Delivery of Learning Design: the Explor@ System's Case

Gilbert Paquette, Olga Marino, Ileana de La Teja, Michel Leonard, Karin Lundgren-Cayrol

Chapter 16

Delivery of Learning Design: the Explor@ System's Case

Primary author(s): Gilbert Paquette and Olga Marino,

Other author(s): Ileana de la Teja, Michel Léonard, Karin Lundgren

ABSTRACT

The IMS Learning Design Specification (IMS-LD) presents new challenges to learning delivery systems. To comply with this specification, delivery platforms must understand different learning strategies and course structures, must manage multi-actor environments, must allow for standard learning objects integration, must deal with conditions and rules to be validated on runtime and must support notifications.

In this chapter, we take a look at these requirements from the viewpoint of an open delivery system, Explor@-2. Explor@-2 is the result of a research stream that started a decade ago at Télé-université's LICEF research center. Explor@ has focused, right from the beginning, on a resource (or learning object) management orientation, making it possible to assemble a set of educational support tools, documents and services to be shared across all programs, courses or activities delivered by an organization. The chapter presents Explor@-2's basic learning design information model and analyses how Explor@-2 can deal with IMS-LD compliant courses – how it can deliver units of learning modelled either with the IMS-LD level A specification or with the IMS-LD level B or C specifica-

tions. The chapter ends with some conclusions on future research and development to be done in order to build a fully IMS-LD compliant delivery system as well as on some promising directions for developing powerful and adaptive distance learning environments.

Introduction

In chapter 6, we have described a methodology, MISA, for designing and developing learning systems as well as two software tools, MOT and ADISA, developed to support this methodology. The relationship between the design products of the methodology and the IMS-LD specification has also been shown. In this chapter, we look at the IMS-LD specification from a delivery viewpoint by presenting the Explor@-2 delivery system (Paquette 2001, 1999). As Explor@-2 delivers courses designed using the MISA methodology or another method, it must represent the four models: knowledge model, instructional model, media model and delivery model.

This chapter is divided into four sections. Section 1 provides a general presentation of Explor@: its evolution and current global architecture. Focusing on the instructional model, section 2 presents an UML model of Explor@-2 learning design information model as well as its instructional activity structure editor. Section 2 shows how we can use this editor to build a representation of an IMS-LD Method that can be delivered using Explor@-2. Further, the components of Explor@-2 that correspond to the IMS-LD specification will be presented. Although integrating IMS-LD level B and C in Explor@-2 should be straightforward, we propose in section 4 an alternative approach to deal with personalization, advising and notification, which suggests further interesting studies on how to design and integrate external global applications (advisors, managers, helping systems, intelligent tutors, etc.) to the IMS-LD specification. The conclusion gives some hints on where to go next and on how to handle the inherent complexity of powerful, flexible distance learning systems.

1- Explor@-2 General Presentation

Explor@-2 is the result of a research stream that started a decade ago at Télé-université's LICEF research center. The initial research efforts (Paquette, 1995) focused on a Virtual Learning Center (VLC) model, architecture and prototypes. To build the VLC model, object-oriented modeling techniques were applied such as Jacobson's use cases methodology (Jacobson, 1993) and the Object Modeling Technique, OMT (Rumbaugh et al., 1991), to identify sets of actions that different actors would do while interacting within a virtual campus. Five actor types were identified then: the learner, the trainer, the content expert (informer), the designer, and the manager. Sixty-three roles that can be played by these various actor types were defined.

Right from the beginning, the ambition was to build a distance learning operating system capable of supporting a variety of roles within a variety of delivery models such as High-tech Distributed Classroom, Web/multimedia self-training, Online training, Community of Practice or Performance Support Systems. From 1995 to 1999, we have conducted various research and development projects supported by the Quebec Information Highway Fund and the Canadian Telelearning Network of Centers of Excellence (TL-NCE). This work has lead to the implementation of our Virtual Learning Campus (VLC) architecture using Web-based technology. In 1999, the Explor@-1 implementation of our VLC model was completed and a number of distance learning courses were developed and delivered through it, mainly at the Télé-université, but also in pilot applications at Hydro-Quebec and in professional associations.

The Explor@-1 system had a set of innovative features that are still pioneering.

- Contrary to the general authoring system paradigm, Explor@-1 focussed on a resource (or learning object) based learning management, making it possible to assemble a set of educational support tools and resources to be shared across

programs, courses or activities delivered by an organization.

- The system had more flexibility compared to the traditional learner-trainer-manager trio, enabling the definition of any set of actors.
- Each course could be designed to meet different needs implementing different pedagogical approaches, by using a variety of proprietary or third-party tools, made available to learners, course designers and other facilitators, such as instructors, content experts (informers), training program administrators, etc.
- An Advisor Editor, enabled the designers to build a set of rules that would trigger help/assistance in various forms (questions, messages, visual cues) when certain conditions were met by values in the user properties tracked by the system.
- The Explor@-1 system was designed to support the integration of existing Web courses without changing their format or assistance structure, thus allowing an organization to transform its training/learning methods progressively.
- Finally, the open modular structure of the system made it possible to significantly reduce design time, speeding up the implementation and allowing periodic updates by the design team or the online tutor. Environment maintenance also became much easier. Once the first course was implemented, each additional course integrated into Explor@ could be limited to a few Web pages and hyperlinks to existing documents.

