Specifying the Pedagogical Aspects of Narrative-Based Digital Learning Games Using Annotations

Frederik Van Broeckhoven
Vrije Universiteit Brussel
Pleinlaan 2
B - 1050 Brussels, Belgium
Frederik.Van.Broeckhoven@vub.ac.be

Prof. Dr. Olga De Troyer
Vrije Universiteit Brussel
Pleinlaan 2
B - 1050 Brussels, Belgium
Olga.DeTroyer@vub.ac.be

ABSTRACT
In this paper, we present an extension of ATTAC-L, a domain specific modeling language to model the narrative content ("story") of virtual experience scenarios or digital games. ATTAC-L is specifically designed to allow the involvement of non-technical people in the modeling process. However, as in learning games (i.e. games with an educational purpose) giving due consideration to the learning aspects is as important as to the game aspects, ATTAC-L is extended to also allow the modeling of the pedagogical aspects of the learning game. To realize this, we opted for an annotation mechanism. In this way, the educational or pedagogical aspects are specified on top of the storyline model. This allows (and even gently forces) at the one hand to integrate learning into the storyline, and on the other hand it prevents that the specification of the two aspects (learning and gaming) are entangled in the specification. It allows for a clear separation between the narrative content and the educational aspects in the story flow. This allows stakeholders to concentrate either on the story flow, or on how the story flow will be used to realize the pedagogical objectives of the game.
This paper explains the principles of the annotation technique, and describes and illustrates the newly introduced modeling concepts.

Categories and Subject Descriptors
• **Software and its engineering~Domain specific languages**
• *Applied computing~E-learning*

General Terms
Design, Human Factors, Languages.

Keywords
Domain Specific Modeling Language, Learning Games, Narrative Content.

1. INTRODUCTION
Already over a timespan of numerous years, Game Based Learning (GBL) has become an increasing popular learning approach as it combines playful elements with learning. In various domains, GBL has been recognized to be an efficient way to submerge the player-learner in the pedagogical material in a playful and motivating way [2].

Several authors argue that to exploit the power of games in GBL, the game narrative and the learning should be tightly integrated rather than one being just an add-on of the other one [2]. However, as a consequence and because a wide diversity of educational curricula that can be offered in the form of GBL, the respective games (here called educational games) must specifically be developed to fit their purpose. Developing these educational games often require the involvement of technical as well as non-technical experts, e.g., pedagogical experts, domain experts, educators. This creates the challenge to effectively involve these non-technical people in the development process without requiring them to deal with low-level implementation details (e.g., programming concepts). In this paper, we concentrate on supporting the involvement of non-technical stakeholders during the design phase.

To support the involvement of non-technical people during design, we propose the use of a Domain Specific Modeling Language (DSML) for modeling the interactive story of the educational game. Some authors also proposed the use of a Domain Specific Modeling Languages to model (certain aspects of) digital (educational) games [3][4][6][7][8]. However, there is a strong call for tools focusing on designing game narratives and integrating educational aspects [9], but research in this field with respect to DSMLs is rather sparse. Often DSMLs are closely tied to a game development framework (and still require non-technical people to deal with the technical details of game development), put hard restrictions on the type, audience or platforms of the games that can be modeled or do not provide a way to specify the educational aspects. Our work especially wants to deal with this last aspect. For games with an educational purpose, giving due consideration to the learning aspects is important to come to learning games that are pedagogically sound.

2. RELATED WORK
WEEV (Writing Environment for Educational Video Games) [7] also proposes a DSML to model the narrative content of educational games. As a proof of concept, this DSML is added on top of e-Adventure, a framework specifically designed for (non-technically skilled) educators to develop educational games of the point-and-click genre. Story modeling is based on an explicit representation of the interactions between the player and the virtual world by means of a state-transition diagram. To reduce de overall complexity, WEEV has language constructs that helps organizing the structure. Opposed to WEEV, which uses a state-transition approach, we use a flow-based approach, a decision that is based on the results of a user experiment [1]. Moreover, we

impose a strict separation between the narrative and the pedagogical aspects, while in WEEV both aspects are interwoven.

