A Model Driven Serious Games Development Approach for Game-based Learning

Stephen Tang and Martin Hanneghan
School of Computing and Mathematical Sciences,
Liverpool John Moores University
Byrom Street, L3 3AF Liverpool, UNITED KINGDOM.
Email: {tang, m.hanneghan}@ljmu.ac.uk

Abstract - Computer games, predominantly a form of interactive entertainment, are having some success being repurposed for educational use. However, this approach is hindered by the lack of availability of experience in serious games tools. Much research is already underway to address this challenge, with some who choose to use readily available commercial-off-the-shelf games and others attempted to develop serious games in-house or collaboratively with industry expertise. These approaches present issues including educational appropriateness of the serious game content and its activities, reliability of serious games developed and the (often high) financial cost involved. Developments in software engineering that enable automatic generation of software artefacts through modelling or Model Driven Engineering (MDE) promises new hope for game-based learning adopters, especially those with little or no technical knowledge, to produce their own serious games for use in game-based learning. In this article, we present our model-driven approach to aid non-technical domain experts in serious games production for use in games-based learning.

Keywords: Model Driven Serious Games Development, Model Driven Engineering, Model Driven Development, Serious Games, Game-Based Learning.

1 Introduction

Game-based learning (GBL) refers to both the innovative learning approach derived from the use of computer games that possess educational value and other software applications that use games for learning and education purposes (e.g. learning support; teaching enhancement; assessment and evaluation of learners etc.) [1]. These computer games are also referred to as educational games or in a more popular term known as serious games. As computer gaming becomes a digital culture deeply rooted amongst the new generation of learners, many educational researchers and practitioners agree that it is now appropriate to exploit gaming technologies in order to create engaging interactive learning content to motivate learners to learn through game-playing using the GBL approach [2].

The preliminary results of GBL have shown some positive impact towards students’ learning [3]. However, the adoption rate of GBL still remains low. One of the barriers to the adoption of GBL is the extremely steep learning curve required to create serious games. Most of the computer games available in the market are designed for entertainment purposes and the majority of content is not fit for educational purposes. This has led some domain experts to create serious games through bespoke in-house development, using open source or royalty-free game engines in collaboration with a team of developers and ‘modding’ (or modifying) commercial-off-the-shelf games by utilising a game editor application. Many of these tools and technology platforms for producing serious games are readily available but most of these tools require substantial technical knowledge in games development which hinders non-technical domain experts from adopting games-based learning. We believe by addressing the absence of high-level authoring environments and support for non-technical domain experts (i.e. teachers) to create custom serious games will be a major factor in the rise of serious games.

Advancements in software engineering are making the creation of high-level serious games authoring environments for non-technical domain experts viable. The MDE (Model Driven Engineering) approach uses abstract models to formally represent aspects of serious games software which is then automatically transformed into more refined software artefacts and subsequently into serious game software applications. This approach provides non-technical domain experts the tools to produce serious games easily and quickly (and possibly at a lower cost) through the use of Domain Specific Modelling Languages (DSML). This therefore lowers the barriers that hinder the production of these applications. MDE offers an increase in productivity, promotion of interoperability and portability among different technology platforms, support for generation of documentation, and easier software maintenance [4]. In addition, it can produce better code quality and improves reliability of the code [5]. From the non-technical domain experts’ perspective, an MDE’s ability to encapsulate the technical aspects of development via a DSML massively lowers the barriers that hinder the production of applications. We believe by marrying games development and the MDE approach, we can provide a technological solution to the aforementioned issues.

In this paper, we present our model-driven serious games development approach. In Section 2, we briefly introduce our model-driven serious games development framework and the underlying models designed to formally represent serious
games. We then explain our model-driven approach and describe our model driven technologies in Section 3. In Section 4, we present a case study to demonstrate the design and production of a serious game using our model-driven approach and follow with a discussion in Section 5. Finally, we draw conclusions on the future of this exciting field of research in Section 6.

2 Model Driven Serious Games Development Framework

Our novel model-driven serious game framework (see Figure 1) is designed to support the production of a variant of serious game software that covers a wide range of technology platforms as well as operating platforms. It consists of nine parts namely: (1) User Interface (UI), (2) Models, (3) MDE Tools, (4) Components Library, (5) Code Templates, (6) Artefacts, (7) Technology Platform, (8) Operating Platform and (9) Software. This configuration loosely couples the modules allowing the framework developer to flexibly substitute modules while maintaining the integrity of relationships among the modules via well-defined interfaces. It also clearly divides the views of entities while promoting structured and systematic workflow [6]. At the core of the framework are three models namely the Game Content Model, Game Technology Model and Game Software Model.