From 1999 to fall 2002, we conducted a third major R&D effort within Technologies Cogigraph, a spin-off from Télé-université research center. The Explor@-2 system was developed and implemented at Télé-université and at Canal Savoir[1] for its SavoirNet delivery infrastructure.

[1] Canal Savoir is Québec's university television channel grouping most universities in Quebec and some colleges. It has started to diversify its educational system to support different combinations of Web and TV delivery models.

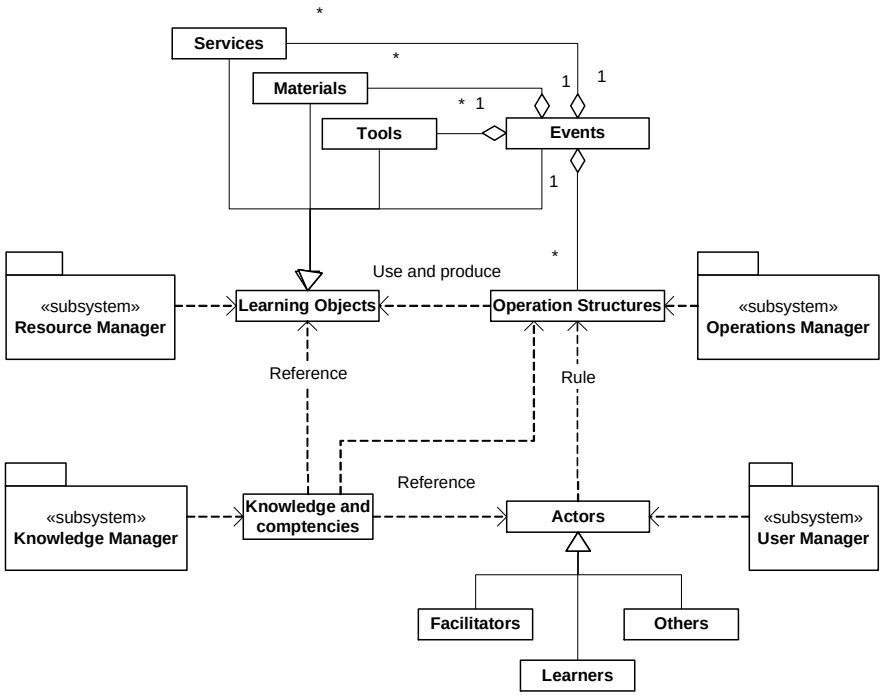

Figure 1 – High Level Architecture of the Explor@-2 System

Figure 1 presents a conceptual view of the core architecture of the Explora-2 system. It deals with four types of objects: actors (or roles), learning objects (or resources), knowledge and competency (or content), and operations structures (or functions). Actors operate functions composed of operations (or activities) where learning objects are used or produced. Knowledge and competencies describe the information owned, produced or processed by actors, processed in operations or contained in resources. Four corresponding managers store and retrieve information in a database, construct information structures and display information to users.

As was stated before, Explor@ has a resource management orientation allowing for the integration of learning objects and services in a learning scenario. The resource manager shown in figure 2 (Paquette

et al, 2004 in press) is the Explor@-2 component in charge of this management. The two upper components, *Learning Object Aggregator* and *Learning Object Launcher*, operate on the learning objects themselves found in one or more repositories, located on servers somewhere on the Web. The six other components all relate to metadata management services. Locally, Learning Object Metadata (LOM) records referencing the resources are stored by the Explor@-2 resource manager in a relational/XML database.

