Journal of Universal Computer Science, vol. 14, no. 17 (2008), 2819-2839
submitted: 30/6/08, accepted: 27/9/08, appeared: 28/9/08 © J.UCS

Authoring Courses with Rich Adaptive Sequencing for IMS Learning Design

Sergio Gutierrez-Santos
(London Knowledge Lab, Birkbeck College, UK
sergut@lkl.ac.uk)

Abelardo Pardo
(Carlos III University of Madrid, Spain
abel@it.uc3m.es)

Carlos Delgado Kloos
(Carlos III University of Madrid, Spain
cdk@it.uc3m.es)

Abstract: This paper describes the process of translating an adaptive sequencing strategy designed using Sequencing Graphs to the semantics of IMS Learning Design. The relevance of this contribution is twofold. First, it combines the expressive power and flexibility of Sequencing Graphs, and the interoperability capabilities of IMS. Second, it shows some important limitations of IMS specifications (focusing on Learning Design) for the sequencing of learning activities.
Key Words: Sequencing Graph, IMS Learning Design, translation, sequencing.
Category: L.2.0, L.2.1, L.3.5

1 Introduction

One of the main advantages of Web Based Education is the large number of different resources that are provided to the user. There is the risk, however, of becoming *lost in cyberspace* [Brusilovsky2001]. Adaptive Educational Hypermedia is one possible technique to solve this problem by customising the learning material to the specific needs of the user. Instead of creating a course with the same resources for every user, the resources can be authored so that an *adapted environment* is created for each learner [Cristea2004], with e.g. its own sequencing of learning activities.

There is, however, another challenge to face. There are many educational adaptive hypermedia systems, and it would be desirable to share resources between them, both at the content level and other levels (e.g. sequences, portfolios, learners' history, etc). With the new scenarios offered by technology, learning can take place anywhere, anytime, and therefore learning content management systems (LCMS) and learning material require higher levels of adaptation, accessibility, interoperability and reuse.

Standards are a way to foster and promote these requirements at international levels. Several international institutions have procedures to deliver technical solutions to current technological problems. Institutions such as IEEE and ISO have numerous working groups elaborating standards (e.g. Learning Object Metadata, LOM [IEEE CS/LTSC2002]), to guarantee that learning management systems offer a common set of functionality and data representation framework so that learning material can be considered platform independent. Other institutions produce specifications that are becoming *de facto* standards, like ADL —that developed SCORM [ADL2004]— or IMS. The latter has created the Learning Design specification (henceforth IMS-LD). The goal of IMS-LD [IMS2003a] is to describe teaching strategies and/or pedagogical approaches, as well as promoting the interchange of them between learning management systems.

IMS-LD is a complex specification. Create a course using the specification, including a rich and flexible sequencing strategy, requires a lot of knowledge of the specification details. This paper tries to lower that barrier by bridging the gap between a technique for creating rich sequencing strategies for personal tutoring, Sequencing Graphs, and the IMS-LD specification.

Therefore, the authoring process could be depicted as follows. The author of an adaptive hypermedia course for personal tutoring, with exigent sequencing needs, authors the course using Sequencing Graphs to describe the sequencing possibilities. Once the resources have been created and the sequencing has been defined in terms of Sequencing Graphs, the course is converted into a Unit of Learning of the IMS Learning Design specification (henceforth UoL). From then on, the course can be run in any LCMS that implements an IMS-LD player, facilitating its distribution and reuse by different stakeholders.

This paper focuses on the second part of this process (but it provides a brief description of Sequencing Graphs, in order to make it easy to understand the whole process). Transforming a course based on Sequencing Graphs to IMS-LD has shown itself to be non-trivial, posing several difficulties that are solved in this paper. Several conclusions are drawn from the procedure about the appropriateness of IMS-LD to express some pedagogical strategies.

The main contribution of the paper is then twofold. First, it describes in detail the process of translating a course authored using Sequencing Graphs to a Unit of Learning defined with IMS-LD. The process has been validated by running the resulting UoL in CopperCore [OUNL], the reference implementation IMS-LD player. Last, but not least, it uncovers several limitations of the IMS-LD specification when it is used to describe complex sequencings (i.e. sets of possible sequences) of learning material. These limitations include the fact that the condition model of IMS-LD presents scalability issues (with implication on maintainability of UoLs), the lack of support for reflexive cycles, and

practical considerations regarding the excessive size of the resulting files when a rich sequencing strategy is authored. Since IMS-LD is the most important IMS specification regarding adaptive sequencing, these considerations are not only relevant to IMS-LD limitations, but comprise a reflection relevant to the whole set of IMS specifications.

The rest of the paper is structured as follows. Section 2 provides a little background on sequencing adaptation and learning standards. Section 3 describes Sequencing Graphs briefly, while Section 4 presents the IMS Learning Design specification, focusing on those aspects more relevant to sequencing. Section 5, the longest of the paper, explains the process for translating the graphs into IMS-LD semantics. Then, Section 6 discusses the appropriateness of IMS-LD for sequencing adaptation in light of this work, and Section 7 shows some other work related to adaptive sequencing or IMS-LD authoring. Section 8 closes the paper with the main conclusions and depicts lines for future research.

2 Background

Results on this paper are specially relevant for the authoring of courses aimed for personal tutoring. Personal tutoring systems have many applications: they can be used by learners as support for normal lessons at school or university [Garrison and Kanuka2004]; they can be used by individuals for self-training, as part of a life-long learning effort [Magoulas et al.2006]; and they can play a significant role in distance learning scenarios [Sherry1996]. A personal tutoring system can mimic the role of a human tutor, adapting the learning process to the needs of the learner and making the learning more successful.

