Production and Deployment of Educational Videogames as Assessable Learning Objects

Iván Martínez-Ortiz[1], Pablo Moreno-Ger[2], Jose Luis Sierra[2], and Baltasar Fernández-Manjón[2]

[1] Centro de Estudios Superiores Felipe II, Aranjuez, Madrid, Spain
imartinez@cesfelipesegundo.com
[2] Department of Computer Systems and Programming, Universidad Complutense de Madrid, Fac. de Informática, 28040, Madrid, Spain
{pablom, jlsierra, balta}@fdi.ucm.es

Abstract. The generalization of game-based Learning Objects as serious learning material requires their integration into pre-existing e-learning infrastructure (systems and courses) and the inclusion of gameplay-aware assessment procedures. In this paper, we propose an approach to the production and development of educational graphic adventure videogames that can be deployed as normal Learning Objects in a Learning Management System. The deployment is carried out using a game engine that includes a built-in assessment mechanism that can trace and report the activities of the learner while playing the game.

Keywords: game-based learning; adventure educational videogames; edutainment; assessment; learning management systems; learning objects; IEEE ECMAScript content to runtime service communication; IEEE Data Model for Content; <e-Game>.

1 Introduction

The e-learning arena is faced with a number of challenges such as the quality of the learning achievements or improving students' motivation to prevent mid-course dropouts. These problems have been addressed in different ways. For instance, a distinctive characteristic of multimedia-based approaches [21, 26] is to catch the attention of the learner by using a lot of eye-candy in the content, while collaborative learning [2, 22] attempts to motivate the learner by generating peer-pressure, group-belonging and close feedback. An alternative approach is to take advantage of videogames as a highly motivating and entertaining activity that can be used to deliver the content.

We will focus our attention on educational games and how they can be integrated in an e-learning system. A serious learning process cannot consist merely of playing, without any kind of supervision or assessment of the activities within the game and the learning outcomes. Another issue with game-based learning is whether it can (or should) be integrated in pre-existent learning processes without altering already established learning procedures.

The most typical e-learning scenario in which we may try to introduce educational videogames is in a Learning Management System (LMS), a web application that

W. Nejdl and K. Tochtermann (Eds.): EC-TEL 2006, LNCS 4227, pp. 316 – 330, 2006.

delivers content and assesses the activity of the learner inside the system. Most LMSs try to comply with a number of specifications and standards regarding content formats or course structure to facilitate the interoperability of content between different LMSs. This interoperability/reusability process is founded on the Learning Object (LO) Model [18].

This work describes <e-Game>, an environment in which educational videogames are integrated in a complex LMS and treated as LOs. This tries to resolve the two key issues presented: assessment within an educational videogame and LMS integration. In order to further integrate the videogames into the learning process, an engine is used to execute the videogames that can profile the activity of the learner during gameplay and generate reports that are notified to the LMS. In addition, the LMS can send data to the videogame so that the game experience can be adapted.

The paper is structured as follows: Section 2 surveys the applicability of computer and videogames to educational processes. Section 3 presents the development model for educational graphic adventure games induced by <e-Game>, while section 4 presents the <e-Game> built-in assessment mechanism that enhances its educational value. Section 5 deals with the deployment of educational games into LMSs using <e-Game> and, finally, section 6 outlines some conclusions and future lines of work. All these sections have figures related to the experimental development of a short adventure game in the field of workplace safety regulations.

2 Computer and Video Games in Education

Applying games in general (not only videogames) to education is not a new idea. Playing is an inherent trait of human beings which is closely tied to the learning process [23]. The basic claim for supporting the introduction of games into learning processes is that games, unlike more traditional educational situations, are fun to play. Playful activities in a classroom (like role-playing events of history in class) [17] can be a very appealing and effective way of learning, although it is not cost-effective and it is very unlikely that a system involving such a high ratio of instructors/learners may ever be implemented. However, technologies bring up new possibilities for gaming environments with computer and video games, hereafter referred as videogames or simply games.