The GLiSMo language (Serious Game Logic and Structure Modeling Language) [3] is specifically designed to model teaching methods directly into the game logic of an educational game. For this, it uses the concept of a serious game brick, a block representing a single, atomic step that can be executed in the context of an educational game-environment, either related to a logical or a pedagogical functionality of the game. The bricks have input- and output-ports and the overall game logic is modeled by linking several bricks through these ports. This inter-linking defines a temporal relationship and dataflow between the bricks, giving the model a dataflow-based structure. An abstraction mechanism is provided, i.e. a serious game composite, which is used in the same way as a brick, but encapsulates one or more inter-linked bricks, consequently providing a way to organize more complex models. Our research, developed in parallel, uses similar principles, but we opted for an explicit flow-based structure that only requires defining temporal relationship between game moves. Pedagogical aspects are expressed using annotations, which allows for a better separation of concerns.

The StoryBricks framework [8] is an interactive story design system. It provides a visual language based on the visual programming language Scratch [6] designed by the MIT lab. Without the need for programming skills, the users can edit the characters in the game and the artificial intelligence of the game that drives the characters. The users can setup characters by using so-called story bricks to give them emotions, items, etc. The bricks can also be used to specify what need to be done at certain points in the game. This way, an interactive scenario is modeled in an implicit way by defining a set of rules expressing which events should be evoked under what conditions. This enables the interaction between the characters in the game without the need for it to be programmed explicitly. The StoryBricks approach allows a great deal of flexibility in defining the rules that make up the game logic, but a story cannot be modeled explicitly. Our user experiment performed [1] shows that a rule-based approach would be less suitable for our target group (social oriented stakeholders). Our work has adopted the brick concept as basic building block for our language from this framework, but in addition, it allows modeling the story flow explicitly. Moreover, we provide a mechanism to model the pedagogical aspects of games.

3. ATTAC-L: NARRATIVE MODELING

To model the narrative content of a learning game, we defined ATTAC-L [1], a Domain Specific Modeling Language, specifically designed to allow non-technical people to model the narrative content of so-called virtual experience scenarios or digital games with some form of interactive narration.

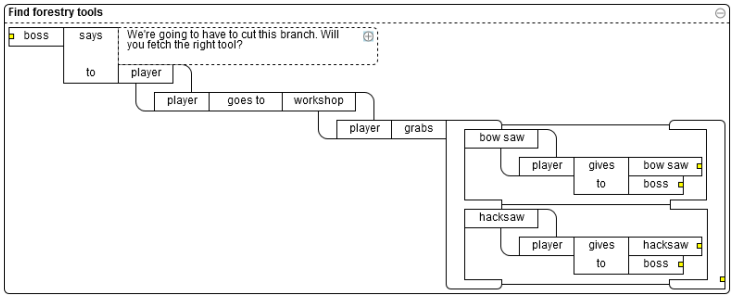

Figure 1: Example of a scenario modeled in ATTAC-L

To make ATTAC-L usable for its target audience, i.e. social-oriented non-technical people, it uses two important principles. First, it uses a flow-based structure to represent the overall narrative content. Each individual step, either to be performed by the player or performed automatically by a Non Player Character (NPC), is represented as a 'game move', a concept adopted from [5]. Game moves are directly or indirectly linked denoting their relative time-relationships. By adopting concepts from UML flow-chart modeling, ATTAC-L is capable of expressing various forms of chronology between game moves: sequences (defining which game move follows another one), choice (or branching, defining alternative story flow paths), and concurrency (story flow paths that are performed in parallel). On top of that, ATTAC-L introduces extra control mechanisms to increase the expressiveness: order independence (story flow paths that must all be performed, but in any order), repetition and optionally. Furthermore, the concept of scenario is used as an abstraction mechanism to deal with the complexity of large models.

The second principle is the use of so-called bricks, a principle that is adopted from the StoryBricks framework [8]. Bricks form the atomic building blocks for a narrative model. Two types of bricks are distinguished: (*regular*) *bricks* and *control-bricks*.