It has been shown that every learning experience benefits from personal adaptation. Benjamin Bloom was the first to measure this, thus exposing the "two-sigma problem": learners with personal tutoring perform two standard deviations better than those without it [Bloom1984]. However, the usual trade-off when it comes to designing an adaptive educational system is between generality and effectiveness. The more generic a system, the more difficult it is to author it (i.e. design it) to be adaptive, both in terms of the content and the sequence of activities. Sequencing authoring is tightly coupled with the notion of content authoring. Selecting different material is indeed a crucial aspect to adapt the learning experience to each learner and make it more effective [Bra et al.1999].

Two adaptation categories are thus to be considered: content adaptation and sequencing adaptation. The former refers to how an activity can be shown in a variety of forms, and the latter, to the possibility of sequencing learning activities in different ways. It must be noted that obtaining a highly personalised content sequencing involves an important challenge. Courses tend to have a high number of learning objects, and designing a personalised sequencing strate-

gies for each learner quickly becomes unmanageable. A solution for this problem has been proposed, based on *Sequencing Graphs* (described in Section 3, see [Gutierrez et al.2004] for a detailed description and an empirical evaluation of their positive effect on learning). This idea is used in this paper as the basis for the definition of the educational sequencing, i.e. the possible sets of sequences of activities that the learners may follow depending on their actions and background. The focus, however, is in describing how these sequencings (defined with Sequencing Graphs) can be expressed with the semantics of IMS-LD.

The fields of adaptive educational hypermedia and the multiple specifications related to learning (IMS Learning Design in particular) have had a reduced intersection until now. This gap was briefly discussed in [Cristea and Burgos2006] at a high level. Some authors [Berlanga et al.2006] have tried to use the elements of IMS-LD to map the usual components of adaptive educational hypermedia systems: adaptive rules, learning elements, etc. The work presented in this document continues this trend of finding common solid ground between both disciplines.

3 Sequencing Graphs

One of the issues that needs to be addressed when authoring the sequencing of a course is how to allow the learning content author to organise and manipulate a significantly large number of possible resources expressing complex sequences (e.g. involving reflection cycles [Zembal-Saul et al.2000]). Sequencing Graphs [Gutierrez et al.2004] tackle this problem and at the same time aim at facilitating the definition of sequences of learning material by mimicking its inherently hierarchical structure.

A Sequencing Graph (henceforth SG) is defined as an oriented multi-graph. Some nodes are bound to actual basic learning resources (e.g. exercises, documents, videos, audio, etc.) and some are logical entities that hierarchically contain another sequencing sub-graph (see Figure 1). Inter-operation on different levels of the hierarchy is achieved by means of *input nodes*, *output edges* and *interfaces*. All edges in a SG connect two nodes in the same level of hierarchy, with the only exception of output edges (see below).

Input nodes are the entry points from a higher level of hierarchy; if there are several of these entry points, one is selected non-deterministically. Input nodes are related to the *initialisation* definition of every graph (i.e. level of hierarchy). When the learner "enters a graph" (that is, the first of its resources is used), this definition sets the variables and their initial values that will determine the sequencing at this level.

Output edges connect a source node and its parent (destination). An input node could be the source of an output edge, if that fits the pedagogical objectives of the sequences defined in a graph. SG have a simple and intuitive hierarchy: the parent node of each node is unique.

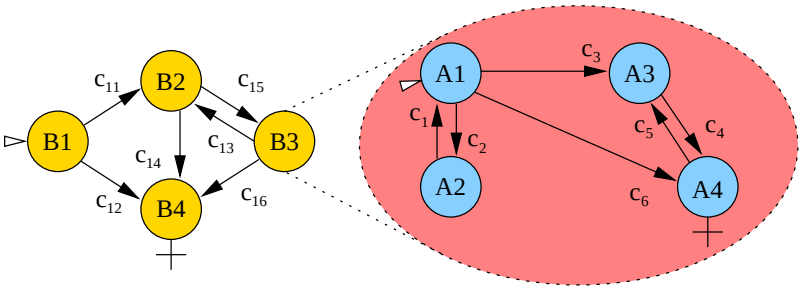

Figure 1: Hierarchical Sequencing Graph example.

Finally, interfaces determine the exchange of variables between hierarchy levels. This mechanism provides a clear separation allowing the designer to concentrate on the authoring of one level separately. The author must specify which variables are exported to the upper hierarchy level when abandoning the current one. Conversely, at the upper level (i.e. container node) it must be defined which variables are exported from the lower level and how should they be named. This variable names are usually changed to maintain its scope and avoid collisions (e.g. "total_time" could be a variable present in two different levels). The restrictions of designing an interface with the variables to export to upper levels, and the edges that interconnect only resources at the same level can be seen as encapsulation mechanisms conceived to encourage the reuse of sub-graphs in different contexts. This type of hierarchical transition structures are not a new concept since they have been already used in hypermedia systems [Turine et al.1997] as well as other areas such as system design [Harel1987].

Every edge in the SG has a *condition* and a set of *actions*. The condition is an arbitrary Boolean expression over a previously defined set of variables. The actions represent a set of operations to be performed over the values of these variables. The semantics of this pair of elements is that the transition represented by the edge is suitable to be performed if the condition is true. If the edge is finally chosen to be followed, the actions are then executed.

The idea behind these graphs is to provide a sequential structure that captures a large set of possible node sequences over a large number of learning objects. These sequences are decided based on the answers obtained from the learners in previous nodes and their current position in the graph. The entity that stores these values (previous answers and variables) is called the "environment".