2.1 Can Videogames Educate?

Videogames allow the immersion of the student in richly recreated environments with a relatively low cost per student. Also, videogames support many educational approaches such as collaborating and/or competing within the game or, alternatively, simulating artificial peers in order to achieve similar results to collective playing in a cost-effective way. In any case, the experiences from in-game activities can then be shared with real peers and monitored by instructors to compensate for the lack of direct human intervention. In addition, the entire process can be fun and appealing to the learner. This description derives from four traits observed in and around modern videogames [9] as follows:

- *Videogames are fun*. And not only for small kids or young male teenagers. As of 2005, the average age of a videogame player is 30 and 43% of players are female [7].This suggests that *entertaining* educational videogames would be appealing for a broad audience and applicable not only for primary/secondary school learners, but also in corporate and long-life learning.
- *Videogames are immersive*. A videogame immerses the player in an environment by using a number of approaches. The most direct one is to allow the player to navigate the environment from a first-person perspective. A second approach is to project the identity of the player onto an avatar. Another alternative is to give the player the power to affect the environment without any visible agent, by clicking on the environment or using a graphic interface (videogames presenting this kind of interaction are often called god-like games, although they may just be simulating the functioning of a corporation or a railroad system). Regardless of the interaction model, videogames transfer the identity of the player to the recreated world and immerse him/her in whatever reality the designer of the game wanted. In [10] the psychological details of this immersion and transfer of identity are elaborated. Educational videogames can thus provide a constructivist and embodied learning process.
- *Videogames stimulate cooperation/competition*. Games are fun and immersive due to their short and efficient feedback loop [22]. Although this feedback loop is manifested in a variety of forms, humans are naturally inclined to interaction with other humans. Thus, the environments in videogames are often populated by characters that provide interaction. Although typical games have the player playing the central role in the story, multiplayer videogames involve several players at the same time and make them collaborate or compete in the achievement of goals. In either case, in most videogames the player interacts directly with other human or human-like entities. When applied to education, videogames can serve as a medium for collaborative learning even without requiring the involvement of actual peers. Alternatively, competitive factors can be employed to motivate learners to perform better.
- *Videogames promote the creation of communities of practice*. Playing a videogame, even a single-player game, is not an isolated activity. Videogames transcend their medium and generate communities around them, proving to be a brilliant example of Communities of Practice [30]. New players soon learn details about the games, strategies, play styles, etc. These communities are essential vaults for learning about a game, and it all happens on a peer-to-peer basis. There are no instructors and no learners, yet some vast vaults of knowledge and learning are formed without external influence. Expert users spend time answering newbie questions and enthusiasts write extensive guides (sometimes above the 300 page count) while only seeking peer recognition. The bottom-line is: Gamers enjoy talking about games after playing them. In the educational field, this suggests potential for discussion and argumentation of the concepts delivered by educational games, ideally directed by an expert instructor (an example of this approach can be found in [31]).

2.2 Integration of Videogames in an E-Learning Process

Regardless of the potential advantages of educational games, it is neither possible nor reasonable to revamp all existing e-learning programs and dispose of the entire existing web-based infrastructure (i.e. LMSs) and content. Educational games should be able to co-exist in the present web-oriented situation. Acknowledging the current trends in web-based education, educational games should be able to fit in an environment in which standards compliancy and the Learning Objects Model are dominant features.

Videogames as Learning Objects

Integrating a huge videogame that can be used to learn an entire domain will usually be difficult. However, if the domain is divided into small pieces of content, it is possible to create and integrate small videogames that focus on a single well-determined concept. This is precisely the concept used in the LO model, where the content is divided into small discrete pieces that can be assembled to form a course.

Many modern LMSs support this model, and their courses are divided into LOs. These LOs are typically made of web documents, but powerful LMSs barely restrict the content format as long as they can be delivered within a web browser. Therefore, this is the perfect situation for the inclusion of small educational videogames in a pre-existing course. The educational videogame can simply substitute one of the learning objects included in the course, as long as it covers the same materials and behaves externally the same way (the last part is not difficult, as most learning objects are passive content).

Videogames as assessable content

The instructors of any e-learning program will usually want to assess the activity of the learner in order to make sure that he/she achieves the results desired. This assessment can be a direct examination of the knowledge of the learner (which typically verifies the results through tests) or an examination of the learning *process* itself (which verifies the activities of the learner, the effort, the learning path, etc.).

Assessing the learning process itself in a web environment consisting of passive web documents is simple and rigid: It is possible to obtain data such as the order in which the themes were displayed and the time that the learner spent reading that content. This kind of measure by itself is not accurate because it does not reflect the real interaction. A sophisticated system may take note of the hour in which a learner accessed a document and when the learner triggered a new event, but this data may be inaccurate since it does not reflect whether the learner actually did anything while the content was being displayed. There is a lot of research related to obtaining assessment information by applying data-mining techniques to get the best out of these measures [13, 14], but with passive web documents, it is not technologically possible to go much further with regard to assessing the learning process.