The regular bricks are used to construct *game move statements*, a formal representation of a game move or a single action that can be performed in the context of the game environment. A regular brick is represented by a rectangular containing a word (denoting its meaning) and corresponds to a smallest, meaningful unit that exists in the context of a story: an act that can be performed, a tangible object that can perform or undergo the act, a state, a mood, etc... Game move statements are constructed by connecting bricks to each other according to some rules. These rules are based on the grammar of natural language. The result is a construct that reads as a simple natural sentence and denotes a game play action.

Control bricks can be used to combine game move statements according to their relative time relationships. As stated earlier, we use a flow-based structure to represent this time relationship, but instead of using a graphical representation like in UML (which is unknown to our target users) we decided to follow the same brick-principle for expressing this control structure.

Figure 1 shows an example of a scenario (part of a story) modeled in ATTAC-L. This simple example is about a player who is instructed to find a correct tool for a woodcutting job. The player gets instructions, goes to the place where he can find tools, chooses one of two tools and gives it to his instructor.

Note that the interpretation of the acts, objects, moods,… in the actual game is left to the code generation. For example, the player grabbing something might be realized by letting the player clicking on the corresponding game entity.

4. MODELING LEARNING

To model the pedagogical aspects of a learning game, we opted for an annotation mechanism. In this way, the pedagogical aspects are specified on top of the storyline model. This allows (and even gently forces) at the one hand to integrate the learning into the story, and on the other hand it prevents that the specification of the two aspects (learning and gaming) are entangled in the specification. It allows for a clear separation between the narrative content and the educational aspects related to the story flow. This allows

stakeholders to concentrate either on the story flow, or on how the story flow will be used to realize the pedagogical objectives of the game.

Using annotations to model the educational aspects means that at certain points in the storyline, extra contextual information (in the form of annotations) are added, which define either pedagogical objectives (e.g., knowledge acquisition) or strategies (e.g., "Trial-and-Error" learning approach), or pedagogical actions that have to be performed. These pedagogical actions might for instance include, providing extra (learning) information, providing assistance to the player, updating the knowledge level of the player, changing roles (i.e. let the player 'experience' the story from another perspective), performing assessments, and provide the possibility to rehearse or retry parts of the story. The annotations not only specify the educational aspects of the game, they also allow validating (at a later stage) that the specification (story flow + annotations) satisfies the pedagogical objectives and strategies specified. This will be done by verifying that the necessarily pedagogical actions and game mechanics are provided to guarantee (to a certain degree) that the pedagogical objectives and strategies specified can be achieved (e.g., if a "Trial-and-Error" learning approach is specified, the player should indeed have the possibility to retry certain actions).

In what follows, we first introduce the general principles for the concept of annotation (section 4.1). This is followed by a more detailed description of some of the annotation types provided in the current version of ATTAL-L. We provide examples of the two major categories: annotations for expressing pedagogical actions (section 4.2), and annotations for expressing the pedagogical objectives and principles (section 4.3).

4.1 General Principles of the Annotations

An annotation is represented as a small square-like brick, called annotation-brick (following the brick-principle of the ATTAC-L language). The annotation-brick contains a symbol that denotes its function, i.e. the type of the annotation, e.g., an information annotation. Each annotation has a set of attributes that can be given values to instantiate the annotation.

Annotations are attached to a game move or to a scenario. Attaching an annotation to a scenario means that the annotation (i.e. its function) applies to the encapsulated storyline part of the scenario as a whole. Whether an annotation can be attached to a scenario or/and a game move depends on its type.

As the amount of information to be specified for an annotation can be quite vast, we opted for a text balloon to list the attributes and to enter or edit their values. This balloon points to the annotation-brick and can be show/hidden, because this detailed information is not required to be visible all the time.

4.2 Pedagogical Actions

In this section, we provide examples of annotation types provided for expressing pedagogical actions.

4.2.1 Changing Roles

An important aspect of game narration is the interaction or the involvement of the player in shaping the plot of the story. The original version of ATTAC-L was using a special brick 'player' to represented the player as an actor in the story. We decided to omit this player-brick and instead provide an annotation type that allows specifying player controls. This annotation type allows specifying that the in-game character used by the player should be changed. This concept gives more flexibility, as it allows specifying that the player should be given the perspective of a different character at a certain moment in the game (and possibly for a short while). This is required to support, for instance an intervention strategy, called changing perspective, used for pedagogical objectives such as behavioral change, and used to let the player experience the scenario from another perspective.

