nivel de Detalle: Alto
INTRODUCCIÓN
¿QUÉ ES UN DIAGRAMA DE CLASES?

Fig. 2. Presentation generated in <e-Aula> following the manifest driven approach on the manifest sketched in Fig. 1b. When the user clicks on an item, the associated resource is loaded

- When the learner needs to access the contents of a course, a tree structure displaying the *active* organization can be presented. Since this information is directly encoded in the manifest file, it will not be necessary to explore the content in order to work out its structure. The term *active* is remarked above because the concept of different organizations in a manifest lets the learner choose the organization that fits her better.
- When the learner selects in the tree each item she wants to read, its associated resource and the content type of this resource can be consulted in the manifest. For each supported resource type there is a specific module capable of processing the resource that will take care of the tasks needed to visualize it.

In Fig. 2 the presentation of the course whose manifest is outlined in Fig. 1b is depicted.

3.2 Course Edition

Course edition can be seen conceptually as a similar process to course visualization. Being the manifest the core of the system, edition actions will be directly reflected in the manifest. More precisely:

- The instructor will be allowed to add new resources to a specific course as well as to remove and modify them. This includes stating the content type of the resource. Such operations will be automatically reflected on the manifest (Fig. 3a).
- For managing the organizations (i.e. for structuring the resources) the instructor is offered with a tree very similar to that seen by learners when visiting a course. The instructor can then add or remove nodes in that tree. Those actions are directly reflected in the manifest by the addition/removal of items within the active organization. During this process, the instructor can assign to each node/item a

resource from the previously gathered resource pool, which is also indexed in the manifest (Fig. 3b).

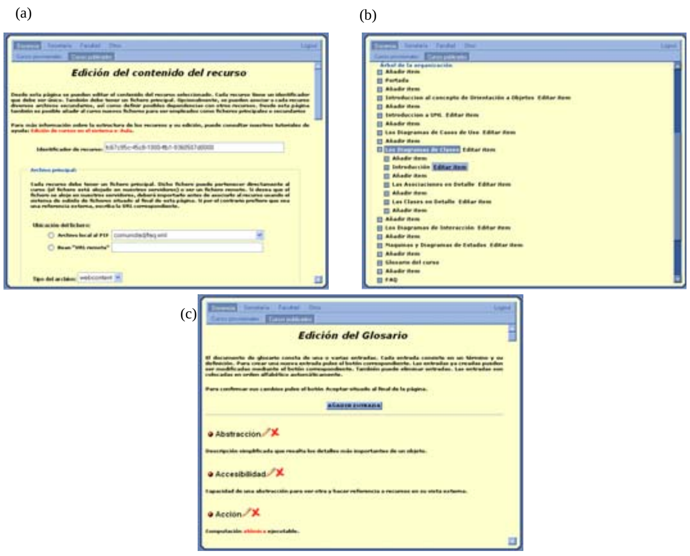

(a)
Edición del contenido del recurso
Identificador de recurso:
Archivo principal:
Ubicación del fichero:
Archivo local al PIF
Bean "URL remota"
Tipo del archivo: webcontent
(b)
Árbol de la organización
Añadir ítem
Portada
Introducción al concepto de Orientación a Objetos Editar ítem
Introducción a UML Editar ítem
Los Diagramas de Casos de Uso Editar ítem
Los Diagramas de Clases Editar ítem
Introducción Editar ítem
Las Asociaciones en Detalle Editar ítem
Las Clases en Detalle Editar ítem
Los Diagramas de Interacción Editar ítem
Máquinas y Diagramas de Estados Editar ítem
Glosario del curso
FAQ
(c)
Edición del Glosario
AÑADIR ENTRADA
Abstracción
Accesibilidad
Acción

Fig. 3. Edition operations: (a) Inclusion of a new resource, (b) edition of the tree of an organization, (c) edition of an <e-Aula> *Glossary* resource.

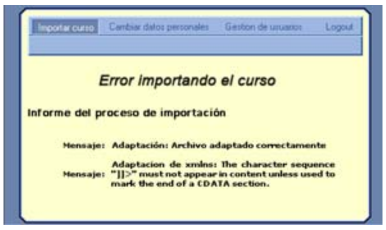
- For creation and modification of actual contents associated with the resources it is possible to adopt a similar strategy to the one that was used to display the selected resources upon learner petition. For every content type there is an associated edition module. When the instructor selects in the tree a resource to be modified, its edition module is launched thus allowing her to modify it using the web interface (Fig. 3c).