However, educational videogames offer new opportunities, without rejecting their nature as LOs, because games are more interactive than standard web content. Since the actual learning will occur through a series of player decisions and interactions, it would be very interesting to assess the detailed activities of the player/learner within the game. Questions like “what decisions did the learner make?”, “in which order?” or “what was the result of the action?” are a key point in the assessment of learner activity inside the videogame.

3 Creating Educational Games: The <e-Game> Project

The <e-Game> Project [19,20] proposes a new model for the development of educational games. Game development demands skilled technicians like designers, programmers or artists. Educational videogames also require the participation of expert tutors that provide the instructional design and decide how to deliver the content within the game. In order to facilitate the collaboration between technicians and tutors, the <e-Game> project applies a well-proven technique: The Documental Approach to Software Development [27, 28].

3.1 A New Development Approach

The <e-Game> project applies the advantages of document-driven developments and XML technologies [4] to facilitate the development of educational games. The key idea behind <e-Game> is to allow instructors to develop educational videogames by following a simplified procedure, without requiring the heavy involvement of experts in ICTs.

However, it is not necessary to permit the creation of *any* kind of game. We are interested in Adventure Games, which have been identified as a valuable genre for learning [3, 32]. Besides, studies like [16] indicate the main traits of adventure games and identify a clear bias for content instead of just plain action. In addition, most adventure games have a very similar structure and functionality, which is very valuable to our objective of simplifying the development process.

Our aim is that an author should be able to specify an adventure (we call them eGames) with only basic knowledge on how to use a computer, a text editor and a few notions of XML and the <e-Game> syntax. The next subsections describe this development process.

3.2 Designing and Marking an <e-Game> Adventure

The <e-Game> development process, described in greater detail in [19], starts with the elaboration of a document with the script/storyboard of the game. After writing the storyboard (or at the same time) the author adds descriptive markup to that document that indicates the conceptual meaning of each part of the script [5, 12]. The <e-Game> engine will receive this document along with a set of art assets (graphics, animations, sounds, etc.) and will directly produce the final executable game.

The storyboard writing process starts by defining the different game scenes and indicating their content (i.e. like in a theater play). Scenes have exits that lead to other scenes and the complete set of scenes defines the environment for the game. However, scenarios in an adventure videogame are static: background artwork and exits leading to other scenarios. Although it is possible to include learning content in the scenarios (embedded in the artwork, for instance), that is not their purpose. The storyboard should instead populate the scenarios with objects and characters.

Once the basic structure of the game is laid out in the form of a network of scenes, the next step is to define the objects and characters that populate the game in detail. From a learning perspective, these objects and characters are the interface that delivers most of the learning content, not the content itself. The actual content is delivered by the potential interactions between the learner and those objects and characters.

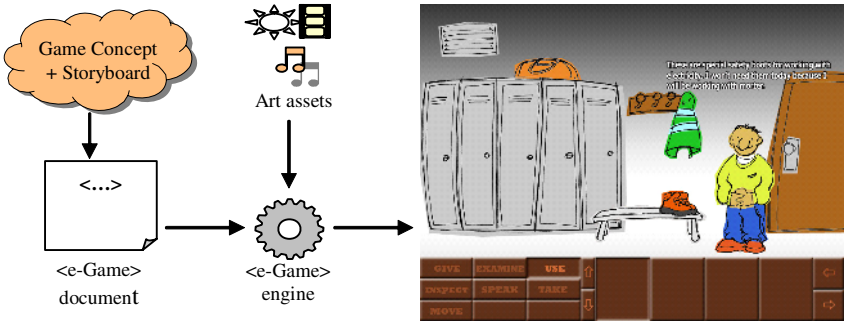

Fig. 1. Outline of the <e-Game> development process model. The storyboard is marked up with the <e-Game> XML syntax and fed to the engine together with the art assets.

```
<scene id="DressingRoom">
  <resources><!-- Art assets --></resources>
  <exits>
    <exit>
      <target idTarget="Corridor"><position x="0" y="300" />
    </exit>
  </exits>
  <elements>
      <item-ref idref="Boots"><position x="40" y="300" /></item-ref>
      <char-ref idRef="Foreman"><position x="485" y="410" /></char-ref>
  </elements>
</scene>
```

