

# Towards Standards-driven Educational Content Providers

Luis Anido, Judith Rodr'iguez, Manuel Caeiro, Juan Santos

*Departamento de Ingenier'ya Telem'atica*

*Universidad de Vigo, Spain*

*E-mail: [lanido](mailto:lanido@det.uvigo.es), [jestevez](mailto:jestevez@det.uvigo.es), [mcaeiro](mailto:mcaeiro@det.uvigo.es), [jsgago](mailto:jsgago@det.uvigo.es)@det.uvigo.es*

## Abstract

*Currently, both public and private institutions involved in the e-learning game are gathering their efforts to obtain standards and recommendations supporting interoperability among heterogeneous platforms. Results from this standardization process allow Educational Content Providers (ECPs) to offer courses ready to use in any learning platform supporting the corresponding standards.*

*This paper presents a set of software components with open interfaces that facilitate the development of distributed, interoperable and standards-driven Educational Content Providers. To materialize these interfaces or services, CORBA was selected as the technological supporting infrastructure.*

## 1. Introduction

Development of educational materials for course delivery at a particular Learning Management System (LMS) is generally a difficult task that involves several actors: learning professionals or pedagogues that decide the elements (text, images, videos, evaluation tests, simulators, etc.) that must be present in the course and how they must be organized; technicians that materialize the outlines from the pedagogues into digital resources, probably with the help of graphical designers; and, sometimes, external expert reviewers that provide instructions, comments or reports to opportunely improve the contents.

At present, most of this work, particularly that developed by the technicians, must be guided by the proprietary data models and formats defined in the "a priori" selected educational platform that will "execute" the final product. Utilization of the developed course in another different learning system implies re-manufacturing of the contents according to the surely distinct formats supported by the second platform. So, the potential clientele of a self-governing Educational Content Provider, i.e. an autonomous entity or institution that develop, assemble, store, publish and dispense pre-

manufactured courses ready to be executed in a LMS, is restricted to the institutions that use a particular e-learning system. Of course, the ECP can develop its products in several formats, but always it will be in a limited number.

Fortunately, in the last years, institutional users of educational software joined their efforts to achieve standards and recommendations that will promote and facilitate the existence of LMS products from different vendors that will be able to launch the same executable learning content. In this way, the proliferation of ECPs, acting as the actual traditional libraries that sell the books used at classrooms, is likely in the near future.

In this paper, we present a set of software components that will facilitate the construction of such providers.

## 2. Involved Standards

Several data models and formats must be standardized in order to allow that interoperability of the learning contents. Next, the most relevant ones are listed, being many of them addressed by different standardization bodies:

- The first needed standards are related to the format of individual contents, i.e. learning objects. Usually it is assumed that any material displayable in a Web browser is interoperable in nature. This is partially true for static contents. However, dynamic contents, that access to data managed by the LMS, require standardized APIs to allow that interchange of information [5]. Additionally, in order to not impose a fixed presentation aspect to the learner, specialized formats are required. Thus, we can find, for example, data models that allow storing of all the information needed to "on the fly" render and process the questions that compose a test [14];
- Individual learning objects are grouped in an organized manner to produce suitable units of instruction or complete courses. So, definition of powerful and versatile formats to represent these structures, with their corresponding behaviors are needed [4, 7];
- Packaging formats to encapsulate all the elements composing a course in a single unit (i.e. a individual

- file) are, also, being standardized [1, 4];
- The format of metadata records to describe and identify the developed contents must be standardized in order to allow the storing, indexing, searching and retrieving of them from a database or repository [6, 10];
- Finally, the available services (in the context of a individual ECP or LMS) for a particular packed course must be completely and conveniently described. So far, there is no concrete proposal by any of the main institutions involved in the e-learning standardization process. The reader interested can find a recommendation to describe services offered over learning objects in [2].

### 3. Reusable Software Components

Development of software products is always accelerated by the use of reusable pre-manufactured components with open and well-defined interfaces. In this sense, we have developed a set of specifications, arranged in a Domain CORBA Facility [13], which allows the construction of such reusable components to be used by the developers of final systems to build interoperable, distributed and standards-compliant Educational Content Providers.

