An eLearning specification meets a game: authoring and integration with IMS Learning Design and <e-Adventure>

Pablo Moreno[1], Daniel Burgos[2], José Luis Sierra[1], Baltasar Fernández-Manjón[1]

[1]Complutense University of Madrid
Department of Software Engineering and Artificial Intelligence
Profesor José Garcia Santesmases s/n
28040 Madrid, Spain
{pablom;jlsierra;balta}@fdi.ucm.es

[2]Open University of The Netherlands
Educational Technology Expertise Centre (OTEC)
Valkenburgerweg, 177
6419AT Heerlen The Netherlands
daniel.burgos@ou.nl

Abstract

IMS Learning Design (IMS LD) is a specification to realize adaptive and interoperable Units of Learning. However, the authoring process remains difficult, mainly due to the lack of high-level authoring IMS LD tools. In order to improve this process, more powerful external resources could be used. Furthermore, the main challenges are a) the authoring process, and b) the integration between these external resources developed with other technologies and the personalized learning experience of an IMS LD Unit of Learning. On the other hand, electronic games and simulations (eGames) are useful for adaptive learning, as they can adapt content, flow and other features based on the user´s inputs. The <e-Adventure> Project provides a full framework to develop conversational games that could be used together with IMS LD Units of Learning to improve personalized learning. The main challenge of this setting is the communication model between both parts, the IMS LD specification and <e-Adventure>, that enables a mutual influence of the adaptive learning experience. In this paper, we describe IMS LD and <e-Adventure>, their integration and communication model, and the authoring process to make the example case of a fully operational IMS LD Unit of Learning with an integrated eGame developed with <e-Adventure>.

Keywords

IMS Learning Design; <e-Adventure>; communication layer; eGames; integration; adaptive learning;Game-based learning

1. Introduction

IMS Learning Design (IMS LD) is focused on providing a flexible specification to model pedagogies. Two of the main objectives of this specification are to model adaptive learning and to provide interoperability. IMS LD can be complemented with externally developed modules integrated in Units of Learning. When this integration is possible, adaptation and learning with IMS LD could be improved and the authoring process on these topics could become easier and more specific.

eGames can provide adaptive learning and fully personalized itineraries, in addition to the motivational enhancements of the learning process. This adaptation could be based on a number of inputs, i.e., the user's behaviour, performance, previous knowledge or personal decision. Within eGames, we distinguish different subgroups, one of these are consisting of the conversational games that have several properties that are worth examining: they are easy to understand and to model, they are well established and keep a long tradition in the game industry, and they provide an engaging and challenging experience where the content and the player's performance keep a key role in the flow. In this line, the <e-Adventure> Project has developed a language, an engine and a player to author conversational eGames.

Within the following sections we show how an adaptive IMS LD Unit of Learning can be modelled and integrated with an external resource, a game created with <e-Adventure>, an XML language. The ultimate goal aims at the full integration of external software applications and modules developed outside IMS LD with the specification in order to improve the users´ experience on adaptive learning and the authoring process. We a) first describe IMS LD, the problems of this objective and the various options when authoring adaptive IMS LD

Units of Learning and when integrating externally developed modules; b) we support the use of eGames on the adaptive learning process; c) we describe the content-centered authoring approach in the <e-Adventure> Project; and d), we describe a specific example case on authoring of an adaptive IMS LD Unit of Learning supporting an external <e-Adventure> eGame.

2. IMS LD and adaptation

IMS Learning Design (IMS LD) [1] is a specification to represent and encode learning structures and methods for learners and teachers. Furthermore, IMS LD is focused on the design of pedagogical methods able to manage learning activities linked to learning objects within a learning flow [2]. This learning flow consists of plays, acts, activities, activity structures and environments and it is flexible enough to provide several personalized itineraries depending on the role assigned or on a set of rules. Several examples are available at [3, 4]. IMS LD consists of three levels: Level A is the main part of the specification as it provides the baseline to build any Unit of Learning with the elements Method, Plays, Acts, Roles, Role-parts, Learning activities, Support activities and Environments; Level B adds some features to create more complex lesson plans using Properties, Conditions, Calculations, Monitoring services and Global elements; and Level C adds Notifications. Every layer improves the previous one [5].

Every single step between the creation and the use of a Unit of Learning needs an IMS LD compliant tool. However, current tools (IMS LD based [6, 7] and general editors [8]) do not allow for an easy editing and a significant effort is still needed to create adaptive IMS LD UoLs [9]. They make the creation of adaptive UoLs technically possible, but too difficult for a non-technical user. A higher level layer with a more visual metaphor is still missing, although some initiatives are taken. For instance, the TENCompetence Project [10] and Complutense University [11] develope a visual LD Editor each.

IMS LD is able to manage six types of adaptation [9] with a different success factor: Learning flow, content, evaluation and interactive problem solving support are fully supported. User grouping is supported by administrative tools and modification of a course on-the-fly can be partially implemented. Also interface-based modification is possible as long as the modifications are made inside the Unit of Learning and not in the player tool itself. Several of these six possibilities are also useful to manage complementary issues to adaptive learning, like active learning, collaborative learning, dynamic feedback, run-time tracking, ePortfolios and assessment [12].