In <e-Aula> we have considered that course structure management (i.e. addition, removal and organization of resources) is more important for our evaluation purposes than content edition itself. Thus, <e-Aula> currently only includes edition modules for the content types specific of this system (e.g. <e-Aula> content pages, course presentations, glossary and F.A.Q.). Other common resource types, although supported for visualization, are not currently supported for edition (e.g. PDF, HTML, etc.). Those resources can be uploaded to the system but once there they cannot be directly edited. Typically, if instructors want to modify such files they will edit them with their usual edition tool and will upload them into the system. Nevertheless, due to the modularity of the manifest driven approach, the incorporation of new editors to <e-Aula> will be straightforward.

3.3 Course Importation

The importation facility allows the incorporation of packages produced in other LMSs. Importation is always a problematic functionality. Even though the IMS CP aims at facilitating this kind of processes, it is very broad. In effect:

- Several content types are permitted and there are no specific restrictions regarding the formats. Therefore it will be usual to find incoming packages including resources of types unknown or unsupported by the system.
- The importation facility must also manage situations that are more difficult. Effectively, the standards are not mature enough and they are subjected to evolution. Moreover, IMS incorporates an extension mechanism allowing vendors to add their own extensions to the standards. Consequently, an imported package can follow an unsupported IMS CP version or include unsupported vendor specific extensions.



Error importando el curso

Informe del proceso de importación

Mensaje: Adaptación: Archivo adaptado correctamente

Mensaje: Adaptacion de xmlns: The character sequence "]]>" must not appear in content unless used to mark the end of a CDATA section.

Fig. 4. A snapshot of the <e-Aula> importation system. Help of the user is required to adapt an incoming package with invalid syntax.

The challenge of a sophisticated importation system, such as the included in <e-Aula>, is to try to understand and adapt the incoming packages. Due to the aforementioned factors this adaptation can require the help of a human user (Fig. 4). The manifest driven approach simplifies this complex process because mostly all the work can be done over the manifest. The manifest is indeed a clean and powerful element in which to centralize the efforts when designing an importation system. Any modification to the standards, any kind of strange resource type and even the version of the standard are always reflected in the manifest. That means that the steps required to import a package can be deduced from a deep examination of its manifest. In addition, many of the changes in the package produced by the execution of these steps can be limited to modifications of the manifest file. For example, when there is no possibility of adapting an alien resource and it must be removed, the action actually performed is to erase from the manifest the references to this resource. The content is still in the package but it does not cause any problems because the resources are always accessed using the manifest. The old manifest can be backed up so that the offending content can be revived when needed (and supported).

Upon completion of all the adaptation procedures during importation, the resulting manifest file is valid from both the standard's and the system's points of view. It can thus be safely displayed and/or edited.

3.4 Course Exportation

On many systems, the process of exporting content following the proposed standards can be a complex task, because it might imply scanning the course's internal representations in order to recover the IMS structures. However, with the manifest driven approach exportation is extremely simple, because the courses are internally represented by their manifests. Therefore the courses maintain full compatibility with IMS CP during their whole life in the LMS, being unnecessary to perform any kind of adaptation or additional processing beyond zipping the content together with the manifest file and storing it all in the file system

4 An LMS Architecture based on the Manifest Driven Approach

During the inception phase of <e-Aula>, some important requirements were identified. Keeping in mind that IMS specifications were still young and thus prone to be changed (perhaps even disregarded by the community) the system should be as flexible as possible. It should not be a fixed application that once developed goes into production, but a lively, ever changing environment. For a few years following its construction, the system should evolve, adapting to the evolution of the standards and the irruption of new concepts. This means that internally complex monolithic systems should be avoided promoting instead modular systems in order to assure maintainability.