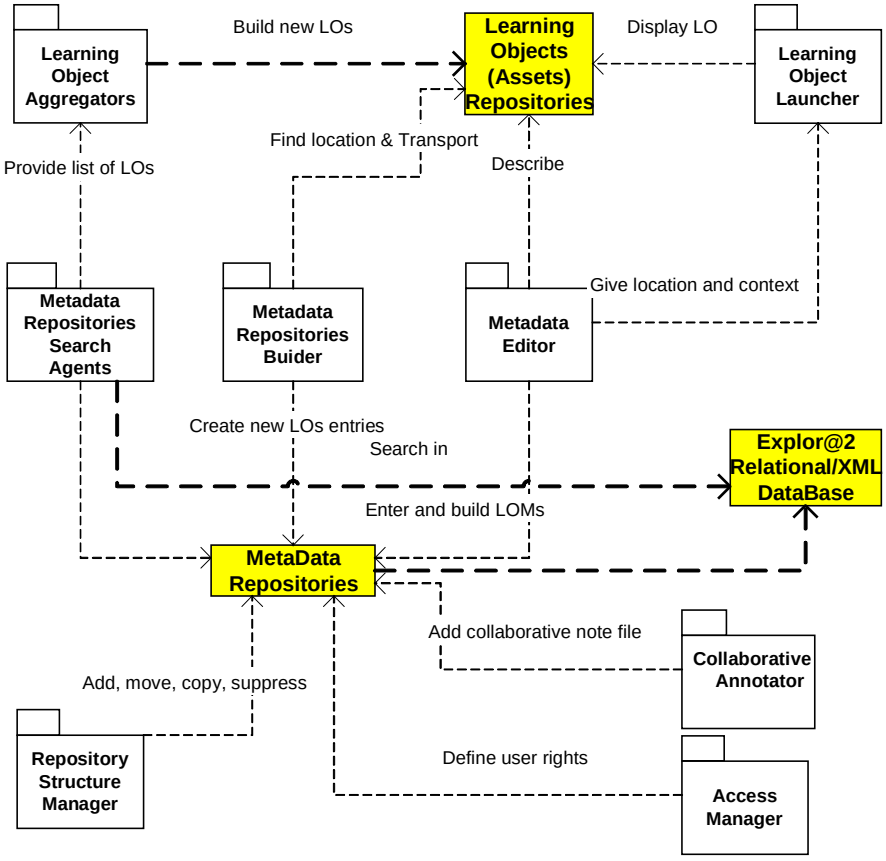

Figure 2 – Main Components of the Explor@-2 Resource Manager

The Explor@-2 system provides designers with three main ways to aggregate learning objects into larger resources. The corresponding designer's tools are the *Resource Aggregator*, the *Role Environment*

Editor and the *Instructional Structure Editor*. The Resource Aggregator is a simple tool to build Web pages filled with hyperlinks to resources found using the metadata repositories search agents. The *Role Environment Editor* aggregates resources into an environment according to the roles of an actor. Using this Editor, a designer identifies the different roles an actor has to play in a course or a Learning Event, and defines it indirectly by creating an environment made of spaces (menus) grouping resources assisting an actor to carry out its various roles.

The most important aggregation tool is the Explor@-2 *Instructional Structure Editor*. It enables a designer to import or build a tree structure describing a Learning Event (or a course scenario) grouping activities where resources are used or produced by a role. This editor is the Explor@-2 version of a learning design editor. It helps designers to construct a runtime learning model. During runtime, a progression tool shows to students their progression through the learning event based on the structure produced by the designer with the activity editor.

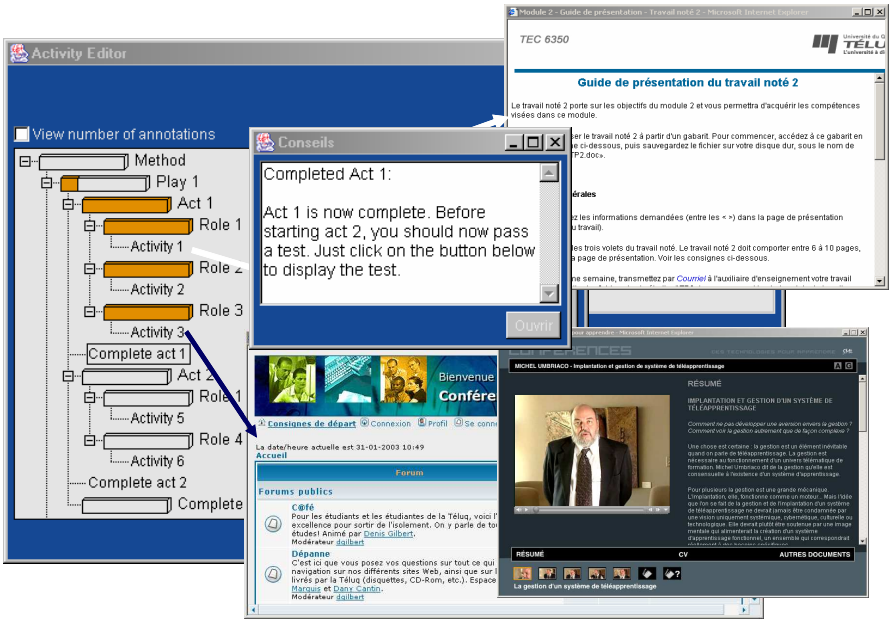

Figure 3 – Screen display of a student delivery environment

The left-hand window, in Figure 3, presents the resulting Instructional Structure corresponding to the IMS LD Method, Play, Act, Activities and Role parts displayed in the Explor@-2 progression tool and produced by the Explor@-2 editor.
For each node and leaf, the user (learner or staff) can access services and learning objects (tools, documents and services) pertinent to the play, the act or the activity by double-clicking on the corresponding title. Three such resources are shown:

- A direct link to an online conference (forum) service
- A video lecture, which can be viewed in segments or as a whole, accompanied by a PowerPoint presentation and other pertinent resources to enhance subject comprehension..
- An exercise guide matched to the Act 1 Activity 1.