In addition to the basic structure of Level A, the elements in Level B are the actual key for more expressive UoLs (for instance, based on adaptation or collaboration), as they combine several features that encourage and make more flexible the content and the learning flow [5, 13]. Furthermore, the combination of these elements allows for modeling several classical adaptive methods (e.g., reuse of pedagogical patterns, adaptability, navigational guidance, collaborative learning, contextualized and mobile distributed learning, adaptation to stereotypes), making also use of different structural elements of IMS LD, like Environment, Content, User groups and Learning flow.

3. Integration of eGames in adaptive IMS LD Units of Learning

3.1 . Types of integration

The use of software applications and modules (i.e., eGames and simulations) is a fact in schools and educational environments, although this use is often isolated from e-learning systems and other information packages (e.g., IMS LD, IMS CP, SCORM) [14]. This issue stresses the disconnection between the educational setting and the authoring and execution of the module. A closer integration allows for pedagogical improvements and allows for a better contextualized learning path [15, 16]. Also, the interoperability of the UoLs and of the modules used is crucial to achieve the best educational goals in a personalized context, without compromising the rationale behind IMS LD. To this extend, we find two main types of integration between the UoL and the module: a) The module as an embedded activity with no communication, and b) The module as a fully integrated resource, with a bi-directional communication with the environment and a state sharing.

For example, if we model a Unit of Learning containing several activities and one of them (for instance, called Activity-Game) is an external module, the first approach a) will run this Activity-Game without any communication with the main flow, but no connection is established with the previous activity or the next activity inside the learning flow. Therefore, the module is incorporated to a learning flow but no further communication is established with it [15]. In the second approach b), the previous Activity to the Activity-Game can provide some input to the module. For instance, the learner answers a quiz and the final score is sent directly to the module (Activity-Game). Then, the module could start with an adaptive setting based on this input. For instance, if the score is less than a specific threshold, the starting level is for beginners; if the score is higher, the starting level is for advanced players. Last, a list of values are sent back to the IMS LD learning flow to provide a detailed report and influence the next action to take (for instance, choosing one learning path out of several possibilities). Therefore, the module is a fully operational part of the learning flow, able to send and receive information to and from the UoL (Figure 1). Some related work has been carried out by Vogten et al. [17] addressing the communication between IMS Learning Design and IMS Question and Test Interoperability through a service integration layer named CCSI (CopperCore Service Integration Layer).

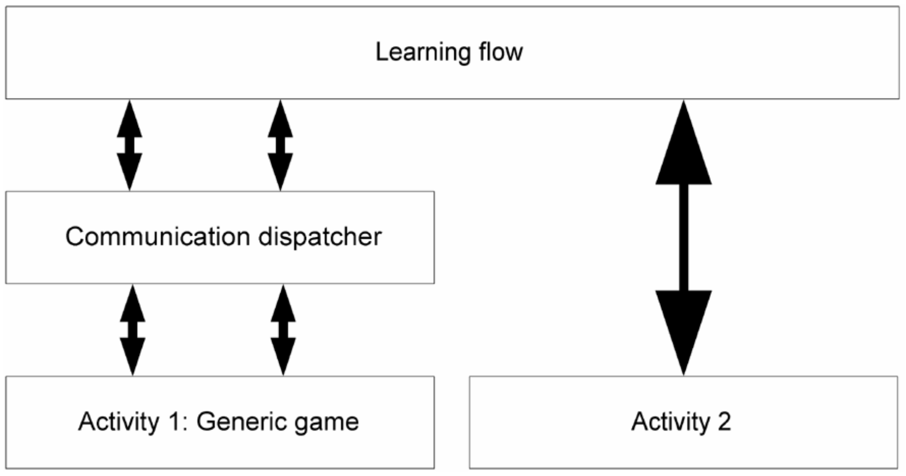

Figure 2. General architecture of communication between the learning flow and the game. The communication dispatcher is supported by the CCSI layer.

Later on we describe the <e-Adventure> Project. We study a) the authoring process with <e-Adventure> and adaptive IMS LD Units of Learning; and b) how this implementation can realize the integration of IMS LD and this external resource for authoring and playing conversational eGames, also with a specific example. We took CCSI as a base to build such integration upon.

3.2 . Adaptive learning and educational eGames

These external software applications and modules can be anything supporting learning, like eGames. Electronic games and simulations (eGames) engage people. EGames improve motivation and involvement, which results in a deeper learning experience [18]. They provide the player with a long list of benefits: fun, interactivity, problem solving, user involvement, motivation and creativity, to mention a few [19]. They also awake personal feelings and emotions in the players, such as wonder, power or aggression, and provide support for the development of personal competences like focused goals, rules, tasks, affiliation, choice, no adverse consequences in case of a wrong choice [20].

