Using a Controlled Natural Language for Specifying the Narratives of Serious Games

Frederik Van Broeckhoven, Joachim Vlieghe, and Olga De Troyer

Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussel, Belgium
{frederik.van.broeckhoven,joachim.vlieghe,olga.detroyer}
@vub.ac.be

Abstract. Creating serious games calls for a multidisciplinary design team, including game developers, subject-matter experts, pedagogical experts, and narrative designers. However, such multidisciplinary teams often experience communication and collaboration problems due to the different terminologies, backgrounds and concerns of the people involved. To overcome these problems, we propose the use of a Controlled Natural Language (CNL) in combination with a graphical notation for modeling game narratives. The use of a CNL provides an easy human-readable, yet flexible and expressive way to specify story-lines. In addition, a CNL provides the possibility for automatically processing story-lines and generating code. As such, incorporating CNL in the design process also contributes to shortening the development time.

Keywords: domain specific modeling language · controlled natural language · game narrative

1 Introduction

The narrative of a video game is a powerful tool to convey knowledge and induce behavior change [3]. However, to come to well-grounded and effective serious games, the development should be a collaborative process involving experts from various fields [16]. Professional narrative designers contribute to the design process by creating compelling and emerging narratives, while pedagogical experts and subject-matter experts collaborate to integrate effective pedagogical methods. Game developers use their experience and expertise to turn all of this into a challenging game. However, narrative designers and pedagogical and subject-matter experts are often not trained or even familiar with the technical matters and jargon used by the game developers, and vice versa. In addition, each domain of expertise represents a different set of concerns [1]. This can seriously hinder communication and collaboration. As different experts usually communicate their requirements to the game developers in natural language, misunderstandings and ambiguities often arise. Furthermore, this form of collaboration and transfer bestows upon the game developers the task of translating the natural language specifications into formal specifications or code.

Suitable techniques and tools could support a more integral form of collaboration, as well as shorten the development process. In light of this observation, various authors [2, 6, 7, 12, 14] have presented arguments in favor of using a Domain Specific Modeling Language [10] (DSML) for specifying (i.e., modeling) (serious) games. A DSML is usually a small language dedicated and restricted to a particular domain [17]. It provides abstractions that make it easier and less time consuming to specify solutions for a particular class of problems. By building on the vocabulary of the problem domain, the DSML enables domain experts to understand, validate and often even develop specifications autonomously. In [10], DSMLs are shown to require less modeling work, that could often be carried out by persons with limited programming experience, resulting in a clear productivity increase. In general, a DSML is a graphical language. Graphical specifications are known to be easier for the communication with non-technical people, and allow people to grasp large amounts of information more quickly than large listings of textual specifications. Despite the use of a familiar vocabulary and a graphical notation, the language constructs in a DSML might still be too much oriented towards programming and therefore difficult to learn and use by people without programming experience. We believe that this problem could be resolved by creating a DSML that builds on a Controlled Natural Language (CNL). A CNL is a strict and controlled subset of natural language that avoids ambiguities. Using a CNL syntax provides an easy and human-readable, yet flexible and expressive way to specify the story-lines of the game, making it significantly easier for non-IT schooled domain experts to understand, and even create models. This way, a better communication and collaboration within multidisciplinary teams would be supported. Compared to the use of natural language, this approach provides the advantages that it allows for the automatic processing of the formally specified story models (i.e., generate code), which also helps to shorten the development time.

In previous publications, we introduced a graphical DSML for modeling (serious) games narrative, ATTAC-L, [14] and discussed the incorporation of educational aspects [15]. In this paper, the focus is on the use of a CNL to open the use of the DSML to a broader range of users. Our target audiences are the different stakeholders involved in the design and development of a narrative-based serious game. However, note that this does not imply that all types of users should be able to actually use the language for modeling narratives. For some users it suffices that they are able to read (i.e., understand) the models. Also note that such a DSML and its associated tool are not aimed to replace the creative contributions of narrative designers or game developers; they rather complement them. The paper is structured as follows: section 2 we discusses the background and related work; section 3 introduces the principles of ATTAC-L; section 4 presents its CNL-based syntax; and section 5 shows how this syntax can be supported by our original graphical language. Section 6 discusses tool support. Section 7 describes a pilot evaluation and section 8 formulates conclusions and discusses future work.

