Challenges in Designing Domain-Specific Modeling Language for Educational Games

Olga De Troyer and Elien Paret

Vrije Universiteit Brussel, Research Group WISE, Pleinlaan 2
1050 Brussel, Belgium
{Olga.DeTroyer, Elien.Paret}@vub.ac.be

Abstract. In this paper, we discuss the challenges related to the design of domain-specific modeling languages to design educational games. A domain-specific modeling language uses the vocabulary of the application domain and provides abstractions that make the specifications of solutions easier and more accessible for domain experts and end users. Although such an approach looks promising for involving domain experts and end users in the design process of educational games, several research questions still need to be solved.

Keywords: Domain-Specific Modeling Language, Educational Game, Domain Expert, End User.

1 Introduction

Educational games are games specifically designed to teach people about a specific subject or assist them in learning as they play. They include board and card games, as well as computer and video games. Our focus is on computer-based educational games. Computer-based (and especially video-based) educational games have historically received quite some criticism because they were often perceived as or associated with issues such as mindless entertainment, pre-school learning, or with violence (e.g., [1]). However, the educational value of computer-based educational games is now more and more recognized (e.g., [2, 3]). Children are growing up today in a society that is increasingly dependent and driven by digital technology. This has produced a new generation of learners (the so-called digital natives [4]) that are much more demanding than the previous generations. Games provide opportunities that traditional methods of teaching cannot offer, e.g., they can provide simulations that immerse the learner into different situations, or can incorporate social aspects. Games also allow incorporating implicit learning or learning-by-doing which entail that learners do not need to be consciously aware of their learning experiences in order to display learning behavior. Although the advantages of integrating games in education or use them for training have already been widely discussed in the literature, in practice, the design and development of an educational game is largely done in an ad hoc way and its success will mainly depend on the experience and insights of the developers involved. Therefore, it is important to think of design methods that could support the development of educational games. There are still a lot of open questions with regards to the development of educational games. For instance, when, for what and for who can games actually lead to increased learning [3]. It is also not yet clear how to combine games and learning. Should the game and the learning be completely integrated rather than one being just an add-on of the other? Should there be a relation between the learning topic (e.g., mathematics) and the goal of the game (e.g., escape from a castle)? How to find the right balance between entertainment and the learning experience [5]? To answer these and other related questions, interdisciplinary research is needed, involving people from domains such as pedagogy, cognitive psychology, and computer science. Experiments need to be set up to investigate the research questions and validate hypotheses. In these experiments, we also need to involve people from the domain for which the educational game will be developed (e.g., for archeology or history, or for societal problems like poverty or bullying) because their domain knowledge is needed during the development process. Also end users are important actors in these research experiments to provide essential input about requirements and the necessary feedback regarding usability. This requires tools and methods that can be easily used and understood by non-technical users (i.e. domain-experts and end users). Such tools are not only important in the context of research, but they will also be valuable (later on) in professional development environments, as it will always be essential to involve domain experts and end users in the development of a successful educational game.

In this paper, we will focus on the use of a domain-specific modeling language to accommodate domain experts and end users in the design process of educational games. Domain-specific modeling languages use a dedicated vocabulary and provide abstractions that make the specifications of solutions easier and more accessible for domain experts. More in particular, we will explore the challenges related to the development of such a language for educational games.

2 Educational Game Development

Educational games can be developed in different ways. Many tools exist that can be used for this purpose. We can make a distinction between programming libraries and authoring tools. With (graphical-oriented) programming libraries a complete application can be programmed from scratch. Examples of such programming libraries are Microsoft DirectX [6], OpenGL [7], and Java3D [8]. These libraries deal with almost all needed low-level functions like access to the graphical devices, sound cards, and input devices. They allow abstracting from the low-level features, however the application still needs to be programmed manually and they also don't provide abstraction mechanisms for features that are common to games, e.g., object behaviors or state transitions. Game engines can also be placed in the category of programming libraries, but compared to the generic libraries mentioned earlier they provide more game-oriented abstraction mechanisms (i.e. basic game functionality) for the programmer. Although they are used a lot for the development of game-oriented application, a major drawback is that the learning curve to master these tools is high and quite some programming skills are required. To be able to use them you need to understand the game architecture used, the interaction paradigm used, interfaces, etc.

