CoSMoS: Facilitating Learning Designers to Author Units of Learning Using IMS LD

Yongwu Miao
Institute of Computer Science and Interactive Systems,
University of Duisburg-Essen, Germany
miao@collide.info

Abstract. IMS LD is a meta-language that can be used to specify a pedagogical method as a learning design. A learning design is a formal process model, encoded in XML, which can be played by any IMS LD-aware-player to learners online. In order to make it easy to edit learning designs, LD authoring tools are absolutely needed. Through a review of currently available LD authoring tools, we see that the development of LD authoring tools is still at an immature stage. This paper describes a LD authoring tool named CoSMoS, which can support editing learning designs at Levels A, B, and C. In this paper the main design ideas are presented and three design issues are discussed.

Keywords: educational process modelling, IMS LD, learning design, authoring tool

Introduction

In the development of learning technology standards, the release of IMS LD (stands for IMS Learning Design specification [7]) signals an exciting paradigm shift from defining, sharing, and reusing learning objects to designing, sharing, and reusing pedagogical strategies. IMS LD is based on the Educational Modelling language (EML) developed by the Dutch Open University OUNL [9]. IMS LD provides a framework for teachers to express the pedagogical meaning and reflect in a deeper and more creative way about how they design and structure activities. IMS LD can be used to fully describe a teaching-learning process in a unit of learning (UOL), including references to the digital and non-digital learning objects and services needed during the process. It enables to describe personalization aspects within a learning design so that not only content but also activities within a UOL can be adapted based on the personal characteristics and situational circumstances. The specification makes it possible that the learning designs been proved as effective are communicated and shared between teachers or archived for reuse on future occasions [10, 11].

IMS LD is a meta-language that is divided into three parts, known as Level A, Level B, and Level C [7]. Separate XML schemas are provided for each level, with Levels B and C each integrating and extending the previous level. By using IMS LD a teaching-learning process can be formalized as a computational model in XML format, which is a platform-independent web-standard notation for describing arbitrary structured data. This means that a learning design, encoded in XML, can be read and run by any IMS-LD-aware player. The problem is that authoring a learning design in XML is a time-consuming and error-prone task. There is an absolute need to research appropriate user interface models to allow easy creation of learning designs.

Since 2003 many efforts have been made to empower learning designers and others to describe and share their teaching-learning designs. Recently several LD authoring tools have been developed. Compared to common XML editors, these LD authoring tools provide
user-friendly interfaces for learning designers to create, reuse, and customise UOLs without the need of editing any XML code. However, most existing authoring tools can only support editing UOLs at Level A. Because the elements added at Level B are something like variables, operators, and control flow statements, how to facilitate learning designers to handle such technical complexity is a challenge for the developers of authoring tools. This paper describes our work that can be regarded as the first attempt to take the challenge to facilitate editing learning designs at Level B.

1. Three Levels in IMS LD

As mentioned above, IMS LD is divided into three levels:

   LD Level A contains the bulk of the IMS LD constructs, including roles, activities, environments, learning objects, services, plays, acts, etc. The main added value of Level A learning design to the domain of e-learning is that it defines activities and roles as reusable components that can be designed into a learning flow using the method element. IMS LD uses a metaphor of a theatrical play to model learning-teaching processes. A method can have one or more plays. Each play consists of a sequence of acts and each act contains a set of role-parts. Each role-part assigns an activity to a role. The specification also allows learning objects (e.g., web-content) and services (e.g., index-searching and conferencing) to be specified at design-time as placeholders within the learning design that will be instantiated by the run-time system.

   LD Level B adds properties and conditions to level A. Properties enable information about learners, roles and the state of the learning design itself to be maintained. The addition of properties is important for recording outcomes of participants and supporting personalization. Conditions enable learning designers to define rules that govern the behavior of the unit of learning as a whole what gets presented to individual participants.

   LD Level C adds notification to level B. Notifications provide a greater level of interactivity and control over a live learning design, as a form of event-driven messaging system within a LD player. A notification can be triggered by an activity completing or by a rule through the global elements and can also trigger a notification to be sent to a design element or human participants. A notification can be used to make a new activity available for a role to perform or to change the value of a property.

2. A Brief Review of Authoring Tools Related to IMS LD

This section provides a brief review of currently available LD authoring tools according to the forms of representing learning designs in the tools. Existing XML editors and authoring tools provided by Learning Management Systems (LMSs) are not taken into account, because they are not LD-specific authoring tools. However, two non-LD authoring tools will be investigated, because these tools are e-learning-specific tools and built around a model of ‘activity’ and ‘learning flow’, rather than around ‘learning object’ and ‘content’.