EGames also provide input, output and feedback in real-time [21], which are used in adaptive learning [22], e.g., choosing the next action to take or the contextualized help provided. In order to achieve the educational objectives, we can use various interactive learning techniques (i.e., learning from mistakes, goal-oriented learning, role playing and constructivist learning [19]) within and-or around the game itself. The main goal is to make of the game a fully integrated activity inside the whole learning process, instead of remaining as an isolated stand-alone resource. In doing so, generic games, as well as specific educational games, can be used as an interlaced element along the learning experience, increasing the educational threshold [23]. However, we still need a better development of an integrated model to improve the use of games in this way, and to make possible a faster and easier authoring for teachers and learning designers, also when describing personalized learning processes [24].

4. The <e-Adventure> Project

The main goal of the <e-Adventure> Project is to apply a documental approach [25] to the development of educational graphical adventure videogames (often also referred as *point and click* adventure games). The idea is to allow an author without a strong technical background to produce and maintain an entire game as a document using an easy-to-understand language which is then fed to a compiler/interpreter (the <e-Adventure> engine) that produces a fully functional game. Nevertheless, <e-Adventure> is not an authoring tool but an environment consisting of an authoring methodology, an XML-based language and an engine supporting the interpretation and execution of the game. <e-Adventure> also supports personalized processes, like adaptive learning.

The language is an XML dialect that the author uses to describe the environment, characters, objects and situations that form the game. The objective is to allow that author to build an executable game without needing a previous background in programming. All he/she would need is basic knowledge on how to use a computer, a text editor, a few notions of XML and a familiarity with <e-Adventure> syntax. In this section we will give a high level description of the philosophy behind the project. Further details on <e-Adventure> (formerly known as <e-Game>) are described in [26].

4.1 . The <e-Adventure> language

The first requirement of the <e-Adventure> language is simplicity. It is designed to be used by instructors without a background in Information and Communication Technologies. Additionally it must fit on the authoring process described above that begins with the elaboration of the storyboard.

On the other hand, the marked documents are also part of the modification, adaptation and evolution processes, and thus should be human-readable and easily processable by a computer at the same time. These needs suggest that XML technologies are a good candidate. XML is focused on describing hierarchically structured content, which is closer to our needs than a full programming language. Human readability is one of XML's design features [27] and it also provides mechanisms such as DTD and XML-Schema [28, 29] that allow the formalization of the language in a machine-readable way. This facilitates the author's work, by providing the means to verify the correctness of the documents before feeding them to the system.

Thus, the <e-Adventure> language is an XML dialect structured as a storyboard. Following the traits of the genre, the basic unit of construction is the "scene". An <e-Adventure> storyboard (and thus the marked document) starts describing all the scenes that form the game, including the associated resources, their connections to other scenes and the description of the characters and objects that populate the scene, as outlined in Figure 3. For further details about the <e-Adventure> language, refer to [30].

```
(...)
<object id="LibraryBookElaborationChocolate">
 <resources>
       <asset type="image" uri="assets-chocolate/book1.png"/>
 </resources>
 <description>
    <name>Practical Elaboration of Chocolate book</name>
       <brief>Examining this book, I can learn about how to
         prepare chocolate</brief>
       <detailed>Just a book, I should speak with the master cook</detailed>
 </description>

 <actions>
       <examine>
        <condition>
              <active flag="StartedLibraryJob"/>
        </condition>
       <effect>
         <activate flag="ReadedBookElaborationChocolate"/>
        </effect>
   </examine>
 </actions>
</object>
(...)
```

```
(...)
<object id="LibraryBookElaborationChocolate">
 <resources>
       <asset type="image" uri="assets-chocolate/book1.png"/>
 </resources>
 <description>
    <name>Practical Elaboration of Chocolate book</name>
       <brief>Examining this book, I can learn about how to
         prepare chocolate</brief>
       <detailed>Just a book, I should speak with the master cook</detailed>
 </description>

 <actions>
       <examine>
        <condition>
              <active flag="StartedLibraryJob"/>
        </condition>
       <effect>
         <activate flag="ReadedBookElaborationChocolate"/>
        </effect>
   </examine>
 </actions>
</object>
(...)
```

Figure 3. Markup for describing scenarios.

4.2 . Game state

Just adding markup to describe the elements that conform the game yields a plain game where every door is always open, every character always says the same things and every exit leads to the same place. For the game to be interesting, it is necessary to support the means to provide a sense of narration. We can achieve this by introducing a notion of “state”. All the actions that we perform in the game should be able to affect future actions. Some objects may be hidden until something happens (e.g. the object appears only if the player has performed action X); some exits may be locked (e.g. you can’t enter the library until you are admitted to the course or until you talk the secretary into letting you in); and a character may offer a different conversation (e.g. the secretary is more friendly after the player gives her a small gift).