Authoring tools, on the other hand, are aimed to make the development process more accessible to a broader public (e.g., Thinking Worlds [9]). These tools try to avoid programming by providing a graphical and easy-to-use interface (e.g., clicking, dragging and dropping). In this way, they try to deal with the "semantic gap" between the author of the game and the implementers who each talk their own language and have their own concerns. Actual, authoring tools have the (ultimate) aim of removing the need for implementers: authors (i.e. designers) can build their applications themselves. Although this sounds very expressive, in practice these tools are often limited either in the type of application that can be specified or in the support that they provide (and often they need to be combined with scripting languages, which again requires programming skills).

To try to overcome the "semantic gap" problem, some researchers [10, 11, 12, 13] propose to consider domain-specific modeling languages (DSML) to develop games. Also in our research, we follow this approach. However, DSMLs proposed for gaming are too limited for the purpose of educational games. A domain-specific modeling language for educational games should not only consider the gaming aspects but also take into consideration the learning aspects, as well as the (learning) domain of the educational game. It must be noted that the technique of domain-specific modeling languages have also been considered in the domain of instructional design [14] but also these languages are too restricted to serve our purpose.

3 Domain-Specific Modeling Languages

A domain-specific modeling language (DSML) [15] is a domain-specific language for the purpose of modeling or designing systems. A DSML can be considered as a special kind of domain-specific language (DSL). A DSL is a, usually small, language dedicated to, and restricted to, a particular domain and a specific class of problems [16]. It provides appropriated abstractions that make the specifications of solutions for this particular class of problems easier and less time consuming. The abstractions are using the vocabulary of the problem domain and as such domain experts can understand, validate, and often even develop themselves specifications expressed in the DSL. In general, modeling languages (and as such also domain-specific modeling languages) are graphical (visual) languages because graphical specifications are easier for the communication with non-technical people; they are also helpful for conveying complex models and designs as they can help people to grasp large amounts of information more quickly than large listings of text. Domain-specific modeling languages have already been proposed for several domains, from insurance products to microcontroller-based voice systems [15]. Luoma et al. [15] performed a study based on data gathered from over 20 cases of DSML creation. They conclude "the task of defining a language seems to become considerably easier when the language needs only to work for one problem domain in one company". The authors also found that "in all cases, DSM had a clear productivity influence due to its higher level of abstraction: it required less modeling work, which could often be carried out by personnel with little or no programming experience". The authors also reported that there was no single approach to construct a DSML; they identified 4 different approaches. van Deursen et al. [16] gives a number of important steps to follow when developing a DSL. A distinction is made between the analysis and design of the language (step (1) to (4)) and the implementation of the language (step (5) to (6)). The steps are: (1) Identify the problem domain; (2) Gather all relevant knowledge in this domain; (3) Cluster this knowledge in a handful of semantic notions and operations on them; (4) Design a DSL that concisely describes applications in the domain; (5) Construct a library that implements the semantic notions; (6) Design and implement a compiler that translates DSL programs to a sequence of library calls. Elliott [17] also emphasizes the importance of tools for the DSL.

4 DSMLs for Educational Games

When we want to develop a (graphical) DSML for the domain of educational games, our language will need to provide high-level modeling concepts (and associated graphical representations) for expressing easily: (1) game aspects; (2) pedagogical aspects; and (3) (learning) domain aspects.