An example of a hierarchical graph is given in Figure 1. The graph has two hierarchical levels. At the top level, node $B3$ contains the sub-graph with nodes

labelled $A1, \ldots, A4$. With the exception of $B3$, all nodes are attached to a learning object. Input nodes at each level ($B1$ and $A1$) are marked with an empty incoming arrow, and output nodes ($B4$, $A4$) are marked with a cross at the bottom. The transition steps through this graph are computed as follows. For example, if the current state is $B2$ and condition c_{15} is true, then the transition function would return $A1$ as the following object to be visited. If more than one possible transition is available (system is in state $A1$ and conditions c_3 and c_6 are true) two traversal variants have been implemented. In the first case the system chooses on of the edges pseudo-randomly. In the second, the title of the two options is shown and the choice is given to the user through an auxiliary step. If the system is on state $A1$ and conditions c_{15} and c_{16} (in the upper level of the hierarchy) are true, the transition ignore these conditions and proceeds through the lower level evaluating c_2, c_3 and c_6 until the output node $A4$ is reached. At that point, the conditions of the upper level are taken into account. A detailed description of the graph traversal algorithm can be found in [Gutierrez Santos2007], page 80, and is omitted here for the sake of space.

These graphs provide a powerful and intuitive authoring framework that permits authors to organise a large number of learning resources by using multiple hierarchy levels. Also, a graph captures a large number of sequences of these resources. This power derives from the hierarchical specification of the sequencing conditions. When designing a graph, an author may focus on the low level decisions on how to sequence resources depending on the results obtained so far by a learner. Once these sequences have been specified for several topics, a second sequencing strategy can be created considering these graphs as atomic units.

4 IMS Learning Design

IMS-LD implements a vocabulary with *plays*, *acts*, *roles*, *activities* and *conditions* that is arguably able to express any pedagogical strategy [Koper2005]. Acts are included into plays, and are means to synchronise different activities that occur simultaneously. Activities are performed by persons playing a role. Roles are to be instantiated by real persons, and are grouped into two built-in role families: student roles and teacher roles. Finally, conditions (*if* something happens, *then* this happens, *else* this other thing happens) allow for variability in the course of the play. Additionally, Learning Design allows the definition and use of *properties* to store information about the learning process. The overall entity including the definition of a pedagogical strategy or learning design, along with the resources or services needed for its enactment is called an *Unit of Learning* (UoL).

Sequencing adaptation in IMS-LD can be done through the use of conditions and properties. Properties have a name and an identifier, a type, a value and some restrictions on the possible values they can hold. Conditions are evaluated against property values. Conditions are evaluated every time the value of

a property changes. When a condition is satisfied, some previously defined action is executed. With this mechanisms, properties can be used to store specific information about the learner and used to adapt the current UoL to particular needs of a learner or a group by means of conditions and actions.

There is one additional set of elements of IMS-LD that are important for sequencing. *Global elements* are defined to be included in XHTML resources. They allow properties to be accessed and modified by the users through forms in XHTML documents. This is the only mechanism for users to modify the properties in a UoL. As a consequence, all other sequencing adaptation decisions must be included in the UoL at design time.

5 From Sequencing Graphs to IMS Learning Design

This section shows how the Sequencing Graphs (henceforth SG) semantics can be captured in IMS Learning Design. The result is a UoL that when enacted produces the same personal tutoring experience captured in the SG.

5.1 The activities

When a learner traverses a SG, activities are shown one at a time. This activity sequence does not have to be linear. The sequence may contain cycles in which a particular exercise is visited multiple times. The sequence may also skip certain exercises depending on the current environment.

IMS-LD does not consider the possibility of repeating an activity several times. Since SGs do contain cycles, this difference is one of the aspects that requires special treatment. In IMS-LD, when a learner has performed and completed an activity, it is assumed that the corresponding objectives have been achieved, so there is no point in repeating it. This philosophy (which has several implications, as discussed in Section 8) is included in the specification:

> *... a control must be available in the user-interface to set the activity status to "completed". A user can do this once (no undo). Once he/she indicated the activity to be completed, then this activity stays completed in the run* ([IMS2003b], page 28).

This fact complicates the translation process from SG semantics into IMS-LD semantics. A set of conditions needs to be created to show the user the <imsld:learning-activity> needed at every moment. A direct translation from exercises in an SG containing cycles to activities in IMS-LD is not possible because of this "complete once, no undo" policy. In the absence of this limitation, the translation from SG to IMS-LD could be performed using the <imsld:on-completion> element of <imsld:learning-activity> to emulate the behaviour defined by the edges.

XML 5.1 Attribute class in a XHTML document

```
<div class="class1">
  <p>Show this paragraph (class 1)</p>
</div>
<div class="class2">
  <p>Show this paragraph (class 2)</p>
</div>
```

XML 5.2 SG condition exported into an IMS-LD condition

```
<imsld:if>
  <imsld:is>
     [Here the desired condition]
  </imsld:is>
</imsld:if>
<imsld:then>
  <imsld:hide>
    <imsld:class class="class2" />
  </imsld:hide>
  <imsld:show>
     <imsld:class class="class1" />
  </imsld:show>
</imsld:then>
```

The solution to this problem requires coupling the *imsmanifest.xml* with the learning resources. In particular, the <imsld:class> property of the <imsld:show> element must be used. These <imsld:class> elements are directly related with the attribute of the same name in a DIV element in a XHTML document. A DIV element is used as a container for other elements. The *class* attribute is used to associate a style defined in the CSS (Cascading Style Sheets) page.

IMS-LD uses the *class* attributes in the DIV elements of XHTML resources for something different. By using show and hide actions, DIV content can be revealed or concealed. This is illustrated in XML excerpts 5.1 and 5.2, the former showing a fragment of the structure of a sample resource document and the latter depicting the corresponding condition. If the condition is true, the IMS-LD player hides the content inside the DIV element with a *class* value of "class2", and show the content inside the DIV element with a *class* value of "class1". Thus, only the message "Show this paragraph (class 1)" is shown on the screen.

