

## 2

# Bridging the Gap Between E-Learning Modeling and Delivery Through the Transformation of Learnflows into Workflows

OLGA MARIÑO, RUBBY CASALLAS, JORGE VILLALOBOS, DARIO CORREAL, AND JULIEN CONTAMINES

**Abstract.** E-learning pedagogical models are described in terms of educational modeling languages (EMLs). IMS-LD is accepted as the standard EML. It allows for the description of multiactor adaptable learning processes. Although some IMS-LD-compatible editing tools are being developed, no delivery platform is yet available. This chapter proposes to bridge this gap by looking at business process modeling languages and execution engines, in particular the Workflow Management Coalition Standard, XPD. The first two sections of the chapter give the introduction and the context of the work. Section 3 describes IMS-LD as well as existing editing and delivery tools. Section 4 describes XPD and some editing tools and execution engines. Section 5 proposes a transformation from IMS-LD to XPD, and Section 6 describes the application developed to implement this transformation. The chapter ends with some conclusions on the work done and on the possibilities it opens to further research and applications.

## 2.1 Introduction

A unit of learning in a virtual learning environment relates various different models: the actors' model, the resources or learning objects (resources and services) model, the knowledge and competency model, and the learning process model. The last one, also called pedagogical model or learnflow model, is the integrating model, the one that orchestrates all the others. It is through this model that the learning strategy is described in the system. In a structured learning situation, learners follow the process described by this learnflow. Other actors, as well as resources and services, are coordinated by this workflow as well.

To describe those models in a platform-independent language, some educational modeling languages (EML) have been proposed. In 2003, the IMS Global Learning Consortium released the IMS Learning Design specification [15]. IMS-LD is becoming an accepted EML standard specification. This specification allows for the description of the pedagogical process or learnflow of a unit of learning in an educational multiactors workflow called the method. Using a theatrical metaphor, a method can be realized by different plays. Each play is composed of sequential

acts; each act includes parallel role-parts, which are associations between roles and activity. Roles are played by one person or a group of persons. Initial role classification distinguishes between learner and staff. The activity is the core of the model. It can be simple or complex (activity structure). It uses an environment composed of learning objects and services and may produce outcomes (products) that enrich the environment. Levels B and C of the specification add variables (properties) and rules to allow for personalization and annotation.

IMS-LD tends to bridge the gap between the design of a course, mainly an on-line course, and its delivery, and opens the way to reuse not only learning objects but also learning scenarios and strategies.

In spite of those promises, IMS-LD raises new challenges, both for the e-learning design and for the e-learning delivery. Although some delivery platforms like COW [44], LAMS [22], and Explor@ [38] are being modified to be level A compatible, no system is yet fully IMS-LD compatible. The IMS-LD community is working on the definition of a standard IMS-LD delivery platform architecture [20,45].

Looking at an apparently different context, business processes are also composed of activities played by actors using resources. Workflow management systems improve business processes by automating tasks, getting the right information to the right place for a specific job function, and integrating information in the enterprise [11]. The integrating component of a workflow management system is the workflow model. A workflow model is an abstraction of a business process. It is a structured organization of individual steps called tasks or activities. An agent or actor is a human being or a machine that can perform a task (enact an activity). Roles are a logical abstraction of one or more actors, usually in terms of functionality. Dependencies determine the execution sequence of activities and the data flow between them [6,23].

Workflow management systems and learning management systems both deal with common issues such as multiactor process modeling and execution, activities synchronization, services and objects integration, role instantiation, etc. Furthermore, both research communities are addressing new challenges such as the process life cycle management, the process evolution, and flexible and open process definition.

In this chapter, we make a comparison of workflow management systems and learning management systems. We focus our discussion on the process (workflow and learnflow) and on the existing tools to support both editing and enacting the process. Section 2 gives the context and the goals of this work. In section 3, we look closer at the learning design standard IMS-LD, both in terms of the language and in terms of the existing tools, while in section 4 we describe this same state of the art for workflow standard, XPDL. We have established a correspondence between IMSLD and XPDL elements, we have developed a transformation from IMSLD to XPDL based on this correspondence, and we have implemented this transformation in an application called LDX. The fifth part describes the transformation and the sixth the LDX set of tools. The chapter ends with a broader view on how this transformation allows us to support automatically the e-learning process, and with some conclusions at the model and language level as well as at the practical tool level.

## 2.2 Context

This section gives the context and a broad description of the work presented in the chapter. We start by describing and comparing the two problems we are dealing with: e-learning support and business process support. Although different in nature and domain, these problems establish some common requirements to supporting tools, while still having particular specific requirements. Section 2.2 sketches the main differences between e-learning and workflow management tools. The third section states the goal of our research, while in the fourth section we present the approach we took to develop this work. We end by summarizing the main results of our work, which will be detailed in the rest of the chapter.

### 2.2.1 Looking at the Problem

Both e-learning system and workflow system tend to solve the same very general problem of having an (or many) actor(s) executing an activity or graph of activities and producing something (Fig. 2.1). Therefore, the three main components of such a system in both cases are actor, activity, and product.

The goal of an e-learning system is the “learning.” The main actor, the learner, is expected to learn, to acquire new knowledge and competencies, through the execution of different structured learning activities. E-learning systems’ main component is the actor-learner. This emphasis on the actor-learner establishes particular requirements on e-learning systems: the process should be defined taking into account the learner’s profile; the system should allow for run-time process adaptation based on conditions of the learner; the learner model must be rich and known to the process. Even resources proposed to the actor to execute an activity should take into account the actor profile.

In the business process context, on the other hand, what matters most is the product, the final outcome of the process. The main aspects of a workflow system are the product and the efficiency of the process regarding this production. Actors are secondary components; they are seen as resources whose importance lies in helping produce the final product. Particular requirements in this context include finding an assignment function from people to roles, decomposing the process in an efficient way, and optimizing the actor’s participation. There are also particular requirements concerning the product: version management, satisfaction of a set of required properties, the possibility to be used and consulted outside the workflow system, etc.



FIGURE 2.1. E-learning and workflow conceptualization.

In terms of the process or main activity, there are also some common issues. Both processes can be described by a graph of activities. Each activity is done by a particular type of actor, called a role. Normally, more than one role (actor) is involved in a process. Multiactor coordination and synchronization is thus required. Time events as well as other external events may affect the process execution.

However, there are also some differences. E-learning processes represent pedagogical models. While an e-learning pedagogical model might be inspired in an existing face-to-face training, new models can exist without a corresponding face-to-face model. In addition, when taking into account personal differences, the model of a same unit of learning is instantiated in different ways for different students. In the business world, on the other hand, workflows do normally represent existing documented enterprise processes. Different instances of the same workflow are identical copies except for the actors' assignment to roles and timing information, and they should produce quite identical products.

### *2.2.2 Looking at the Solution*

The requirements identified in the preceding section are translated in some properties of the corresponding support systems. **Both e-learning and workflow management systems have a formal way to describe multiactor processes.** Activities in these processes may be organized with different patterns (sequence, choice, parallelism, and synchronization points). Responsibility for the execution of activities in the processes is defined in terms of actor types, also called roles. In an instantiation of the process, those roles are associated with actual actors. To do an activity, the actor has available services and resources that are partially or completely determined in the process model. To execute (enact/deliver) the multiactor process, both e-learning and workflow systems have to have control and synchronization mechanisms.

Having the actor as the heart of the process imposes particular requirements for e-learning systems. The model of the learner is normally part of the system, or at least there are ways to pass learner information into the system. Learner information must include knowledge and competencies. The whole process and possibly each activity should also have a reference on prerequisites and objectives. Thus, e-learning system should have some kind of knowledge modeling component.

Having the product and its production inside a business context, as the heart of the process, imposes particular requirements for the workflow management systems. To satisfy those requirements, workflow management systems normally include product management as well as different services and tools to manipulate the process (evaluation, measurement, audit, etc.). To support a broad variety of business processes, workflow management systems and workflow models offer a large span of control and synchronization patterns. Finally, they offer well-defined interfaces to communicate with other business applications.

From a practical point of view, the workflow management community has well-established process languages and various open-source enactment tools. On the other hand, the e-learning community is actively working on the development of pedagogical model editing tools that help produce IMS-LD models, but no e-learning platform is yet fully compatible with the standard e-learning modeling language, IMS-LD.

### *2.2.3 The Goal: Bridging the Gap Between E-Learning Editing and Delivery*

The goal of our research is to benefit e-learning systems from advances made by the workflow management systems community, and at the same time to explore personalization and knowledge management integration in workflow management systems.

The work presented in this chapter addresses the first goal: more precisely, to bridge the currently existing gap between learnflow editing and learnflow execution by allowing a learnflow expressed in a standard educational modeling language to be executed by a workflow engine, capable of executing the process described in a standard process description language. To reach this goal we have defined three subgoals:

- To study e-learning and workflow models so as to identify common issues and differences and to establish a common vocabulary to describe them
- To define a translation schema, allowing us to express the control aspects of e-learning models in terms of the richer set of control elements of process models.
- To build a tool to implement this translation and to validate the results

### *2.2.4 Methodological Approach*

Our starting hypothesis is twofold: we propose that pedagogical models can be described and delivered using the standard educational modeling language IMS-LD, and that workflow processes can be described and executed using the standard business process definition language, XPDL.

Based on this hypothesis, we have defined the following project steps:

- To express both languages (IMS-LD and XPDL) in a common language to facilitate their comparison and translation from one to the other
- To abstract all components that are not part of the control model, and to propose a translation schema from IMS-LD to XPDL. This translation is guided by IMS-LD syntactical structures; it takes into account every important aspect of the source language (IMS-LD).
- To identify IMS-LD elements that do not have a direct and natural translation into XPDL, and to propose a way of using XPDL extended attributes to describe them so that an XPDL compatible tool may handle the document.