2 Background and Related Work

The easiest way to specify the narrative of a game is through natural language (NL). However, plain NL is difficult to process by a computer. Therefore, usually a strict subset of NL (a CNL) that avoids ambiguities is used. The formal use of a NL syntax has been studied extensively (see [8] for an overview) and various CNLs have been developed for a wide range of applications. The main challenge of using CNL is to deal with the trade-off between 'preciseness' (i.e., the degree to which the meaning can be directly retrieved from its textual form), 'expressiveness' (i.e., the range of propositions that the language can express), 'naturalness' (i.e., how much effort a person needs to do to interpret or understand the text), and 'simplicity' (i.e., the level of complexity of syntax and semantics) [8].

The CNL selected is Attempto Controlled English (ACE) [5]. ACE expressions are formulated much like plain English sentences. These sentences are written in the 3rd person, singular simple tense, and 'describe' logical terms, predicates, formulas and quantification statements. ACE-sentences are built based on the definition of two word classes: function words (determiners, quantifiers, negation words, ...) and content words (nouns, verbs, adverbs and prepositions). ACE defines a strict and finite set of unambiguous constructions and interpretation rules to create ACE sentences. We opted for ACE because it has a sound fundament (first order logic) and is well suited for our purpose.

The use of a CNL in relation to the modeling of narratives for video games is not new. The most notable works are *Inform* and *StoryBricks*.

The *StoryBricks*[1] framework is an interactive story design system. It provides a graphical editing language based on *Scratch* [11]. Without major programming skills a user can edit the characters in the game and the artificial intelligence that drives them. Through the use of so-called story bricks, users can place characters in a virtual environment and give them emotions and provide them with items. Story bricks are blocks containing words that read like NL sentences when placed together. The story bricks can also be used to define rules that specify what needs to be done under certain conditions during gameplay. In this way, the flow of an interactive story is modeled implicitly. Experiments have shown that people with limited programming knowledge and skills found it difficult to understand this form of implicit flow specification [13]. In our work, we adopted the bricks principle from StoryBricks, but not its rule-based approach. Instead, the story flow is explicitly specified.

Inform[2] is a toolset targeted toward professional narrators. It allows them to create interactive fiction (e.g., adventure games). Since version 7, *Inform* includes a DSML to define all aspects of an interactive fiction, including (scene) setting, character setup, and story flow. The DSML uses a CNL. Although not explicitly stated, *Inform* shares a lot of similarities with ACE. In contrast to *Inform*, we opted for a graphical language language as most DSML's do. Also, our DSML also does not allow to define aspects such as environment settings and

[1] `http://www.storybricks.com`

[2] `http://inform7.com`

low-level implementation aspects. Instead, it focuses on the specification of the narrative, thereby reducing the complexity and increasing the understandability. Other tools can be used to specify other aspects.

3 Principles of ATTAC-L

In this section we briefly explain the principles of ATTAC-L. A more elaborated explanation can be found in [14, 15].

The principle of a 'game move', adopted from [9], represents an individual step in the game narrative performed either by the player (interaction) or by a non-playable character (NPC). Game moves are linked to each other according to their temporal order. By using flow-chart modeling principles, the overall flow of the story is defined. As such, ATTAC-L can express sequences (game moves following each other), choice (branching, defining alternative story flow paths), concurrency (story flow paths that are performed in parallel), and order independence (story flow paths that must all be performed, but in any order). Combinations of game moves that represented parts of the story can be defined as scenarios and re-used in other scenarios or stories. Scenarios help users to maintain an overview of large models.

The 'brick' principle, adopted from StoryBricks, represents a unit or building block used for composing story models and is used for the graphical representation of game moves. We discern between two types of bricks: *(Regular) bricks* and *Control-bricks*. Regular brick represents the smallest meaningful unit that exists in the context of a story: an act to be performed, a tangible object that can perform or undergo the act, or a state. These bricks are used to construct game moves by connecting bricks to each other according to a set of syntax rules (see section 4). As we use a CNL syntax, the constructs created read as NL. Regular bricks are illustrated in Fig. 1a. Control bricks represent the control structures inside the story, i.e. these bricks are used to express the temporal relationships between game moves, composing the overall flow of the story. We decided to use the brick principle also to represent the control structures because our target users might not be familiar with the typical flow-chart notation used in modeling languages such a UML. Control bricks are shown in Fig. 1. Fig. 2 shows how regular bricks are used to form game moves and how they are connected by control bricks.

