Production and Maintenance of Content-Intensive Videogames: A Document-Oriented Approach

Iván Martínez-Ortiz[#], Pablo Moreno-Ger[*], José Luis Sierra[*], Baltasar Fernández-Manjón[*]
(#) *Centro de Estudios Superiores Felipe II, Aranjuez, Madrid, Spain.*
(*) *Dpto. Sistemas Informáticos y Programación. Fac. Informática. Universidad Complutense, Madrid, Spain.*

Abstract

<e-Game> is a tool for the rapid development of adventure videogames with an educational purpose. It provides a markup language (the <e-Game> language) for structuring documents containing storyboards, and a processor (the <e-Game> engine) for executing games from these marked documents. This paper describes how <e-Game> facilitates new development models for the production and maintenance of content-intensive applications with domain-specific markup languages by applying our ADDS approach (Approach for Document-oriented Development of Software)
Keywords: *Development Approach, Adventure Videogames, Documental Paradigm, Edutainment, XML*

1 Introduction

Videogame development has become a prominent field in software development. The average development project nowadays employs hundreds in personnel and has a budget of several million dollars [1]. That's the reason why it is imperative to apply sophisticated Software Engineering methodologies, and that is precisely what the videogame industry is successfully doing [5]. However, in genres such as graphic adventure games, what is attractive about the game lies more in the brilliance of the dialogues or the surprising situations than in the graphics or the technology. Typical Software Engineering techniques in the field of videogames do not necessarily confront the special needs of such content-rich games, because if the spotlight is on content instead of on technology, then the quality of the product depends on scriptwriters, not programmers.

Beyond the domain of videogames, the development of content-rich applications has found similar situations where it was imperative to have experts in the domain of the application involved in the development process. Several development processes have been proposed but we want to focus our attention on a specific methodology: the application of the *Documental Paradigm*. In the documental paradigm, developers and domain experts agree on the format of a set of documents which will describe the main features of the application (e.g. contents, relevant properties of the user interface, etc.). In addition, they agree on a descriptive markup language specifically designed for describing the relevant structure of these documents: the Domain-Specific Markup Language (DSML) [8][10]. Once there is an agreement on the document types and on the markup language, the domain experts focus their effort on writing the contents of the application in documents following the specified language. On the other hand, developers prepare a processor for the DSML that receives the marked documents and runs the application.

However, the documental paradigm is a generic approach. We would like to focus our attention on the specific process suggested by the ADDS approach (Approach to Document-based Development of Software). This approach, which is described in more detail below, suggests the necessary processes, tools and interactions needed to successfully apply the documental paradigm from a variety of perspectives.

This work shows how the ADDS model can be applied to the production and maintenance of graphic adventure games using the <e-Game> tool, which was devised according to the documental paradigm. The paper is structured as follows. Section 2 summarized the ADDS approach. Section 3 describes how ADDS has been applied in the <e-Game> context. Section 4 outlines the main features of <e-Game>: the <e-Game>