### 2.2.5 Main Results

The main results of our project, which will be detailed in sections 5 and 6 of this chapter, include:

- A translation schema to translate IMS-LD into XPDL
- A clear identification of the principal difficulties of this translation
- The implementation of a set of tools to support the learning design lifecycle

## 2.3 IMS-Learning Design as an Educational Modeling Language

This section presents the educational modeling languages that allow for the description of e-learning processes and more specifically the IMS-LD specification. We start by giving a generic reference model for e-learning systems, one of whose components is the learnflow or pedagogical model. The pedagogical model is described by an educational modeling language (EML). The second section describes the main elements of an EML. In section 3.3, we present the particular case of the educational modeling language standard IMS-LD. Section 3.4 presents some of the tools to support course design using IMS-LD and its delivery.

### 2.3.1 E-Learning System Reference Model

There is no widely accepted e-learning reference model. In spite of that, most e-learning systems include the following components, not always well differentiated: A kernel, is the e-learning operating system, the learnflow execution engine. This engine executes a process described by a pedagogical modeling component. Actors are described elsewhere and actors' profiles are taken into account by the engine too. Resources, tools, and services are managed by one or more components in charge of finding, installing, launching them when needed. Finally, some systems also include a more or less elaborated knowledge component to describe at a knowledge level the activities, the resources, and the actors. Figure 2.2 shows this conceptual reference architecture model.

### 2.3.2 Educational Modeling Languages

A learnflow or pedagogical model is described using an EML, which includes a vocabulary or set of words to describe a process like activity, activity structure, role, and outcome, as well as pedagogical concepts like objectives, learner, and support activity. It also includes a grammar, a set of relations between these concepts, as well as consistency rules. An EML normally has an XML representation. The XML representation is used by the delivery system or learning management systems to run the described e-learning process. Thus, the XML file is the interface between the modeling of a learning process and its execution.

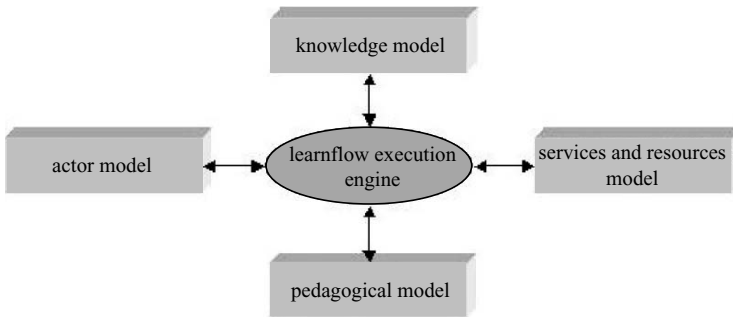


FIGURE 2.2. E-learning conceptual architecture model.

The main elements of an EML are the concepts of process, also called method, activity graph, or unit of learning, and the activity.

- The process is the structure describing the learnflow, the way in which activities are organized. It reflects a pedagogical strategy and is the backbone that connects the other elements of the model. Most EMLs allow the modeling of sequences of activities, of choice and decomposition structures, and of distribution and synchronization of parallel activities.
- The activity is the building element of the process. It describes the actual pedagogical action. An activity is related to other elements such as the type of actor who must execute the activity and the resources and services proposed to him to do it. The activity information includes pedagogical objectives and prerequisites, proposed time and eventually outcome information, and evaluation criteria and weight.

The EMLs are expected to have the expressive power required to describe any pedagogical model and to be neutral regarding pedagogical theories. Although it is true that any model can be described in terms of activities, relations between activities, roles and actors, resources and services, and objectives [14], EMLs' focus on process and activities makes them more suitable to represent pedagogical models centered on the activity than, for instance, those centered on collaboration and actors' interaction [27].

Languages such as MISA-MOT [36], COWL [43], and EML [21] are educational modeling languages in the sense that they offer a vocabulary to describe pedagogical models, a grammar to relate this vocabulary, and an XML binding. Nevertheless, their scope is local: their XML binding is understandable only by their own delivery platforms.

In the next section, we describe the educational modeling language IMS-Learning Design (IMS-LD) [15], which is a specification proposed as an EML standard by the IMS Global Consortium in 2003 [15]. As the IMS Global Consortium groups the main players in the e-learning community, one might expect that IMS-LD will be widely accepted and that model editing tools as well as e-learning

delivery platforms will be developed or transformed to be compatible with this new standard.

### 2.3.3 Conceptual Elements of IMS-LD

IMS-LD is the standard educational modeling language approved by the IMS Learning Consortium in 2003 [15]. It is inspired by the educational modeling language EML [21], developed by the Open University of the Netherlands. Its goal is to provide a framework for the formal description of any education and learning process. It helps relate other specifications such as LOM [12] for the learning objects, RDCEO [13] for competency modeling, and LIP [17] and e-Portfolio [16] for the learner model.

To ease the implementation of these tools, the standard defines three levels. Level A ensures a basic behavior, level B adds properties and conditions, and level C adds notifications. The basic concepts of IMS-LD are shown in Figure 2.3.

**Process** (or the method): The process or learnflow is built into IMS-LD on a theatrical metaphor. A learning situation called a *method* in IMS-LD corresponds in the metaphor to a theatrical piece. As such, a piece can be played in different ways (the plays). A *play* is composed of sequential acts. In an *act*, the different roles or characters of the piece execute in parallel their script. In IMS-LD, the script to be executed can be an *activity* or a group of structured activities (*activity-structure*).

**Activity:** IMS-LD distinguishes between two types of activity: *learning activity* and *support activity*. The people involved in a learning situation may thus be

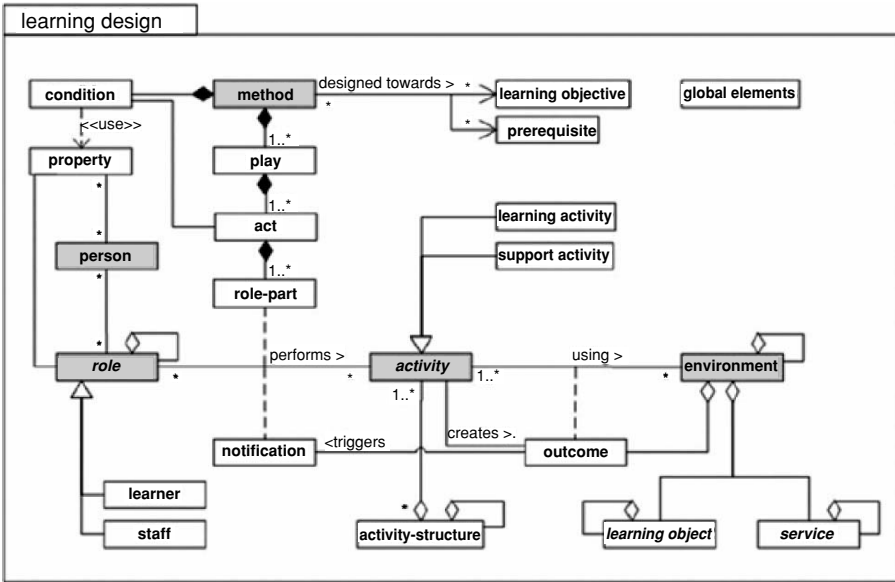


FIGURE 2.3. IMS-LD conceptual model [15].



assigned the role of a *learner* or a supporting role as *staff* member. Each activity is executed by a role (role-*part* association) using an *environment* composed of *learning objects* and services. The product of an activity, its *outcome*, may be integrated into the environments of other activities. A set of *prerequisites* and *learning objectives* may be tied both to the method and to the activities.

**Role:** The role *learner* can be modeled in a more explicit and rich way using other standards such as the portfolio, the LIP, or the competency model. IMS-LD allows for the description of learner information through the elements called properties. As explained in the next section, properties may be local or global and thus they help describe, among other data, learner information that is passed to or from the process and learner information that is generated and used only inside the process, possibly to personalize the training. During execution, the group of data that correspond to the learner properties is called the *dossier*.

**Local Properties, Global Properties, and Conditions:** Properties and conditions are part of the level B specification. Local properties allow for the management of information of persons, of roles, and of the learning process during the delivery of a learning situation (the execution of an LD run), while global properties act as parameters to interface the execution of the process with its context (institutional norms and conditions, learner portfolios, academic program data, etc.). Property values may be changed when an act or an activity is finished or when a rule is activated—its condition is satisfied. The modification of a property value may activate further changes, such as the ending of an activity, the hiding or showing of an activity, the generation of a notification, or the modification of other properties. In this way, by combining rules (called conditions in IMSLD) and properties, one can personalize the e-learning process and synchronize activities inside an act [27].

**Notifications:** Level C of the specification adds the concept of notification. A notification is a message sent automatically to a role when a condition is satisfied. This condition may be the ending of an activity, an act, a play or the whole method, the modification of the value of a property, or having an expression evaluated as true in the if part of a condition.

### 2.3.4 Tools for Learnflows

Many initiatives are being developed around the IMS-LD specification since its release in 2003. Research and development teams are working on the development or adjustment of editing, visualization, and delivery tools. Yet no full set of tools exists to support the whole process, and much of the work is still in a prototyping phase. In this section, we present some already-available tools for creating, manipulating, and delivering LD.