2.1 Game Content Model

Our novel Game Content Model improves on the existing work Game Ontology Project (GOP) [7], Rapid Analysis Method (RAM) [8] and Narrative, Entertainment, Simulation and Interaction (NESI) [9]. It combines our study on game design, game development and serious games with a selection of concepts from the aforementioned existing works to form a robust formal model. Our Game Content Model covers all the essential game design concepts for documenting serious game design in the role-playing and simulation genres initially (but this can be easily expanded upon to support other genres). The top level of our game content model consists of ten interrelated key concepts that best represent the rules, play and aesthetic information of a computer game. These are Game Structure, Game Presentation, Game Simulation, Game Rules, Game Scenario, Game Event, Game Objective, Game Object, Game Player and Game Theme [10].

In brief, the game structure provides the form and organises the game into segments of linked game presentations and game simulations. The interactions between a game object and the results of an interaction in a game simulation are defined using game rules. A game simulation can be used to host multiple game scenarios aligned with the storyline. Each game scenario is set up using a selection of game objects to create an environment, a sequence of game events and a set of game objectives that challenges the game player’s skills and knowledge about the game domain. The game player can control game object(s) and interact with others via hardware controllers or a graphical user interface. And finally, the game theme describes the “look and feel” of the game. Detail on our Game Content Model is available in our previous work in [10].

2.2 Game Technology Model

Our Game Technology Model is based on a data-driven architecture and includes the essential game specific systems and core components of software which facilitates the operation of serious games. The Game Context System handles the transitions between contexts and works cooperatively with the Game Simulation System to simulate a scenario. For ease of processing, a scene graph is used to organise renderable and updatable objects such as media components, GUI components, front-end display components, game objects and lights. These objects are processed using
the platform independent core components such as renderer, animation, audio, input, game physics, user interfaces, video player, game resource manager, networking and artificial intelligence [11].

Supporting these core components are the helper components namely the math library, random number generator and unique object identifier management. The functionality of each component defined in the Game Technology Model are specified as interfaces that each wrap a different implementation of a game technology. This allows serious games software to be produced on different technology platforms through code generation which reads the Game Technology Model and translates it into software artefacts. More details on our Game Technology Model can be found in [11].

### 2.3 Game Software Model

Our Game Software Model is the platform specific software representation of the game software. The Game Software Model is designed by a technical person who possesses great understanding of the model driven framework and the technology platform. Developers of the Game Software Model may choose one of the two different perspectives; (1) to bridge the Game Technology Model to an existing game software framework (e.g. Microsoft’s DirectX) or (2) to implement game software from scratch for an intended technology platform. Both these exercises may require platform specific details or components to be added which has been omitted in the Game Technology Model.

Implementing the Game Software Model for an existing game software platform will require framework developers to map the components presented in the Game Technology Model to the chosen game software framework. Often it is likely to achieve a one-to-one mapping of Game Technology Model components with game software framework components with possible inclusion of information that is required by the game software framework. Figure 2 illustrates the overview of Game Software Model with component wrappers (shaded in grey colour) which map components of Game Technology Model to the appropriate component in a game software framework.

Unlike the former approach described above, designing a Game Software Model for a specific software technology platform will require the addition of certain platform specific components which are used by the core components. These are identified by Gregory [12] as window management, file system, timer, graphics wrapper and physics wrapper. Most game software frameworks would have these platform specific components implemented in low-level code that is coupled to a specific operating platform to ensure it delivers the performance required of the game software. In our approach, these platform specific components have been omitted from the Game Technology Model and are only added in the final stage of the model transformation to achieve true operating platform independence. Implementers of our Game Software Model will have to define these platform specific components so they can be mapped to the right implementation during the generation of program code. This makes the Game Software Model structure differ from the earlier version described above as the component wrapper is replaced with platform specific components (shaded grey in Figure 3).

### 3 Model Driven Serious Games Development Approach

Our model-driven serious game development approach, based on our framework presented in Section II, is made-up of a modelling environment, model translators and an artefacts generator. An overview of this is shown in Figure 4. In the following subsections, we briefly discuss the implementation of our model-driven approach.
3.1 Serious Games Modelling Environment (SeGMEnt)

Our modelling environment, referred to as Serious Games Modelling Environment (SeGMEnt), is designed to allow non-technical domain experts to model a serious game using both graphical notations and step-by-step guides. We have chosen to develop a web-based modelling environment due to the wide-access the web can offer to the game-based learning community. This approach can also lower the barrier of entry for adoption of game-based learning as practitioners don’t necessarily require a high-performance multimedia computer to produce serious games. We have chosen the Adobe Flash platform as our initial development platform of the modelling environment as it has a rich range of facilities to support our requirements. Non-technical domain experts will use this modelling tool to author serious game content, collating the art assets and defining the necessary game mechanics that make up the serious game.