As stated in [11] a well-conceived service or facility is always based on an underlying semantic model that is independent of the target platform (operating system, implementation language, etc.). So, in order to maximize the utility of the domain facility, we have conveniently separated the modelling and specification of the services into Platform Independent Models, abstracting away any technical detail, and Platform Specific Models, taking into account the characteristics and restrictions of the target implementation platform, CORBA in this case. All the resultant models are expressed in UML [8].

#### 3.1. Platform Independent Models

The obtaining of the developed models was based on the Unified Software Development Process [9] and guided by the recommendations by other authors [3]. Thus, a series of phases derived into a set of models, from the most abstract to the most concrete.

Once established and defined the high level components of the system in a Reference Model, the analysis of the functional requirements from both clients and designers viewpoint allows the identification of the elements (or analysis classes) needed to deploy the identified functionality. We just consider that analysis classes offering basic common

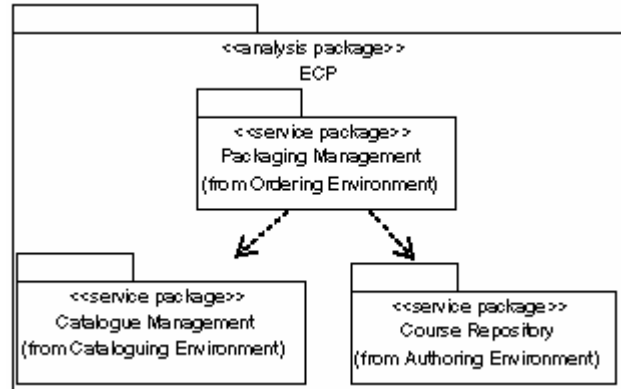


Figure 1. Reference Architecture.

services useful for the development of distributed and interoperable systems. These classes were grouped into service packages where they have a strong relationship among them, manage the same underlying information models and, therefore, tend to change together. Figure 1 shows the identified services packages, defining our Reference Architecture:

- *Course Repository*: It handles the creation, storage and management of learning objects. ECPs include facilities to ease the development of new learning objects, including fine-grain management. Available course structure format specifications have been considered.
- *Catalogue Management*: This subsystem offers services to support the development of catalogues for the learning objects provided by the ECP, including searching services, both external (to locate courses to be transferred to clients) and internal (to locate those learning objects that can be reused to build new contents). Metadata specifications are used to describe the learning objects available at the ECP.
- *Packaging Management*: Developers of ECPs use this subsystem to create applications to transfer all the resources related to a course (learning objects, course structure, metadata descriptions) among heterogeneous platforms in encapsulated units ensuring there is no loss of information. Specifications for content packaging are used to create packages.

All the responsibilities of the classes included in each service package are completely identified and textually described. Figure 2 shows the identified analysis classes and their relations. To complete the Platform Independent Models, collaboration diagrams showing how a specific functionality, identified at the requirements capture phase, can be realized in terms of analysis classes.