2.1 Tree-based Authoring Tools

A tree-based authoring tool presents the elements of LD as a tree structure. An interface is provided for users to navigate through the tree and to enter values for the elements. LD authoring tools adopting a tree-based approach are ALFANET, CopperAuthor, and RELOAD.
ALFANET (stands for Active Learning for Adaptive Internet), an EU project aims to develop new methods and services for active and adaptive learning [1]. In ALFANET, IMS LD is used to represent learning scenarios for supporting personalization. The menu buttons in the tool are organized as a hierarchical structure according to IMS LD element structure. It provides basic support for authors to fill the forms for defining elements at Level A.

CopperAuthor is a tree-table-based authoring tool. It allows learning designers to construct and navigate the structure of the edited learning design. The generic information of each element can be entered in the cells of the table. This tool provides a primitive and direct manner to edit a learning design at Level A.

RELOAD (stands for Reusable E-Learning Object Authoring & Delivery [18]) is a project, funded by JISC, UK. RELOAD LD editor consists of several edit pages and each page supports editing a type of element such as role, activity, environment, method, etc. An element tree on each page enables to navigate through all elements of a certain type. If selecting an element, the tool presents a user-friendly interface for authoring the element in a series of panels. The RELOAD LD editor is regarded as a reference implementation of LD authoring tool. In June the newest version of RELOAD LD editor was released, which can support learning design at Levels B and C as well. However, it supports editing conditions in a way that the representation structure of expressions and actions exactly reflects the element structure of IMS LD specification, rather than the conventions of viewing and writing expressions. In addition, at any point of time users can only view and edit sole operand. It is not convenient for users to perceive and define conditions.

2.2 Diagram-based Authoring Tools

Diagram-based authoring tools represent IMS LD elements and their relations as graphical objects such as nodes and arrows. A tool in this category provides a graphical language for users to edit a learning design as diagrams.

MOT+ system [17], developed by the CIRTA Research Centre, Tele-university, Montreal, is an interesting example of diagram-based authoring tools, which enables users to navigate in and construct hypertext documents that make up a learning design. A learning design represented as hypertext documents can be exported as IMS LD XML files. Now the developers of MOT+ are working on LD Level B and C editing functions.

Another existing diagram-based authoring tool is ASK-LDT (Advanced e-Services for the Knowledge Society Research Unit [2, 8]), developed by the department of technology education and digital systems, University of Piraeus, Greece. The ASK-LDT editor provides a graphical language. The components of the graphical language include several types of nodes representing several predefined activity types (e.g., lesson, discussion, assessment and so on) and two types of arrows representing preceding and rollup relations between activity nodes. A learning design can be modeled as a diagram representing a learning flow by using the graphical notations. The resulting diagram can be translated into a XML representation automatically by using IMS LD Level B in a way to create a property with a “boolean” type for each activity and creating a simple condition clause for each arrow. Therefore, users can define a learning design with features of LD Level B without need to define properties and conditions. However, this tool offers a relatively easy path into LD authoring and can not really support learning designers to edit UOLs at Levels B and C.

2.3 Two IMS LD Relevant Tools

Two non-LD authoring tools reviewed in this section are LAMS and COMPtILE. Both are diagram-based authoring tools that are intended for normal teachers. Although these tools
do not implement the IMS-LD specification, they embody the core ideas behind the specification in terms of a focus on creating sequences of activities, rather than learning objects. An additional common feature of these two tools is to be integrated with run-time environments.

LAMS (stands for Learning Activity Management System [12]) was developed by MacQuarie University in Sydney and WebMCQ Ltd. It provides a simple and highly intuitive user interface that allows the course designer to drag and drop typed activity nodes into the edit area and use connecting arrows to organize the activities into a sequential workflow [4]. LAMS provides a variety of software tools that are intended to support the creation and management of specific learning activities such as synchronous discussion (e.g., Chat), concept-mapping (e.g., Mind-Map) or role-play scenarios (e.g., Kartouche). The activity types are fixed, but these cover many basic activities carried out in the classroom. This use of familiar elements makes the application easy for teachers to comprehend, as this is the way that conventional lessons are planned.