Fig. 2. Definition of a scene. There is a section for external assets, a section for exits (leading to other scenes), and a section with references to the elements that populate the scene (objects and characters). Objects and characters are detailed separately.

```
<item id="Boots">
  <resources> <!-- Art assets --> </resources>
  <description>
    <brief>A pair of sturdy boots</brief>
    <detailed>These are special safety boots for construction. The regulations
     demand that they be worn at all times.</detailed>
  </description>
  <actions>
    <use-with>
      <target idRef="Locker" />
      <effect>
        <speak-player>I think I shouldn't leave them here until I go home</speak-player>
      </effect>
    </use-with>
    (...)
  </actions>
</item>
```

Fig. 3. Definition of an item. There are two descriptions, a section for external assets, and a set of actions that can be performed on this element. Every action contains a description of its outcome.

The definition of an object must mention what happens when an object is examined roughly (a brief description), examined in detail (a more detailed description), combined with another object or given to another character. On the other hand, interactions with characters also include offering objects and, most important of all, conversations. The different interactions and conversations that the user performs are the main element for delivering content.

3.3 Adding Narrative Flow and Customization

The previous section has presented the means to describe a simple game, where every door is always open, every character always says the same thing and every exit leads to the same place. We can enhance this by introducing the 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 or some exits may be locked (e.g. you can't use the elevator unless you have been instructed to do so and received the key). The <e-Game> syntax allows each interaction (with an object or character) to activate a condition. The state at any given point in the game is the set of actions that have already been performed. We can then add preconditions to anything we want as a list of conditions that indicate which actions must have been performed.

```
<exit>
  <target idTarget="Rooftop" />
  <condition>
    <active flagRef="SAFETY_EQUIPMENT_READY" />
  </condition>
</exit>
```

(A)

```
<conversation id="CheckEquipment">
  <speak-char>Are you wearing all the safety equipment?</speak-char>
  <speak-player>Yes, sir</speak-player>
  <speak-char>In that case, go to the rooftop and help David</speak-char>
  <end-conversation>
    <activate flag="SAFETY_EQUIPMENT_READY" />
  </end-conversation>
</conversation>
```

(B)

Fig. 4. A conditioned exit (A) and a conversation that triggers it open (B). After the conversation ends, the flag "SAFETY_EQUIPMENT_READY" is activated.

```
<conversation id="FirstGreeting">
  <speak-char>Ah, you must be the worker. What's your name?</speak-char>
  <speak-player>Yes, sir. My name is <param name="username"/> </speak-player>
  <speak-char>Welcome on board, <param name="username"/> </speak-char>
  (...)
</conversation>
```

Fig. 5. Text customization. The `<param name="username"/>` markup will be substituted by a value written in the configuration file when the game is executed.

Finally, most of the content of these games is text, and it may be desirable to customize some pieces of text in order to enhance the gaming experience. The <e-Game> syntax supports this kind of personalization with the inclusion of a special markup indicating that a piece of the text should be read from a configuration file.

This customization mechanism can be used for trivial modification like including the name of the user in conversations. However, the system can be pushed further: if all the texts are derived to a configuration file, we are actually implementing an internationalized game.

4 Assessment in <e-Game>

<e-Game>, as described thus far, is a tool for the documental development of adventure videogames in general. However, <e-Game> is actually oriented to the development of *educational* adventure videogames. For this purpose, <e-Game> has a built-in assessment mechanism. When an author defines an eGame, learning designers can include in the documents information about which tasks are relevant for the learning process. When the videogame is executed, the activation of the relevant tasks is logged and a report can be elaborated for the instructor.

```
<assessment id="WEAR_HARDHAT" type="achievement" importance="high">
    <condition>
      <active flagRef="WEARING_HARDHAT"/>
    </condition>
    <effect>
     <assessment-concept>The user equipped the hardhat</assessment-concept>
     <assessment-text>
       The user grabbed the hardhat before entering the construction site
     </assessment-text>
    </effect>
</assessment>
```

Fig. 6. Assessment action conditioned to the activation of a flag that indicates that the user is equipped with the safety hardhat

4.1 The Assessment Syntax

As depicted in Fig. 4, the state of the game is characterized for the set of flags that are active. <e-Game> allows learning designers to determine which game states are relevant to the assessment process. For this purpose, they include *assessment actions* using `assessment` elements. As depicted in Fig. 6, these assessment actions are associated to conditions in the game state.