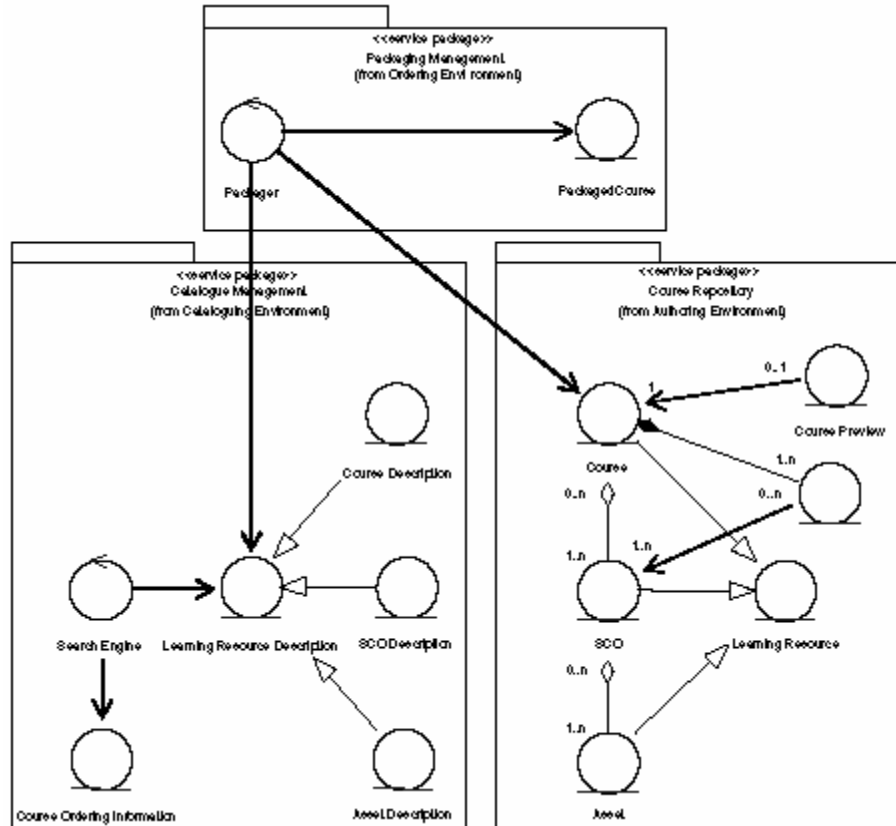


Figure 2. Analysis classes included in the three service packages.

### 3.2. Platform Specific Models

As the Reference Architecture includes only analysis classes, it is implementation independent and purely conceptual. It is the starting point in the elaboration of a Design Model using the constructs and concepts of a specific development environment for building distributed applications. As our final objective is the definition of a potential Domain CORBA Facility, we chose CORBA as the implementation/ deployment environment.

In the Design Model the service packages identified in the Reference Architecture derive in a set of design service subsystems that materialize the responsibilities of the analysis classes in the concrete programming language, CORBA Interface Definition Language (IDL), in our case. All the resultant artifacts were also modeled using the UML Pro- file for CORBA [12]. This profile, adopted by the OMG in 2000, specifies how to use UML in a standard way to define CORBA IDL interfaces, structs, unions, etc.

As an example, Figure 3 shows the interfaces identified for the *Packaging Management* design subsystem. The

Package Factory interface acts as a factory of objects.

It defines “introspection” methods to discover what data models, both for course structures and for packaging, are supported by the final implementation. In this way, the objects implementing the interfaces can implement data models already defined and others that can appear in the future.

The *Package* interface defines methods for packaging the contents and the structures supplied in the create methods of the *Package Factory*. It also defines methods for adding new structures to the resources the course is composed of. Clear interface definition is a key point to guarantee interoperability. Therefore, each method is properly described using English text and UML diagrams. For example, below we show the IDL interface definition for the method:

```

createPackage().
Package createPackage(
in CL_CommonTypes::CourseIdSeq course_id,
in CL_CommonTypes::RepositoryId content_rep,
in CL_CommonTypes::RepositoryId metadata_rep,
in CL_CommonTypes::CourseStructureIdSeq
course_structures,
in CL_CommonTypes::SchemaIdSeq
course_structure_schemas,
in CL_CommonTypes::SchemaId
packaging_schema)
raises (CL_CommonExceptions::InternalError,
CL_CommonExceptions::InvalidParameter,

```

```

CL_CommonExceptions::UnsupportedSchema,
MultipleStructuresNotAllowed,
CL_CommonExceptions::XMLError,
InvalidCourseStructure,
CL_CommonExceptions::SchemaError);

```

This method is used to obtain a reference to a Package object in order to wrap up a course into a standardized single file. There is no reference in the method signature to any specific packaging model. Therefore, IMS proposal or other coming in the future can be used without changing the method signature. Input parameters are:

1. *course id*. Identifier for the course to be packaged used through the ECP.
2. *content rep*. Reference to the content repository the course contents must be read from.
3. *metadata rep*. Reference to the catalogue repository where the metadata description on this course must be read from.
4. *course structures*. The different course structures for this course to be included in the package. The same contents may offer different external behaviors depending on the actual course structure being used.
5. *course structure schemas*. Data models used to define each previously mentioned course structures to be packaged.
6. *packaging schema*. Identifier for the actual content packaging data model to be used (e.g. IMS Content Packaging). There is no restriction in terms of the proposals for content packaging that can be used.