COMPILE (stands for Cooperative Open Multimedia-based Process-centered Integrated Learning Environment) was developed at Muenster University, Germany. It was developed by using a content-based and process-centered approach [13]. The aim of the system is supporting learning projects (formal collaborative learning processes [14]) and learning programs (informal collaborative learning processes [15]). The system provides two graphical languages for teachers to edit teaching-learning processes by using nodes such as courses, lessons, instruction activities, test activities, projects, project activities, and artifacts. Notations such as temporal-arrow, artifact-arrow, start, end, branch, fork, and join are used to connect elements as a learning flow and an artifact flow. The authoring tool enables to define properties and to define a transition condition for each branch. Because this tool is intended for teachers, the properties and conditions defined in COMPILE are much simpler than those in IMS LD.

3. Design and Implementation of CoSMoS

This section briefly presents the design and implementation of a LD authoring tool and gives a quick overview of its particular features and functionality.

CoSMoS (stands for Collaboration Script Modelling System) was originally designed and implemented for supporting formalization of collaborative learning processes [16]. The modified version is compatible with the IMS LD specification. This tool is developed for learning designers, who may be pedagogic experts, course planners, and instructional designers with knowledge about IMS LD. Therefore, the tool is designed in a way adhering rigidly to IMS LD.

As shown in Fig. 1, the user interface of the tool consists of three components. At the top of the window, the menu bar and the toolbar of the tool list the main functions. The left pane shows learning designs in a hierarchy structure. It is allowed to edit multiple learning designs at the same time. Such a design can help to reuse definitions of elements across learning designs and makes it easy to specify the relations between multiple learning designs. The structure of a learning design is hierarchically ordered as the same as the structure of major elements of IMS LD. Such a structure can help learning designers to navigate in the element tree easily. Users can create new elements by adding leaves of the tree. The right pane is an edit area showing forms for editing elements. If a user selects an editable element node such as a role, a learning activity, an environment, an act, and so on, a corresponding edit form will be shown in the edit area. Users can enter values in editable input fields. If a value is a reference to an existing element, users can drag the tree node representing the element and drop it into the input field. Such a reference can be removed by
dragging it and dropping into the garbage icon in the tool bar. Using drag&drop to do constructive and destructive work intuitively is an important feature of the tool.

Fig. 1: The User Interface of CoSMoS

Fig. 2 shows the form for editing a condition clause. The schema used to define Level B elements can be seen in the left pane. The form consists of three edit panels for editing conditional expressions, then actions, and else actions. Our approach is to provide a user-friendly representation of expression and to allow passing parameters by using drag&drop. When defining an expression, a learning designer can drag an operator from the schema tree (e.g., <, not, and etc.) and drop it in a blank field in the ‘If’ panel. The blank will be replaced by the dropped expression. For instance, if selecting an ‘and’ symbol, a dialog window will pop up to ask how many operands will be defined in this ‘and’ expression. If the answer is ‘3’, three blanks connected by two ‘and’ operators will appear. A blank field can be further specified by dragging an operator and dropping in the blank. For example, if a ‘complete’ symbol is dropped in the first blank. The blank will be replaced by ‘[an undefined process] is completed’. Dragging a process element (e.g., a learning activity titled “Enter your initial thoughts”) and dropping in this blank, the title of the learning activity will be shown in the blank (see Fig. 2). It is possible to define an expression by using a property. Fig. 2 also shows such an example property titled “What I think greatness is”. By using drag&drop, learning designers can define actions as well. The Fig. 2 shows examples of ‘then’ and ‘else’ actions: showing and hiding classes and sending a notification, an element at Level C. An ‘else’ has three options: no action, else actions, and else if condition. Note that the tool can support recursively defining ‘else if’ conditions.

In order to aid learning designers to create UOLs effectively, the tool can validate the definition of any LD element at design-time and show warning messages and error messages of the definition. The value of the status of an element will be assigned by the tool
automatically according to the validation result. The tool can show the XML representation of any element in the edit area on demand. Experts can modify the XML representation directly in the edit area, which can be imported to the tool. The tool supports a deeper copy (rather than a reference copy) and paste in the same learning design and even in other learning designs. These functions help to reuse the definitions of elements easily.

The tool is implemented by using pure JAVA. A defined UOL can be exported as an imsmanifest.xml file, which can be read and played by an IMS LD-aware player such as the CopperCore engine [3].

![Fig. 2: Defining a Condition Clause](image)

4. Discussions

CoSMoS has been demonstrated and used in an UNFOLD (stands for Understanding New Frameworks of Learning Design [18]) workshop in April by participants including learning designers, experts, and the contributors of IMS LD. UNFOLD is an EU project aiming at promoting and encouraging the development and use of e-learning systems related to IMS LD. In the workshop, most comments were positive and some were valuable suggestions. Based on the comments and experiences, we can discuss some design issues, which should be taken into account when designing a LD authoring tool.