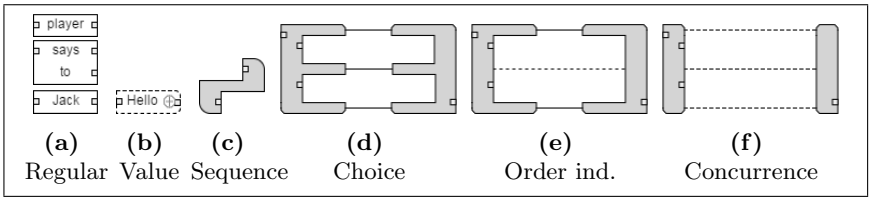
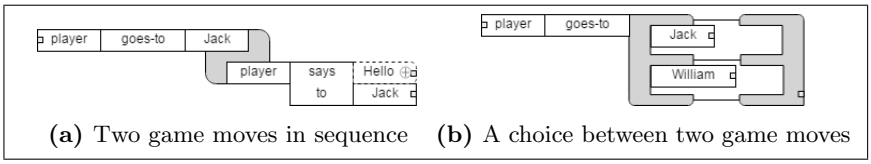


Fig. 1: Several types of bricks



(a) Two game moves in sequence **(b)** A choice between two game moves

Fig. 2: Connecting bricks to form story-lines

4 ACE Based Syntax

As motivated in the introduction, we now use ACE for the syntax of game moves. This implies that game moves, i.e. sentences for events and actions (including the players actions) are expressed in 3rd person simple tense. In other words, narratives are modeled as if they were told from a narrator's point of view. In addition, and similar to ACE, we distinguish between function words and content words. Our set of content words consists of nouns, verbs, and adjectives, corresponding respectively to entities, acts, and states in the story-line. Our set of function words consists of determiners, negation words, pronouns, and the copula be. Unless otherwise specified, words are always expressed in lower case letters. Lastly, in accordance with ACE, the sentences are always composed of two main parts: a *subject*, followed by a *predicate*. The former describes the entity that invokes the action and is a *noun phrase*. The latter describes the action itself and is expressed as a *verb phrase*.

Noun Phrases. The simplest form of a noun phrase is a *proper noun* that refers to an entity by means of a name. Nouns always start with an upper case letter and may contain upper and lower case letters or digits (e.g., 'X1', 'John'). Spaces are not allowed, but hyphens can be used to form word-groups (e.g., 'Mr-Smith'). The proper noun 'Player' is reserved to refer to the player or the character that he or she is controlling.

A noun phrase can also refer to a game entity in an indirect way by means of a *countable common noun* which is a common noun preceded by a determiner or a quantifier. This type of noun phrase can be used to refer a single entity (e.g., 'the house', 'a door', 'some person', 'one key') or multiple entities (e.g., '2 doors', 'some persons', 'all keys'). This type of noun phrase is used to refer to one or more entities in a general way. This means that on different 'runs' of the story flow, different entities conforming to this noun phrase could be used. This allows narrative designers to create less predictable stories.

Adjectives. Adjectives can be used to make a countable common noun phrase more specific. As such, the group of entities to which the noun refers can be narrowed down. Adjectives can be positive, superlative or conjoined (e.g., 'the last person', 'two highest trees', 'a sad and angry person').

Variables. Variables are a way to assign a proper name to a countable common noun phrase. When a proper noun is appended to such a noun phrase, e.g., 'a person Mr-X', the proper noun can be used to refer to the same entity in subsequent game moves. This corresponds to an in-line and implicit declaration of a variable. For people without programming knowledge or experience, this approach is likely to be more easily than the use of explicit variable declarations.

Genitives. Genetives are used to refer to nouns that have a possessive association with another noun. Genitives are constructed by appending the noun phrase of the possessor to the noun phrase of the possession using the preposition 'of' (e.g., 'the back-pack of Jack', 'two items of a person'). Note that in Standard English, the use of the preposition 'of' might sound odd in some cases (e.g., 'two apples of a tree' as opposed to 'two apples from a tree'). Nonetheless, we refrain from introducing extra prepositions in order to keep the syntax as simple as possible.

Values. Values (text, numbers, lists and associative maps) can be used as noun-phrases when their noun can be deduced from the context of the game move. For example, when the game move expresses a speech action, the action can only involve a text-value. The type of a value can be specified more directly by prepending a common noun (e.g., 'a message "Hello!"').