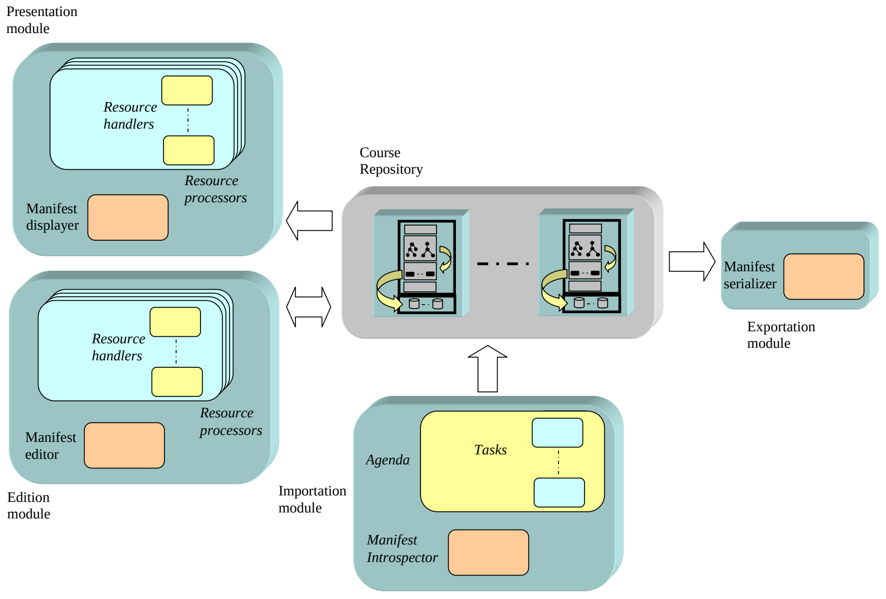


Fig. 5. Architecture of a manifest driven LMS

The manifest driven approach described in the previous section can be implemented by means of a modular and powerful architecture meeting the aforementioned requirements. This architecture has been implemented in the <e-Aula> system and tested in the development of several courses at *Complutense* University of Madrid (Spain). The next subsections describe this architecture. Subsection 4.1 gives an overview of the architecture. Subsections 4.2, 4.3 and 4.4 detail its different aspects. Finally, subsection 4.5 summarizes how this architecture is implemented in <e-Aula> using well-known web development standards.

4.1 Overview of the Architecture

In Fig. 5 the architecture for an LMS based on the manifest driven approach is depicted. The architecture must support the four main tasks described in section 2. Therefore a distinct module is included for each one of these tasks.

The architecture is organized around a *storage* module where the different courses are stored. This module maintains an explicit representation of the IMS manifest for each course. According to the manifest driven approach, all the operations performed on the courses are reflected on these representations.

The architecture of the presentation and the edition modules are similar and based on a delivery policy with *resource processors* able to select the most suitable *resource handlers* for each type of resource. They are detailed in subsection 4.2 and subsection 4.3. On its turn, the importation module exhibits an *agenda based architecture* [2], which is suitable to cope in a modular way with the complexities of this operation (see subsection 4.4). Finally, as mentioned before the structure of the exportation module is straightforward due to the explicit representation of the manifests in the storage module.

4.2 The presentation module

The previous section has already outlined the difficulties faced when building a LMS supporting the broad range of content types that an IMS based course can contain. Indeed, a mechanism that can react to the different content types and process them accordingly is required. In the LMS vocabulary, such a mechanism is called a *delivery system*. Following the underlying concepts of the manifest driven approach a three-step process is suggested which is parallel to the three layers defining the content type of a resource:

- The first step begins when a course is loaded. The action taken is to consult the required *application profile* in the manifest. The concept of *Application Profile* (AP) is the IMS term to designate a customization of a standard to meet the needs of particular communities of implementers with common application requirements. For each supported AP there is a different *resource processor*. From the perspective of the implementation, the presentation module will be equipped with a table listing the relations between the different APs and their corresponding resource processor objects. When the course is loaded, the corresponding resource processor is invoked. That processor will be responsible of handling all the requests related with the resources until a new course is loaded. After this step is completed, the system examines the manifest, loads the default organization and displays a tree reflecting the structure of the items. This is made by a component called *manifest displayer*.
- The second step is triggered whenever the learner clicks a node of the tree (which represents an item). The system queries the manifest about whether that item has an associated resource. If it does, the content type of the resource is consulted in the manifest. That information is transmitted to the active resource processor. Just like in the previous step, the resource processor contains a table relating each

content type to an appropriate *resource handler*, which is a module capable of processing that content type.
- In the third step, the resource handler gets control and performs all the operations required by the content type. Such operations could include, among other things, formatting the content, adapting it to the learner's profile, adapting the content to the client's device or storing statistical data about the learner's visit to the item before and after the visit itself.

Notice that this organization allows the incorporation of new resource processors and new resource handlers in a modular and transparent way without interfering with the behavior of the presentation module.