From the perspective of the author, these interactions are conceptually modelled by allowing each interaction (with an object or character) to activate conditions or, in the <e-Adventure> terminology, *flags*. Then, the author can add preconditions to any element of the game. Intuitively, the state at any given point of the game is the set of active flags, which are an indication of which relevant actions have already been performed. Figure 4 provides a simple example of this mechanism.

(a)

```
<actions>
  <use-with idTarget="ceramic_pot">
    (...)
    <effects>
      <activate flag="FINISHED_LAST_SAUCE"/>
    </effects>
  <use-with idTarget="ceramic_pot">
</actions>
```

(b)

```
<conversation id="proceed_to_final_stage">
  <condition>
    <active flag="FINISHED_LAST_SAUCE"/>
  </condition>
  (...)
```

Figure 4. Use of flags and conditions. (a) The effect of an action can be the activation of a flag. (b) A conversation with a specific character that will only be played if the condition holds

5. The communication between <e-Adventure> and IMS LD

As described in section 2, the integration of eGames (implemented with <e-Adventure> or any other authoring methodology) is a complex task and raises a number of authoring issues. The basic problem is that when a learner is interacting with a specific Unit of Learning, the specification demands that the player keeps record of the state of a number of variables, called properties that can be used to alter the path of the learning flow. On the other hand, eGames are often analyzed in terms of game states, which may or may not be directly expressible in terms of IMS Learning Design properties.

It is necessary to provide the means to communicate and to translate the information used within the Unit of Learning and the information used within the eGames, which may require a strong programming background for the author.

However, <e-Adventure> supports a clear and narrow eGame model, in which information is stored and interpreted in a declarative fashion. The task of implementing the game is facilitated by the use of a domain-specific language that can be understood and applied without a programming background. The same ideas can be applied to the authoring of the information flow, thus allowing the non-technical author to specify the communications that should take place between the eGame and the Unit of Learning in a declarative fashion. These specifications written by the author are interpreted by the communication dispatcher described in Figure 2, integrated in the CopperCore Service Integration Layer [17]. In addition, our tests have been performed using a modified version of the open source SLeD player, designed to support any kind of eGames and not only <e-Adventure> games [31].

The rest of this section describes the documents that should be created by the author to specify the translations between properties in the Unit of Learning and game states that should be performed.

5.1 . Mapping UoL properties to <e-Adventure> game states

While authoring an adaptive eGame using <e-Adventure>, the game designer is required to implement that adaptation in terms of conditions over the state of the flags, since that is the mechanism used within <e-

Adventure> to make conditional decisions at any point. Indeed, if the state of a number of flags is modified as indicated in section 3.2, the game can exhibit a completely different behaviour. If defined carefully, these different behaviours can correspond with different adapted versions of the same game.

The document identifying the relations between sets of properties and states is an XML file with the syntax outlined in figure 5.

```
<adaptation-rule>
  <uol-state>
    <property id="PASSED_EXAM" value="true" />
    <property id="PREVIOUS_KNOWLEDGE" value="HIGH" />
  </uol-state>
  <game-state>
    <activate flag="FINISHED_CHAPTER_1" />
  </game-state>
</adaptation-rule>
<adaptation-rule>
  <uol-state>
    <property id="LEARNING_STYLE" value="EXPLORATORY" />
  </uol-state>
  <game-state>
    <activate flag="SKIP_TUTORIAL" />
  </game-state>
</adaptation-rule>
```

Figure 5. Example of adaptation rules. The first rule omits the first and most basic chapter of the game. Also, the initial tutorial describing the controls of the game and the tasks to be performed are skipped if a learner preference for exploratory learning has been identified.

This configuration file closes the gap that separates the internal representation of the state of a Unit of Learning and the internal representation of different initial game states, thus enabling the communication from the Unit of Learning towards the eGame. When the game is launched, the communication dispatcher depicted in Figure 2 checks the state of the properties within the Unit of Learning and sees if their value triggers a specific initial game state. In case it does, the dispatcher informs the game of which flags should be activated (i.e. which is the initial state for the game).

5.2 . Mapping <e-Adventure> game states to UoL properties

Once the eGame has been designed and written using <e-Adventure>, instructors or learning designers can also prepare separate documents identifying those game states that are relevant from a pedagogical perspective and that should affect the state of the current Unit of Learning. When the <e-Adventure> engine is running, every change in the state of the game is notified to the dispatcher. The dispatcher, in turn, checks the new state against the list of pedagogically relevant states. If the state is listed, the dispatcher notifies the Unit of Learning to set the indicated properties to the corresponding values.