These elements <imsld:show> and <imsld:hide> are used for implementing the behaviour defined by SG within IMS-LD by showing some activities and hiding others depending on some conditions. This is the equivalent behaviour to the learner traversing the SG. Resources are shown or hidden according to their relation to a node. A single resource master file in XHTML is generated containing all the resource in a SG. All these resources are surrounded by a DIV element with the *class* attribute equal to the name of the original node. Traversing an edge in a SG is translated into one <imsld:show> action for the selected resource and multiple <imsld:hide> actions for the rest of the resources.

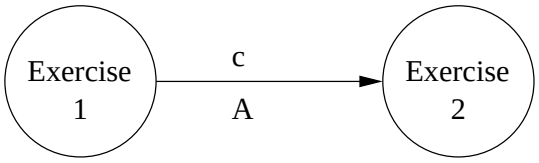

Figure 2: Sample graph of only two exercises

XML 5.3 Sample of resource master file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is common -->
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:imsld="http://www.imsglobal.org/xsd/imsld_v1p0">
 <head>
   <title>Activities</title>
 </head>
 <body>

   <!-- This is specific to each SG and each UoL -->
   <div class="Exercise 1">
          [XHTML code of the resource associated to node "Exercise1"]
   </div>
   <div class="Exercise 2">
          [XHTML code of the resource associated to node "Exercise2"]
   </div>
   ...
   ...
   <!-- This is common again, and to the end of the file -->
   <div class="classExitDiv">
     <p>You have succesfully completed this module.</p>
     <p>Congratulations!</p>
   </div>
  </body>
</html>
```

A simple example of a graph composed of only two nodes is illustrated in Figure 2 and XML excerpt 5.3. Each node is linked to a XHTML file to be shown to the user.

The file presents an invariable part marked in XML excerpt 5.3 using XML comments. These fragments are always present in the *imsmanifest.xml* created during the translation process of a SG. The rest of the file depends on the number of nodes <sg:exercise> contained in the SG, as well as the content of the XHTML files linked to them.

An important side effect of the translation process is that the hierarchical structure of the SG disappears. All nodes are exported to activities on the same level. Conditions in the edges are written in the same condition section of the *imsmanifest.xml*. This consequence has several implications: a mechanism to avoid collisions between property names is needed, and the size of the manifest

file increases polynomially with respect to the number of nodes.

5.1.1 The manifest

The tool developed to translate a SG into a UoL uses a template for creating the *imsmanifest.xml*. This template contains the information about the learning design that is common to any SG and is divided as follows:

- There is only one role: Student. SG are designed for personal, unassisted tutoring.
- There is only one <imsld:learning-activity> always visible with name "Sequencing graph". This activity is associated to the file *resourceFile.xml*.
- The method element contains one single <imsld:play> itself containing one <imsld:act>. The <imsld:act> contains only one <imsld:role-part>, and it is composed of the "Student" role and the corresponding <imsld:learning-activity>.

The rest of the manifest is derived from the SG being translated. The main blocks of this structure are:

Title of the learning design. This element is generated with the same name as the root of the original SG.

Properties. They are the equivalent in IMS-LD to the variables in SG. The number of properties is proportional to the number of variables in the graph. Although there is no formal relation between the number of variables and the size of the SG description file, the graphs studied in this work have shown a linear dependency.

Conditions. They are the equivalent of the conditions in the graph. There must be at least the same number of conditions in the manifest as in the original graph. The number of conditions is linearly related to the number of properties, but is proportional to the square of the number of nodes. Also, the length of the set of actions associated to IMS-LD conditions is proportional to the number of nodes. This means that the *total* number of actions follows the cube of the nodes. Most of the resulting manifest is occupied by conditions and their associated actions.

5.1.2 The "continue" box

The only mechanism provided by IMS-LD to indicate that an activity has finished is to complete it. As activity completion cannot be undone, the influence

XML 5.4 Invariant set-property for export_SG2LD-last-unit