Method definition also includes descriptions of those situations when an exception may be raised. These exceptions may indicate that a specified data model is not supported by the implementation (UnsupportedSchema, content packaging model not supported; InvalidCourseStructure, course structure schema not supported). They may report on semantic violations (MultipleStructuresNotAllowed, the content packaging model being used does not allow for multiple course structures to be included in the same package). They may indicate syntactic errors (XMLError, SchemaError, InvalidParameter) or even unexpected implementation-dependent exceptions (InternalError).

#### 4. Implementation Prototype

Development of new interoperable Educational Content Provider involves two stages: (1) the development of components, or off-the-shelf purchasing, implementing standardized interfaces providing interoperability; and (2) the customization of concrete applications adding a wrapper over the underlying interoperable components. The former provides

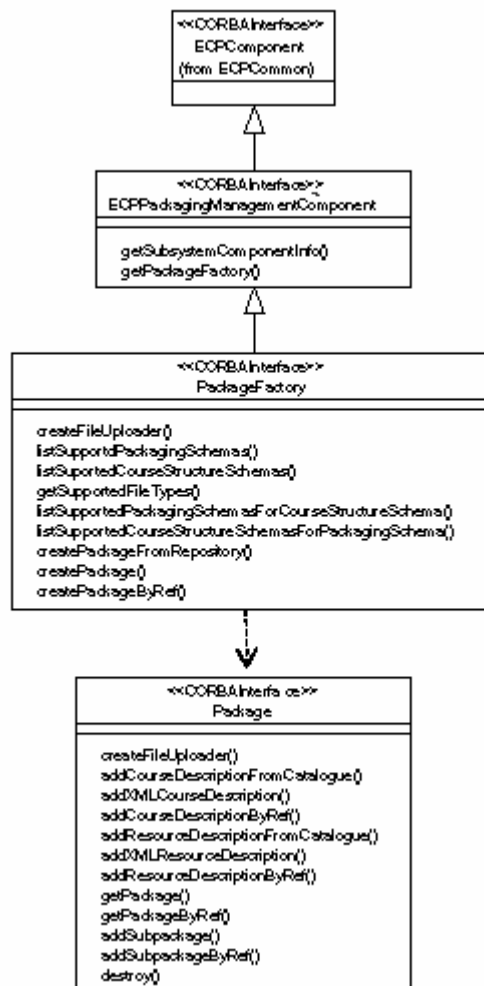
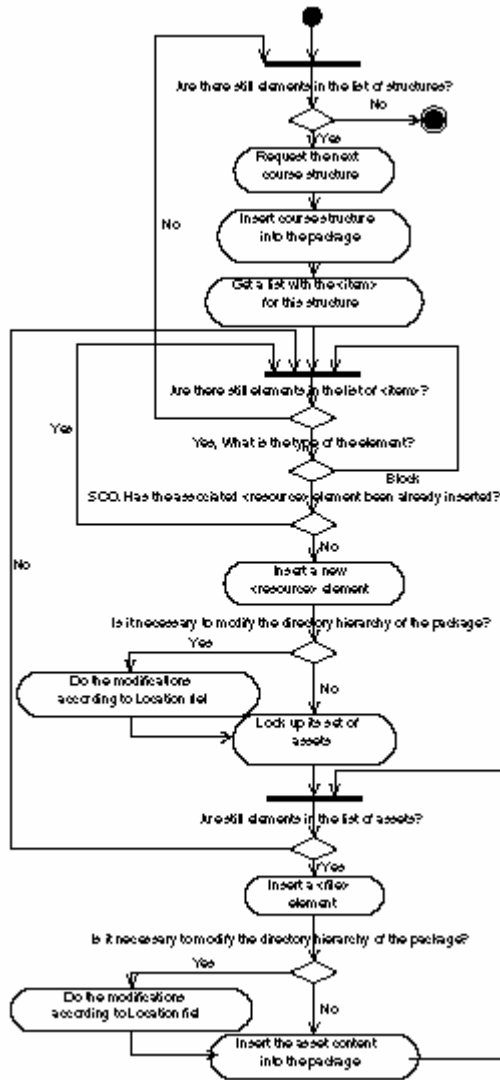