The document identifying the relations between states and sets of properties is an XML file with syntax outlined in figure 6, which is an extension of the internal assessment engine implemented by <e-Adventure> and described in [32].

```
<assessment-rule>
  <condition>
    <active flag="FINISHED_WORK"/>
  </condition>
  <effect>
    <set-property id="GRADE" value="50" />
  </effect>
</assessment-rule>
<assessment-rule>
  <condition>
    <active flag="FINISHED_WORK"/>
    <either>
      <active flag="SECONDARY_OBJECTIVE_A"/>
      <active flag="SECONDARY_OBJECTIVE_B"/>
    </either>
  </condition>
  <effect>
    <set-property id="GRADE" value="75" />
  </effect>
</assessment-rule>
```

Figure 6. Example of assessment rules. If the work is finished, we set the property GRADE to have a value of 50%. Achieving at least one of the secondary objectives raises that mark to 75%

Each entry in this file is a mapping between a game state and a set of values for some of the properties present in the Unit of Learning. The game state is represented as a Boolean expression on the flags as used in the <e-Adventure> language itself (see section 3.2). Meanwhile, the properties in the Unit of Learning that should be modified are expressed with a list of set-property elements identifying the property to be set and its new value.

Given the nature of this process, it is important to note that the separation between this mechanism and the definition of the game in terms of states conditioned by flags is wide enough to take an authoring approach in which the writer of the game and the instructor identifying the pedagogically relevant states need not be the same person. This mechanism can thus cater to a scenario in which the instructor is creating a game on his or her own and to a scenario in which the instructor is part of a team in which there are professional writers designing the game.

6. A practical example of authoring and integration: "The art and craft of chocolate"

In order to show the approach aforementioned we have developed an adaptive IMS LD Unit of Learning with an integrated eGame externally modelled with <e-Adventure> and a bi-directional communication flow resulting in an personalized learning path based on two inputs: the previous knowledge and the performance of the learner [33].

6.1 . Basics and layout

In this game, called "The art and craft of chocolate" (available at [4]), the final goal is to learn more about the world of chocolate from a practical side. The student must know the properties of the ingredients and the history of this product to make tasty sauces that can match with the appropriate selection of meals and with the expectations of the customers.

The eGame pursue several didactic objectives, focused on learning 1) how to make the right combination of the basic elements of chocolate to produce the base mix, 2) how to elaborate different chocolate-based sauces, and 3) how to marry a few chocolate sauces with a selection of dishes. Every objective is related to one stage of the game. The first stage (Library) deals with the origins of chocolate and the basic elaboration; the second stage (Kitchen) is more creative and allows for the elaboration of different chocolate sauces; and the third part (Restaurant) is a practical exercise with customers, where the learner should get the perfect marriage between the dishes selected by these customers and the available sauces. The final grade of the students depends on the satisfaction of every customer attended. The player decides on his own when he is ready to face the customers, although his level of previous knowledge influences the access point. There is no minimum number of sauces required, although there is a restriction: (S)he needs to collect and sort several ingredients and objects, and to make several sauces before being granted to go to the last part (Restaurant) for the live test.

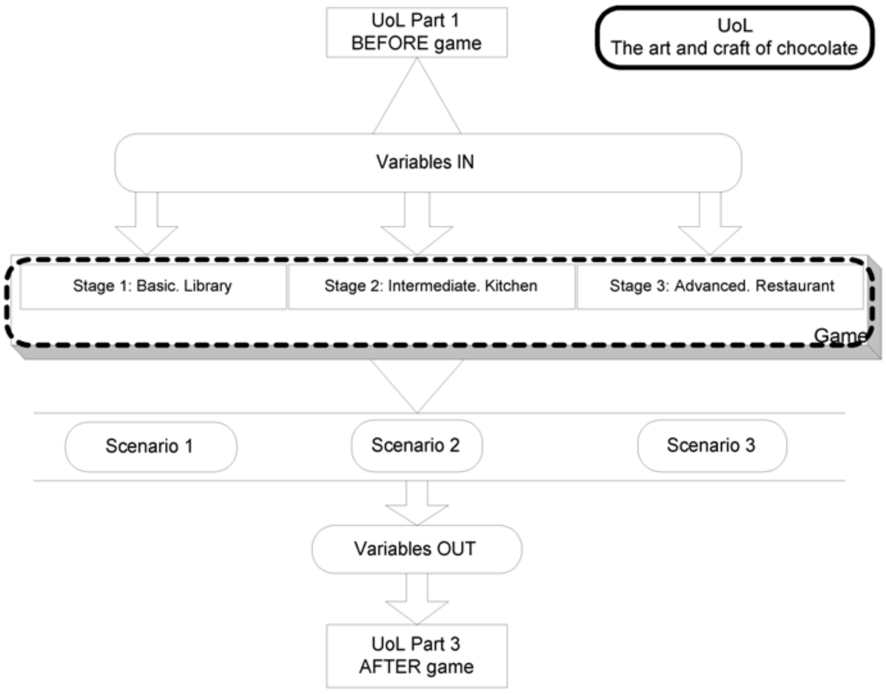

Figure 7. Structure and dependencies of the UoL. Definition of the 3 stages in the game

The structure of the eGame and the UoL is as follows: the game itself is embedded in a full Unit of Learning with a previous quiz and post-adaptive learning path (see Figure 7). Depending on the score out of the quiz, the student accesses directly one of the three stages, where there are control questions. When the answer to one of these questions is not right (s)he receives the right answer and is sent back to the previous stage. When the control question for a direct access to stage 2 is right, the student automatically receives 2 mixes (dark and milk). The student is granted to come back to the stage 1 to make more mixes at any time. When the control question for a direct access to the stage 3 is right, the student receives automatically 3 sauces to be used in the game. The student is allowed to come back to stage 2 to make more sauces at any time (Figure 8).