**Editing Tools:** Editing tools are applications that produce an IMS-LD document, that is, an XML document that is IMS-LD compliant. This document describes a learning design in terms of its different components: its global or pedagogical structure, which includes the method, the plays, the acts, the activities, and the activity structures; its learning environments with the associated learning objects

and services; and its properties, prerequisites, learning objectives, expressions, notifications, and roles.

Editors may also produce partial IMS-LD documents. It may produce a “content independent” pedagogical structure, that is, an XML document still compliant with IMS-LD but with only the pedagogical structure and possibly the roles, the services, and the properties defined; neither the prerequisites nor the objectives or the learning objects are specified. We call this content-independent LD an LD template. Its main interest concerns the possibility of having a repository of frameworks of pedagogical strategies from which to create a learning unit. Actually, there is no editing tool supporting the management of LD templates. The LAMS system [22] use templates, but the system is not yet fully compliant with IMS-LD.

### MOT+ Editor

At the Laboratoire en Informatique Cognitive et Environnements de Formation (LICEF) research center, development efforts have been placed on authoring methods editors and tools to facilitate delivery. The Méthode d’Ingénierie pédagogique de formations à distance (MISA) method is a mature instructional design method produced and refined in the last 10 years [37]. It uses a graphic educational graphical modeling tool, MOT, and it is supported by a Web-based design system, Atelier distribué d’ingénierie de systèmes d’apprentissage (ADISA).

In the context of the *Lorner project* [26], the IMS-LD learning level A specification has been transposed using the MOT+ [4] graphical modeling tool to develop an IMS-LD editor (Fig. 2.4). All IMS-LD objects [method, play, act, activities (three types), roles (two types), environment with services and learning objects] have been specified. Parallel to this work, a built-in parser to the MOT+ tool is being completed to produce an IMS-LD-compliant XML output.

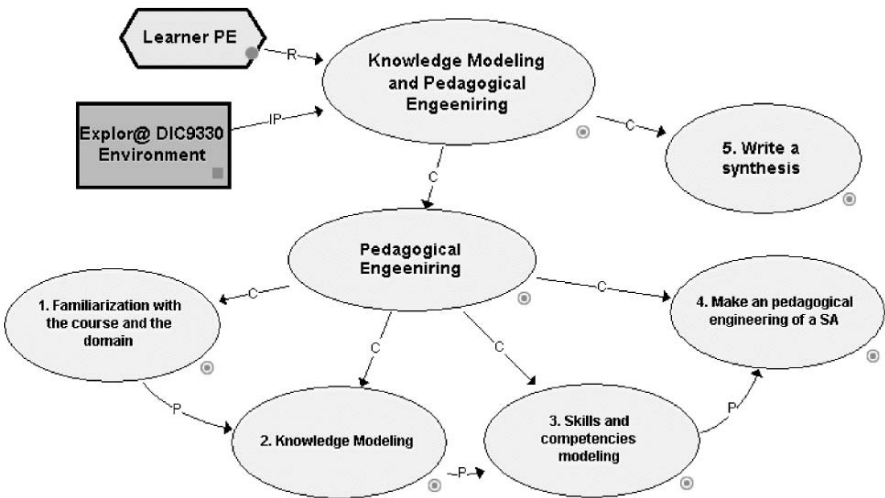


FIGURE 2.4. An LD in MOT+ editor.

Besides that, work will be undertaken, on the one hand, to adapt the MISA method to help designers produce compliant IMS-LD designs and, on the other hand, to extend the Explor@2 delivery system to support the production of run-time design instances. Currently, this delivery platform is not fully compliant with IMS-LD, although it does already take into account some multiactor issues. The goal is to enable Explora@2 to import IMS-LD-compliant courses.

Mot+ allows exporting LD in the XML format-compliant IMS-LD but not in a content package compliant with *IMS Content package* specification. In addition, the user cannot describe resources with *LOM specification* or plug a learning object stored in a learning object repository (LOR). However, the *LICEF research center* developed a learning object repository within the *eduSource project* [8] and a LOM editor (*LomPad* [24]). In the future, these three systems will be linked.

The validation process happens during the exportation in XML format compliant with IMS-LD schema. In addition, a manual is available that presents the graphical language and a light methodology to develop an LD.

### RELOAD Editor

RELOAD [19] is a project funded by the *Joint Information Systems Committee* (JISC) of Great Britain. RELOAD is also the name of the project tools. RELOAD is initially dedicated to the development of learning object management tools and to research on learning object management in collaborative online environments. During 2003, plans to develop a LD editor based on their SCORM [39] editor were put forth. Now, this editor supports all the three levels (A, B, and C) of IMS-LD.

RELOAD allows describing resources with LOM metadata, using these resources to elaborate an instructional model. This model can be compliant with SCORM 1.2 or IMS-LD. Besides, it makes it possible to generate the package in according to *IMS content package* specification.

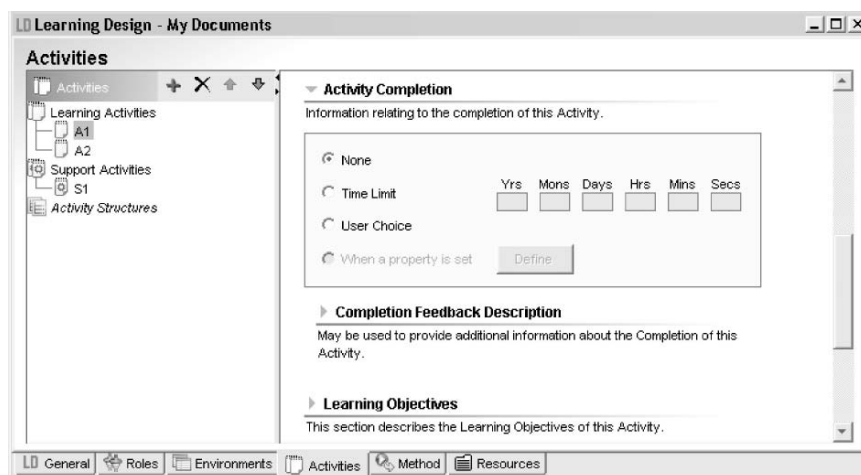


FIGURE 2.5. An LD in RELOAD editor.

As opposed to MOT+, RELOAD is a frame-based editor, not a graphical one. Different tabs allow defining the different components of the LD (Fig. 2.5). The editing process is constraint with frame-based solution. There is no validation functionality because syntax mistakes are inhibited. For help, a manual is available in which users can find a light process<sup>1</sup> to build an LD.

### Alfanet Editor

Alfanet Editor [41] is being developed by UNED (Madrid, Spain) within the European *Alfanet project*. Alfanet aims to create methods and tools for active and adaptable learning. Alfanet Editor is a module of the Alfanet LMS prototype. The user can create courses compliant with e-learning standards. Like RELOAD, this editor is frame-based (Fig. 2.6); users can produce a content package, and local resources are managed with LOM specification. The first difference is that there is a connection with a learning object repository. During the editing process, the user can retrieve resources within this repository. The second difference is that Alfanet editor is only compliant with the level A of the IMS LD specification. A user manual explains the editor functionalities; no methodological design approach is proposed in the manual.

Whereas MOT+ and RELOAD installations are easy and without specific preliminary setup, the user must install the *Groove Workspace* application to launch Alfanet Editor. On the other hand, once installed, this system proposes communication tools and allows collaborative work.

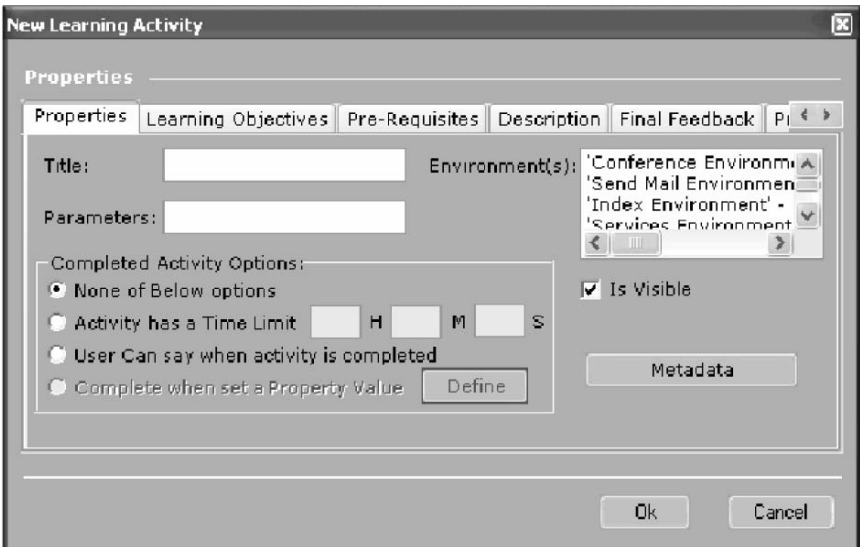


FIGURE 2.6. An LD in Alfanet editor.

<sup>1</sup> Light process is a process to construct main IMSLD components and elements.

**Execution tools:** A valid IMS-LD document may be used to actually produce a learning environment. The same IMS-LD document can be used to produce many particular learning environments in the same way as the same face-to-face course may be offered in different sessions at different times, for different groups of students with different teachers. As for those courses, a process of describing the particular session is needed. In terms of IMS-LD, this production process makes the association between an LD document and a session or run: starting time, learning community, etc. Such a session is called an *IMS-LD instance* or an *IMS-LD run*. This production stage can be supported by a specific tool or by the execution tool itself.

An IMS-LD run is executed or delivered to the different members of the learning community (learners, teachers, tutors, experts, etc.) through a delivery platform. Each one of these actors has his/her own view of the run, which includes his/her personal properties, activity trace, etc. Each of these views is called an *IMS-LD personalized run* or *IMS-LD personalized instance*.