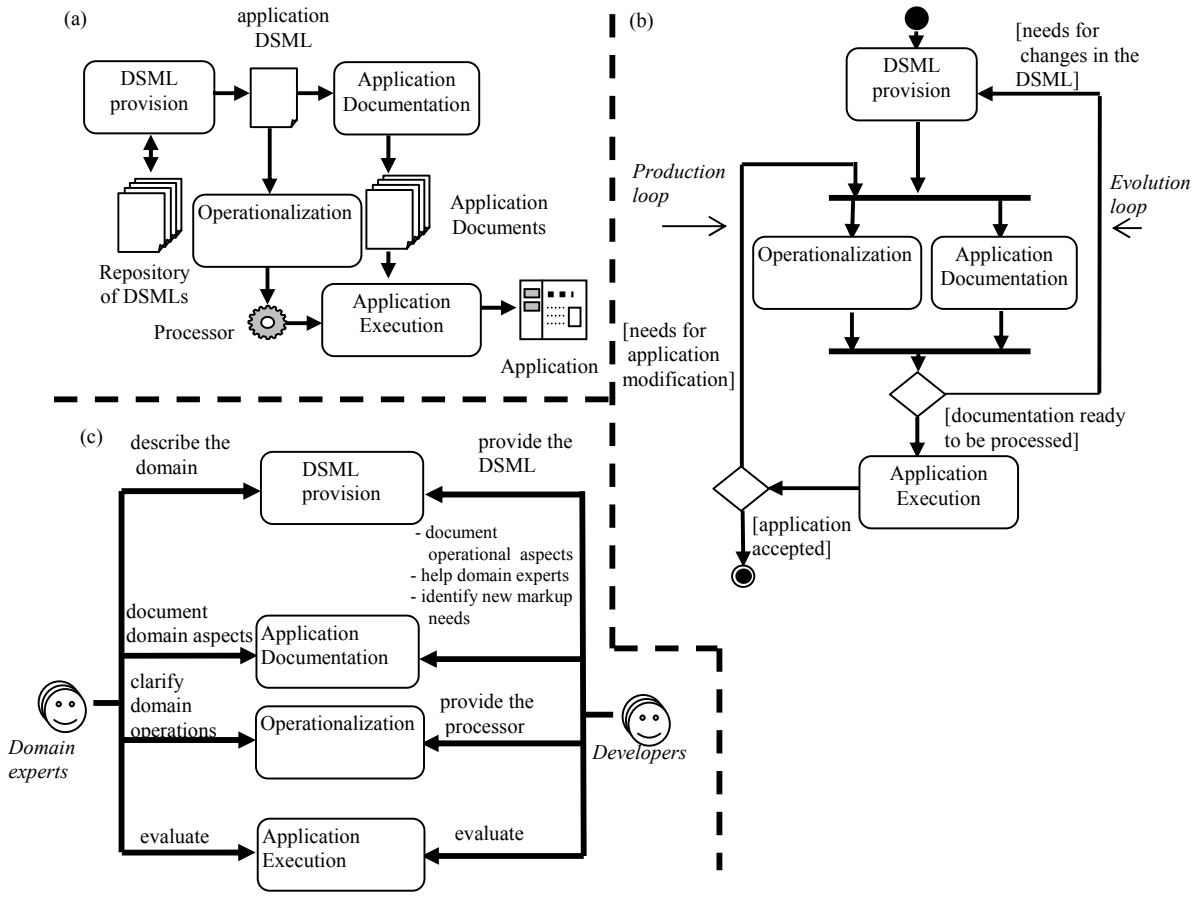

Fig 1. The three views of ADDS: (a) products and activities, (b) sequencing of the activities, (c) participants and their responsibilities in each activity

language and the <e-Game> engine. Finally, section 5 gives some conclusions and lines of future work.

2 The ADDS Approach

This section summarizes the main aspects of the ADDS approach, which is described with more detail in [8][10]. As outlined in Fig 1, ADDS comprises three different views, which are briefly presented in the following subsections.

2.1 Products and activities

The *products and activities* view shows the activities in the approach together with the products produced and consumed by these activities (Fig 1a):

- During the *DSML provision* activity, a grammatical characterization of an *application DSML* is produced. The cost of this activity can be decreased by reusing DSMLs pre-stored in a *repository of DSMLs*.
- During the *Documentation* activity, the main features of the application will be described in documents and the application DSML will be used to make the structure of the resulting *application documents* explicit.
- During the *Operationalization* activity a suitable *processor* for the application DSML will be built.
- During the *Application Execution* activity the application itself is produced by using the processor on the marked application documents.

2.2 Sequencing

The *sequencing* view shows how the activities are sequenced (Fig 1b). This view produces two different cycles or loops. On the one hand, in the *production loop* the application's features are documented, those documents are marked up, and the running application is automatically generated from these marked

documents. On the other hand, the *evolution loop* is initiated when new markup needs not covered by the current DSML are discovered. This leads to an evolution of the DSML to incorporate these needs, which implies an evolution of the corresponding processor. ADDS encourages the use of suitable mechanisms to manage the evolution of the languages and their processors. In [9] two of these mechanisms are described.

2.3 Participants and responsibilities

The *participants and responsibilities* view outlines the participants in the activities along with their responsibilities (Fig 1c). The *domain experts* are experts on the different aspects of the application's problem domain. In turn, the *developers* are experts in computer science. The responsibilities of these two participants in the different activities are as follows:

- During DSML provision, domain experts describe the application domain to developers, who formulate the grammar for the application DSML.
- During Documentation, domain experts document and mark up the application's features assisted by the developers, who also detect new markup needs and initiate new iterations of the evolution loop when required.
- During Operationalization, the main responsibility is for developers, who build the processor for the DSML with the assistance of experts regarding the nature of the domain operations.
- Finally, during Application execution both domain experts and developers evaluate the application generated with respect to functional and other non-functional (e.g. performance) requirements, initiating new iterations of the production loop when needed.