```
<imsld:set-property ref="export_SG2LD-last-unit"/>
```

of this fact on the translation process is important. The specification does not allow to express a sequence in which a learner solved an exercise, the results are graded, a second activity is solved and then the first exercise is solved again. Using other tools as examples [Semet et al.2003, Gutierrez et al.2006], there is no "Continue" or "Next" button in IMS-LD and therefore, a special technique has to be used to implement this behaviour.

In the resource master file previously described, each DIV element is concatenated with a <imsld:set-property>[1]. This element is shown in XML Excerpt 5.4. When the IMS-LD player finds this tag in the XHTML resource, it generates a text box in which the user can introduce directly a value for the property called "export_SG2LD-last-unit".

The element <imsld:set-property> is one of the four global elements in IMS-LD. Global elements are the resources included in IMS-LD to allow users to interact with the learning design through the activities. In particular, <imsld:set-property> permits to introduce data, e.g. the answers to an exercise. When the learning design is created from a SG, these elements are used to obtain the answers as well as any other input data needed from the learner.

The "continue" box just described achieves the same behaviour as a "Continue" button. When the user has finished an activity and wants to advance to the next one the word continue must be entered in the text box, and then press "OK". Figure 3 shows the look-and-feel of the box in the RELOAD IMS-LD player[2].

Figure 3: "Continue" text box in RELOAD

When the "OK" button is pressed, the value "continue" is stored in property *export_SG2LD-last-unit*. This change fires the evaluation of all conditions that containing this property, i.e. conditions related to the edges. The evaluation of those conditions prompts the update of several variables (must notably

[1] Not shown in XML excerpt 5.3 for the sake of simplicity. It is included in the section marked as "Code XHTML of the resource".
[2] Available at http://www.reload.ac.uk/new/ldplayer.html (last accessed June 2008).

export_SG2LD-last-unit and *currentActivity*) and a change on the visible state of the DIV elements. This procedure is covered in more detail in Section 5.2.1.

5.1.3 Resource master file

The resource master file is always created with the same name: *resourceFile.xml*. Therefore, UoLs created with this application always have two files: *imsmanifest.xml* and *resourceFile.xml*.

The size of the master file is proportional to the size of the aggregation of all resources used in the graph. This might present problems for the web browser if the number of resources is high and/or they are heavy documents. Before they are integrated, most likely the resources need to be downloaded and renamed (see [Gutierrez Santos2007] for more information).

5.2 Conditions

The most delicate phase of the process is translating all the semantics of a sequencing graph into IMS-LD conditions and actions. The main difficulty is derived from the change in paradigm. SG follow an *imperative* paradigm: *if* some condition applies, *then* follow this edge; *after that*, *if* you have followed an exit edge, *then* look for more conditions, etc; *after that*, *if* you arrive to a node, *then* go down one level, etc. The conditions in IMS-LD, on the other hand, follow a *functional* paradigm [Mauny and Cousineau2003]. All conditions are evaluated at the same time; if a <imsld:condition> is satisfied and some action is taken, one or several <imsld:property> elements change and all relevant conditions are evaluated again. There is no logical or chronological order or precedence among <imsld:condition> instances.

Conditions in IMS-LD must then capture all the semantics of those in the SG, plus that of the spatial distribution of activities, i.e. current position, existence of edges, etc. To achieve this, some additional special properties have been defined.

5.2.1 Special properties

These special properties are generated in the translation process to provide a behaviour equivalent to that of traversing the SG. More details are provided in [Gutierrez Santos2007].

entering A Boolean property to indicate if the traversal algorithm is entering the node. If true, the *actions* described in the initialisation section of the node have to be executed.

currentActivity Its mission is to record the current node for the learner. This information is evident in SG, but not IMS-LD; thus, a set of conditions and actions has to be deployed to emulate the movement of the learner. The initial value of this variable is the root node of the SG.

export_SG2LD-last-unit Its mission is to record which node the user is going to leave, so that only those conditions associated to the edge's that start on that node are evaluated. Additionally, this property takes a role in the process of moving from an activity: it is the property that is changed when the user presses the OK button in the "continue" box (Figure 3); the word "continue" is then stored. This change in value fires a new evaluation of the conditions.

validMove Boolean variable. The value "true" indicates that the user has introduced the word "continue" in the text box shown by the player (see Figure 3). When this happens, the value of *currentActivity* is copied to *export_SG2LD-last-unit*. This fires the evaluation of those conditions related to the edges that start in the current node; the main effect is that the next activity is shown. If a value different from "continue" is introduced, nothing happens. Property *validMove* has an initial value of "true", so the first activity can be shown directly.

As a summary, edges, conditions and variables in a SG are converted into IMS-LD conditions and properties that combine the SG conditions, topological information about edges (*currentActivity*, *export_SG2LD-last-unit*), hierarchical information (*entering*) and information regarding the navigation decisions of the user (*validMove*, and the "Continue" box).

5.3 Other elements

Aside from nodes and conditions, the additional SG elements that need to be exported are: variables, edges and actions. This section presents the issues in their translation process.

5.3.1 Variables and properties

In addition to the special properties described before, the exporting process must convert all variables in the SG to variables in the UoL. In an SG description file variables are declared at the initialisation section of the node in which those variables exist. If a variable is to be used in different levels of the hierarchy, it must be declared at every one of them.

XML excerpts 5.5 and 5.6 show an example of a SG variable and the corresponding property in IMS-LD. The data type of the property is *integer*. The

XML 5.5 A variable declared in SG

```
<init>
  <set parameter="Qualification" value="0"/>
</init>
```

XML 5.6 The IMS-LD property equivalent to 5.5

```
<imsld:locpers-property identifier="qualification">
 <imsld:datatype datatype="integer" />
 <imsld:initial-value>0</imsld:initial-value>
</imsld:locpers-property>
```

property type is found during the exporting process analysing the actions performed (on the edges) on that variable. For instance, if there is a mathematical operation involving a variable [Gutierrez Santos2007], the corresponding property is marked as integer.

The name of the property is the same as the variable except for the prefix. The prefix implements a name space [W3C] to avoid collisions between properties (i.e. variables). The exporting process requires also flattening the hierarchical structure. As a consequence, two SG variables with the same name in different locations in the graph collide once they are put at the same unique level on the *imsmanifest.xml*. To avoid this problem, a prefix is added to the name of each IMS-LD property with the name of the node in which the original value was declared as well as the name of every ascendant node up to the root.

For example, if the variable *qualification* is declared in a node *Exercise3*, included in *Cache* and included in a node *Computer Architecture Course*, the resulting portion of the *imsmanifest.xml* is presented in XML excerpt 5.7.

The final step with respect to variables includes their initialisation. In a SG, the initialisation section contains the variables and their initial values. Every time the learner enters a node, the initialisation actions must be performed. Therefore, some additional XML code must be added to the *imsmanifest.xml* to perform the same operations every time the learner "enters" a node (i.e. goes to one of its input nodes). This action translates into an *if* condition as shown in XML excerpt 5.8. The value of the special property *entering* compared to be *true* to assure that the variables are only initialised when the learner enters the node and not when returning from from a sibling node.

XML 5.7 The IMS-LD property equivalent to 5.5, with its prefix