The "Completed Act 1" window, at the center, is where feedback is provided to the user when Act 1 is completed either because the learner clicks a box or when the time-limit set by the designer is exceeded. The progress bar shows whether or not the user has com-

pleted the act. As a user progresses from one activity to another, the completion level is calculated for the Play level as well as the Method level, all according to rules set forth by the course designer in the Explor@-2 Instructional Structure editor.

2- The Explor@ Learning Design Information Model

Explor@-2 provides designers with a set of tools to build a runtime learning design specification and support learner and staff using Web-based instances of this learning design. In Explor@-2, using the instructional structure editor, a designer can import (from ADISA, MOT, or any useful XML tree structure editor) an instructional structure or build it from scratch, associate resources to the structure, describe time, collaboration and evaluation rules, associate knowledge and competencies, add advices and assessment questions, specify a progress/completion mechanism and finally, describe advisor/assistant rules governing actions in the environment.

The Instructional Structure in Explor@-2 starts with a root representing the main Learning Event: a program, a course, a module, etc. (the method element in IMS-LD).The second level is composed of smaller Learning Events nodes (plays in IMS-LD) that can be decomposed (through IMS-LD acts and activity structures) at any number of levels until we reach terminal nodes corresponding to Learning Units (activity structures in IMS-LD with no sub activity structures). Below are terminal nodes that correspond to activities (learning or staff single activities in IMS-LD) in the MISA instructional scenario. Finally, below these terminal nodes there are the input and output resources from an activity (the environment in IMS-LD).

A corresponding conceptual model is shown on figure 4. Tree leaves are special kind of nodes. Any node may have associated resources, advice and assessment questions. They can also hold a progression rule that specifies if the sub-nodes are to be processed in sequence or

in parallel, possibly with options, such as do 2 out of 4 nodes. The completion of sub-nodes will affects the progression level of a parent node, according to the progression rule associated to the parent node.

Additional elements can be associated to the leaves of the instructional structure, corresponding to properties, such as required completion time, collaboration time and type, assessment tag and weight (percentage of the evaluation). The system adds these elements values and propagates the cumulative value to the all upper levels of the instructional structure corresponding to Learning Units and Learning Events[2].

Besides the Instructional Structure, the designer can build a knowledge and competency tree structure and assign knowledge and competencies to activities that are regrouped upward and assigned to larger activity structures. This association informs the learner on which learning events, learning units, and/or activities will have him work on certain knowledge and competencies. An alternative way to associate knowledge is to use the instructional structure editor to add a text description of the competencies to any node or leaf of the structure or to recover a learning object describing the knowledge from a learning object repository.

Figure 4 also displays the actor's environment concept (produced with the role environment editor presented above). Any environment in the learning system groups the resources for each actor into one or more spaces like self-management, information, resource production, collaboration or assistance. Figure 4 also indicates rules that can be assigned to any node to build an advisory system for the users. This important aspect corresponds to IMS-LD levels B and C and will be discussed later.

[2] See chapter 6 for a correspondence between MISA terminology and IMS-LD.