3 Applying ADDS to the Domain of Adventure Videogames: the <e-Game> project

The main goal of the <e-Game> project is to provide an effective method for developing educational games [4]. If we want our videogames to be truly educational, it is necessary to have pedagogues play a key role in the development process. Therefore, we need to facilitate the potentially difficult communication and collaboration between experts and developers. In <e-Game> we propose to use ADDS to organize these interactions.

The ADDS approach would allow a teacher to *specify* the contents of an educational game without requiring practically any knowledge in Computer Science beyond basic use of XML-based markup languages and some exposure to the particular DSML provided. The teacher writes the documents that define the game and embeds the educational content there. These documents are then fed to the engine which produces and runs the videogame.

Instead of applying this model to the entire domain of educational videogames, the specific genre of graphic adventure games was chosen. This was a design decision, based on a number of reasons:

- A DSML for the broad domain of videogames was not a feasible solution given the diversity of genres and game styles. Therefore we needed to narrow the domain to specific genres.
- Graphic adventure games are the most content-rich genre in videogames. Their success relies strongly on their scripts and storyboards, and these are a perfect starting point for the documents required to drive the development process.
- The bias for content instead of action made this genre a perfect solution for our educational objectives.
- The genre lived a golden age in the 90s with many successful and very similar titles published by Sierra™ and LucasArts™. These titles set the foundations of the genre and many of their traits are nowadays considered commonalities and expected in any videogame of the genre. All these commonalities can be assumed and thus abstracted from the DSML.

The remainder of this section will discuss the details of the application of the ADDS model in <e-Game>

3.1 Products and activities in <e-Game>

The particularization of the products and activities view to the <e-Game> project is depicted in Fig 2.

In the initial *DSML provision* activity, the first draft of the <e-Game> DSML was produced. Starting with previous experience in the genre of adventure games, the focus was on creating a descriptive language rather than an operational one. This decision was based on the notion that the authors (probably teachers) would find this approach easier than the more programming-like operational approach. The DSML grammar was formalized on a DTD and a XML Schema [12]. Although the DTD is simpler and more compatible with several tools, some of the aspects and restrictions of the DSML can only be reflected in the XML

Schema.

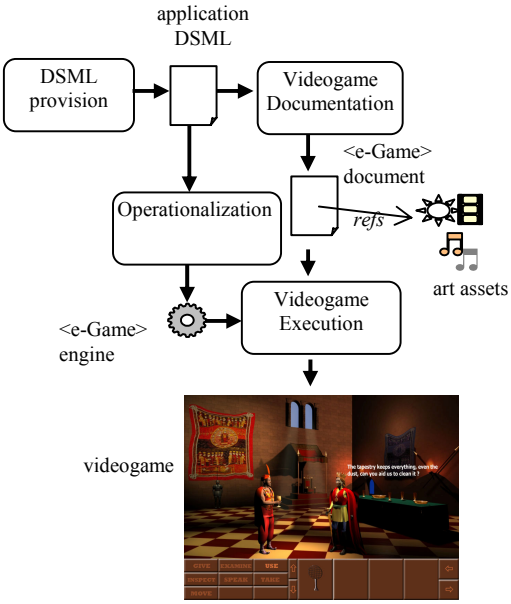

Fig 2. Products and activities in <e-Game>

The Operationalization activity involves creating a processor for the <e-Game> DSML called the <e-Game> engine. This processor was built using the Java platform due to portability concerns. The engine reads documents describing adventure videogames and produces the final videogame.

The Documentation activity (which in this project is called the *Videogame Documentation* activity) is driven by the production of the game's storyboard. As for products, the <e-Game> documents created by the authors must also be complemented with a number of external art assets (music, background images, character animations). These assets are referenced in the documents, but can be developed separately. This activity is further refined during the production loop, as indicated in the next subsection.

During the Application Execution activity (which in this context is called *Videogame Execution*) the games documented are run by the <e-Game> engine.

3.2 Sequencing in <e-Game>

The production loop of <e-Game> is quite straightforward and clearly defines the development process for this kind of games (Fig 3):

- The author informally writes a storyboard for the adventure desired. This storyboard would include descriptions of the different rooms and its contents. These contents are objects and characters, and they should be described too. The storyboard should also include the possible interactions with these characters and objects (combination of objects, using an object with another, conversations with characters, etc.).