We have implemented five unique user interface (UI) components that can assist non-technical domain experts when modelling the aspects serious games design in the SeGMEnt modelling environment. These UI components are:

- **Flow visualisation** – this provides visual tools to represent the flow of states within the serious game using a state diagram notation that has been extended to include additional information;
- **Dynamic option interface** – this is a list of options generated from data fetched from within SeGMEnt to simplify the data entry process and to prevent error in data entry;
- **What-you-see-is-what-you-get (WYSIWYG) visualisation** – this aids users to visually position media, graphical user interface (GUI), front-end display (FED) and game objects strategically on a 2D space through drag-and-drop interaction and edit properties of media, GUI, FED and game objects via a simple data entry interface;
- **Statement construction interface** – this guides users in constructing an acting statement, which is an English-like sentence that defines a game act, by selecting the verb, noun or conjunction from the options presented in dynamic option interface; and
- **Guided data entry interface** – this provides a step-by-step guide for users to document aspects of serious games systematically via common GUI components to avoid overloading requests for information from users.
Our SeGMEnt tool consist of seven different viewpoints designed to separate ten aspects of serious games design described in our Game Content Model into smaller and more manageable clusters of data. Each of the viewpoints uses a combination of UI components described earlier to aid users in visualising aspects of serious game design. These design viewpoints are:

- **Game structure designer** which allows users to model the flow and the structure of a serious game (see Figure 5);
- **Game scenario designer** where users specify the flow of events in a scenario through the definition of game scenario, game events and game objectives;
- **Game object designer** which provides the interface to define a game object’s identity, specify the associated attributes, assign an appearance, define actions and define associated intelligence; **Game simulation designer** which provides the facilities for users to specify game simulation concepts by adding and positioning FEDs on the virtual canvas and defining the game tempo and physics via a data entry interface;
- **Game presentation designer** which offers the viewpoint for users to model off-game user interfaces such as a menu screen, a screen that presents the game player with the game objectives and a screen that presents the victory or loss results to the game player;
- **Game environment designer** where users model a game environment in which a given scenario takes place by strategically positioning game objects, proximity triggers and checkpoints to construct the environment in which the game-play will be set and place virtual cameras visually on a 2D space; and
- **Game player designer** which provides the viewpoint where users can specify the player’s avatar, the inventory size which limits the storage of virtual items, the game attributes associated, the data to be tracked and the mapping of game controls to a game object’s action.

Our SeGMEnt tool only serves as an interface aid for domain experts to document the design of a serious game that is compliant with our Game Content Model. Underlying the UI is the data which needs to be saved, processed and exported into eXtensible Markup Language (XML) format which are then transformed into more refined models using our MDE tools.

### 3.2 Model representation

In our model driven approach we have chosen to use XML as the format for representing our models. This offers great flexibility for defining the data format for representing models. In addition, XML can easily accept additional information from the automated transformation process between the models for MDE. Furthermore it is also well supported by MDE technologies such as Eclipse Modelling Framework (EMF) and Generic Modelling Environment (GME) [13] making it the ideal choice for representing our data-model.

### 3.3 Model Translation and Code Generation

The Game Content Model generated by our SeGMEnt tool needs to undergo a transformation process to be translated into the Game Technology Model (a computational model independent of platform) using an MDE tool. The MDE tool can be developed using existing MDE technologies such as EMF and GME as described in [14] or implemented using any programming language with XML parsing capability.

In our model-driven serious games development framework, we have developed custom transformation and generation tools in PHP. The transformation from Game Content Model to Game Technology is mainly a process of refining data and reformatting it into a computation...
independent model by reorganisation of data into programmatic structures. This also involves the addition of programmatic statement calls to the relevant Game Technology Model component’s function to process the relevant data. The transformation from Game Technology Model to Game Software Model further refines the data by adding in platform specific data.

Our implementation follows a simple approach by traversing through the entire source model to locate the required token of information (XML element) and a new target model is composed by structurally reformatting data in the source model and adding in the additional information. This approach does not constrain us to the structure of the source model. A similar approach is used to transform the Game Technology Model to the Game Software Model by appending additional tokens of information that mark the interfaces of the components to be included to enable code pairing during the software artefact generation process.

Our basic code generation tool is also implemented using an approach similar to that described for the transformation tool. Each of the marked tokens are located and then translated into code for a specific platform. Each token of information either maps to a single line of code or a segment of codes defined in some code template. The final code is built up based on the structure of Game Software Model and generated as a textual artefact.