```
<imsld:locpers-property identifier="Cache_Exercise3_qualification">
 <imsld:datatype datatype="integer" />
 <imsld:initial-value>0</imsld:initial-value>
</imsld:locpers-property>
```

XML 5.8 Condition to initialise the properties when the user enters in a node

```
<imsld:if>
 <imsld:and>
  <imsld:is>
   <imsld:property-ref ref="currentActivity" />
   <imsld:property-value>Exercise3</imsld:property-value>
  </imsld:is>
  <imsld:is>
   <imsld:property-ref ref="entering" />
   <imsld:property-value>true</imsld:property-value>
  </imsld:is>
 </imsld:and>
</imsld:if>
<imsld:then>
 <imsld:change-property-value>
  <imsld:property-ref ref="Cache_Exercise3_qualification"/>
  <imsld:property-value>0</imsld:property-value>
 </imsld:change-property-value>
</imsld:then>
```

XML 5.9 SG edge

```
<exercise name="This exercise" ...>
 <edge destination="A neighbour exercise">
  <condition>
   <!-- * * * Condition here * * * -->
  </condition>
  <actions>
   <!-- * * * Actions associated to this transition * * * -->
  </actions>
 </edge>
</exercise>
```

5.3.2 Edges, conditions and actions

In a SG, transitions between nodes are determined by the edges, their conditions and their actions on the environment. To reproduce this behaviour in IMS-LD it is necessary to create equivalent structures of conditions and actions. The edge structure in SG is illustrated in XML excerpt 5.9. The equivalent structure in IMS-LD is a pair *if-then* in the section of conditions of the *imsmanifest.xml*, as shown in XML excerpt 5.10.

Informally, this transition can be described as: "*If* going out of "This exercise", *and* conditions for this edge are met, *then* perform actions associated to this edge and *then* set "A neighbour exercise" as current activity. For any edge in the SG, a similar structure is created in the *imsmanifest.xml*. The first step must be checking if the last activity delivered was the origin of the edge. Therefore, conditions and actions are not taken into account unless their corresponding edge might be the one selected by the learner; in other words, the learner can only follow those edges that start at the current node.

In a SG, actions are used to modify the value of variables in the environment and are mostly equivalent to <imsld:change-property-value> actions in IMS-LD.

XML 5.10 IMS-LD equivalent to 5.9