Verb Phrases. A verb phrase corresponds to the predicate of a sentence and describes the actual activity that is performed in a game move. Generally, a verb phrase starts with the conjugated verb. Depending on the type of verb that is used, direct and indirect passive objects can be included in the verb phrase. While an intransitive verb stands on its own (e.g., 'to wait'), a mono-transitive verb involves only one direct passive object (e.g., 'to see *something*'), whereas a ditransitive verb involves both a direct and indirect passive object (e.g., 'to give *something* to *somebody*'). Some verbs can also be associated with a particle. In this case, the combination verb-particle is considered as a whole and is written as a hyphenated combination (e.g., 'X walks-to Y', 'X looks-at Y').

It is possible to use an adjective phrase as part of a verb phrase. Similar to verbs, an adjective phrase can be intransitive (e.g., 'happy', 'sad') or transitive (e.g., 'angry with *somebody*' or 'afraid of *something*'). In the latter case, the adjective is always associated with a preposition and must be written as hyphenated combination (e.g., 'angry-with X').

The verb 'to be' serves as a copular verb and thus represents a special case. Copular verbs establish a link between the meaning of a predicate of the sentence and its subject. This means that they can be used to set a state for the subject. In order to do so, the copular verb is suffixed by an adjective phrase (e.g., 'X is happy', 'X is angry-with Y').

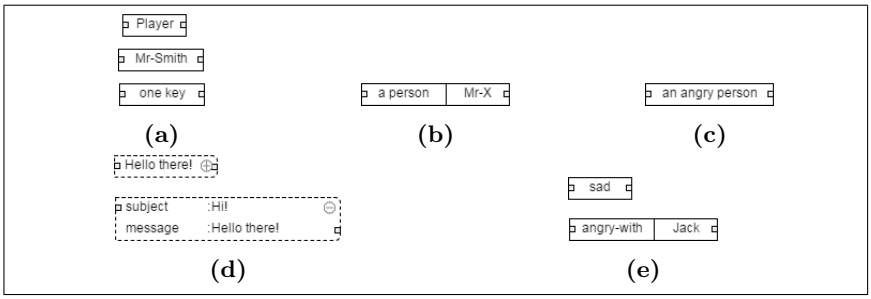


Fig. 3: Noun and adjective phrases

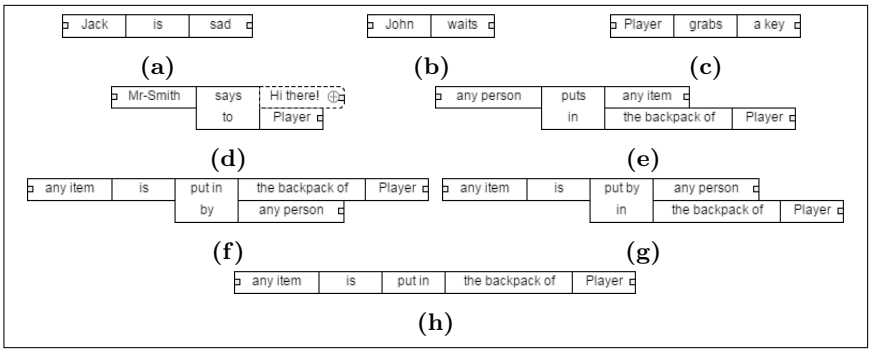


Fig. 4: Verbs connecting noun and adjective phrases forming game moves

Special Sentence Structures. Some special structures are added to provide the modeler with extra flexibility or expressive power. The phrases 'there is' and 'there are' are introduced to allow the modeler to *explicitly declare variables*. These phrases are followed by a noun phrase with a variable (e.g., 'there is a person Mr-X', 'there are two persons The-Johnson-Brothers').

The modeler can express any sentence containing a transitive or ditransitive verb phrase in a *passive voice*. A passive voice sentence uses the copular verb to be in combination with the past participle tense of the verb from the original sentence. The placement of the subject and the direct passive object are then swapped (e.g., 'X steals an item of Player' becomes 'an item of Player is stolen by X'). This option increases flexibility by allowing the modeler to create sentences when the character that performs the action is not defined, i.e., when no subject would be present in the active equivalent of the sentence (e.g., 'an item of Player is stolen').