Finally, all those states may be applied to IMS-LD in all three levels, A, B, and C, thus having, for instance, IMS-LD instantiations of IMS-LD documents that are level A compliant.

Efforts were concentrated on edition tools since 2003. In the last years, some companies and research centers start adapting their delivery system in order to import LD documents and execute them. For example, Blackboard Inc. is adapting EduBox Player [7] and integrating it in its popular e-learning platform. EduBox Player is an execution environment for EML-compliant pedagogical model. IMS-LD was inspired in EML, so adapting EduBox Player to be IMS-LD compliant should be straightforward.

In Canada, the e-learning delivery system Explor@2 of LICEF research center is being modified to import the LD documents produced by the MOT+ Editor.

Probably the most advanced effort in this direction is CopperCore [32]. CopperCore application is a sequel to research and tool development carried out by OUNL (Open University of the Netherlands) researchers in the EduBox project, sponsored by the EU Alfabet project, and launched in February 2004. CopperCore is an open-source project, consisting of a set of Application Program Interface (APIs) allowing the production and delivery of an IMS-LD unit of learning. These APIs cover publication, administration, and a run-time engine for all levels of IMS-LD. With the collaboration of SLED Project [40], CopperCore can be used like a Web service simple object access protocol (SOAP). For the time being, no available e-learning delivery system embeds CopperCore engine.

## 2.4 XPDL as a Business Process Language

There exist many languages and models for representing business processes. Some of the most widely known are BPMN [3], BPEL4WS [1], WSBPEL [31], and XPDL [46]. These languages have different notations to describe data flows, transitions, and control of the process being modeled. Some of these notations describe

these elements with graphical languages. Most of them propose an XML binding to facilitate the export of the process to different enactment engines.

Various groups and associations are actively working on the definition of standard models and languages that allow for the description of both the business process and of its instantiation and execution. OASIS [31] is refining the language WSBPEL, WfMC [11] has proposed XPDL, and BPMI [2] is responsible of a notational model for process description called BPMN [3].

As for e-learning systems, the main interest of workflow or business process modeling languages is to provide a vocabulary and a grammar sufficiently rich as to express any business model in a way that can be understood and implemented by a workflow management system.

In this section, we study in detail the process modeling language XPDL, XML Process Description Language. XPDL is the language proposed by the Workflow Management Coalition and it is part of a broader architecture proposed as the reference model for the definition, implementation, and interrelation of application intended to support the modeling and execution of business process workflows. Section 2.4.1 presents the WfMC reference model. Section 2.4.2 presents the XPDL language, and in section 2.4.3, we describe various existing support tools.

### *2.4.1 Workflow Reference Model*

The Workflow Management Coalition (WfMC) [11] defines a workflow as being the total or partial automatization of an industrial process. The workflow helps automatically support processes that include the circulation of information, documents, or tasks between participants based on a set of rules. A workflow management system defines, manages, and executes workflows. Workflow execution, also called enactment, consists of the execution of its composing activities as well as of the applications and environments associated with these activities. The workflow control model determines the execution order of those elements.

The workflow reference model proposed by the WfMC is described in terms of a central component, the system engine or workflow enactment service, and of five interfaces between this engine and the other components of the system (Fig. 2.7). Interface 1 makes the link between the engine and process definition tools. It is through this interface that a process description—an XPDL document—is passed to the engine to be executed. Interface 2 makes the link with the client applications. Interface 3 helps connect and launch the tools that are used in the different activities of the workflow. Interface 4 allows for the interaction between different enactment engines. Finally, interface 5 describes the process audit and management services.

### *2.4.2 Process Description Languages or Workflow Models*

Interface 1 of the workflow reference model gives the enactment engine the definition of a process it has to execute. The definition of a process is “the representation of a business process in a form that support automated manipulation, such as modeling, or enactment by a workflow management system” [46]. This definition or model includes a structured organization of individual steps called tasks or activities; the definition of agents or human actors who can execute a particular task;

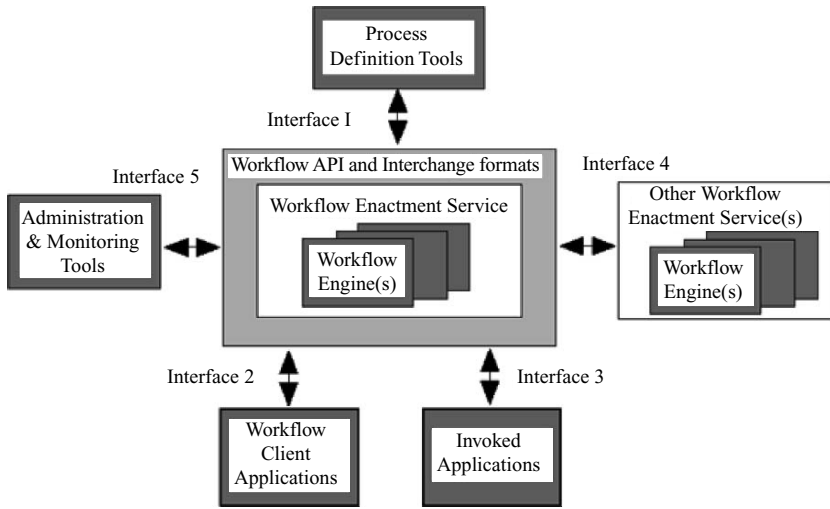


FIGURE 2.7. Workflow reference model [11].

the roles or actor types and the dependencies or transitions that describe the activities' execution order and conditions, and finally the resources; and the external applications that are available to the actor of an activity [6,23].

The main component of the workflow model is the workflow itself. This flow includes all the activities of the process as well as the transition rules between these activities and the data flow between them. The concept of transition and control pattern is central to the model. Definition process languages propose different kinds of transitions and control pattern (sequence, parallel paths, synchronization, multiple choice, single choice, etc.).

### 2.4.3 The Conceptual Elements of XPDL

The XML Process Definition Language (XPDL) was produced by the Workflow Management Coalition (WfMC) in 2002 [46]. Version 2.0 of the language is being defined and a draft was published in February 2005. This new version is fully compatible with version 1.0 and its main modification is its intended compatibility with the Business Process Modeling Notation (BPMN) defined by BPMi [2].

The goals of XPDL are (1) to provide a business process representation language in a way that allows for automated manipulation: modeling, instantiation, simulation, visualization, audit, documentation, etc., and (2) to clearly distinguish between the process definition and the process execution and to provide an interface between the process modeling and the process execution so as to be able to link different modeling tools with different execution engines.

XPDL describes the high-level entities that appear in a process definition (Fig. 2.8).

The rest of this section presents the main elements of the XPDL language, in its version 1.0. First, we present the elements that have a scope global to various

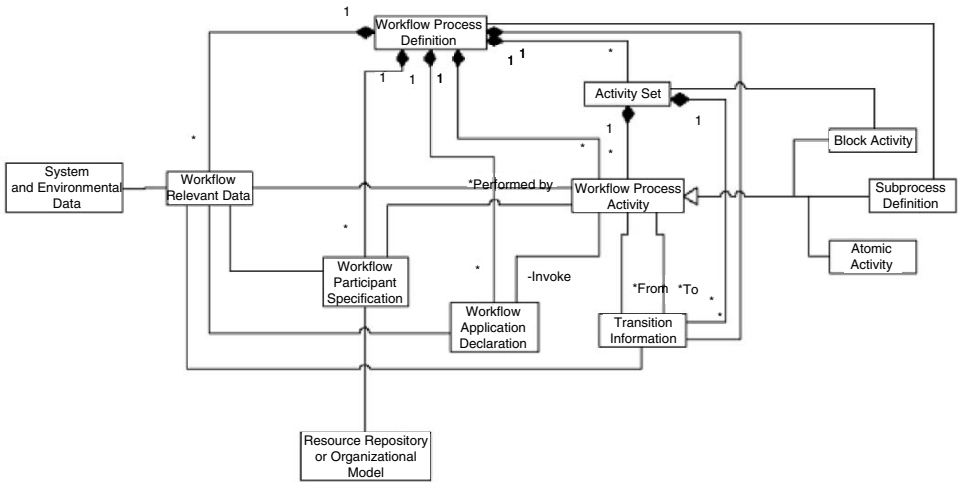


FIGURE 2.8. XPDL metamodel [46].

process definitions, then the ones that are local, whose scope is only one process definition.

**The Package Structure:** In a XPDL definition, the element with the broader scope is the package. It contains elements shared by all the processes inside it as well as the definition of each of these processes. Table 2.1 describes the most relevant elements of a package.

**The Structure of a Process:** The building piece of a process is the activity. A *workflow process activity* represents a task to be made. An activity is executed by a participant or by a resource (link with the *workflow participant specification* or with the *workflow relevant data*) eventually using an application (link with the *workflow application declaration*).

An activity may be a whole subprocess having its own activities and related to the main process by the subprocess definition elements. An activity may be a structured

TABLE 2.1. Elements of a XPDL package. The graphic icons are the graphical representation of these elements in the graphical editor, JaWE [9]











Element	Name	Description /Use
	Workflow Participant Specification	Participants are part of the organization model.
	Workflow Application Declaration	XPDL allows the declaration of tools or applications that will be invoked by the workflow process.
	Workflow Relevant Data	Represents the variables in a package definition, used by activities, processes, and subprocesses.
	External Packages	Definitions in other XPDL packages that could be referenced in this package.