Tree- and diagram-based tools: A tree-based authoring tool provides an overview of all major elements of a learning design. Such a tool makes it convenient to navigate in the hierarchical structure and operate on any element. The main disadvantage of the tree-based approach is that the relations between the elements can not be explicitly represented in a tree except for component relations. However, the diagram-based approach has strengths to represent relations between elements, and to construct and modify the relations within a diagram. The most attractive feature of diagram-based tools is that the visualized notations can help users to understand and represent learning designs intuitively. If a learning flow is drawn as an individual diagram by the tool (e.g., LAMS and ASK-LDT), the diagram
provides an overview as well. However, if the number of graphical objects increases, the limitation of the space and the increased complexity of the diagram will be a serious problem. A layer-diagram approach will be a natural choice. For examples, MOT+ has two layers and COMPILE can have arbitrary layers. Users have to navigate in a set of hypertext documents. It may lead to a classical problem ‘lost of way’ in hypertext documents. An additional difficulty is representing a relation between two nodes located in different diagrams. Finally, the diagram-based tool has a problem to load and present a learning design created by using other tools, because the graphical information is missing.

Higher- and lower-level tools: According to [11], the judgment of higher- or lower-level tools is based on how much knowledge about the LD the users of the tools should possess. LD experts would like to use a tool that enables them to access the parts of a LD document without a need to enter repetitive text and to have their document checked for integrity. Learning designers need tools that enable them to represent pedagogic methods by defining roles, resources, and the flow of activities together with the various branching conditions. Normal teachers, who may have no knowledge about LD, need to be able to represent their pedagogic experience and instructional plans by using their conventional vocabularies such as course, lesson, presentation, discussion, voting, test, exercise, homework, and so on. It is a challenge to develop authoring tools for normal teachers. These tools should be based on a higher-level learning process modelling language and can translate the learning designs represented by the higher-level language into the IMS LD. For example, if a teacher creates a ‘voting’ activity in a learning design, it is not necessary to create a detail specification about roles, learning activities, environments, voting tools, properties, conditions, and so on. The teacher just need to select or assign a value or use a default value for each attribute such as the type of voting, voting issue, voting items, voting policy (e.g. majority agree or common agree) and so on. The authoring tool can automatically interpret such a higher-level description into a formal specification of a ‘voting’ activity in IMS LD. It is expected that a spectrum of authoring tools can be developed from lower-level to higher-level for different kinds of users. CoSMoS is a lower- and middle-level authoring tool. ASK-LDT, COMPILE, and LAMS can be regarded as, to some extent, higher-level tools. Recently, some attempts are tried to develop higher-level languages and tools [6, 7]. It is important to note that there is a widespread misunderstanding that diagram-based tools are somehow higher-level tools. For example, MOT+ has been categorized as a higher-level tool in [11]. Whether a tool is higher-level depends on whether the tool provides a set of higher-level notations. MOT+ is not a higher-level tool because the vocabularies used by the MOT+ LD editor are exactly those of IMS LD.

Stand-alone and integrated tools: Most existing LD authoring tools are stand-alone and single-user tools. Perhaps it is because the development of the tools is still in the initial phase. Authoring a UOL is usually not an independent and individual work. An integrated and shared authoring environment is expected. Users with different roles in an organization or in a virtual community can cooperate to construct and share learning objects and to create and share learning designs. Furthermore, it should be encouraged to go one step more – build an integrated authoring, publication and delivery environment.

5. Conclusions and Future Work

This paper briefly reviewed authoring tools related to IMS LD. The conclusion drawn from the review is that the development of LD authoring tools is still at an immature stage although several LD authoring tools have been developed. In particular, most LD authoring tools can only support editing learning designs at Level A. This paper described the
development of a LD authoring tool named CoSMoS, which can support editing learning designs at Level B and C. Recently RELOAD LD editor can support editing learning designs at Level B and C as well. Comparison with the RELOAD LD editor, CoSMoS supports editing learning designs at Level B in a way that is more close to the convention of viewing and writing condition clause. Our approach to allow easy creation of learning designs can be characterized by providing a user-friendly representation of condition clauses and by using drag&drop to do constructive and destructive editing work intuitively.

The paper discussed three design issues. This discussion will be helpful for developers of IMS LD authoring tools to make decisions when designing their own authoring tools. The discussion has revealed that there are three more challenges related to LD authoring tools in the future: to develop a higher-level modelling language for describing teaching-learning processes, to develop a diagram-based authoring tool based on the higher-level modelling language, and to develop an integrated environment for authoring, publication, and delivery of learning designs.

References