For specifying this player perspective, we have defined the *playable*- ( ) and *not playable* ( ) annotation types. When attached to a game move, the playable annotation has no attributes; it only specifies that the associated game move should be performed by the player, and consequently that, at that moment, the character performing the game move is the player's controlled character. When annotating a scenario, it has an attribute 'role' that states the character the player should control. An example is given in Figure 2. To (temporarily) disable the player's controls, the *not playable* annotation (without attributes) can be used in a similar way.

4.2.2 Providing Information

At certain points in the storyline, it could be necessary to give the user some extra information, ranging from instructions to play the game to (extra) course material. To indicate this, the game moves where this extra information should be presented can be annotated with the *inform*- ( ) annotation. Its single attribute 'what' defines the material to be presented to the user (and also its form). The information will be shown to the player right before the game move is performed. An example is given in Figure 3.

4.2.3 Providing Assistance

Sometimes, a player might need guidance in making the right decision. For example, when the player is confronted with a choice and has difficulties with making the choice (note that in this case, the game must be able to detect whether the player has difficulties). This might be when the player is taking too long to decide or when he is taking the wrong decision systematically. Also, assistance can exist in the form of emphasizing a certain entity in the game environment to draw the user's attention.

To support this, we have defined a set of annotations. An *assist*- ( ) annotation annotates a game move where assistance might be needed. Its attributes specify the assistance the player will get, in what form, and under what conditions (for example, when the player has not yet reached a certain level of progress). A *wrong path*– ( ) annotation can be used to identify incorrect choices that players may take. Both annotations are illustrated in Figure 4.

4.2.4 Other Annotation-types

More annotation-types are defined for pedagogical actions, e.g., for specifying emphasizing, a scoring-system (rewarding and punishment), and a checkpoint-system (retry and rehearsal). Due to space-considerations, we will not elaborate them.

4.3 Pedagogical Objectives and Principles

In this section, we present annotation types provided for expressing pedagogical objectives and strategies. This concept is still under development; therefore we will only describe it briefly.

4.3.1 Pedagogical Objectives

A *(pedagogical) objective-annotation* is used to explicitly relate pedagogical objectives to scenarios. A pedagogical objective is defined as the learning goals together with the expected learning