TABLE 2.2. Elements of an XPDL process. The graphic icons are the graphical representation of these elements in the graphical editor, JaWE [9]

Element	Name	Description /Use
	Workflow Process	This element represents a workflow process definition.
	Generic Activity	Represents indivisible steps in a workflow process.
	Route Activity	This kind of activity is used for control flow and synchronization of other activities.
	Block Activity	Set of activities and transitions enclosed in a single activity.
	Subflow	This kind of activity represents a new process definition. This process could be invoked in a synchronous or asynchronous way.
	Transition Information	A transition represents the link between two activities. It contains a condition to determine if it will be activated or not.

block of activities, the *activity block*, composed of a set of more basic activities, the *activity set*. Finally, an activity can be an atomic task, the atomic activity. Activities can be *generic*, if they make part of the process, or *route* activities, used for control purposes. One particular route activity is the empty activity (Table 2.2).

**Participants and Roles:** *Workflow participant specification* contains information on the potential actors of the different activities. These participants may be human or machine agents. The specification includes the possibility of linking this element with a bank of participants or resources. Another concept, the *workflow application declaration* establishes the interface between the workflow and the applications and services it needs to call to support a particular task.

**Relevant Data:** XPDL specifies two types of data: the workflow relevant data, which are data generated during the enactment of the process, and the system and environment data, which help describe the execution context. Both types of data can be used in the activities and in the transition condition as well as interface parameters to pass data to other system components.

**Extended Attributes:** These elements represent information that can be interpreted and used by the tools, allowing for the particularization of the process to some tool properties.

#### 2.4.4 Tools for Workflows

In this section, we analyze some existing tools that help define and execute process described in XPDL. Each tool is described in terms of its functionality. For each

one, we present the main extended attributes used to implement particular aspects not specified in the standard.

**JaWE: A Process Editing Tool:** XPDL representation of processes may be complex and difficult to visualize and trace in a textual way. Graphical editing tools should be used. The new version (version 2.0) of XPDL includes information that helps manipulate the display of the process main elements by any design tool. Each application will be able to create a special element called *nodegraphinfo* to store information on the presentation of the elements. Information generated by one application could be reused by another one.

However, existing process editors still work with version 1.0 of XPDL. In this version, the only way to deal with the presentation of elements is the use of extended attributes.

Enhydra JaWE [9] is a tool to design workflows compatible with XPDL version 1.0. JaWE is a graphical workflow editor. With JaWEi+ is possible to design a process in a graphical way and generate its XPDL representation. It is also possible to import XPDL process description and to visualize them graphically.

Being built on Java 1.4, JaWE installation requirements are minimal. It runs on UNIX and Windows. Additional required libraries are distributed with the application.

A process in JaWE is contained into a project. The definition of a process in JaWE is made top-down in two steps. Entities at the package level, such as workflow application declaration, workflow relevant data, workflow participant specification, extended attributes, external packages, and data types, are defined in the first step. In the second step, one can create workflow processes inside this package. Inside a workflow process, one can define the following entities: process definition, application declaration, relevant data, transitions, and process activities.

JaWE offers the possibility of validating the process defined, according to XPDL version 1.0 specification. Thus, while building the package and the composing process flows, the designer may validate it and identify the wrong parts.

JaWE uses extended attributes for the activities, the transitions, and the workflow process. These attributes are explained in Table 2.3.

TABLE 2.3 XPDL extended attributes in JaWE

Extended attributes in the activities		Use
Participant ID		Role in charge of the activity
X Offset		Offset in the x axis of the activity
Y Offset		Offset in the y axis of the activity
Extended attributes in transitions		Use
Routing Type		Transition type
Extended attributes in the process		Use
Start of workflow		Initial activity of the workflow
End of workflow		Final activity of the workflow
Participant visual order		Order of presentation of the participants in the editor

**Process Execution Tools:** A second group of workflow tools concern workflow execution engines. These engines can import business process definitions, described in XPDL, and launch and execute an instance of the process described.

There are quite a few workflow-editing tools, and various execution engines have been developed, many of which are GPL licensed, and many others are commercial products. In this section, we describe two of these execution models: WFMOpen [5] and Enhydra Shark [10]. Both are widely used; they are GPL and LGPL licensed, respectively, and both use J2EE technology in their implementation.

### WFMOpen

WFMOpen is a tool developed on J2EE that offers a component for process execution based on a set of Java interfaces. This component is based on the WfMC specifications as well as on the specification *Workflow Management Facility V 1.2* of OMG [34].

From a functional perspective, WFMOpen offers services to import process definitions defined in XPDL, to instantiate them, to manage the created instances, to assign participants to the instances, to assign products to the activities, to call external tools, and to control the workflow of these instances. In addition, it offers workflow monitoring and management services as well as some services for the construction of client applications. The workflow components may manage an unlimited number of instances of the original process. These instances, once launched, are independent of the original process definition.

From a logical perspective, the tool has a central component called Danet's Workflow Component that offers the basic functionalities of the execution engine in charge of the execution of the process instances. This component has three sub-components: Workflow Core, Workflow Engine, and Resource Assignment Facility.

The Workflow Core facilitates process creation and instantiation and controls the execution of the created instances. The Workflow Engine subcomponent is in charge of the activation of the activities, and the subcomponent Resource Assignment Facility is in charge of the resource assignments.

The main component, as well as its subcomponents, is implemented through five packages that compose the API of the Workflow Engine. Those APIs are the component workflow API, the workflow management system API, an API for the invocation of application control agents, an API to use the resource assignment service, and an API to use the resource management system.

The workflow component uses a set of extended attributes inside the imported XPDL process definitions. The list of extended attributes used by this component is presented in the Table 2.4.

### Enhydra Shark

Enhydra Shark is a workflow engine based on WfMC and object management group (OMG) specification; thus, it used XPDL 1.0 as the process description language.

TABLE 2.4 XPDL Extended attributes used by the WFMOpen workflow component

Attribute	Use
Implementation	Extension used inside the declaration of applications. This attribute is used to specify the behavior of the tool to be used.
Remove Closed Process	Establishes how the engines will discard the ended processes.
Debug	Indicates if the processes are started in a debugging mode or not.
Audit Event Selection	Indicates which events will be audited during the process execution.
Store Audit Events	Indicates if the selected events will persist in an event log or not.
Deferred Choice	Indicates if a transition AND-Split must be executed following the pattern Deferred Choice.

Shark may be used as an embedded library in different types of applications, either in Web environments or in more traditional client/server applications.

The most important feature of Shark is the fact that it does not use extended attributes for the process definitions. The process definitions handled by Shark are easily transferred to other editing or execution tools.

Moreover, thanks to its common object request broker architecture (CORBA) [33] interface, Shark may be used by different clients. It offers also a mechanism to make the integration with LDAP [25] for the use of information on established organizational structures.

## 2.5 Translation Scheme

The scope of the translation scheme is the model of control from IMS-LD to XPDL. Only level A is considered.

This section presents a common vocabulary in order to describe, in the same terms, the subjacent models of control of IMS-LD and XPDL. The reason to introduce this vocabulary is to avoid the ambiguities raised because of the use of the same terms with different meanings in the models.

The first subsection presents the elements of the static model; the second subsection presents the elements of the dynamic model; the third subsection describes the model of control of IMS-LD; the fourth subsection describes the model of control of XPDL; and, finally, the fifth subsection shows the proposed translation.

### 2.5.1 Static Aspects of the Common Model

This subsection introduces some static elements of a process model. They are an abstraction of the main characteristics of the e-flows and the w-flows:

**Role:** A role defines a type of actor by characterizing its responsibilities and/or abilities, and by associating a name. An actor can play various roles and many actors can play the same role.

**Datatype:** A datatype defines the structural characteristics (syntactical) that represent the possible elements in the domain of application. There are (predefined) simple datatypes and complex datatypes defined by the use. A simple datatype has a name and a set of values. A complex datatype has a name and a description of its composition.

**Activity:** An activity is a discrete task with a start, an end, and a well-defined objective. The activity has associated a role, which describes the responsible actor of the task, and, it can have a set of services and tools to be used by the actor, if required. Moreover, an activity is characterized by a set of input data (to be used by the actor) and a set of output data (to be produced by the actor). An activity can be instantiated only once or multiple times simultaneously.

**Transition:** A transition defines an order relationship between two activities.

**Process:** A process is a structured set of activities that share a defined common objective. *Structured* means that the activities are executed following an established order defined by some rules in the model of control and based on transitions. A process has an execution context consisting of a set of data (name and type). This context allows the manipulation of information shared by all the activities in the process; the data in the context are used to make decisions at run time.

**Services and Tools:** Services and tools are elements not controlled by the execution engine; nevertheless, an actor can use them, based on his own decisions, to achieve the objective of the activity. The tools are used by the actor to transform data, whereas the services are used by the actor to acquire knowledge, communicate something, participate in a discussion, etc.

### 2.5.2 *Dynamic Aspects of the Common Model*

This subsection introduces some concepts that appear at run time.

**Actor:** The actor is responsible for transforming or creating data to accomplish the objective of an activity by means of services and tools to perform the associated task. The execution model should include a policy to assign actors to instances of activities.

**Data:** Data are the values that elements in the application domain can assume. These elements are typed and have a name.

**Process Instance:** A process can have several instances at run time, and each one serves to accomplish the defined objective in the process definition. Each instance has its own execution to be used by the activity instances of the process.

**Activity Instance:** When a process is instantiated, at the same time all its activities are instantiated too. An activity instance can be in one of the three states: inactive, active, or finished. Each activity instance has its own execution context consisting of the received input data and the global data of the process instance to which the activity belongs.