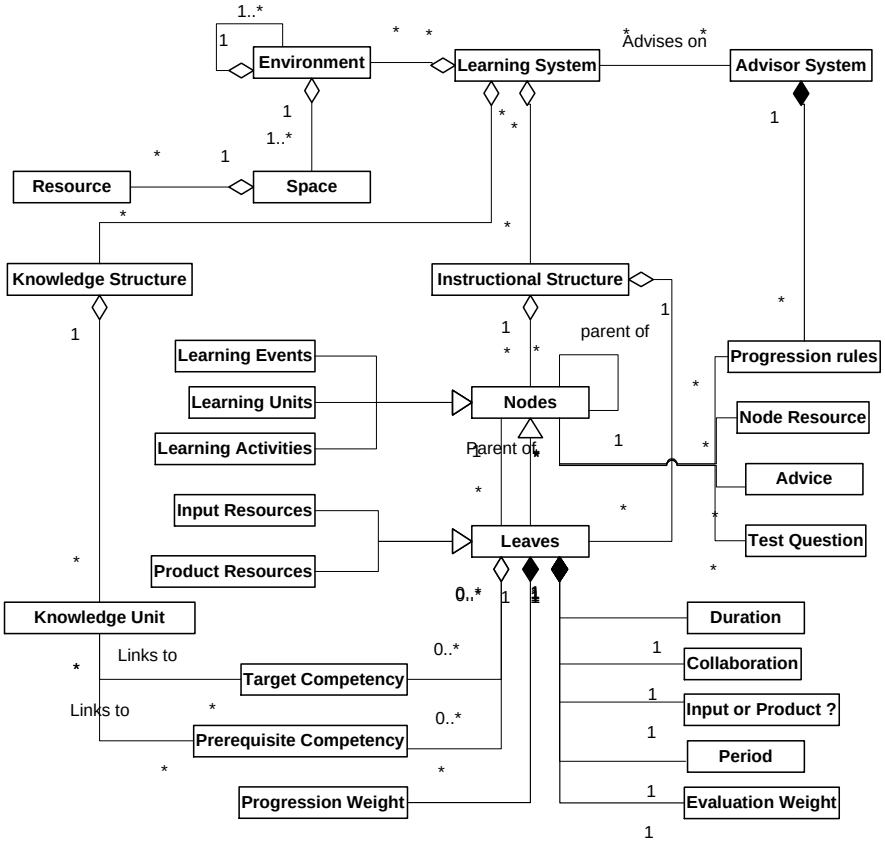

Figure 4 – The Instructional Structure of Explor@-2

3- Integrating IMS-LD (level A) specification in Explor@-2

We will now focus on the Instructional Structure Editor presented in figure 5. On the bottom left side of the window, we see functions to add or suppress nodes and leaves of the instructional structure (Add node, add leaf, remove). It is also possible to import an XML structure built with the MOT+ Editor embedded or not in the ADISA instructional design support system to MISA (see chapter 6).
Selecting any node, a designer can assign progression rules on how

to proceed within the corresponding event, unit or activity, either in sequence, in parallel or with options. Designers can also use the editor to assign other node and leaf attributes such as duration, evaluation weight, assignment, advice, annotation capability. They can also associate to nodes in the instructional structure, learning objects pointers stored as LOM records, to be launched at run time.

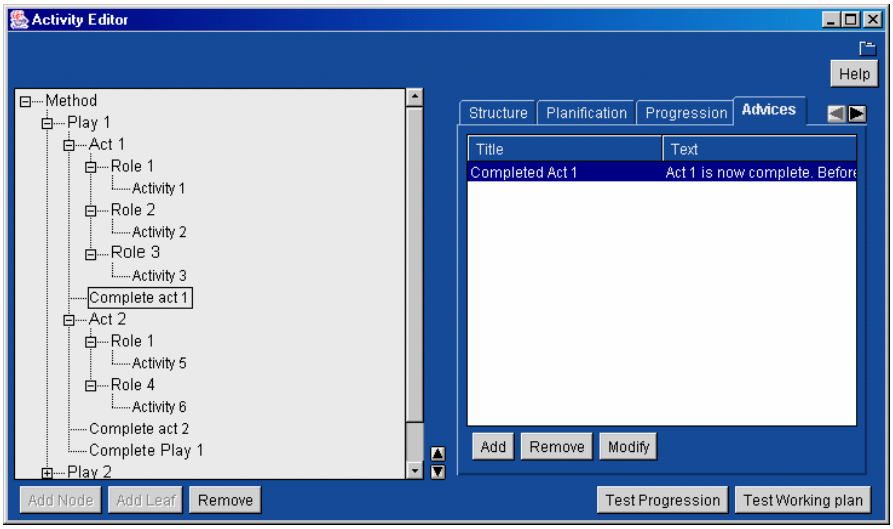

Figure 5 – The Instructional Activity Structure Editor

Using this editor it is possible to build a representation of an IMS-LD Method and an Explor@-2 user progression as the one displayed in figure 3. Figure 6 presents a concrete instantiation of the activities of that structure. Here, the Method corresponds to a Learning Unit called Module C and the plays present two alternative course delivery models from which a learner has to choose one: Web delivery (play 1) or classroom delivery (play 2). Play 1 consists of two Acts in sequence. In the first Act, learners prepare a seminar by consulting resources, participate in a discussion forum and produce a presentation; tutors animate the forum; experts provide advice to learners in and outside the forum. In the second Act, learners deliver the presentation while assessors take note to produce an evaluation report (this activity could figure in a third act). Figure 6 shows that two of the three role-parts in Act 1 have been completed; one of the

learners has still to produce a text. If the learner clicks the check box of this activity, the system displays a validation questions with two possible answers, each triggering an advice on what to do next.