An assessment action includes information using different attributes and nested elements that will subsequently be used in the production of reports:

- The `type` attribute: When a report is generated, the assessment elements can be grouped by their type. Initially, <e-Game> suggests the values `achievement` (important events are triggered), `behaviour` (things that the user does within the game and reflect his/her playstyle), `stuck` (points of the game that suggest the user got stuck at a certain moment) and `impossible` (points that should never be reached, mostly for testing purposes). However, the different possible values can be decided by the author for each specific game.
- The `importance` attribute: Reports can be generated with different levels of detail. Thus, the author can decide on the relative importance of each action and assign it a value. Later, the instructor will be able to generate reports with different levels of detail in which elements of lower importance are not included. Again, it is possible to customize the names and order of the possible values, although <e-Game> suggests using `verbose`, `low`, `average` and `high`, with the first one only being used for testing purposes.
- The `assessment-concept` element: It is a description of the action being assessed. Different assessment may have the same concept associated.

- The `assessment-text` element: A text describing how the learner triggered the assessment action.

Of course it is not necessary to fill in all these fields and attributes. Indeed, all assessment actions are assumed to be of type `achievement` and of `average` importance. Also notice that assessments can be included in a separate document. Therefore, the same game can support different assessments tailored to different learning scenarios.

4.2 Assessment Logs and Reports

When the eGame is executed, the <e-Game> engine will keep track of all the assessment actions and log their activation. During the play session or after its completion it is possible to access the assessment log that allows the instructor to understand the behaviour of the student within the videogame. This log is marked up using descriptive markup in order to facilitate its subsequent processing and presentation. For each action triggered, there will be a timestamp and the reference to such an action.

```
<assessment-log>
   <assessment idTarget="WEAR_HARDHAT" timestamp="00:09:30"/>
   <assessment idTarget="FOREMAN_IS_DECEIVED" timestamp="00:21:30"/>
    (...)
</assessment-log>
```

Fig. 7. An assessment log generated by <e-Game>. Assessments actions have an associated timestamp. The XML markup permits the generation of formatted reports on demand.

The logs, together with the assessment actions, can be used to generate enriched HTML reports for the instructors (Fig. 8). To view a report, the instructor can select a level of detail and all actions with a lesser importance will not be displayed. This way, if the instructor selects “average” as the detail level, actions marked as having “average” or “high” importance will be displayed while “verbose” and “low” importance actions will be filtered out.

ASSESSMENT OF ACTIVITIES

Achievements
00:04:23 - Flag ID: DISCOVER_HARDHAT
Event: The user found the hardhat inside the Locker
Description: --

00:09:30 - Flag ID: WEARING_ HARDHAT
Event: The user equipped the hardhat
Description: The user grabbed the hardhat before entering the construction site

Behaviour
00:21:30 - Flag ID: DECEIVE_FOREMAN
Event: The user tried to deceive the foreman
Description: The user lied about the height of the scaffold

Fig. 8. An assessment report where assessments are classified by their types

The open structure of these reports allows the author to use them for a variety of purposes. On one hand, when the author is concerned about the path followed by the learner, the reports can be used to trace the activities of the learner in detail with thorough descriptions. On the other hand, if the objective is to give a single mark based on the last objective achieved (for eGames with multiple possible endings), it is possible to mark only those final actions and put a numeric mark in the text of the assessment element. In the latter case, the report will only contain the name of the final state of the game, the score associated and the time it took the learner to complete the game.

5 Integration of <e-Game> Educational Games in a LMS

As described in the previous sections, the main goal of the <e-Game> Project is to allow authors to create small games that can replace some pieces of static learning content such as an HTML or a PDF file. eGames, like Learning Objects, tend to be small and cover a specific set of learning objectives. Once an eGame has been fed to the <e-Game> engine, it can be executed as a stand-alone program. However, it can also be deployed in a web-based LMS.

5.1 Delivering Games in a Web-Based LMS

As the <e-Game> engine uses Java as supporting technology, the eGames produced can actually be packaged as Java Applets that can be easily embedded into a HTML page. Thus, an eGame can be deployed into practically any LMS that supports web pages with embedded objects as content. In fact, any LMS compliant with the "de-facto" standard IMS Content Packaging [15] can easily have the game deployed as an IMS Resource, with the (applet version of the) <e-Game> engine, the <e-Game> document and the art assets as support files, and a container HTML file as the main file.

This integration, although very straightforward and compliant with almost all the existing LMSs, only offers the motivational advantages of game-based content but does not offer any assessment advantages over alternative simple formats such as a PDF file or a standard applet. The next subsection offers some insights on how an eGame can have a more sophisticated integration with a LMS.