5 Mapping to a Graphical Language

In this section, we explain how the CNL syntax defined for ATTAC-L can be expressed by means of our graphical language constructs, i.e., bricks. We have mapped game moves to bricks by assigning words or phrases to regular bricks.

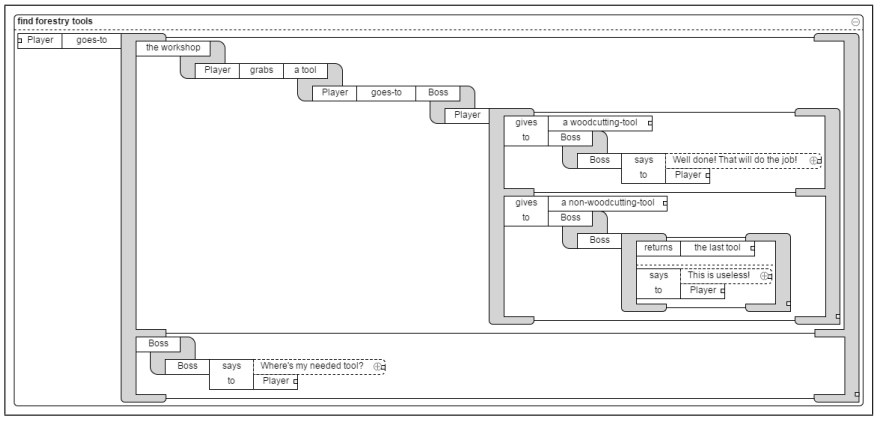


Fig. 5: Example story-line model in the CNL-based DSML

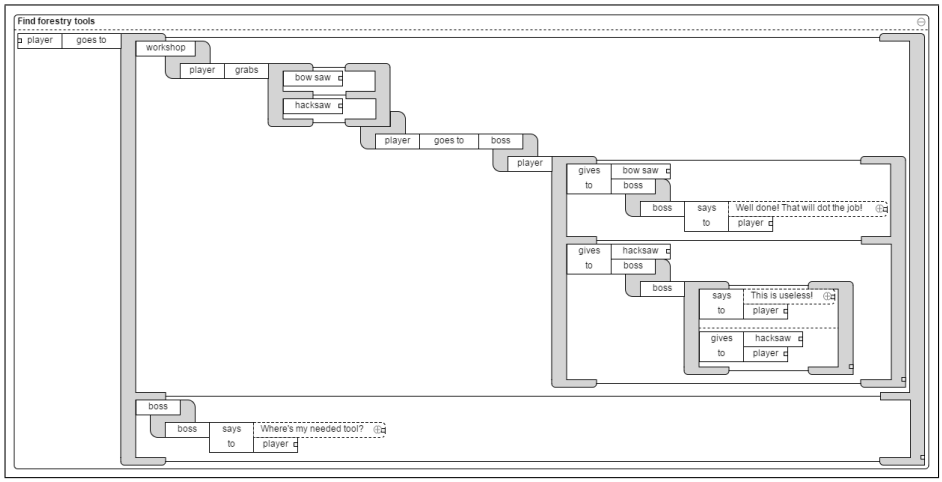
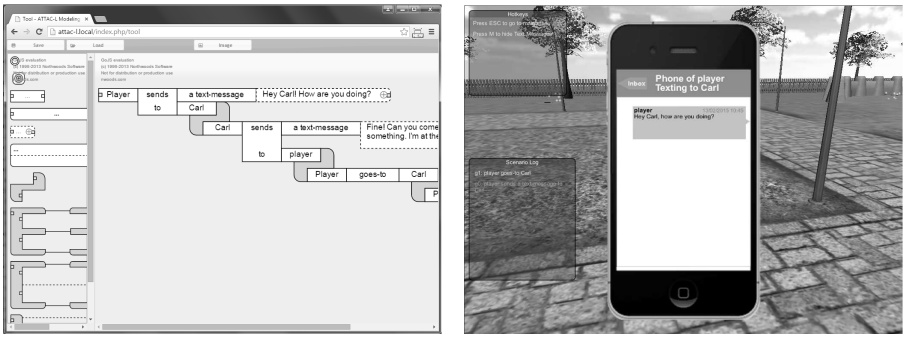


Fig. 6: Example story-line model using the original syntax

These regular bricks can then be connected to each other in accordance with the syntax rules defined in section 4.