4.3 The edition module

The edition module must handle two different problems: the edition of the structure of the course (edition of the manifest) and the edition of the course content (edition of the associated resources):
- For the edition of the resources it is possible to adopt the same delivery strategy followed in the edition module. Therefore, the content edition environment can be architected in a similar way, reusing the ideas used in the presentation module to trigger the appropriate resource handler.
- The edition of the structure of the course is carried out by a *manifest editor* component. This component reuses the ideas employed to visualize the navigation tree in the presentation module, adding the functionality to add/edit/delete nodes and link them to the resources edited.

As with the presentation module, the architecture of the edition module allows the incorporation of new edition facilities in a straightforward manner.

4.4 The importation module

The architecture of the importation module must be flexible enough to cope with the complexities of the importation process. This must implement a flexible behavior capable of reacting when confronted with different problems, even with the possibility of querying the user when more information is needed to perform the importation. Because of this, we propose an implementation based on an *agenda* similar to the proposed in [2] to simulate discrete systems.

According to the agenda based organization, when a package is imported the system parses the manifest, adding new tasks to the agenda to resolve the troubles encountered during the scan. These tasks (especially those that involve a query to the user) can create other tasks and add them to the agenda if it is needed for their resolution. In this manner, a complex process that requires a heterogeneous set of actions is dynamically split into simple tasks. More precisely:
- The importation begins with a deep scan of the manifest. This is carried out using a component called the *manifest introspector*. This generates a report that profiles, among other things, whether the manifest is a well-formed XML document, which version of the standard it follows, which other schemas (if any) are needed to

understand it and under what AP it has been developed. This report is presented to the user, who decides how to continue.
- Depending on the user's response, the system generates a list of initial tasks for the agenda and starts working on them. Such tasks may include the modification of the manifest file, modification of the AP, adaptation of some resources or physical installation of the package.

This agenda based implementation makes it possible to add new types of adaptation tasks during importation in a transparent way (e.g. to adapt a new content type). Such an organization dramatically enhances the modularity and maintainability of this complex subsystem.

4.5 Implementing the Architecture

This architecture has been implemented in <e-Aula> using Java technologies in order to maximize maintainability, extensibility and robustness of the resulting implementation. More precisely, we have based our implementation on the Sun Microsystems' J2EE platform [11] complemented with the Apache Foundation's Struts framework [3]. From J2EE we adopt the multi-tier organization according to which applications are layered in different tiers (*client*, *web*, *business* and *persistence*). From Struts we adopt the organization in terms of the classical *Model-View-Controller* (MVC) design pattern. With this, each one of the presentation, edition, importation and exportation modules has its own view and a controller, which interact with a common model represented by the course repository and the manifests within it, as suggested by Struts. In addition they are disposed on a layered organization, as suggested by J2EE.

5 Related Work

LMSs have been widely adopted by institutions and instructional designers in order to fulfil certain needs and requirements in a field of ever increasing demands for effective education and training [4]. In many of these systems considerations like adherence to standards or content exchange are often secondary goals because, although being desirable, there are not usually a key point for customer satisfaction (be it a learner or an institution interested in buying the system to deploy their own content). On the contrary, we follow a more scholarly approach. Therefore, we focus on the evaluation of standards and the research in modular architectures, being aspects like user friendliness or the support for a wide spectrum of high quality content formats a secondary objective.

Many initiatives have adopted IMS as a basic interoperability mechanism. They range from commercial product like WebCT [18], which can be purchased by companies in order to deploy their own content, to initiatives like ADL-SCORM [16], which provides complementary specifications for obtaining high-quality content and systems in a variety of fields. While these initiatives pay only attention to IMS standards when it comes to interoperability issues (e.g. content exportation/importation), our manifest driven proposal goes a step forward. Indeed, we defend the use of the specifications (in particular, the IMS manifest) as central mechanisms to conceive, architect, design and implement the system.

There are alternative approaches to architect an LMS. One of the most relevant is proposed by the SAKAI project [14]. SAKAI's goal is to join different open source course management systems and related tools in a single open source architecture. To achieve this goal the SAKAI software is based in the MIT Open Knowledge Initiative (OKI) [13], which is a service-oriented framework to develop a Course Managment System. This approach focuses on the general architecture of a LMS, allowing the addition of the existing tools. The choice of specific standards is delegated on the different tools that implements local services. Our manifest driven approach, being more oriented to the development of such local services, can be complementary to this kind of proposals.