5.2 Communication of the Assessment Reports to the LMS

Games developed with <e-Game> can let a LMS know about the activities of the learner within the game. Depending on the standards supported by the LMS and its implementation, the integration of an eGame within the LMS can be more or less sophisticated:

Blackbox Integration: As described above, this is the most straightforward methodology and almost any LMS that supports web-pages can integrate an eGame. From the LMS point of view, games-based resources are treated as black boxes, with no knowledge of what happens within the game. In this scenario, the best that the LMS

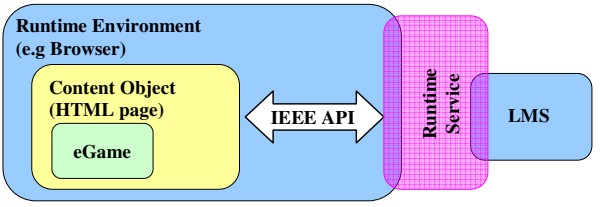

Fig. 9. Use of the IEEE ECMAScript API for communicating assessments logs to the LMS

can do is to obtain the same set of external measures that can be obtained from a simple content file (number of accesses, duration of the visit, etc.).

Standardized communication, not <e-Game>-aware: When the LMS supports it, the IEEE ECMAScript API for Content to Runtime Service Communication [24] can be used for communicating assessment data to the LMS. This Runtime Service is designed as a bridge between the environment (typically a web browser) where the learning content runs and the LMS itself. This communication mechanism can be used with different kinds of LOs, such as, for example, SCORM SCO (Shareable Content Objects) [1]. In general terms, this standard provides a communication protocol between a Content Object and the LMS, but does not specify what kind of information is actually interchanged between the content (in this case, an eGame) and the LMS. To tackle this issue, a recent IEEE Standard, the IEEE DMC (Data Model for Content) to LMS Communication [25], provides a general Data Model for communication exchange.

```
(a) (assessment-identifier,
     Timestamp:type:importance:assessment-concept:assessment-text)

(b) (WEAR_HARDHAT, 000930:achievement:high:
                        The user equipped the hardhat:
                        The user grabbed the hardhat before
                        entering the construction site)
```

Fig. 10. Attribute-value pairs transmitted to the LMS. (a) General syntax. (b) Sample entry according to the example in figure 6.

A LMS with even basic support of the IEEE ECMAScript API will be able to receive basic information from the eGames deployed, as depicted in Fig. 9. For this purpose, the applet version of the <e-Game> engine processes the assessment log and uses the ECMAScript API[1] to push basic information to the LMS in the form of simple attribute-value pairs, as in Fig. 10.

In this way, assessment information is passed to the LMS, which is responsible for storing it. Of course, if the LMS is not <e-Game>-aware, it is very unlikely that the system will be able to do much with all the information received other than simply displaying it without type or level filtering as described in Section 4.2.

[1] Bidirectional communication between Java Applets and ECMAScript is feasible when using for instance the API supported by the Java™ Plug-in.

The LMS is expected to have a mechanism that puts together all the information gathered from a specific resource (in this case, all the attribute-value pairs for a specific game). A simple and comfortable use of this approach is to have the learning designer assign a final score to a specific action.

Standardized communication, <e-Game>-aware: The simple data described above is highly compatible, but it does not exploit the full potential of <e-Game> assessment mechanisms. Even if the engine pushes all the relevant information towards the LMS, the LMS will not necessarily know what to do with that information. However, instead of using a generic inspection tool, it is possible to create a specific application profile aware of the information provided by the engine and get the most out of the <e-Game> assessment system. A prototype of such a tool has been tested in the <e-Aula> LMS [29].

When this tool is present, it lets instructors generate reports for the activity of each learner during each play session. The tool queries the instructor about the types of assessment elements that should be included and the level of detail desired, always according to the default nomenclature and semantics presented in Section 4.1. The tool gathers the information from the LMS storage layer and generates enriched HTML reports that are displayed to the instructor.

In the runtime initialization, the engine can discover if there exists a communication mechanism with the Runtime Service, and if there does, the engine verifies what features are supported. In particular, the engine will check whether the <e-Game> Data Model is supported. If there is no support for any kind of communication, the logs are never generated. Otherwise, if there is support for basic (standard-compliant) communication, or the LMS is <e-Game>-aware, the engine will push all the relevant information contained in the log document using the standard IEEE ECMAScript API.