The execution engine follows the order defined by the transitions and uses the process and activity context to determine if a transition can take place or not. The set of transitions arriving at an activity determines if the activity can be initiated

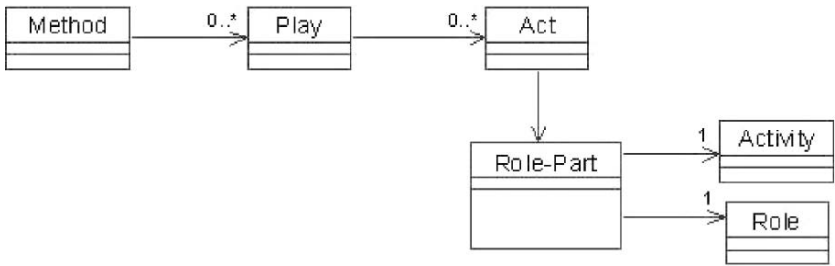


FIGURE 2.9. Conceptual structure of the model of control in IMS-LD.

or not. Furthermore, it should be defined if the activity instance will be executed in sequential or parallel mode.

**Model of Control:** The model of control includes the elements whose life cycle is administrated by the execution engine. It means that the elements whose execution is controlled by the actor are not part of the model of control.

2.5.3 Model of Control of IMS-LD

IMS-LD uses a theatrical metaphor to define pedagogical processes in a standard way. Furthermore, the metaphor predefines some control structures assuming similar to every pedagogical model.

In the theatrical metaphor, the notion of transition is not explicit, although the metaphor establishes an organization among the activities, given as a result a predefined topology of control. Figure 2.9 shows the conceptual structure of the model of control in IMS-LD.

The execution engine interprets the structure shown in Figure 2.10 as follows: the plays, in a method, are executed simultaneously. The acts in a play are executed in a sequential way following the order in which they were defined, i.e., an act is not initialized until its predecessor was finished. The role-parts are executed in parallel. The activity-structure, in a role-part, can have *sequential* or *selection* type. If the type is *sequential*, the set of activities associated with the activity-structure is

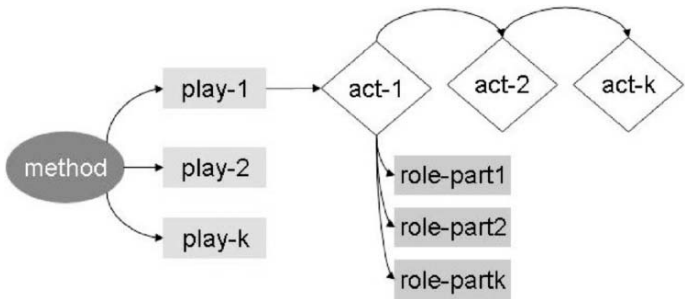


FIGURE 2.10. Execution of a method.

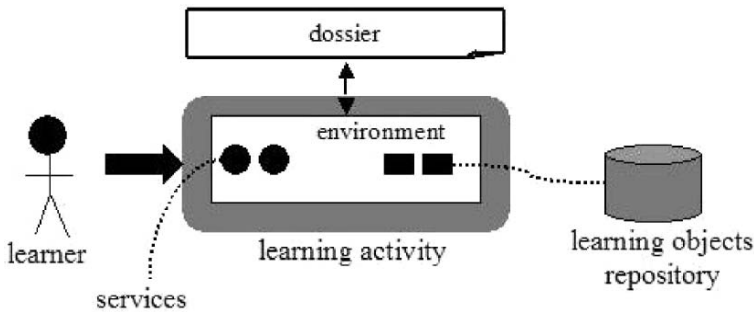


FIGURE 2.11. Execution of a learning activity.

executed in a sequential way. If, on the contrary, the type is *selection*, the role-part finishes when a number of activities, greater or equal to a predefined minimum, finishes. Finally, if any of the activities in the role-part is a support activity and this has associated a group of actors, the execution engine has to create an instance for each actor in the group, and the support activity finishes when all the instances finish.

Figure 2.11 illustrates a learning activity being performed by a learner. In a structure called the *environment*, the actor has available services, tools, and learning objects. The use of the elements in the environment is decided on and controlled by the learner and not by the execution engine. The input data to the activity are values of *properties* and they are in a set called a *dossier*. The actor performing the activity can modify the data in the dossier. The actor decides when the activity is finished.

Levels B and C of IMS-LD imply an extension to the model of control. In that case, the execution engine should take into account that some elements, in particular, the activities can become visible or invisible inside a process instance. It means that the graph of control is changed at runtime.

### 2.5.4 Model of Control of XPDL

The XPDL model does not have a common metaphor because there are a vast number of application contexts for the workflow process. It can range from a process to fulfill a request for a credit in a financial company, to a process to develop software. For this reason, the language has to provide the necessary structures to describe different control flows.

In XPDL a process is defined as a set of activities and a set of transitions, where the transitions define a partial order among the activities. There are four types of basic activities: (1) simple, which represents a punctual activity or a routing activity; (2) a block activity, which is a group of activities perceived as a unit, (3) a subflow, which represents a synchronic or asynchronous execution of an independent process; and (4) an application, which encapsulates the execution of an external application.

A transition relates two activities and has a condition associated with it. At run time, if the evaluation of the condition is true, the transition can be done.

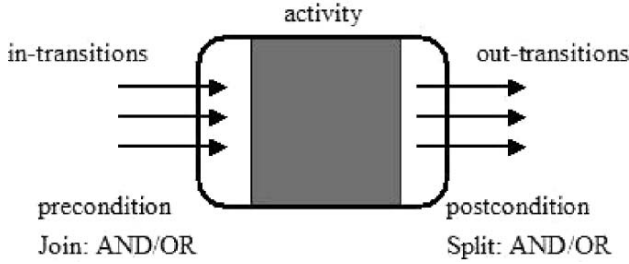


FIGURE 2.12. Model of control of XPDL.

The model of control of XPDL is described inside the activities by means of preconditions and postconditions (Fig. 2.12). A precondition determines if an activity can be initiated, taking into account the set of transitions arriving to it. There are two possible cases: (1) the activity is initiated when, for all the transitions arriving to it, the condition is true (Join-AND); or (2), when for at least one transition, its condition is true (Join-OR).

A postcondition determines the set if activities to be executed after the current activity. There are two possibilities: (1) all activities reached by a transition from the current activity are executed (Split- AND); or (2) only the activities reached by a transition from the current activity, whose condition is true, are executed (Split-OR).

### 2.5.5 The Proposed Translation Scheme

Our translation scheme is centered on the model of control. This subsection describes the translation of the theatrical used by IMS-LD to the XPDL language constructors and the limitations of a direct translation.

**Translation of the Theatrical Metaphor:** This subsection presents the translation scheme in a top-down approach. Furthermore, we use the workflow patterns proposed in [42] to help explain the translation.

#### Method

A method is translated to a Block Activities in XPDL; each activity represents one of the plays defined in the method. Block Activities are executed in parallel and do not need to be synchronized. Figure 2.13 illustrates the translation. This is an

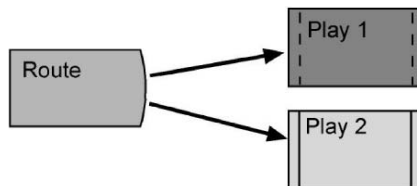


FIGURE 2.13. Translation of a method to XPDL.



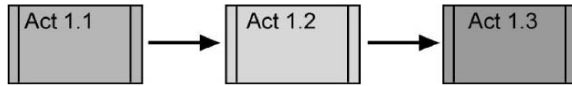


FIGURE 2.14. Translation of a play to XPDL.

example of a method with two plays executed in parallel. This type of execution corresponds to the parallel split pattern, and to the implicit termination pattern [42].

### Play

A play is translated to a Block Activities in XPDL; each activity represents one of the acts defined in the play. The activities are executed in a sequential order, using the sequence pattern [42]. Figure 2.14 presents a play composed by three acts; they are executed in sequence. The conditions of the transitions are all true, the pre-condition is a Join-OR and the postcondition a Split Join-OR.

### Act

An act is translated to a set of Block Activities in XPDL; one for each role-part. They are executed in parallel and synchronized when all are finished. Figure 2.15 shows an act composed of two role-parts and synchronized using two routing activities. This translation follows the parallel split and synchronization merge patterns [42].

The first routing activity has as its postcondition a Split-AND on the role-parts included in the act. The second routing activity synchronizes the termination of the role-parts executed in parallel, and its precondition is a Join-AND.

### Role-Part

A role-part is translated as a Block Activity that contains the activities of its activity structure. There are two cases to be considered according to the activity-structure type (*sequence* or *selection*).

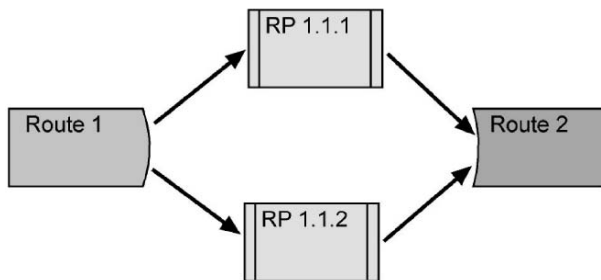


FIGURE 2.15. Translation of an act in XPDL.

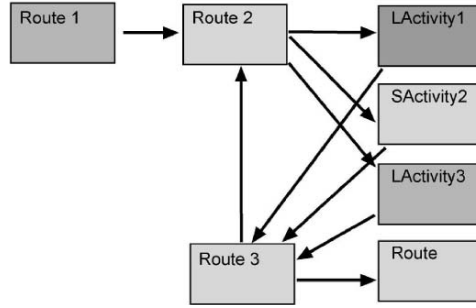


FIGURE 2.16. Translation of an activity structure (selection type) in XPDL.