Once the game is over several variables are sent back, stating the satisfaction level of both customers and which of the possible sauces were actually prepared. The UoL takes these results about the performance of the learner and provide an adaptive learning path out of three possible alternatives.

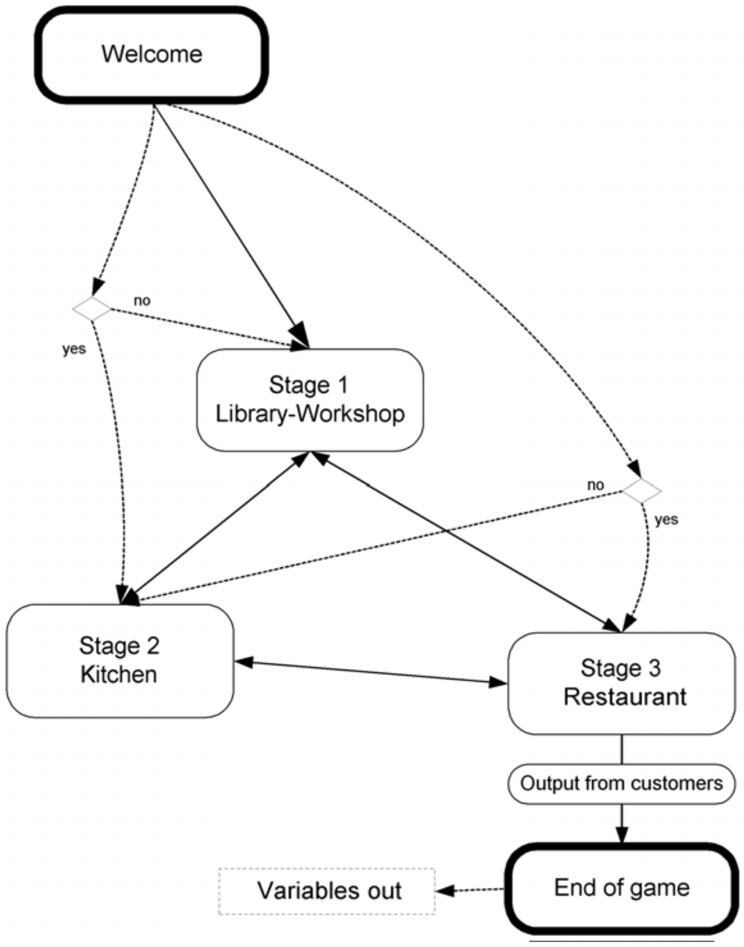

Figure 8. EGame layout and communication flow.

6.2 . Authoring the IMS LD Unit of Learning

The game is embedded in a full Unit of Learning consisting of three parts. The first part is focused on a quiz about the topic where the learner has to answer five multiple-choice questions to establish his-her previous knowledge. Based on this score, the learner will be granted access to a different area of the second part, the game. During the game, the performance of the learner is stored in several variables that will be sent back to the IMS LD part of the UoL. Once the game is finished, and based on the values of these variables, the UoL will provide one or another learning path to follow. In this case, the adaptation is present in the three parts of the UoL, based on previous knowledge and actual performance, although it could be authored to use other different types of inputs. The first part, the quiz, is based on the UoLs Geo-Quiz 1 [3] and they deal with Properties, Conditions and Calculations to define the questions and get the appropriate results out of the user's answers. It also works with adaptive content, showing and hiding different areas depending on the actual moment of the run, and providing an adaptive feedback based on the performance of the user.

The third part, the adaptive learning path after the game, is based on Geo-Quiz 3 [3] and it defines up to three different possible itineraries as learning activities with different content. All of them are hidden until the game is finished. Afterwards, only one of the activities is shown, directly related to the performance of the user during the game.

6.3 . Authoring the adaptive <e-Adventure>

The <e-Adventure> game itself can be defined in parallel, because the adapter can handle the necessary transformations to align (i.e. to translate) the values and variable names used in the Unit of Learning and in the <e-Adventure> game.

The process of writing the adventure starts with the elaboration of the storyboard following a number of guidelines that will facilitate the markup process following the principles of descriptive markup [34]. After the storyboard is approved, there follows an iterative process in which the storyboard is marked up and reviewed and in which the different art assets are produced in parallel. The development of the game itself is a well-studied development process and it is described with greater detail in [26].

However, it has already been mentioned that the communication with the IMS Learning Design raises two additional authoring issues: It is necessary to adapt the values used inside the Unit of Learning and the <e-Adventure> documents for both directions of the communication.