More similar to the <e-Aula> LMS are the large amount of e-learning tools based on scripting technologies (e.g. systems like ILIAS [9], Moodle [12] or Dokeos [6] are written in PHP, while WebCT Campus Edition [18] is written in Perl). While these products prove that it is possible to build large scale LMSs based on the mentioned technologies, in our opinion it requires a large amount of effort by developers when it comes to adding new functionalities and it affects their maintainability. Indeed, new developments, in particular large scale LMSs such as WebCT Vista [18], are being developed using Java technologies. On its turn, Java technologies are more robust and they meet better complementary requirements like modularity and maintainability. These have led us to adopt these technologies in <e-Aula>.

6 Conclusions and Future Work

In this paper we have described our manifest driven approach to architect IMS based LMSs. This approach is the result of our efforts in the development of <e-Aula>, a system aimed to evaluate different e-learning standards and e-learning modular architectures.

The manifest driven approach facilitates the main tasks contemplated in the LMS: course presentation, edition, importation, and exportation. In addition, this approach leads to a modular architecture that facilitates extensibility. Adding support for a new content type is just a matter of writing the code needed to prepare and display content in that format. This is also true when adding a new content editor to the system, or a new task for the agenda. The high degree of modularity of the architecture also enhances maintainability. It is easy to find the points in the source code where any changes could be needed, and these changes can be done will little impact on the rest of the system. While the complexity of the architecture is high (in terms of number of classes and files in the resulting implementation), this is the price to pay for achieving a very high degree of modularity, and therefore a better extensibility and an easier maintenance.

Our Java based implementation preserves the benefits of the architecture and also adds a high degree of robustness. Nevertheless we should point out as a drawback the need of more computing power on the server when compared with lighter applications based on scripting solutions (like PHP). While the choice between a large and robust system and a lightweight application will depend on the exact needs of each learning environment, the J2EE / Struts based solution is well suited for our purposes because it allows a continuous evolution of the system to accommodate our evaluation and research needs.

Currently the architecture based on the manifest driven approach is fully implemented in the <e-Aula> system. This LMS is fully functional and adheres strictly to the IMS CP standard. There is also partial support of the IMS QTI standard, which is also functioning as a stand-alone application, and some built-in adaptability features for different client platforms and user levels. As future work we are planning to incorporate advanced user modeling capabilities. The IMS LIP (Learner Information Profile) standard may be a supporting aid in this task. In addition, we are planning to apply further the *document oriented* approach promoted in [17] for both the incremental definition of new types of resources and the incremental construction of their associated handlers. Finally, we are planning to involve to field experts in the development of high-quality content that fully exploits the functionalities of our system. The advanced importation features incorporated in <e-Aula> should facilitate this task.

References

1. <e-Aula>. eaula.sip.ucm.es
2. Abelson, H.; Sussman, J.; Sussman, J. Structure and Interpretation of Computer Programs - Second Edition. MIT Press. 1996
3. Apache Struts. struts.apache.org
4. Avgeriou, P.; Papasalouros, A.; Retalis, S.; Skordalakis, M. Towards a Pattern Language for Learning Management Systems. Educational Technology & Society, 6(2), pp. 11-24. 2003
5. Coombs, J. H.; Renear, A. H.; DeRose, S. J. Markup Systems and the Future of Scholarly Text Processing. Communications of the ACM, 30 (11), pp. 933-947. 1987
6. Dokeos. www.dokeos.com
7. Fernández-Manjón,B.; Sancho, P. Creating cost-effective adaptive educational hypermedia based on markup technologies and e-learning standards. Interactive Educational Multimedia 4. 2002
8. IEEE Standard for Learning Object Metadata. IEEE Standard 1484.12.1-2002. 2002
9. ILIAS. www.ilias.uni-koeln.de/ios/index-e.html
10. Instructional Management System Global Consortium. www.imsglobal.org
11. Java 2 Enterprise Edition. java.sun.com/j2ee/
12. Moodle. moodle.org
13. OKI Project. www.okiproject.org
14. SAKAI Project. www.sakai.org
15. Sancho, P.; Manero, B.; Fernández-Manjón, B. Learning Objects Definition and Use in <e-Aula>: Towards a Personalized Learning Experience. Edutech:Computer-Aided Design Meets Computer Aided Learning, pp177-186. Kluwer Academic Publishers. 2004
16. Shareable Content Object Reference Model SCORM. www.adlnet.org
17. Sierra, J.L.; Fernández-Valmayor, A.; Fernández-Manjón, B.; Navarro, A.. ADDS: A Document-Oriented Approach for Application Development. Journal of Universal Computer Science, 10(9), pp 1302-1324. 2004
18. WebCT. www.webct.com