In the *sequence* case, the activities in the Block Activity are executed according to the sequence pattern [42]. In the *selection* case, the translation uses the arbitrary cycle's pattern [42], as shown in the example of Figure 2.16.

Figure 2.16 shows a role-part composed of three activities: two learning-activities (LActivity1, LActivity3) and, one support activity (SActivity2). The flow of control starts with a routing activity (Route 1) followed by a second routing activity (Route 2). Route 2 uses a Split-Or postcondition (with exclude conditions) to choose during an iteration, with only one possible transition. Once the selected activity is finished, Route 3 is initiated. Route 3 decides if a new iteration has to be executed or not, by verifying if all the selected activities have been executed.

In the case of multiple and simultaneous instances at run time, it is possible to make the translation but with a restriction: a predefined maximum number of instances to be created at run time has to be known. This scheme follows the multiple-instances run-time pattern [42].

**Limitations of the Direct Translation:** There are at least two characteristics of the model of control of IMS-LD that cannot be translated directly to XPDL: the possibility to create at run time an arbitrary number of instances of the same activity, and the possibility to modify the visibility of the activities at run time.

To solve the first problem, we have defined the restriction on the maximum number of instances to create. The second problem appears in levels B and C of IMS-LD, which are out of the scope of our work.

## 2.6 LDX-Flow Tools

This section describes LDX-Flow. This is a set of tools to support the life cycle of a learning design. The strategy used to define the tools was the transformation of learnflows into workflows. The tools were developed as a part of a collaborative project between the University of Los Andes and the Téléuniversité du Québec (TELUQ).

The section has four subsections: the first three describe the functional, logical, and physical architectures, and the fourth presents some of the results achieved during the experimentation.

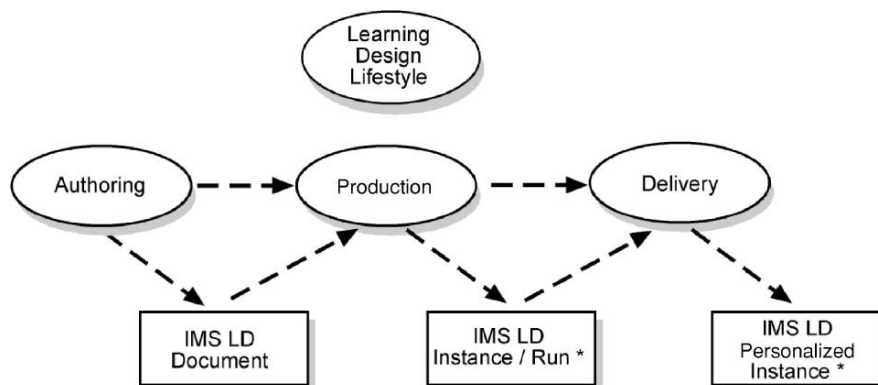


FIGURE 2.17. Learning design life cycle.

### 2.6.1 Functional Architecture

Figure 2.17 presents the life cycle of a learning design.

LDX-Flow allows the creation, editing, translation, and execution of a unit of learning defined using IMS-LD. LDX-Flow is a set of five tools that support the life cycle presented in Figure 2.17. In Table 2.5, we summarize their main functionality. Currently, our tools are limited to manage learning design descriptions compatible with level A of IMS-LD. Levels B and C are not yet supported.

**Functionality of the LDX-Editor:** By means of the LDX-Editor (Fig. 2.18), it is possible to create and edit learning designs using the IMS-LD standard. A user (for example, the instructor) can manipulate, in an interactive way, concepts like method, play, component, activity structure, learning activity, support activity, etc. LDX-Editor facilitates the visualization of the course through a graphical

TABLE 2.5 The set of tools of LDX-Flow and their relationship with the process in the life cycle of a learning design

Process	Tool	Description
Authoring	LDX-Editor	Allows the editing of learning designs using the IMS-LD elements. Once the information is edited, this could be stored in an XML file following the standard specification.
Production	LDX-Translator	Partially supports the production of a learning design. It corresponds to the translation of the control model defined in IMS-LD and to the control model of the XPDL language.
Delivery	LDX-Enactor	Executes instances of XPDL processes, which were generated using the LDX-Translator tool.
	LDX-Client	Allows the diverse actors of a learning unit to interact with their execution instances.
	LDX-Admin	Offers services of administering and monitoring the elements involved in the execution.
	LDX-Resource	Offers services of assigning resources to the activities.

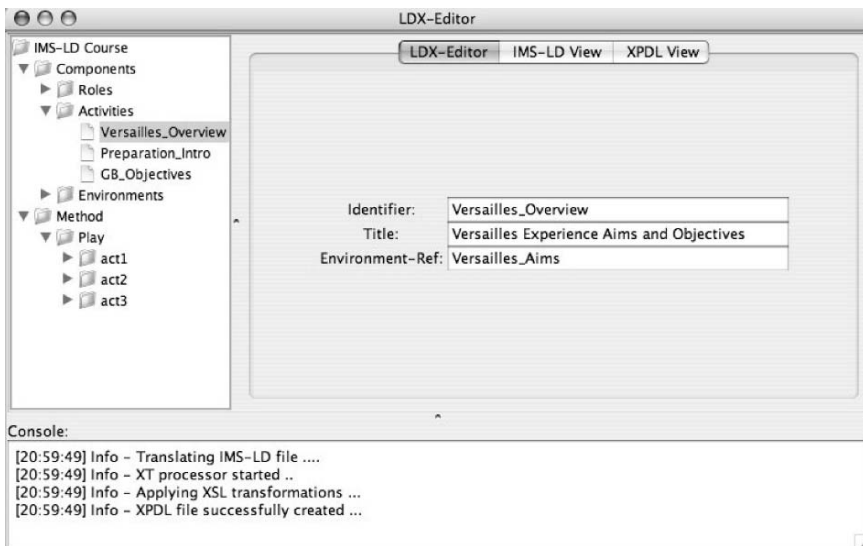


FIGURE 2.18. LDX-Editor.

presentation based on a tree that shows the composition of the different elements. Furthermore, the tool guarantees that the course built are structurally correct with respect to the standard.

**Functionality of the LDX-Translator:** LDX-Translator performs the translation from IMS-LD to XPDL. The translations are done according to the schema presented in section 2.5. Moreover, the tool can add to the output XPDL file the set of extended attributes required for the JaWe editor to visualize the XPDL process.

**Functionality of the LDX-Enactor:** This tool allows the instantiation and execution of the processes created in the LDX-Editor and translated by the LDX-Translator. The tool is responsible for the management of the enactment of the processes. This offers services to import XPDL process definitions. Once the tool imports the processes, they can be instantiated and executed.

**Functionality of the LDX-Resource:** This tool offers services to manage resources. These resources are the participants (human actors) in the activities.

**Functionality of the LDX-Client:** This tool offers services to allow actors to interact with an instance of a process being executed by the LDX-Enactor. Using this tool, users can know the instance of the process in which they are participating, the assigned activities already finished, and the assigned activities pending initialization.

The tool has a specialized interface according to the type of role. For instance, if the user is an instructor (staff role), he can consult the state of advancement of every learner in his course.

**Functionality of the LDX-Admin:** This tool offers services for the administration and monitoring of the LDX-Enactor using JMX technology [30]. For example,

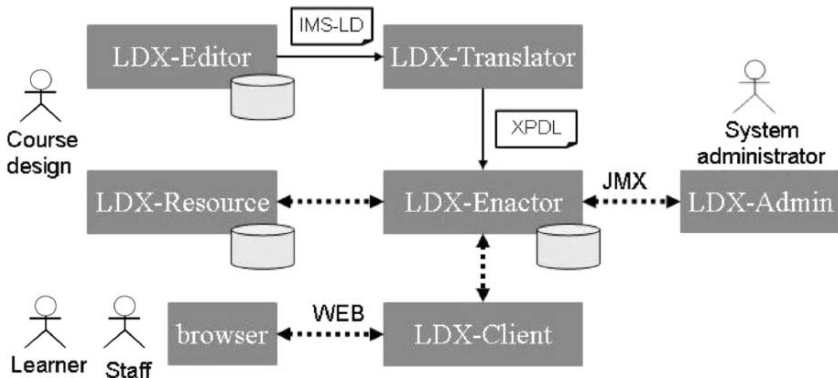


FIGURE 2.19. Logical architecture of LDX-Flow.

by using this tool, it is possible to consult how many instances are in execution, to review the log of the system, and to monitor the communication between the connected clients.

### 2.6.2 Logical Architecture

Figure 2.19 shows the logical architecture of the suite of tools of LDX-Flow.

The goal of the design of the architecture was to allow each tool to support a process in the life cycle of a learning design. The editor allows the edition of learning units in IMS-LD and the connection to the translator to produce XPDL files. The translator receives learning designs and applies the transformation needed to produce correct XPDL documents. The enactor imports these documents and allows the instantiation and execution of processes defined there.

The LDX-Enactor tools serves as a bridge between the other tools composing the architecture. First, the enactor offers services to the client components (LDX-Client). Users through a browser access the client components. Second, the LDX-Enactor offers services to the administration and resource tools.

### 2.6.3 Physical Architecture

The logical architecture, presented in the preceding section, has been implemented using the Java 2 Platform, Enterprise Edition (J2EE) platform [28], specifically on the JBoss Application Server (JBoss) container [18].