On the one hand, for the communication from the Unit of Learning towards <e-Adventure> with adaptation purposes a document using the syntax described in figure 10 was created. In particular, the first part of the Unit of Learning sets two very specific properties indicating the level of the student according to his or her responses to the quiz questions. The input configuration file for the adapter indicates which internal flags in the eGame should be activated to alter the behaviour of the game so that the simpler parts can be skipped.

On the other hand, for the communication from the <e-Adventure> engine to the Unit of Learning with assessment purposes, a different document was created, using the syntax described in Figure 9. The eGame keeps Boolean flags indicating which dishes with what sauces were served to each customer. The output configuration contains rules that associate dishes delivered with punctuations that will be reported to the IMS Learning Design environment, thus affecting the path followed in the third part of the Unit of Learning.

7. Conclusions and future work

Since the beginning of IMS LD in 2003, the relation with other surrounding technologies (such as Learning Management Systems, other non-IMS specifications, databases or simple stand-alone executable modules developed with any programming language) has been a pending issue. There has been a big research effort to improve the pedagogical expressiveness but not to solve this kind of technical issues and to achieve integration, with exception of some specific work, like the one referred to, combining IMS LD and IMS QTI [17]. The connection between IMS LD and <e-Adventure> and the bi-directional flow of properties that are able to modify the learning flow and the information in both sides shown in this article is a contribution in this direction. Furthermore, this contribution improves a) the authoring means to develop more powerful adaptive IMS LD Units of Learning integrating externally developed eGames; and b) the communication flow between specification and software applications and modules that allows such integration.

However, the introduction of this communication mechanism raises new and important issues when it comes to authoring the adaptive courses and-or Units of Learning. With the objective of allowing teachers and Learning Designers to model rich lectures, full of interactive learning objects, educational objectives and tasks it is necessary to provide the proper authoring tools and communication mechanism.

As it has been described, there are several tools and approaches to IMS Learning Design authoring that cover the authoring needs as far as the description of the general learning model is regarded. Regarding the authoring of adaptive eGames, <e-Adventure> facilitates the definition of conversational eGames for every person with a very low technical threshold. Following a simple and well-documented process, any person can design and implement small to medium-sized conversational games with special features that enhance their educational value, including adaptation and dynamic assessment.

The two authoring processes (Unit of Learning and eGame) are related and their joint use is relatively straightforward, although still difficult to understand for a regular user with little technical background. Furthermore, to achieve a communication between the Unit of Learning and the game there is a need for defining some technical settings, like properties to exchange information. However, there is no high-level editor for either the <e-Adventure> games or the additional configuration files that enable this communication and thus the technical gap could be significant. On the other side, although there is the service layer CCSI to enable communication. The contribution of our research working with CCSI and SLeD improves this communication focused on eGames developed with <e-Adventure> and integrated with IMS LD UoLs.

Given this situation, two main lines of work are foreseen. The first one points to the real use of this authoring setting for adaptive educational games with actual teachers and students. This goal is double-sided: Teachers should practice the process and the making of IMS LD Units of Learning with embedded eGames made with

<e-Adventure>, and students should follow the created course. Both user-groups should provide extremely valuable feedback based on practice.

Second, the development of a graphical editor that makes the creative process easier and suitable for every creator-teacher-learning designer, no matter his technical expertise. This is especially relevant for the definition of the rules that relate the state of the learning process with the internal state of the eGame, which is technically difficult and error-prone and represents the main obstacle in the implementation of our approach.

Acknowledgements

This paper is partially supported by the European projects TENCompetence (IST-TEL/2004-2.4.10, www.tencompetence.org) and ProLearn (IST Contract number 507310, www.prolearn-project.org), and the research group <e-Ucm> (www.e-ucm.es). The Education and Science Spanish Committee has also supported this work partially by means of the projects MetaLearn (TIN2004-08367-C02-02) and OdA Virtual (TIN2005-08788-C04-01). Special thanks to Bruno Torijano Bueno for his participation in the creation and preliminary tests of the sample game.