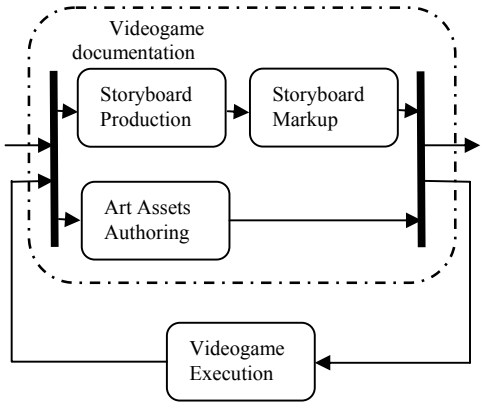

Fig 3. The production loop in <e-Game>

- The author marks up the storyboard. The DSML includes markup for scenarios, characters, objects, interactions and conversations. Thus, marking the storyboard should be a relatively simple step.
- The artists provide art assets for the scenarios, objects and characters. These assets include background images, images for objects, animations for characters and different sound effects (or background music). The approach permits the independent development of these assets, so they can be acquired from a repository of artistic content instead. Otherwise, authors and artist can work in parallel.
- The storyboard marked, along with the above-mentioned assets, are fed to the engine in order to produce the final game.

These steps can be iterated until the resulting videogame is satisfactory and the author considers it finished.

As for the *evolution loop,* the project is still on the first iteration and the possible evolutions of the DSML and the engine will be studied after some feedback is gathered from the initial game developments. However, the engine was designed with such evolutions in mind.

3.3 Participants and responsibilities in <e-Game>

The participants in the <e-Game> project can be divided roughly into three categories (Fig 4): *programmers*, *authors* and *artists*.

The *programmers* are the developers responsible for the initial provision of the <e-Game> DSML. That

initial version is accompanied by the corresponding version of the engine.

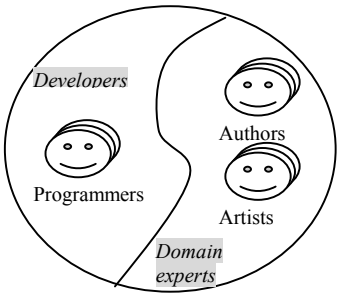

Fig 4. Participants in <e-Game>

The *authors* are the domain experts who mainly use the tools provided by the <e-Game> project. From the original perspective, the authors are usually teachers or experts in a specific domain, but, actually, <e-Game> can be used to develop any kind of adventure game. Thus, the author can be any person interested in developing his/her own adventure games without needing prior programming skills. The author is responsible for writing the storyboard and marking it using the grammar of the <e-Game> DSML.

Finally, the *artists* are the experts who create the different assets that embellish the game and make it attractive. These assets can be prepared with external authoring tools.

4 The <e-Game> features

All this development process is supported by two main entities, which were identified as products in the *products and activities* view: The <e-Game> language and the <e-Game> engine. The following subsections describe these features.

4.1 The <e-Game> language

The language was proposed with two main objectives in mind: the syntax should be as simple as possible and the markup philosophy should be as close to a typical storyboard as possible.

Since the main entities of adventure videogames are the scenarios, characters, objects and actions, these are the main XML elements suggested in the language. The elements are defined separately and linked, using XML's built-in identifier-reference mechanism (Fig 5)

```
<scenario id="SCE-1">
  <character-ref idRef="CHA-1">
    <position>...</position>
      ...
  </character>
    ...
</scenario>
<character id="CHA-1">
  <conversation-ref idRef="CON-1" />
    ...
</character>
<conversation id="CON-1">
    ...
</conversation>
```

Fig 5. A simplified XML excerpt showing the basic structure of the language. Conversation CON-1 is triggered when speaking with character CHA-1 who appears in scenario SCE-1

A *scenario* contains references to the corresponding art assets (background image, background music and hardness map) and a number of exits that connect it to the other scenarios of the game. In addition, it will contain references to the identifiers of the characters and objects that populate the scenario.

Correspondingly, *objects* and *characters* will have a unique identifier and will contain references to the corresponding art assets (sequences of images for the animation). In addition, each character or object will contain a number of references to the identifiers of the possible interactions which are related to them.

Finally, the interactions must have a unique identifier and the description of the actions and the outcomes.

4.2 The <e-Game> engine

The <e-Game> engine is used to execute the games from their descriptive <e-Game> documents. This processor was designed to facilitate its maintenance and evolution. As depicted in Fig 6, it is architected in terms of:

- A *tree builder*. This artefact is based on a standard DOM parser [12], and it builds a tree representation of the input <e-Game> document.
- A *component repository*. This repository contains a set of *game components*, whose component model is an evolution of that described in [7], and which are assembled in the production of videogames.
- A *game generator*, which is the core of the engine and which processes the document tree to assemble the game components. This artefact is architected according with the operationalization model described in [9].