We used session and entity Enterprise Java Beans (EJB) components to implement the tools: LDX-Translator, LDX-Enactor, and LDX-Resources. Oracle database [35] is in charge of the management of the persistence data. The LDX-Client was implemented as a set of Web components (JSPs and Servlets). The LDX-Admin was implemented using JXM technology [30]. Finally, we used Java [29] and the swing framework to implement the LDX-Editor.

### 2.6.4 *Evaluation of the Tools*

The master's program of the systems and computing engineering department of the University of Los Andes offers a seminar whose purpose is for the students to describe the state of the art of their research topic. The process defined, to achieve the goal, consists of a series of activities performed by instructors, students, and teaching assistants. The mechanics are based on the sequential assignment of a set of readings, preparation of reviews, discussion of the reviews in small groups, and publication, for the whole class, of a synthesis. The instructors have as task, the assignment of the readings, the participation in the discussion sessions, and the validation of the results. The teaching assistants have the administrative support to everybody in the class as their task.

During the first semester of 2005, we defined, using IMS-LD, a learning design to model the process described above. The process was executed with a group of 15 students, two teaching assistants, and four instructors. We followed a completed process of production and delivery using the LDX-Flow tools. This exercise allowed us to validate the translation scheme and to test the implementation of the tools in a real context.

From the gained experience until now, we have defined three lines for our future work on this project. In the first place, we know that it is necessary to perform a validation on a larger scale, to consolidate the tools, and to validate, in other contexts, the translation scheme proposed. In the second place, we would like to evolve the translation scheme to use the version 2.0 of XPDL. Finally, we would like to study some ways to use the extended attributes of XPDL to support levels B and C of IMS-LD.

## 2.7 Conclusion

Current educational modeling languages, in particular the standard IMS-LD, offer a rich framework for the description of multiactors and adaptable pedagogical models. These properties create new requirements for e-learning delivery engines. While some IMS-LD editing tools are being released, no delivery engine is yet fully IMS-LD compatible.

On the other hand, workflow management systems also manipulate workflow modeling languages to describe business processes. One broadly accepted process description language is XPDL, the language proposed by the Workflow Management Coalition. XPDL is supported by various commercial and shareware editing and execution tools.

This chapter has explored the differences and similarities between e-learning and workflow systems. The establishment of a common vocabulary between IMS-LD and XPDL has helped identify translation possibilities, has shown the expressive power of workflow control pattern, and has clearly identified particular elements of the e-learning world that might be difficult to translate into workflow modeling languages and to execute by workflow enactment engines.

The translation schema proposed as well as the LDX-Flow set of tools that have been developed, although reduced to level A IMS-LD components, are a first attempt to implement the support of the whole life cycle of an IMS-LD-compatible learning design: design, production, and delivery are automatically supported by LDX-Flow. The evaluation made has proven the validity of the process and the compatibility with the standards involved.

It is worth noting that the translation step opens broad possibilities for both e-learning and workflow management communities. In fact, one can imagine various life cycle paths such as editing an LD with the graphical editor MOT+, publishing it with LDX-Flow, and delivering on a workflow process engine.

The new process engine will take advantage of the analysis we have done to understand the subjacent models of IMS-LD and XPDL. The classification of concepts as the static ones and as those that only appear at run time has allowed us to better understand the requirements a process engine should meet to fully support the execution of an LD expressed on IMS-LD.

More conceptual research concerns the analysis of the differences encountered and of the possibilities of enriching both models with features included in the other model.

## References

1. Andrews, T., Curbera, F., Dholakia, H., et al. (2003) Business process execution language for Web services version 1.1. Technical report. <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>
2. Business Process Management Initiative (BPMi). <http://www.bpmi.org>.
3. Business Process Modeling Notation (BPMN). <http://www.bpmn.org>.
4. Centre de Recherche LICEF : Mot+ Editor. <http://www.licef.teluq.quebec.ca/francais/real/demot.htm> (20 Jun. 2005).
5. Danet.: WfMOpen Project. <http://wfmpopen.sourceforge.net>.
6. Eder, J., Gruber, W. (2002) A meta model for structured workflows supporting workflow transformation. In: Lecture Notes on Computer Science, 2435.
7. EduBox. <http://www.ou.nl/info-alg-edubox>.
8. eduSource Project. <http://www.edusource.ca>.
9. Enhydra Java Workflow Editor (JaWE). <http://jawe.objectweb.org>.
10. Enhydra Shark Open Source Workflow. <http://shark.objectweb.org>.
11. Hollingsworth, P. (1995) The workflow reference model. Technical Report TC00-1003, Workflow Management Coalition (WfMC).
12. IEEE. (2002) Draft standard for learning object metadata 1484.12.1.
13. IMS Global Learning Consortium. (2002) IMS Reusable Definition of Competency or Educational Objective Information Model. Version 1 Final Specification.
14. IMS Global Learning Consortium. (2003) IMS Learning Design Best Practice and Implementation Guide. Version 1. Final Specification.
15. IMS Global Learning Consortium. (2003) IMS Learning Design Information Model. Version 1 Final Specification.
16. IMS Global Learning Consortium. (2004) IMS ePortfolio Information Model. Version 1.0 Public Draft.

17. IMS Global Learning Consortium. (2005) **IMS Learner Information Package—Information Model Specification**. Version 1.0.1 Final Specification.
18. JBoss Inc.(2005) Jboss Enterprise **Middleware System**. <http://www.jboss.org/products/index>.
19. Joint Information Systems Committee (JISC). **Reload editor**. <http://www.reload.ac.uk>.
20. Kluijfhout, E. (2002) Stimulating the widespread use of EML. OTEC2002/22, <http://hdl.handle.net/1820/200>.
21. Koper R. (2001) **Modeling units of study from a pedagogical perspective, the pedagogical meta-model behind EML**. Technical report first draft, version 2. Educational Technology Expertise Center—Open University of Netherlands, Heerlen, Netherlands.
22. **Learning Activity Management System (LAMS)**. <http://www.lamsinternational.com/>.
23. Lei, K., Singh, M. (1997) A Comparison of Workflow Metamodels, Proceedings of the ER-97 Workshop on Behavioral Modeling and Design Transformations: Issues and Opportunities in Conceptual Modeling, Los Angeles, November 1997.
24. LICEF : Lompad. <http://demo.licef.teluq.quebec.ca/LomPad/en/index.htm>.
25. **Lightweight Directory Access Protocol (LDAP)**. <ftp://ftp.rfc-editor.org/innotes/rfc3928.txt> (2004).
26. Lornet. Lornet Project. <http://www.lornet.org>.
27. Marino, O., Contamines, J. (2004) **La modélisation de Scénarios Collaboratifs d'apprentissage: possibilités et limites du standard IMS-LD**. Communication au Colloque CIRTA, Congres ACFAS.
28. Sun Microsystems. **Java 2 Platform Enterprise Edition (J2EE)**. <http://java.sun.com/j2ee>.
29. Sun Microsystems. **Java API Specification**, <http://java.sun.com/reference/api>.
30. Sun Microsystems. **Java Management Extensions (JMX)**. <http://java.sun.com/products/JavaManagement>.
31. OASIS. **Web Services Business Process Execution Language (WSBPEL)**. <http://www.oasis-open.org/committees>.
32. Open University of the Netherlands (OUNL). CopperCore. <http://www.coppercore.org>.
33. **OMG. CORBA**. <http://www.corba.org>.
34. OMG. (2000) **Workflow Management Facility Specification Version 1.2**. Needham, MA: Document number bom/00-05-02.
35. Oracle Corporation. **Oracle Database**. <http://www.oracle.com/database>.
36. Paquette, G. (2002) **La Modélisation des Connaissances et des Compétences, pour Concevoir et apprendre**. Presses de l'Université du Québec, 2002, 352 pages.
37. Paquette, G. (2003) **Using learning object repositories: the eduSource suite of tools. eduSource Industrial Forum communications**. <http://www.edusource.ca/english/resources/eduSource-Moncton-AN.ppt>.
38. Paquette G., Marino, O., de la Teja, I., Léonard, M., Lundgren-Cayrol, K. (2005) **Delivery of learning design: The explor@ system case**. In: Koper, R., Tattersall, C. (eds.), **Learning Design: A Handbook on Modelling and Delivering Networked Education and Training**. New York: Springer-Verlag.
39. **SCORM. Sharable Content Object Reference Model (SCORM)**. <http://www.adlnet.org/scorm/index.cfm>.
40. SLED. <http://www.jisc.ac.uk/?name=sblds>.
41. **UNED.: Alfabet Editor**. <http://rtd.softwareag.es/alfanet>.
42. Van der Aalst, W. (2003) Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. QUT Technical report, FIT-TR-2003-06, <http://www.citi.qut.edu.au/pubs/ce-xpdl.pdf>, Queensland University of Technology, Brisbane, 2003.



43. Vantrois, T., Peter, Y. (2002) **Un système de workflows flexible pour la formation ouverte et à distance**. In: Frasson, C., Pecuchet, J.-P. (Dir.), *Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie*. Villeurbanne, France: Institut National des Sciences Appliquées de Lyon.
44. Vantrois, T., Peter, Y. (2004) Cow, a flexible platform for the enactment of learning scenarios. *International Workshop on Groupware CRIWG 2003, Lecture Notes on Computer Science*. New York: Springer-Verlag.
45. Vogten, H. (2005) **Designing a learning design engine as a collection of finite state machines**. *International Journal on E-Learning*. <http://hdl.handle.net/1820/303>.
46. Workflow Management Coalition (WfMC). (2002) **Workflow Process Definition Interface—XML Process Definition Language**. Version 1. Final Draft WfMC-TC-1025.