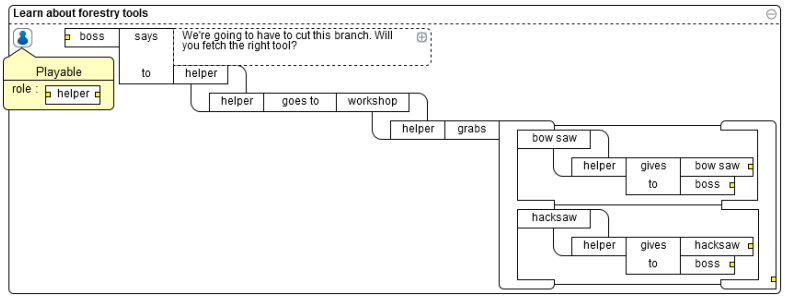

Figure 2: Example use of the 'Playable' annotation

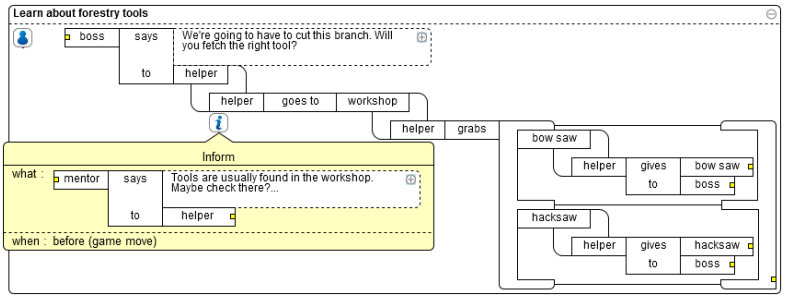

Figure 3: Example use of the 'Inform'-annotation

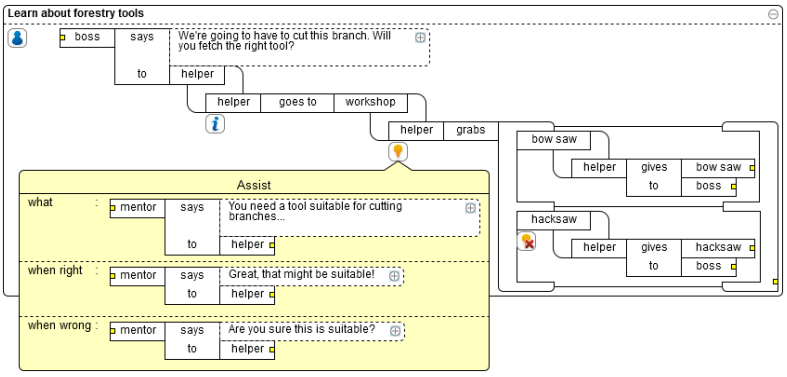

Figure 4: Example use of the 'Assist'-annotation

outcomes (obtained from playing the scenario). The information in this annotation will be used to validate the model, i.e. to verify that the listed objectives actually can be met in the specified scenario-model, but it can also be linked with assessments or, evaluations that check whether a learning goal is achieved.

4.3.2 Pedagogical Strategies

A pedagogical strategy refers to a concrete set of methods that can be used to assure a certain degree of learning. A scenario can be annotated with a *(pedagogical) strategy-annotation* to specify which strategy is used in the scenario. Also this information can be used to validate the model (i.e. verifying that the model is in accordance to the strategy), but it can also be used to make suggestions to the modeler e.g., on what pedagogical actions or objectives (not) to use.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an extension of ATTAC-L, a domain specific modeling language to allow the modeling of narrative content for learning games, specifically designed for non-technical people. The language combines the brick principle used in StoryBricks with a flow based structure for the representation of overall narrative model. The extension consists of an annotation-system to specify educational aspects on top of the storyline. This allows for a clear separation of concerns.

The next step consists of further elaborating the annotation types for expressing pedagogical objectives and strategies, in order to provide more support to the modeler. Possible types of learning outcomes and strategies can be presented to the modeler to select from, and for common pedagogical strategies possible pedagogical actions can be suggested. This requires a further investigation of how principles used in education can be mapped onto game mechanics. Other work is oriented towards the validation of the model against the objectives and strategies specified. Furthermore, a profound evaluation of the complete ATTAC-L language is planned. Tool support is currently already under development.

6. ACKNOWLEDGEMENT

This research is funded by IWT (Institute for Science and Technology) (www.iwt.be) (Friendly ATTAC project).

7. REFERENCES

[1] Van Broeckhoven, F. and De Troyer, O. 2013. ATTAC-L: A Modeling Language for Educational Virtual Scenarios in the Context of Preventing Cyber Bullying. 2013 IEEE 2nd International Conference on Serious Games and Applications for Health (SeGAH) (May 2013), 1–8.

[2] Dondlinger, M. 2007. Educational Video Game Design: A Review of the Literature. Journal of Applied Educational Technology. 4, 1 (2007), 21 – 31.

[3] Hirdes, E.M., Thillainathan, N. and Leimeister, J.M. 2012. Towards Modeling Educational Objectives in Serious Games.

[4] Kudo Programming Language: http://research.microsoft.com/en-us/projects/kodu/. Accessed: 2013-12-14.

[5] Lindley, C.A. 2005. Story and Narrative Structures in Computer Games. Developing Interactive Narrative Content. Bushoff and Brunhild, eds. High Text. 1–27.

[6] Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E. 2010. The Scratch Programming Language and Environment. ACM Transactions on Computing Education. 10, 4 (Nov. 2010), 1–15.

[7] Marchiori, E.J., del Blanco, Á., Torrente, J., Martinez-Ortiz, I. and Fernández-Manjón, B. 2011. A Visual Language for the Creation of Narrative Educational Games. Journal of Visual Languages & Computing. 22, 6 (Dec. 2011), 443–452.

[8] StoryBricks: http://www.storybricks.com. Accessed: 2013-10-12.

[9] Zyda, M. 2005. From Visual Simulation to Virtual Reality to Games. Computer. 38, 9 (Sep. 2005), 25–32.