Expert Systems with Applications 39 (2012) 12799–12813

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Petri net-based engine for adaptive learning

Juan C. Vidal *, Manuel Lama, Alberto Bugarín

Centro de Investigación en Tecnoloxías da Información (CITIUS), Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Spain

ARTICLE INFO

Keywords:
Adaptive learning
Petri nets
Workflows
IMS Learning Design

ABSTRACT

In this paper, an IMS LD engine based on a Petri net model that represents the operational semantics of units of learning based on this specification is presented. The Petri nets of this engine, which is called OPENET4LD, verify the structural properties that are desirable for a learning flow and also facilitate the adaptation of the engine if potential changes in the IMS LD specification were proposed. Furthermore, OPENET4LD has an open and flexible architecture based on a set of ontologies that describe both the semantics of the Petri nets execution and the semantics of each learning flow component of IMS LD. Furthermore, the implementation of this architecture has been exhaustively validated with a number of UoLs that are compliant with the levels A and B of IMS LD.

1. Introduction

The need to manage reusable resources has led to the development of several metadata specifications in order to represent learning content, educational resources and learning design methodologies. The specifications for the learning design, known as Educational Modeling Languages (EMLs) (Rawlings, van Rosmalen, Koper, Rodríguez Artacho, & Lefrere, 2002), are models of aggregation and semantic information that describe both the content and the educational activities. These languages are used to describe from a pedagogical perspective the instructional design of a teaching activity, such as a subject, a course, a presentation, a seminar, and so on. In this situation teachers usually ask themselves things like the learning objectives, the content of the activities to be carried out by the students, or the pedagogical methodology to apply. EMLs usually provide an environment to represent the instructional design for a particular training activity and to show this design to the rest of people involved (teachers, students, etc.). Moreover, its elements are organized in units of learning (UoLs) in order to enable reuse and interoperability and to facilitate the description of the pedagogical aspects that are related to learning objects in educational processes (Koper & Manderveld, 2004).

Of all EMLs, IMS Learning Design specification (IMS LD) (IMS Global Learning Consortium, 2003; Koper & Tattersall, 2005) has emerged as the standard *de facto* for representing the design of a UoL, known as the *learning design*. Thus IMS LD facilitates the design, communication and reuse of processes of learning. It is primarily a specification focused on online learning (e-Learning) that allows the representation of a large variety of pedagogical models, and the adaption of their resources in a flexible way. Therefore, IMS LD is defined as an instructional model which aims to describe activities to a group of people so they may achieve a set of learning objectives in the context of a specific knowledge domain. The underlying principle of IMS LD is that the teaching plan of a classroom, that is, the description of actors, learning activities, resources and services, may be transformed into a UoL.

However, to support the IMS LD specification is needed to develop two types of tools: *IMS LD engines*, that coordinate the execution of the IMS LD activities carried out by the participants of the UoL; and *IMS LD players*, that enable students and teachers to access to the learning objectives, activities and resources through an advanced graphical interface. Thus, there are some developments that represent directly the learning flow execution in a *programming language* (Escobedo del Cid, de la Fuente Valentin, Gutierrez, Pardo, & Delgado Kloos, 2007; Gaeta, Gaeta, & Ritrovato, 2009; Hagen & Kinshuk, 2006; Molina, Jurado, de la Cruz, Redondo, & Ortega, 2009; Vogten, 2008). This feature makes these engines difficult to adapt to potential changes or extensions (Burgos, 2008) to the semantics of the IMS LD specification: a change, even small, would involves the re-implementation of the engine. Furthermore, as they do not define a *computational formal model* for describing the learning flow that coordinates the execution of the learning activities, it is not possible to check the consistency of the structural properties and behavior of that learning flow (for instance, whether it is deadlock-free and liveness). To deal with these drawbacks, some approaches that translate the IMS LD learning flow model into a workflow specification language have been proposed (Takayama, Ghiglione, Wilson, & Dalziel, 2007; Vantroys & Peter, 2003). However, these workflow languages do not directly support the evaluation of the structural properties, since they are

* Corresponding author. Tel.: +34 881813564; fax: +34 981528012.
E-mail addresses: juan.vidal@usc.es (J.C. Vidal), manuel.lama@usc.es (M. Lama), alberto.bugarin.diz@usc.es (A. Bugarín).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.05.013

not based on a formalism that models the execution semantics of the control branchs. Furthermore, none of the current IMS LD engines support changes to the learning flow at runtime (Burgos, Tattersall, & Koper, 2007), preventing teachers to introduce new learning activities that are not defined in the UoL design.