Actually, it will not be possible to come up with one single DSML for educational games. First of all there are different types (genres) of games with different characteristics [18]: action and adventure games, strategy games, role-playing games, real world simulations, puzzle games…. According to Dobbe [11], the game genres do not influence the general structure of the DSL and he proposed a single DSL for all game genres. However, we think that it is better to provide some specific modeling concepts for the different genres as these may be easier to understand by the non-technical people than more generic modeling concepts like object, event, interaction. In addition, by using specific modeling concept for specific genres we may also provide better guidance to the designers, as the modeling concepts available will already suggest the type of information that need to be specified. The fact that all game genres have some common vocabulary can be solved by having a layered set of modeling concepts (e.g., modeled as a concept hierarchy), where layers are turned on and off depending on the game genre selected at design time.

Secondly, there exist different learning and teaching styles and strategies. A discussion on different learning and teaching styles is outside the scope of the paper (see e.g., [20, 21]) but it should be clear that understanding the learning style of the target audience and choosing the most appropriated teaching style is beneficial for the effectiveness of the learning process. Also the teaching style should fit the domain as well as the game genre. In addition, the educational goals (e.g., drilling, practicing skills, understanding…) need to be taken into consideration. Different models of instructions [22] can be used (e.g., direct instruction, role playing). The different choices made in this respect will also have an impact of the modeling concepts that should be provided to the designer. Similar as for the game modeling concepts, we could opt to provide a layered set of modeling concepts.

Thirdly, as we may want to specify educational games for different learning domains, it will be necessary to also provide modeling concepts for the learning domain (e.g., mathematics, social skills). This means that this part of the DSML should be open; it should be possible to plugin different sets of modeling concepts for different learning domains. Unlike for the two other aspects (gaming concepts and pedagogical concepts), this part of the language cannot be pre-engineered.

Furthermore, we need to consider how all these different modeling concepts can be combined to come to the specification of an attractive educational game. A question that needs to be answered is for instance: Is it possible to specify the game aspect and the learning aspect independently and weave them together in a later stage (and if so, how should this be done) or will this only complicate the modeling process? In any case, the two aspects need to be related somehow and how this could be done needs to be investigated.

Other issues that need to be considered when designing a DSML for educational games is the level of detail we want to consider. We can opt for a very detailed level, such that in principle all information is available to generate the actual code for the educational game (Model Driven Engineering principle [23]). However, from experience in the field of Virtual Reality ([24]), we known that this is not feasible and also not desirable because the (graphical) models will become so large and complicated that they loose their benefit. End users and domain experts cannot develop them anymore and even have trouble to understand them. Also, the time needed to create (and verify) them tends to become equal to the time of an actual implementation, and developers will then prefer to do the implementation manually, as they will have more control on the result. If our purpose is to actively involve domain experts in the design process, the focus should be on the overall design of the educational game, which can be used as input for the actual implementation but still need to be complemented with all kind of information, e.g., graphical design. Related and other pitfalls when creating a DSML can be found in [25].

5 Conclusions and Future Work

In this paper, we discussed the use of a domain-specific modeling language for the design of educational games. Domain-specific modeling languages use the vocabulary of the target domain and provide abstractions that make the specifications of solutions easier and more accessible for domain experts. Although such an approach looks promising, several research questions still need to be solved. Many of the research questions have to do with the lack of fundamental knowledge on how to design educational games that are successful and effective from a learning point of view. In current research projects we are investigating different of these research questions. We are also working on the development of a particular DSML, where we will limit the language to one particular learning domain, one particular game genre, and one particular instructional method. Later on, we will try to generalize the language to different game genres and instructional methods.