```
<imsld:if>
 <imsld:and>
  <imsld:is>
   <imsld:property-ref ref="export_SG2LD-last-unit" />
   <imsld:property-value>This exercise</imsld:property-value>
  </imsld:is>
  <!-- * * * Condition here * * * -->
 </imsld:and>
</imsld:if>
<imsld:then>
 <!-- * * * Actions associated to this transition * * * -->
 <imsld:change-property-value>
  <imsld:property-ref ref="currentActivity" />
  <imsld:property-value>A neighbour exercise</imsld:property-value>
 </imsld:change-property-value>
</imsld:then>
```

As a consequence their translation procedure is trivial. One complete example can be found in [Gutierrez Santos2007] that illustrates the process.

6 Discussion

The design of this translation process, its implementation and subsequent tests have drawn some interesting insights about the limits of the IMS-LD specification. It must be noted that the specification is being used for a purpose that is different from its main goal. IMS-LD aims at describing pedagogic strategies in which collaborative learning and synchronisation of different roles are very important. However, in this work it has been used to describe a personal tutoring process in which the sequencing of learning activities is adapted. This strains the specification to its own limits, revealing clues about its current possibilities and future extensions.

The first realization is related to the condition model of IMS-LD. Its functional paradigm does not scale well if used "as-is". When the number of conditions in a SG grows, it becomes more and more difficult to keep track of all conditions and the possibility of unwanted side effects increases dramatically. Every condition must be checked for every possible case, leading to a design process that is both tedious and error-prone. Although this may be adequate for small control tasks, the creation of large set of conditions and actions that control an equally large number of activities becomes unfeasible unless some automatic tool is used to help. This clearly shows the value of higher level tools that produce IMS-LD output[3].

The immediate consequence of this is that complex sequences, like those involved in any non-trivial tutoring process, are impossible to create and/or

[3] Our group has created a graphical tool for the creation of Sequencing Graphs.

maintain using IMS-LD conditions directly. In other words, while it is possible to create a UoL with a rich sequencing strategy, maintaining it directly is a very difficult task (impossible in practical terms in most cases); making changes in such a UoL will thus require that they are done in another tool (e.g. with Sequencing Graphs) and then translated into IMS-LD (i.e. creating a new UoL that is a revision of the former one).

Another important issue is that IMS-LD does not provide support for reflexive cycles when accessing the material. This is a substantial limitation, because it has been shown that going over some known material for revision improves learning, and this is specially true when long periods of time are considered (like in Life Long Learning scenarios). However, IMS-LD is oriented towards a sequence of activities that is strongly linear. This is originated on the metaphor of the theatrical play, and produces consequences such that an activity cannot be uncompleted once it has been set as "completed" by the learner. This paper has shown that it is possible to express a SG, even with cycles, but this translation comes at a cost: the resulting *imsmanifest.xml* files are immense and a peculiar text box has to be introduced that may distract the learner introducing noise in the learning process.

With regard to the issue of the size, our studies show that even trivial (e.g. one single cycle) deviations from linear sequences produce very large *imsmanifest.xml* files. The size of this file grows polynomially with the number of nodes and edges on the SG. A fully detailed example can be found in [Gutierrez Santos2007], pages 147–156, but is ommitted here for the sake of space. Although this is not a theoretical limitation of IMS-LD, it imposes a practical limitation on the number of learning units that can be present in a course with a flexible sequencing strategy.

Finally, the policy that, once an activity is completed, it "stays completed in the run" [IMS2003b], has been found to be excessive and limiting. Such a behaviour is justified when dealing with synchronisation elements (e.g. <imsld:act>), but not in the case of activities, when it could be interesting to come back and perform them again (e.g. exercises). This restrictions hampers the use of elements like <imsld:on-completion>, as they can be used only once. A possible solution to this problem is to include a distinction between the state "completed and unable to repeat" and the state "completed, but repetition allowed". It is our impression that a specification that aims at describing any pedagogical process should provide for such a distinction. Both completion events would allow actions <imsld:on-completion>) when they occurred. This would simplify the adaptation of graph-based sequencing definitions to IMS-LD, making it possible to use some formalism like the Sequential Graphs to define a complex sequencing strategy without producing enormous XML files that are heavy to process.

7 Related Work

AHA! [Stash and Bra2002] is a general-purpose Web-based adaptive framework for building hypermedia systems that puts the focus on simplicity and allows many types of adaptation rules. The framework tries not to enforce a specific adaptation style, allowing complete presentation freedom for the system designer. There are two authoring tools for AHA!, the Concept Editor and the Graph Editor. The Concept Editor is a low-level tool in which the author creates a predefined set of attributes and adaptation rules. There are rules of two types: *generate rules* and *requirement rules* and are attached to a newly created concept, using a template. The Graph Editor uses a graph metaphor, similar to the one presented in Section 3 in which the nodes are concepts and the arcs represent different dependency rules between concepts such as knowledge propagation, prerequisite knowledge, etc. An automatic process transforms these high-level constructs into low-level adaptation rules. Thus, the result of the Graph Editor may be further refined with the Concept Editor. The rules and relationships are then translated into the combined domain and adaptation model of AHA! During this translation process some transitions between resources are enabled or disabled according to the rules defined in the previous step.

Both the first phase of the process presented in this paper (see Section 3) and the authoring process in AHA! are similar to the CADMOS-D methodology [Papasalouros et al.2004], that uses UML to describe sequences of activities. The methodology is part of *CADMOS*, a system designed to develop web-based educational applications.

RELOAD Learning Design Editor[4] is a comprehensive and flexible editor for the creation of Units of Learning (UoL). Unfortunately, the user of this editor requires a deep knowledge of the IMS-LD specification. The high complexity of this specification translates into a higher than desired barrier for widespread use. Some authors have tried to tackle this problem providing templates [Hernandez-Leo et al.2006] for the most common UoL structures. Although these templates hide part of the possibilities of the specification, they greatly simplify the process to create UoLs. The use of templates with IMS-LD is complementary to the approach described in this paper. The sequences of resources processed can be obtained by using templates. Other proposed solutions to lower the adoption barrier of this specification are oriented toward the reuse of elements of previously created UoLs [Berlanga and Garcia2005]: objectives, prerequisites, method, learning activities, adaptive rules, personalisation properties, etc.

There is another IMS specification that is aimed specifically at the problem of sequencing: IMS Simple Sequencing [IMS2003c]. Curiously enough, it is not adequate for the work presented in this paper. The specification describes a

[4] Available at http://www.reload.ac.uk/ldeditor.html (last accessed June 2008).