6 Conclusions and Future Work

The <e-Game> Project, as presented in this work, offers a valid alternative for the gradual inclusion of small games into pre-existing online courses and LMSs. Its documental approach to the development and maintenance of educational adventure games allows authors without an extensive programming background to develop these adventure games.

The games developed by those authors can be executed by using an engine equipped with a built-in mechanism that supports the assessment of the activities performed within the game. For this purpose, <e-Game> includes a flexible syntax that allows the assessment of different aspects of these activities. In addition, the games can be integrated into almost any modern LMS although the depth of the integration will depend on each specific LMS. Nonetheless, <e-Game> supports the three different levels of integration described in this work.

This flexibility and the viability of the development approach have undergone some preliminary testing which suggests a great potential. A group of volunteer authors participated in the development of short adventure games covering different

aspects of workplace safety regulations. The authors were provided with a number of art assets and basic storyboards and managed to mark up those documents and generate the corresponding eGames with hardly any assistance. It must be noted however that all the authors had previous experience with XML applications, which is actually the main concern regarding the applicability of the development model: developing an eGame is an XML-intensive task and many authors without any experience may find the process excessively complicated. On the other hand, the process still has a great advantage over a traditional game development and many authors in the e-learning arena are familiar with XML technologies as long as these support most standardization processes in the field.

Regarding the integration of eGames into different LMSs, the trivial Blackbox integration model described in Section 5.2 was tested in publicly available LMSs such as Moodle [6] and Sakai [8] as well in the commercial WebCT [11] deployed at the Complutense University of Madrid. As for more sophisticated integration mechanisms, the <e-Game>-aware inspection tool developed for the <e-Aula> experimental LMS has demonstrated the viability of the approach.

It is interesting to note that the aforementioned IEEE ECMAScript API and Data-Model Communication standards also support communication in the other direction: The content can query the LMS for specific pieces of information with different purposes. Next steps in <e-Game> are to support this type of communication with different objectives:

- The basic approach is to customize the game with data pulled from the LMS. As described in Section 3.3, the <e-Game> syntax allows the configuration of eGames from external data read by the engine from a configuration file. This data may be stored within the LMS instead and be pulled in real time with this mechanism.
- An <e-Game>-aware LMS could store data corresponding to preconditions of a specific eGame. The potential of such a mechanism is almost unlimited: Since preconditions are the basic utility for game flow, it would be possible to configure the entire eGame from the LMS itself. It would be possible to adapt the game to different learner profiles (since the preconditions can open and close game branches), to save the game in the LMS itself (since preconditions are the current state of the game) and even to allow the result of previous LOs and learning paths to affect the behaviour of the eGame.

Acknowledgements

The Projects TIN2004-08367-C02-02 and TIN2005 08788 C04-01 and the Regional Government of Madrid (4155/2005) have partially supported this work.

References

1. ADL. Advanced Distributed Learning Sharable Content Object Reference Model (ADL-SCORM). 2006; Available from: http://www.adlnet.org/ 20th January, 2006.
2. Alavi, M., Computer-Mediated Collaborative Learning: An Empirical Evaluation. Management Information Systems Quarterly. Vol. 18(2). (1994) 150-174.