References

[1] IMSLD, "IMS Learning Design." Boston: The IMS Global Learning Consortium, 2003.
[2] R. Koper and C. Tattersall, "Learning Design - A Handbook on Modelling and Delivering Networked Education and Training." Heidelberg: Springer Verlag, 2005.
[3] LN4LD, "Units of Learning developed by several authors at Learning Network for Learning Design of The Open University of The Netherlands," vol. 2006, 2005.
[4] OUNL, "DSpace," Open University of The Netherlands, 2002.
[5] R. Koper and D. Burgos, "Developing advanced units of learning using IMS Learning Design level B," *International Journal on Advanced Technology for Learning*, vol. 2, 2005.
[6] Y. Miao, *Cosmos LD Editor*, 2005.
[7] Bolton, "Reload Project." United Kingdom: The University of Bolton, The University of Strathclyde and JISC, 2004.
[8] Altova, "XML Spy Editor," 2006.
[9] D. Burgos, C. Tattersall, and R. Koper, "Representing adaptive eLearning strategies in IMS Learning Design," presented at International Workshop in Learning Networks for Lifelong Competence Development, TENCompetence Conference, Sofia, Bulgaria, 2006.
[10] TENCompetence, "TENCompetence Project," 2005.
[11] UCM, "Complutense University. <e-Ucm> Research Group," 2006.
[12] D. Burgos, C. Tattersall, and R. Koper, "Utilización de estándares en el aprendizaje virtual," presented at II Jornadas Campus Virtual, Madrid, 2005.
[13] M. Specht and D. Burgos, "Implementing Adaptive Educational Methods with IMS Learning Design," presented at Adaptive Hypermedia 2006 [www.ah2006.org], Dublin (Ireland), 2006.
[14] P. Cobb, M. Stephan, K. McClain, and K. Gravemeijer, "Participating in classroom mathematical practices," *Journal of the Learning Sciences*, vol. 10, pp. 113–163, 2001.
[15] G. Richards, "Designing Educational Games," in *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*, R. Koper and C. Tattersall, Eds. Germany: Springer Verlag, 2005, pp. Chapter 13.
[16] D. Burgos, C. Tattersall, and E. J. R. Koper, "Can IMS Learning Design be used to model computer-based educational games?," *Binaria [www.uem.es/binaria]*, vol. 5, 2006.
[17] H. Vogten, M. H., R. Nadolski, C. Tattersall, v. Rosmalen, P., and R. Koper, "CopperCore Service Integration, Integrating IMS Learning Design and IMS Question and Test Interoperability," presented at 6th IEEE International Conference on Advanced Learning Technologies, Kerkrade, The Netherlands, 2006.
[18] R. Garris, R. Ahlers, and J. E. Driskell, "Games, motivation, and learning: A research and practice model," *Simulation & Gaming*, vol. 33, pp. 441-467, 2002.
[19] M. Prensky, *Digital Game-based Learning*. New York, USA: McGraw-Hill, 2001.
[20] K. Squire, "Video games in education," *International Journal of Intelligent Simulations and Gaming*, vol. 2, 2002.
[21] D. Laurillard, "Multimedia and the learner's experience of narrative," *Computers in Education*, vol. 31, pp. 229-243, 1998.
[22] D. Leutner, "Guided discovery learning with computer-based simulation games: Effects of adaptive and non-adaptive instructional support," *Learning and Instruction*, vol. 3, pp. 113-132, 1993.
[23] D. Burgos, C. Tattersall, and R. Koper, "Re-purposing existing generic games and simultations for e-learning," *Computers in Human Behavior*, 2006.

[24] D. Williamson, K. Squire, R. Halverson, and J. P. Gee, "Video games and the future of learning," *WCER Working paper*, vol. 2005-4, 2005.

[25] J. L. Sierra, B. Fernández-Manjón, A. Fernández-Valmayor, and A. Navarro, "Document Oriented Development of Content-Intensive Applications," *International Journal of Software Engineering and Knowledge Engineering*, vol. 15, pp. 975-993, 2005.

[26] P. Moreno-Ger, I. Martínez-Ortiz, J. L. Sierra, and B. Fernández-Manjón, "Language-Driven Development of Videogames: The <e-Game> Experience," presented at 5th International Conference in Entertainment Computing (ICEC 2006), Cambridge, UK, 2006.

[27] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, "Extensible Markup Language (XML) 1.0," in *W3C Recommendation*, vol. 2006, 2000.

[28] D. Lee and W. W. Chu, "Comparative Analysis of Six XML Schema Languages," *ACM SIGMOD Record*, vol. 29, pp. 76-87, 2000.

[29] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, "Taxonomy of XML schema languages using formal language theory," *ACM Transactions on Internet Technology*, vol. 5, pp. 660-704, 2005.

[30] P. Moreno-Ger, J. L. Sierra, I. Martínez-Ortiz, and B. Fernández-Manjón, "A Documental Approach to Adventure Game Development," *Science of Computer Programming*, vol. In press, In press.

[31] OUUK, "Sled player." United Kingdom: Open University of United Kingdom, 2005.

[32] I. Martinez-Ortiz, P. Moreno-Ger, J. L. Sierra, and B. Fernández-Manjón, "Production and Deployment of Educational Videogames as Assessable Learning Objects," presented at First European Conference on Technology Enhanced Learning (ECTEL 2006), Crete, Greece, 2006.

[33] D. Burgos, C. Tattersall, and E. J. R. Koper, "Representing adaptive and adaptable Units of Learning. How to model personalized eLearning in IMS Learning Design," in *Computers and Education: E-learning - from theory to practice*, B. Fernández Manjon, J. M. Sanchez Perez, J. A. Gómez Pulido, M. A. Vega Rodriguez, and J. Bravo, Eds. Germany: Kluwer, 2006.

[34] C. F. Goldfarb, "A Generalized Approach to Document Markup," *ACM SIGPLAN Notices*, vol. 16, pp. 68-73, 1981.