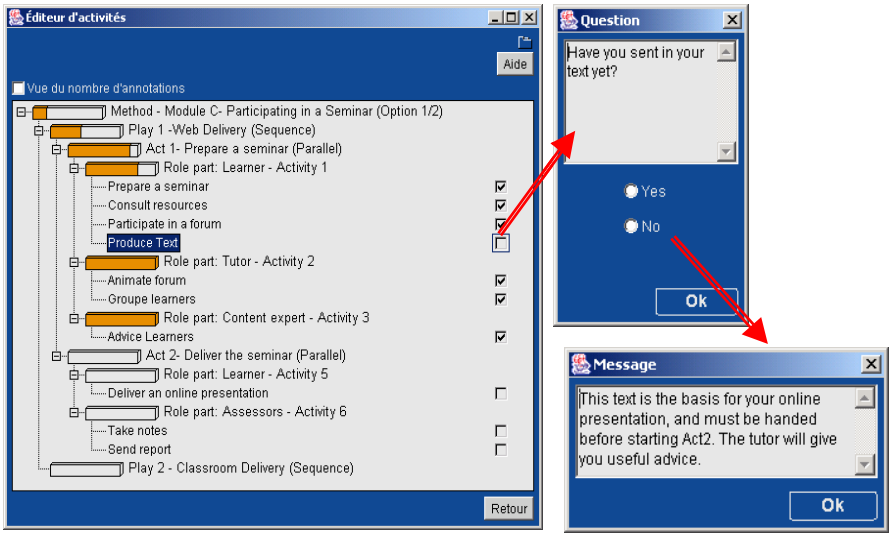

Figure 6 – The Instructional Activity Structure at Runtime

Explor@-2 has a built-in bottom-up propagation mechanism to assign a progression level to each node of the instructional structure calculated from its leaves, which can be used to provide feedback using completion requirements for acts, plays or the method as specified in IMS-LD.

When the user selects a leaf of the tree structure, he can declare it completed. If the designer has prepared an assessment question, only a right answer will turn on the completed requirement flag; if there is no question, the flag will be on by default or after a certain time limit selected by the designer. If all the role-parts in an act are completed, in whatever order, the act is completed. If all the acts are completed in the specified sequence, the play is completed. If the required number of plays is completed, the method is completed. When an act, a play or a method is completed, a feedback message

can be displayed[3].

This example can be generalized to any method, showing that the Instructional Structure in Explor@-2 is generic enough to build any unit of learning modelled with the IMS-LD specification. In practice the corresponding XML files could be produced either by a MOT model or a slightly modified activity editor, and read into the instructional structure.

Actually, in Explor@-2, each actor or role has its own activity structure (which is not multi-role) and its own resource environment, so additional functionalities will have to be built to exploit the multi-actor capabilities of the IMS-LD specification. These include synchronization mechanisms when the completion of an Act requires verifying all or some other roles have also completed the Act. We will then provide an IMS-LD activity editor as an option, generate role environments automatically and activity structures for each type of actor, and provide contextual alternate views to help an actor situate the activities within a play. A way to do this using the concept of a function model has been presented in (Paquette and Rosca 2003).

On the other hand, Explor@ can produce and deliver instructional structures that are more complex than an IMS-LD Method since it is possible, at any level, to assign to any node a progression mode specifying that the sub- nodes are to be completed in sequence, in parallel or with options. This might pose certain problems when we want to translate an Explor@ Instructional Structure into an IMS-LD specification to increase reusability and interoperability with other delivery systems. This problem will need further investigation.

[3] In the actual version of Explor@-2, that message is entered by the designer in the assignment attribute of a node and is displayed only if the user asks for it. In a previous version such a message could be displayed at the initiative of the system; this functionality will be re-introduced in the next version.

4- Integrating Level B and C Specifications in Explor@-2 or taking an epiphyte approach

The IMS-LD levels B and C specifications give additional possibilities to a simple feedback produced by completed parts of a method. On the other hand, they are minimal to provide adaptation and role coordination capacities in a distributed learning environment. As we see in the IMS-LD best practices document (IMS-LD 2003), conditions and properties allow for the personalization of pedagogical treatments. Instructional designer may, for example, personalize the activities a student has to do, as a result of his/her profile and pretest scores (ex: 2.1 and 2.3 from the best practices document) or previous experience (ex. 2.7) or as a result of recognizing particular learners needs (ex: 2.10: obtaining learners profile from a human resources database) the resources to be used in a particular activity (ex. 2.2: the systems find adapted resources according the student cultural group) the composition of groups, taking into account students' profiles (ex. : 2.2) or the selection and sequencing of activities (ex: 2.14). This personalization is achieved by inserting actions (show, hide, notify and change property) into the learning structure, which are to be triggered when conditions on properties are met. Those conditions are inserted in different parts of the learning design, at the Method, Play, and Act level.

It might be worthwhile to look at another possibility which would be, to leave the design free of conditions and actions and to have an external advising agent monitoring it and eventually taking control when needed. This is the approach taken in Epitalk (Paquette et al. 1996), which has been applied both to support instructional engineering in MISA (Paquette and Tchounikine 2002 and to assist learners using Explor@ (Girard et al 1999; Lundgren et al 2001).

This approach is based on an external advisory system, a set of software agents that can be grafted to an existing host system. As was

shown in these articles, Epitalk has many advantages over the more traditional "branching-like" approach where conditions are wired in the host system. The following principles guide this type of system:

- the actions giving advices or adapting the environment can be added to an existing host system without having to change its code;
- the actions and the conditions are based on a model of the host system constructed by the designers using a terminology that he/she chooses for some intended purpose (this aspect is accessible to an instructional designer without programming skills);
- an advisory editor can be built to support instructional designers in the difficult task of building an adaptive assistance system: to build an instructional model and assign conditions and actions to the model;
- since the assistance is mediated by a model constructed by the designer, it enables him/her to address assistance issues from different viewpoints; for example, one agent could manage the resources proposed to the learner, while another one would assist on the coherence of a tutor's interventions.