In this paper we present an ontology-based and Petri net-based engine, called OPENET LD, for the execution of UoLs designed following the IMS LD specification. In this engine the IMS LD learning flow is represented through a set of high-level Petri nets (HLPNs) (Jensen, 2003) that capture *formally* the semantics of execution for each IMS LD component, so that the coordination of the learning activities is carried out by a Petri net engine (Vidal, Lama, & Bugarín, 2010). With this strategy we achieve a dual purpose: on one hand, the semantics of the IMS LD is not implemented in a programming language, and therefore to extend the engine with new IMS LD components means to create a Petri net that models its behavior. On the other hand, the structure of each Petri net has been created to verify formally the behavioral properties that are desirable to guarantee the execution of the learning flow. Furthermore, OPENET4LD has a layered and flexible architecture that use ontologies to describe both the knowledge for executing HLPNs and the semantics of the IMS LD components. This architecture has been validated with 32 UoLs with different structural complexity and adaptive features (level B of IMS LD).

The rest of the paper is organized as follows: Section 2 describes the IMS LD specification. Section 3 analyzes the main features of the IMS LD engines of the current state of the art. Section 4 is the core of the paper, since the Petri net model for representing IMS LD learning flows is presented. Section 7 presents the validation of the model from the point of view of verifying the structural properties of a workflow. Section 6 describes the ontology-based service-oriented architecture (SOA) of OPENET4LD. Section 8 emphasizes the main advantages of the OPENET4LD when comparing with other IMS LD engines. Finally, Section 9 summarizes the main achievements of the paper.

2. IMS Learning Design

The IMS LD specification is a metadata standard that describes all the elements of the design of a teaching–learning process (IMS Global Learning Consortium, 2003). This specification is based on: (i) a well-founded *conceptual model* that defines the vocabulary and the functional relations between the concepts of the learning design; (ii) an *information model* that describes in natural language the semantics of every concept and relation introduced in the conceptual model; and (iii) a *behavioral model* that specifies the constraints imposed to the software system when a given learning design is executed in runtime. In other words, the behavioral model defines the semantics of the IMS LD specification during the execution phase. Furthermore, the IMS LD specification defines three levels of implementation depending on whether the learning design is adaptive or not:

- *Level A*: This first level defines the main components of a IMS LD-based UoL: participants (roles), pedagogical objectives, resources (services and contents), and learning design. This last component is understood as the description of coordination of the learning activities to be performed by the participants to achieve the pedagogical objectives. To describe this learning design, the IMS LD specification follows a theater metaphor where it exists a number of plays, that can be interpreted as the runscripts for the execution of the UoL and that are *concurrently* executed, being independent of each other. Each of these plays consist of a set of acts, which can be understood as a module or chapter in a course.These acts are performed in *sequence*, and in each of them the participants in the UoL carry out *in parallel* an activity or a structure of activities, which are executed in *sequence* or by *selecting* a number of them.
- *Level B*: This level adds *properties* and *conditions* to level A. It also adds monitoring services and global elements which allow users to create more complex structures. The properties store information about people (preferences, outcomes, roles, etc.), personal information, or even about the learning design itself. Level B also establishes (i) the visibility of the elements of the learning flow; (ii) if properties are transient or should persist across multiple sessions; and (iii) the set of operators and expressions that may transform the value of properties and the visibility of elements.
- *Level C*: The last level incorporates notifications to level B. Notifications fire automatically in response to events triggered in the learning process. For example, if a student submits a job, the learning flow might automatically send an email to report the event to the teacher.

Therefore, a learning design can be considered as a learning workflow (or *learning flow*) that consists in the execution of a set of activities carried out by the participants in the UoL. Through a set of plays and acts the IMS LD specification imposes a given structure to that define the learning flow. This structure is independent of the pedagogical methodology followed in the UoL, but is particularly suitable to collaborative learning. Taking this into account, it seems natural to model the semantics of the IMS LD learning flow through the workflow formalism.

3. Related work

The development of an IMS LD engine is a complex project that shares many similarities with the implementation of workflow engines, since the design of learning is understood as the coordination of the activities performed by users (called students and teachers) in an educational context (in order to achieve their educational objectives). In addition, the IMS LD specification itself is very complex when compared with other Educational Modeling Languages: most of these languages only take in consideration the activities to be undertaken by students, while IMS LD also introduces elements of control (plays, acts, role parts, learning activities and structures of activities) that are restricted to each other when they are executed (for example, acts of a play are executed in sequence). These restrictions determine the implementation of these elements, and they customize the educational journey.

Relatively few execution engines have been developed because of the IMS LD complexity, and some points in the design and implementation of an engine remains an open issue. In this sense, making the personalization of the learning unit to the students easier and, verifying the properties of the workflow to determine, for example, if there are dead points, or if all states are reachable, are two of these open issues. With this in mind, we highlight the following execution engines:

- *CopperCore* (Vogten, 2008) has been developed by the OUNL to test the viability of the IMS LD specification and is the reference implementation for IMS LD. It has therefore been used in several European projects such as TENCompetence or UNFOLD and has allowed the development of technologies based on IMS LD, as has been provided for researchers in the development of tools that have evaluated the specification. Coppercore allows the execution of IMS LD documents specified in any of its three levels of modeling, and internally translates the IMS LD learning flow to finite state machines that are interpreted by the Java