simple strategy for sequencing, based on a tree structure that is traversed in preorder (i.e. top to bottom, left to right). IMS Simple Sequencing lacks a general user model, making it impossible to express most sequencings that depend on some degree of user modelling. For example, the sequencing cannot be influenced according to a "skill level" of the student, because there is no such concept in the specification and there are no means to express it.

8 Conclusions and Future Work

There are two important challenges when creating an adaptive hypermedia course: designing a flexible sequencing strategy and reusing the course in different systems. This paper focuses on the second problem, while keeping an eye on the other. A translation process has been presented to translate sequencing strategies modelled by a Sequencing Graph into a UoL described using the IMS Learning Design specification. A Sequencing Graph allows for the definition of complex sequences (including reflexive cycles), and is conceived to ease the process of authoring a large course including a high number of resources due to its inherent hierarchy.

The size of the resulting UoLs is not excessively big when the exported SG is small. This suggests that SGs could be used as an authoring formalism for small UoLs designed for micro-tutoring (small courses composed of few items, oriented more towards remembering than effective teaching).

IMS-LD claims that it can describe any learning design, with any pedagogical approach, involving a single or multiple users ([IMS2003b], Section 1.1). Although this work has not proven otherwise theoretically, it has found several practical limitations to this claim.

Several problems appeared when trying to express instructional designs expressed in SG in terms of IMS Learning Design. The translation process described in this paper shows that it is possible to define a flexible set of adapted sequences of learning material with a SG, translating it later into IMS-LD to use it in many platforms. However, the cost of the process is high: it introduces noise in the approach (i.e. the "continue" box) and the resulting files (the manifest and the master resource file) are so big that they present memory problems in some systems, even when the original graph is not very complex. This suggests that the goal of IMS-LD —that is, being able to express any instructional design— is only partially fulfilled.

The next step is to investigate how these graphs could be used for collaborative activities. Collaborative learning has been shown by cognitive psychology as having a positive effect on learning [O'Donnell et al.2006]. Besides, one of the main goals of IMS-LD is being able to define a learning design in which collaborative learning processes take place.

Acknowledgments

We thank Gemma Herrera Recio and José Manuel Durán Vivancos for their support in this research. This work was partially funded by the "Programa Nacional de Tecnologías de la Información y de las Comunicaciones", project TSI2005-08225-C07 (*mosaicLearning*).

References

[ADL2004] ADL (2004). *SCORM 2004 Sharable Content Object Reference Model*. Advanced Distributed Learning.

[Berlanga and Garcia2005] Berlanga, A. J. and Garcia, F. J. (2005). IMS LD reusable elements for adaptive learning designs. *Journal of Interactive Media in Education (Electronic Journal)*, 2005/01.

[Berlanga et al.2006] Berlanga, A. J., Garcia, F. J., and Carabias, J. (2006). Authoring adaptive learning designs using IMS LD. In *Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 4018 of *Lecture Notes in Computer Science*, pages 31–40. Springer-Verlag.

[Bloom1984] Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Research*, 13:3–15.

[Bra et al.1999] Bra, P. D., Brusilovsky, P., Eklund, J., Hall, W., and Kobsa, A. (1999). Adaptive hypermedia: Purpose, methods, and techniques. In *Proceedings of the 10th ACM Conference on Hypertext and hypermedia*, pages 199–200.

[Brusilovsky2001] Brusilovsky, P. (2001). Adaptive educational hypermedia. In *International PEG Conference*, pages 8–12.

[Cristea2004] Cristea, A. (2004). Authoring of adaptive and adaptable educational hypermedia: Where are we now and where are we going? In *IASTED International Conference in Web-Based Education*.

[Cristea and Burgos2006] Cristea, A. and Burgos, D. (2006). Authoring adaptive hypermedia and ims learning design: A possible understanding? In *Int. Conf. on Advanced Learning Technologies*, pages 1190–1191.

[Garrison and Kanuka2004] Garrison, R. and Kanuka, H. (2004). Blended learning: Uncovering its transformative potential in higher education. *The internet and higher education*, 7:95–105.

[Gutierrez et al.2006] Gutierrez, S., Pardo, A., and Delgado Kloos, C. (2006). A modular architecture for intelligent web resource based tutoring systems. In *Int. Conf. on Intelligent Tutoring Systems*, volume 4053 of *Lecture Notes in Computer Science*, pages 753–755.

[Gutierrez et al.2004] Gutierrez, S., Pardo, A., and Kloos, C. D. (2004). An adapting tutoring system based on hierarchical graphs. In *Adaptive Hypermedia*, volume 3137 of *Lecture Notes in Computer Science*. Springer.

[Gutierrez Santos2007] Gutierrez Santos, S. (2007). *Sequencing of Learning Activities Oriented Towards Reuse and Auto-Organizarion for Intelligent Tutoring Systems*. PhD thesis, Carlos III University of Madrid.

[Harel1987] Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3).

[Hernandez-Leo et al.2006] Hernandez-Leo, D., Villasclaras-Fernandez, E. D., Asensio-Perez, J. I., and Dimitriadis, Y. (2006). COLLAGE: A collaborative Learning Design editor based on patterns. *Educational Technology & Society*, 9.

[IEEE CS/LTSC2002] IEEE CS/LTSC (2002). IEEE Standard for Learning Object Metadata. Final Specification v.1.0.

[IMS2003a] IMS (2003a). IMS Learning Design. Final Specification v.1.0, IMS Global Consortium.

[IMS2003b] IMS (2003b). IMS Learning Design Information Model. Final Specification v.1.0, IMS Global Consortium.

[IMS2003c] IMS (2003c). IMS Simple Sequencing. Final Specification v.1.0, IMS Global Consortium.

[Koper2005] Koper, R. (2005). Designing learning networks for lifelong learners. In *Koper, R. and Tattersall, C. (eds.), Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*.

[Magoulas et al.2006] Magoulas, G. D., Papamarkos, G., and Poulovassilis, A. (2006). A services-enabled environment for personalising lifelong learning pathways. In S. Weibelzahl, A. C., editor, *Proc. of Workshops held at the* 4^{th} *Int. Conf. on Adaptive Hypermedia and Adaptive Web-based Systems*, pages 140–147.

[Mauny and Cousineau2003] Mauny, M. and Cousineau, G. (2003). *The Functional Approach to Programming*. Cambridge University Press.

[O'Donnell et al.2006] O'Donnell, A. M., Hmelo-Silver, C. E., and (Eds.), G. E. (2006). *Collaborative Learning, Reasoning, and Technology*. Lawrence Erlbaum Associates, Inc.

[OUNL] OUNL. CopperCore, The IMS Learning Design Engine (http://coppercore.sourceforge.net, last accessed Feb 2007).

[Papasalouros et al.2004] Papasalouros, A., Retalis, S., and Papaspyrou, N. (2004). Semantic description of educational adaptive hypermedia based on a conceptual model. *Educational Technology and Society*, 1.

[Semet et al.2003] Semet, Y., Yamont, Y., Biojout, R., Luton, E., and Collet, P. (2003). Artificial ant colonies and e-learning: An optimisation of pedagogical paths. In *10th International Conference on Human-Computer Interaction*.

[Sherry1996] Sherry, L. (1996). Issues in distance learning. *Int. J. of Educational Telecommunications*, 1:337–365.

[Stash and Bra2002] Stash, N. and Bra, P. D. (2002). AHA! a general-purpose tool for adaptive websites. In *World Wide Web Conference*.

[Turine et al.1997] Turine, M. A. S., de Oliveira, M. C. F., and Masiero, P. C. (1997). A navigation-oriented hypertext model based on statecharts. In *ACM conference on Hypertext*, pages 102–111.

[W3C] W3C. Namespaces in XML (http://www.w3.org/tr/rec-xml-names, last accessed feb 2007).

[Zembal-Saul et al.2000] Zembal-Saul, C., Blumenfeld, P., and Krajcik, J. (2000). Influence of guided cycles of planning, teaching, and reflection on prospective elementary teachers' science content representations. *Journal of Research in Science Teaching*, 37(4):318–339.