3. Amory, A., Naicker, K., Vincent, J., and Adams, C., The Use of Computer Games as an Educational Tool: Identification of Appropriate Game Types and Game Elements. British Journal of Educational Technology. Vol. 30(4). (1999) 311-321.
4. Birbeck, M. et al, Professional XML 2nd Edition. Wrox Press.(2001).
5. Coombs, J.H., Renear, A.H., and DeRose, S.J., Markup Systems and the Future of Scholarly Text Processing. Communications of the ACM. Vol. 30(11). (1987) 933-947.
6. Dougiamas, M. and Taylor, P. Moodle: Using Learning Communities to Create an Open Source Course Management System, in World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003. Honolulu, Hawaii, USA: AACE (2003).
7. ESA, E.S.A. Essential Facts about the Computer and Videogame Industry. 2005; Available from: http://www.theesa.com/files/2005EssentialFacts.pdf 20th February, 2006.
8. Farmer, J. and Dolphin, I. Sakai: eLearning and More, in 11th European Univeristy Information Systems (EUNIS 2005). Manchester, UK (2005).
9. Garris, R., Ahlers, R., and Driskell, J.E., Games, Motivation and Learning: A Research and Practice Model. Simulation & Gaming. Vol. 33(4). (2002) 441-467.
10. Gee, J.P., What video games have to teach us about learning and literacy. New York; Basingstoke: Palgrave Macmillan.(2003) 225 p.
11. Goldberg, M.W. and Salari, S. An Update on WebCT (World-Wide-Web Course Tools) - a Tool for the Creation of Sophisticated Web-Based Learning Environments, in NAUWeb '97 - Current Practices in Web-Based Course Development. Flagstaff, United States (1997).
12. Goldfarb, C.F., A Generalized Approach to Document Markup. ACM SIGPLAN Notices. Vol. 16(6). (1981) 68-73.
13. Heathcote, E. and Dawson, S. Data Mining for Evaluation, Benchmarking and Reflective Practice in a LMS, in E-Learn 2005: World conference on E-learning in corporate, government, healthcare & higher education. Vancouver, Canada (2005).
14. Huang, H.M. and Wu, C.H., Applying Data Mining in E-learning Environment. Learning Technology Newsletter. Vol. 6(2). (2004) 31-32.
15. .IMS. IMS Content Packaging Specification v1.1.4. 2004; Available from: http://www.imsproject.org/content/packaging/ March 17th, 2005.
16. Ju, E. and Wagner, C., Personal computer adventure games: Their structure, principles and applicability for training. The Database for Advances in Information Systems. Vol. 28(2). (1997) 78-92.
17. Keller, C.W., Role Playing and Simulation in History Classes. The History Teacher. Vol. 8(4). (1975) 573-581.
18. Koper, R., Combining re-usable learning resources and services to pedagogical purposeful units of learning, in Reusing Online Resources: A Sustainable Approach to eLearning, A. Littlejohn, Editor., Kogan Page: London. p. 46-59 (2003).
19. Martinez-Ortiz, I., Moreno-Ger, P., Sierra, J.L., and Fernández-Manjón, B. Production and Maintenance of Content Intensive Videogames: A Document-Oriented Approach, in International Conference on Information Technology: New Generations (ITNG 2006). Las Vegas, NV, USA: IEEE Society Press (2006).
20. Moreno-Ger, P., Martinez-Ortiz, I., and Fernández-Manjón, B. The <e-Game> project: Facilitating the Development of Educational Adventure Games, in Cognition and Exploratory Learning in the Digital age (CELDA 2005). Porto, Portugal: IADIS (2005).
21. Oz, E. and White, L.D., Multimedia for better training. Journal of Systems Management. Vol. 44(5). (1993) 34-38.

22. Pivec, M. and Dziabenko, O., Game-Based Learning in Univeristies and Lifelong Learning: "Unigame: Social Skills and Knowledge Training" Game Concept. Journal of Universal Computer Science. Vol. 10(1). (2004) 4-12.
23. Prensky, M., Digital Game Based Learning. New York: McGraw-Hill.(2001).
24. Richards, T., IEEE Standard for Learning Technology - ECMAScript API for Content to Runtime Services Communication. (2004).
25. Richards, T., IEEE Standard for Learning Technology - Data Model for Content to Learning Management System Communication. (2003).
26. Schank, R.C., Learning via multimedia computers. Communications of the ACM. Vol. 36(5). (1993) 54-56.
27. Sierra, J.L., Fernández-Manjón, B., Fernández-Valmayor, A., and Navarro, A., Document Oriented Development of Content-Intensive Applications. International Journal of Software Engineering and Knowledge Engineering. Vol. 15(6). (2005) 975-993.
28. Sierra, J.L., Fernández-Valmayor, A., Fernández-Manjón, B., and Navarro, A., ADDS: A Document-Oriented Approach for Application Development. Journal of Universal Computer Science. Vol. 10(9). (2004) 1302-1324.
29. Sierra, J.L., Moreno-Ger, P., Martinez-Ortiz, I., Lopez-Moratalla, J., and Fernández-Manjón, B., Building learning management systems using IMS standards: Architecture of a manifest driven approach. Lecture Notes in Computer Science. Vol. 3583. (2005) 144-156.
30. Squire, K., Video games in education. International Journal of Intelligent Simulations and Gaming. Vol. 2(1). (2003) 49-62.
31. Squire, K. and Barab, S. Replaying History: Engaging Urban Underserved Students in Learning World History through Computer Simulation Games, in 6th International Conference of the Learning Sciences. Santa Monica, United States: Lawrence Erlbaum Assoc. (2004).
32. Young, M. An ecological description of videogames in education, in Annual Conference on Education and Information Systems: Technologies and Applications. (2004).