4 Producing Serious Games for 
Games-based Learning using A 
Model-Driven Approach

The production of serious games for game-based learning is centred on design for learning where rules and game-play are design to support the defined learning objectives. In our previous work [15], we proposed an educational game design methodology consisting of thirteen activities that are grouped into three phases:

- **Planning Phase**: (1) Define learning objectives and design goals, (2) understand learners, (3) identify learning activities for learning objectives defined in activity 1, (4) Sequence learning activities in increasing complexity order, (5) design the story to set the scene and link learning activities defined in activity 3;
- **Prototyping Phase**: (6) design game mechanics for learning activities defined in activity 3, (7) Design game components and associated behaviours, (8) Design scenarios and game-play for learning activities defined in activity 3 using activity 4 and activity 5, (9) prototype game level, (10) evaluate prototype against learning objectives, (11) refine the game level;
- **Finalising Phase**: (12) finalise educational game and (13) quality assurance test on educational game.

The model-driven approach follows the same process where non-technical experts conduct the design activities 1 to 7 on paper. Once the details of the serious game have been decided, domain experts can then model the serious game in our SeGMEnt tool. Modelling in SeGMEnt follows a bottom-up approach in which it requires a modeller to model basic elements with a view to creating more complex composite elements from these basic elements later on. There are seven successive stages to follow when modelling a serious game in SeGMEnt (see Figure 6). The first stage involves modelling of game objects and this is done using the game object designer. Once all game objects have been defined the game environment can be set up using the game environment designer. This is followed by modelling of the game scenarios and definition of game rules and game objectives using the game scenario designer. The next stage involves the modelling of various game presentation contexts using the game presentation designer. Then the domain expert can model the game simulation parameters via the game simulation designer. The order of stage 4 and stage 5 can swapped as there is no precedence dependency. Once the game presentation context, game simulation context and game scenarios have been modelled, domain experts can now focus on modelling the game flow. The final stage of modelling of the serious game involves the definition of the game player via the game player designer.

### Stages of Modelling in SeGMEnt

| Stage 1 | Modelling game object |
| Stage 2 | Modelling game environment |
| Stage 3 | Modelling game scenario, game rules and game objectives |
| Stage 4 | Modelling game presentation |
| Stage 5 | Modelling game simulation |
| Stage 6 | Modelling game structure |
| Stage 7 | Modelling game player |

Figure 6: Stages of serious game modelling in SeGMEnt.

The transformation of models and generation of code in our model driven approach is automatic and can be initiated through the “Export” command in the SeGMEnt tool. This will first generate an XML file compliant with the Game Content Model by passing the data from SeGMEnt to Game Content Model Creator. After the file has been created, the Game Content Model Creator will pass the control to Game Technology Model Translator which will read the Game Content Model and transform it into a programmable format which is also in XML file format. Once the Game Content Model has been transformed to Game Technology Model, the control is then passed to the Game Software Model translator which will read the Game Technology Model and add in platform specific information to the Game Technology Model to form the Game Software Model. Once the transformation is complete, the control is then passed to the code generator which generates code in the form of text output presented on a web interface. At present, our code generator supports the generation of ActionScript 2.0 code for execution on the Adobe Flash platform.

5 Discussion

This model-driven approach changes how serious games are developed traditionally. Instead of developing software based on a set of given design requirements, our model-driven approach demands software developers to produce
assets and tools which non-technical domain expert can use to produce serious games without worrying the technical aspects of games development. The complexity of serious games development is now hidden behind the SeGMEnt tool and driven by the models and MDE tools that interprets and refine the models for generation of software artefacts.

Using the model-driven approach in serious games development does not imply that every aspect of serious games production is automated. Domain experts are expected to understand serious game design and required to adhere to the methodical approach of serious game modelling in SeGMEnt to ensure that the Game Content Model produced by experts using our SeGMEnt tool is valid. We acknowledge that serious games design is still a creative process and it demands specialised skill despite the tools being a guide and aid for non-technical domain experts. Therefore we cannot expect our model-driven approach to instantly transform a novice tutor to a serious game designer capable of designing interesting and creative problems for game players.

Serious games as real-time applications demand lag-free performance and generated codes can reduce opportunities for code optimisation in some instances. This can limit the level of complexity of a serious game and the amount of dynamic objects the software can process during runtime. Unlike generated code, manual hand-coding permits expert game developers to apply clever solutions to improve performance. We believe this trade-off in code performance is far outweighed by the fact that more and more non-technical game designers will be able to create and reuse serious game resources.

6 Conclusions

Game-based learning is a highly desirable learning approach for the “PlayStation-driven” generation of learners. However, it lacks technological solutions that can help non-technical domain experts to author custom interactive learning content to support this innovative learning approach. Our model-driven framework supports the development of serious games through the use of our MDE tools. The model-driven approach helps non-technical domain experts to produce serious games quickly, easily and affordably (in the long term). Our vision is for our model-driven serious game development framework and our model driven approach to serve as a basis for more implementation of high-level serious game authoring tools designed specifically for non-technical domain experts who wish to produce serious games. We look forward to a future where serious game artefacts are published freely and openly by non-technical practitioners around the world to build upon and improve opportunities for learning.

7 References