Epitalk can in principle be applied to activity models for any actor or sets of actors, thus making it possible to address the multi-actor aspects of an IMS-LD method. In Explor@-1 [Girard et al. 1999] and ExploraGraph [Dufresne 2001], Advisor editors made it possible to build a model of the host system and to use it to maintain a user model and define rules triggering actions when certain conditions were met. We are now in the process of re-introducing such functionalities in the actual Explor@-2 system.

As shown above, the Explor@-2 advisory component of the activity editor actually includes a simpler advisory system than in Explor@-1 focused on the student progression in the learning design. It has two components: the Advice Editor and the Student Advisor. The Advice Editor allows the designer to tie to each node in the learning design; its weight of importance; its type of progression (sequential, modular, parallel or optional); pop-up advices and assessment questions. The student advisor in Explor@-2 actually supports three

functions:

- It displays diagnostic questions and pop-up advices while navigating in the course site (proactive advisor – dynamic advice);
- It makes available contextual advice in an assistance space of the user environment where the user can trigger pieces of advice (passive advisor – static advice);
- It displays viewers, for example a progress bar showing the students progression both in the instructional structure and the cognitive structure (student self-monitoring).

To give dynamic advice and to display the student progress bar, the advisory system builds dynamically a simple student model, tracing student interaction, both with the learning system and the advisor. Rules in the Advisor Editor are actually specialized: their conditions involve properties on the user's progression, navigation and answers to the diagnostic questions; their actions are mainly to trigger an advice or a question, and to update progression viewers.

In spite of this specialization, those rules do already have the structure required to implement in Explor@-2 the levels B and C of IMS-LD specification. Indeed, triggering advices could be transformed into sending a message by including email names and addresses. Showing and hiding is already possible. Property modification could be made by generalizing the modification of the progress bar to other properties, as was the case in Explor@-1.

From an implementation method point of view, this discussion leads us to propose that a next version of the IMS-LD specification should consider an approach similar to Epitalk, basically a multilevel design allowing grafting the advisory system on to the host system instead of including it. This could be done either by changing the XML binding to address multilevel designs, or alternatively, to limit IMS-LD to its actual level A and to add a new companion specification for an assistance system that can be grafted on an IMS-LD (level A)

learning design.

Conclusion - Where to go next …and further

Educational Modeling Languages and the IMS-LD specification bring important innovation to the e-learning toolset and propose new technical challenges. Next step, on our part, will be to analyze the specification from a delivery point of view to adapt our Explor@-2 system so that it can fully process all three levels of the IMS-LD specification. Within the eduSource[4] project, we will also define generic services that any delivery system should provide to fully exploit this specification.

Looking further, we believe that a new era of more powerful and flexible distance learning system is starting. IMS-LD is a cornerstone in this direction. Its proposed model of a method leads directly to delivery models of a distributed learning system seen as a set of multi-actor process models. Pushing this idea further, our knowledge, delivery and assistance models are also basically process models in the sense that they describe and relate activities, objects and actors. In [Paquette and Rosca 2002] we have developed this idea under the name of function models. Function models are models that aggregates resources used or produced by users, with operations that these users perform and possibly other functionalities such as assistance services. Function models are promising components to describe, model and manipulate the different processes that take place in a distance learning course and their relations. They allow for the description, not only of the anatomy of a learning system, but also of its physiology, as a dynamic set of interactions.

In the LORNET project[5] we intend to develop and to tool the con-

[4] The eduSource project, in an ambitious Canadian project that aims to implement a functional network of learning object repositories, based on the international standards and providing a software suite of tools to find, reference an use learning objects in educational applications.

[5] LORNET (Learning Object Repositories Networks) is a major 5 years research network heavily funded by the Canadian government to address these questions in a Semantic Web and Knowledge Management perspective. It groups 5 of the

cept of function model to provide a solution to the inherent complexity of a distance learning system and to encourage the evolution of the delivery systems towards greater flexibility. As part of the project, we will build a collection of Learning Designs integrated to Learning Object Repositories and we will provide different ways to aggregate these learning designs with Knowledge Objects and with Assistance objects in a unified way through function models implemented as multi-actor coordination interfaces. These goals correspond well to the research agenda set forth by Duval and Hodgins (2003), where they outline that authoring by aggregation and design for content reuse are research issues that must be addressed in a near future, if reusability and interoperability among learning resources are to be attained. Furthermore, by allowing function models to mutate, change and evolve, we expect to be able to produce flexible, personalized, evolving and even emerging learning situations.

References

Booch.G., Jacobson, J. & Rumbaugh, I. (1999). *The Unified Modeling Language User Guide.* Addison-Wesley.

Bruner, J. (1966). *Toward a Theory of Instruction*. Cambridge, MA, Belknap Press/Harvard University Press.