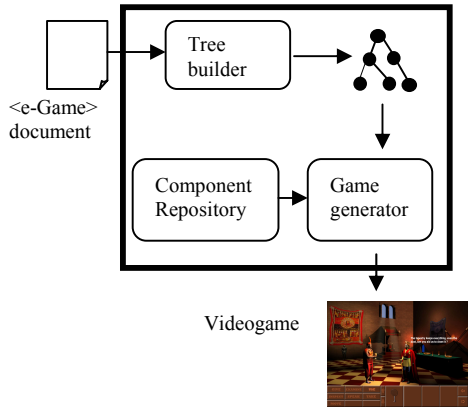

Fig 6. High-level architecture of the <e-Game> engine

5 Conclusions and Future Work

<e-Game> promotes a document-oriented approach to enhance the production and maintenance of adventure videogames with an educational purpose. The document-oriented principles followed in <e-Game> make it a more usable tool for authors without a deep knowledge of computer science than other programming-based technologies for game development (e.g. [2]).

The current state of <e-Game> allows any teacher to develop relatively simple adventure games provided that he/she has access to art assets (either by contacting an artist or by accessing a repository) and that he/she receives a certain exposure to XML in general and the <e-Game> language in particular.

In the near future, it will be interesting to explore the potential of this methodology in fields outside education where rapid development of simple games is required. In particular, the field of casual gaming [3] can benefit from this approach.

Since adventure games are as engaging as their scripts, without any need for stunning graphics and technology, we also envision a great potential in the development of adventure games for devices of limited capability, like PDAs or mobile phones [6] .

Finally, it is important to point out the growing interest in games whose purpose is not to entertain, but to transmit political ideas, propaganda and even product advertisement [11]. The simplicity of <e-Game> allows any person or institution to develop a small adventure game with a low cost with the sole purpose of transmitting one of such ideas.

Acknowledgements

The Spanish Committee of Science and Technology (TIC2002-04067-C03-02 and TIN2004-08367-C02-02) has partially supported this work. The Spanish National Center of Information and Educative Communication (CNICE) provided the art assets displayed in the examples.

References

[1] Entertainment Software Association, Sales & Genre Data. http://www.theesa.com/facts/sales_genre_data.php (last visited 21 October 2005)

[2] Harbour, J and Smith, J. *Beginner's Guide to DarkBasic Game Programming*. Premier Press. 2003

[3] Mills, G. "Casual Games". *International Game Developers Association White Paper*, 2005. Available at http://www.igda.org/casual/IGDA_CasualGames_Whitepaper_2005.pdf

[4] Moreno-Ger, P, Martínez-Ortiz, I, and Fernández-Manjón, B. "The <e-Game> Project: Facilitating the Development of Educational Adventure Games". *Int. Conference on Cognition and Exploratory Learning in the Digital Age* CELDA'05, Porto, Portugal, December 14-16, 2005

[5] Rucker, R. *Software Engineering and computer games*. Addison-Wesley. 2002

[6] SCUMM Virtual Machine, PocketPC version, available at http://www.scummvm.com, September 2005

[7] Sierra, J. L, Fernández-Valmayor, A, Fernández-Manjón, B, and Navarro, A. "Developing Content-Intensive Applications with XML Documents, Document Transformations and Software Components". *31th Euromicro Conference on Software Engineering and Advanced Applications*, Porto, Portugal, August 31 – September 2, 2005

[8] Sierra, J. L, Fernández-Manjón, B, Fernández-Valmayor, A, and Navarro, A. "Document-oriented software construction based on domain-specific markup languages". *Int. Conf. on Information Technology: Computing and Coding* ITCC'05, Las Vegas, USA, April 4-6, 2005

[9] Sierra, J. L, Navarro, A, Fernández-Manjón, B, and Fernández-Valmayor, A. "Incremental Definition and Operationalization of Domain-Specific Markup Languages in ADDS". *ACM SIGPLAN Notices*, in press

[10] Sierra, J. L. Fernández-Valmayor, B. Fernández-Manjón, and A. Navarro. "ADDS: A Document-Oriented Approach for Application Development". *Journal of Universal Computer Science* 10(9), 2004, pp. 1302-1324

[11] Water Cooler Games web site. http://www.watercoolergames.org/ (last visited 21 October 2005)

[12] World wide web consortium technical reports. www.w3.org (last visited 21 October 2005)