Figure 3. Interfaces from the *PackagingManagement* design service subsystem.

interoperability, the latter offers value-added services. Underlying standardized components can be developed using different programming languages and may offer different performance. However, provided they are compliant with the defined interfaces, interoperability would be guaranteed and new systems could be developed incrementally. Also, updates and upgrading of already existing ECPs is easier as older components can be replaced by new ones. This components exchangeability can be used to replace components implementing a given standardized data model for a different one. Note, that our defined interfaces do not impose any concrete data model and provides introspection mechanism to discover at runtime which data model is actually implemented. This is especially important in a so active field as e-learning standardization with different proposals for standards in progress. We have developed a prototype whose main aim was to prove that our defined interfaces could be implemented and used successfully. For example,



**Figure 4. Packaging algorithm.**

Figure 4 describes the algorithm we implemented to package courses using the IMS Content Packaging Specification [1].

## 5. Conclusions

Currently, e-learning standardization comes up with several different data models to allow interoperability among heterogeneous platforms. However, few institutions propose software components implementing those standards and offering access to them via standardized interfaces. This paper proposes a service software architecture offering development of new Educational Content Providers from reusable software components. Open interfaces are defined for each of them allowing heterogeneous systems to interoperate in a distributed environment like the Internet.

## 6. Acknowledgements

We want to thank “Xunta de Galicia” and “Ministerio de Ciencia y Tecnología” for their partial support to this work under grants “Arquitecturas distribuidas para Teleservicios” (PGIDT00TIC32203PR) and “CORBA Learn: Interfaz de Dominio guiada por Estándares para Aprendizaje Electrónico” (TIC2001-3767) respectively.

## 7. References

- [1] T. Anderson and M. McKell. IMS Content Packaging Information Model. Technical Report Version 1.1.2, IMS, August 2001.
- [2] L. Anido. *Contribution to the Definition of Distributed Architectures for E-learning Systems using CORBA*. PhD thesis, Departamento de Ingeniería Telemática. E.T.S.I Telecomunicación, December 2001.
- [3] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, April 1999.
- [4] P. Dodds. The SCORM Content Aggregation Model. Technical Report Version 1.2, ADL Initiative, Oct. 2001.
- [5] P. Dodds. The SCORM Run-Time Environment. Technical Report Version 1.2, ADL Initiative, Oct. 2001.
- [6] W. Hodgins. Draft Standard for Learning Object Metadata (LOM). Draft Version 6.0, IEEE LTSC, March 2002.
- [7] J. Hyde. CMI Guidelines for Interoperability. Technical Report Version 3.5, AICC, April 2001.
- [8] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Modelling Language User Guide*. Addison Wesley Longman, 1999.
- [9] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.
- [10] M. McKell and S. Thropp. IMS Learning Resource Meta-Data Information Model. Technical Report Version 1.2.1, IMS, September 2001.
- [11] J. Miller and J. Mukerji. Model Driven Architecture (MDA). Technical report, OMG, July 2001.
- [12] OMG. UML Profile for CORBA Specification. Technical specification, OMG, October 2000.
- [13] OMG. Catalog of OMG Domain Specifications. Information available at [http://www.omg.org/technology/documents/formal\\_2002](http://www.omg.org/technology/documents/formal_2002).
- [14] C. Smythe, E. Shepherd, L. Brewer, and S. Lay. IMS Question & Text Interoperability: An Overview. Technical Report Version 1.2, IMS, February 2002.