Collins, A & Stevens, A.L. (1983). *A Cognitive Theory of Inquiry Teaching*. In Reigeluth, C.M. (ed.) Instructional Design Theories and Models: An Overview of their Current Status. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Duval, E. and Hodgins, W. (2003). *A LOM Research Agenda*. The Twelfth International World Wide Web Conference, Budapest, Hungary. Retrieved from http://www2003.org/cdrom/papers/alternate/P659/p659-duval.html.html

Dufresne, A. *ExploraGraph : Improving interfaces to improve adaptive support.* Paper presented at the AIED'2001, San Antonio, Texas.

major Canadian laboratories in the field, headed by Télé-université's LICEF research center.

Feldstein, M. (2002). *How to Design Recyclable Learning Objects*. eLearn Magazine. Retrieved August 2002 from http://elearnmag.org.

Girard, J., Paquette, G. , Miara, A., & Lundgren, K. (1999). *Intelligent Assistance for Web-based TeleLearning*. Paper presented at the Proceedings of AI-Ed'99, in AI and Education, open learning environments, S. Lajoie et M. Vivet (Eds), IOS Press, pp 561-569.

IEEE (2002). *Draft Standards for Learning Object Metadata*. Learning Technologies Standards Committee of the IEEE. IEEE 1484.12.1 – 2002. Retrieved from http://ltsc.ieee.org/doc/wg12/LOM_1484_12_1_v1_Final_Draft.pdf.

IEEE (1999). *Learning Object Metadata*, Draft Document v3.6 (1999). IEEE Learning Technology Standards Committee (LTSC). September 1999. Retrieved from http://ltsc.ieee.org/doc/wg12/LOM3.6.html.

IMS 2002. *IMS Reusable Definition of Competency or Educational Objective - XML Binding, Version (2002).* 1.0 Final Specification, IMS Global Learning Consortium, Inc. Revision: 25 October 2002.

IMS-LD (2003). *IMS Learning Design. Information Model, Best Practice and Implementation Guide, Binding document, Schemas*. Retrieved October 3, 2003, from http://www.imsglobal.org/learningdesign/index.cfm

Koper R. (2002). *Modeling units of study from a pedagogical perspective – The pedagogical metamodel behind EML* Retrieved march 2002 from http://www.eml.ou.nl/introduction/articles.htm.

Lundgren-Cayrol, K., Paquette, G. Miara, A., Bergeron, F., Rivard, J. & Rosca, I. (2001) *Explor@ Advisory Agent: Tracing the Student's Trail*, Paper presented at WebNet'01 Conference, Orlando, 2001.

Merrill M.D (1994). *Principles of Instructional Design*. Educational Technology Publications, Englewood Cliffs, New Jersey, 465 pages.

Paquette, G. (1995). *Modeling the Virtual Campus*, Innovative Adult Learning with Innovative Technologies A-61, 1995

Paquette, G., F. Pachet, S. Giroux & J. Girard (1996). *EpiTalk: Generating Advisor Agents for Existing Information Systems*. Artificial Intelligence in Education, USA, summer 1996.

Paquette G. (2001) *TeleLearning Systems Engineering – Towards a new ISD model*, Journal of Structural Learning 14, (pp. 1-35).

Paquette G. (2001). *Designing Virtual Learning Centers.* In H. Adelsberger, B. Collis, J. Pawlowski (Eds) Handbook on Information Technologies for Education & Training within the Springer-Verlag series "International Handbook on Information Systems", (pp. 249-272).

Paquette,G., Rosca, I. , De la Teja, I., Léonard, M., & Lundgren-Cayrol, K. (2001). *Web-based Support for the Instructional Engineering of E-learning Systems*, Paper presented at WebNet'01 Conference, Orlando.

Paquette, G. & P. Tchounikine. (2002). *Contribution à l'ingénierie des système conseillers : Une approche méthodologique fondée sur l'analyse du modèle de la tâche*. Science et Techniques Educatives, 9/3-4/2002, (pp. 409-435)

Paquette, G. & I.Rosca. (2002). *Organic Aggregation of Knowledge Objects in Educational Systems*, Canadian Journal of Learning Technologies, Volume 28-3, Fall 2002, (pp. 11-26)

Paquette, G., Lundgren-Cayrol, K., Miara, A and Guérette, L (2004, in press). The Explor@-2 Learning Object Manager. Chapter in Rory McGreal (Ed.) Online education using learning objects. London : Tourledge/Falmer.

Rawlings, A., P. Van Rosmalen, R. Koper, M. Rodriguez-Artacho & P. Lefrere. (2002). *Survey of Educational Modelling Languages (EMLs)*, version 1, September 19th, CES/ISSS

Rumbaugh, J., M. Blaha, W. Premerlani, F.Eddy and W. Lorensen (1991). *Object-Oriented Modeling and Design*. Prentice Hall, Englewoods Cliffs, N.J.

Wiley D.A. (2002). *Connecting learning objects to Instructional design theory: a definition, a methaphor, and a taxonomy*. In Wiley (Ed) The Instructional Use of Learning Objects. Agency for Instructional Technology and Association for

Educational Communications of Technology, Bloomington, Indiana, , 281 pages.