References

1. Anderson, C. A., and Bushman, B. J. (2002): Media violence and societal violence. Science, 295, pp. 2377-2378 (2002)

2. Klopfer, E., Osterweil, S., and Salen. K.: Moving Learning Games Forwards. The Education Arcade (2009).
3. Van Eck, R.: Digital game-based learning: It's not just the digital natives who are resless.... Educause Review, 41(2), pp. 16-30 (2006).
4. Prensky, M.: Digital Natives, Digital Immigrants, Part II: Do they really think differently? On the Horizon, 9(5), pp. 1-6, MCB University Press. Online: http://www.marcprensky.com/writing/ (2001)
5. Salen, K., and Zimmerman, E.: Rules of play: Game design fundamentals. MIT Press, Cambridge, MA, USA (2003).
6. Microsoft Corp., http://www.gamesforwindows.com/en-US/directx/
7. OpenGL, http://www.opengl.org/
8. Java 3D Graphics, http://www.java3d.org/
9. Thinking Worlds, http://www.thinkingworlds.com
10. Furtado, A. W. B., and Santos, A. L. M.: Using Domain-Specific Modeling towards Computer Games Development Industrialization. In: Jeff Gray, Juha-Pekka Tolvanen, Jonathan Sprinkle (eds.) 6th OOPSLA Workshop on Domain-Specific Modeling (DSM'06), Computer Science and Information Systems Reports Technical Reports TR-37, Jyväskylä University Printing House, Jyväskylä, Finland, pp. 1-14 (2006)
11. Dobbe, J.: A Domain-Specific Language for Computer Games. Master Thesis, TU Delft (2004)
12. Guerreiro, R., Rosa, A., Sousa, V., Amaral, V., and Correia, N. : UbiLang: Towards a Domain Specific Modeling Language for Specification of Ubiquitos Games. In: Luís S. Barbosa, Miguel P. Correia (eds) INForum 2010 - II Simpósio de Informática, pp. 449–460 (2010)
13. Marchiori, E.J., Torrente, J., del Blanco, A., Moreno-Ger, P., and Fernández-Manjón, B.: A Visual Domain Specific Language for the Creation of Educational Video Games. Learning Technology, Vol 12 Issue 1, IEEE Computer Society's (2010)
14. Laforcade, P.: A Domain-Specific Modeling approach for supporting the specification of Visual Instructional Design Languages and the building of dedicated editors. Journal of Visual Languages and Computing 21, pp. 347-358 (2010)
15. Luoma, J., Kelly, S., and Tolvanen, J.-P.: Defining Domain-Specific Modeling Languages: Collected Experiences. In: Proceedings of the 4th OOPSLA Workshop on Domain-Specific Modeling (DSM'04), Vancouver, British Columbia, Canada (2004)
16. van Deursen, A., Klint, P., and Visser, J.: Domain-specific languages: an annotated bibliography, ACM SIGPLAN Notices, Volume 35 Issue 6, (2000)
17. Elliott, C.: An Embedded Modeling Language Approach to Interactive 3D and Multimedia Animation", IEEE Transactions on Software Engineering, Volume 25 Issue 3 (1999)
18. Rollings, A., and Adams, E.: On Game Design. New Riders (2003)
19. Felder, R.M., and Henriques, E.R.: Learning and Teaching Styles in Foreign and Second Language Education. In: Foreign Language Annals, vol. 28, pp. 21--31 (1995)
20. Felder, R.M., Silverman, L.K.: Learning and Teaching Styles in Engineering Education. In: Engr. Education, vol. 78, pp. 674-681 (1988)
21. Coffield, F., Moseley, D., Hall, E., Ecclestone, K.: Learning Styles and Pedagogy in Post-16 Learning: A Systematic and Critical Review. Learning and Skills Centre, LSRC, U.K (2004)
22. Joyce, B., and Weil, M.: Models of teaching (5th edition). Englewood Cliffs, NJ: Prentice-Hall (1996)
23. Schmidt, D.C.: Model-driven Engineering. IEEE Computer 39 (2) (2006).
24. WISE Research Group – VR-WISE research, http://vr-wise.vub.ac.be/
25. Kelly, S., and Pohjonen, R.: Worst Practices for Domain-Specific Modeling. IEEE Software Vol.26, No.4 (2009)