A regular brick containing a noun phrase is called a noun-brick and refers to a tangible entity, i.e., an object, a NPC or a player (see Fig. 3a). Adjectives used for noun-phrases are contained in the noun-brick because they are part of the noun-phrase (see Fig. 3c). Adjectives used as adjective phrases are placed in an adjective-brick followed by a noun-brick in case it is a transitive adjective phrase (see Fig. 3e). To introduce a variable, an extra noun brick containing the proper name of the variable is joined at the end of the noun brick (see Fig. 3b). This emphasizes the variable declaration and allows the variable to be used independently in subsequent game moves. Values are represented by the value-brick (see Fig. 3d) which can only be connected to the end of a regular brick.



(a) Screen-shot of the modeling tool

(b) Simulating the player sending a text message to Carl

Fig. 7: Modeling and simulating scenario's

A regular brick containing a verb is called a verb-brick and connects to noun phrases and/or adjective phrases to form a game move. In the case of a ditransitive verb, the associated preposition serves as an extra connecting point for the indirect passive object (see figures 4a, 4b, 4c, 4d and 4e for illustrations of the most common cases). As explained, game moves containing a transitive or ditransitive can also be expressed in a passive form (Fig. 4f represents the passive equivalent of Fig. 4e). In passive sentences, the placements of the indirect passive object and the direct passive object (i.e., the original subject) can be interchanged (Fig. 4f and 4g can be used interchangeably to represent the same game move). In some cases, the 'by'-part of the sentence can be omitted (see Fig. 4h). The subject of the game move is then assumed to be any NPC.

6 Tool Support

We developed a (web-based) toolset for ATTAC-L that consists of a graphical editor including an export module, and a simulator. The editor (see Fig. 7a) allows users to specify a story model by means of drag-and-drop. Extra assistance is provided by means of automatic layout management and an auto-complete suggestion mechanism for entering names. The export module generates a machine interpretable data structure (JSON) for the modeled story. This structure can be used by existing game engines as input for generating code or by an interpreter to execute the story. A code generator is currently under development.

The simulator is a separate module. The simulator is a kind of interpreter that executes the story directly. The execution is performed, however, in a simple and predefined 3D environment with predefined NPCs and behaviors. As such, the simulator provides a fast way to verify and test the modeled stories (Fig. 7b shows a screenshot from the simulation of the example from Fig. 7a) and therefore serves as a fast prototyping tool. The current simulator mainly targets scenarios for cyber-bulling, which was the focus of the project for which ATTAC-L was originally developed [4]. By providing other predefined NPCs and behaviors, the simulator can of course easily be adapted for use in other domains.

7 Evaluation and Discussion

Because of space restrictions it is not possible to provide here an elaborated comparison between the original ATTAC-L syntax and the ACE-based syntax. However, we will illustrate differences with an example. Fig. 5 shows an example expressed with the new syntax, while Fig. 6 gives the same scenario expressed in the old syntax). We see that the model in the new syntax is more compact and allows a more dynamic story flow. First, it is not required to uniquely identify every entity as it was the case with the old syntax. Moreover, uniquely identified entities are now clearly distinguished by the use of proper nouns. Secondly, by using common nouns, actions can be specified more generally, which results in more flexibility for the story-line. For example, the character Boss will react in the same way regardless the player presents him with a hammer, a hacksaw or a non-woodcutting tool. This was not possible with the previous syntax without explicitly mentioning all the non- woodcutting tools. Similarly, the new syntax makes it possible for the player to grab woodcutting tools in places other than the workshop in order to successfully finish the scenario. Within the previous syntax, 'hacksaw' would refer to exactly one entity, i.e., the hacksaw positioned in the workshop. This example illustrates how the use of a CNL has increased the flexibility and expressive power of the DSML, whilst also enhancing its readability.

To investigate whether our CNL-based DSML satisfies our objectives, i.e., appropriate for people with and without programming knowledge and having the potential to enhance the collaboration in interdisciplinary design teams, we conducted a pilot evaluation. Four non-computer schooled people but with background in game narrative design participated. None of these people had prior experience in modeling stories with a DSML. First, the participants received a small introduction on the purpose of the DSML and how to use it. They also received an example (Fig. 5) and an outline of the syntax rules (in text). They were then asked to specify (i.e., model) a small prescribed story-line (given by means of text), namely a scenario about the player who has to escape from a locked house and having three possible ways of doing so. They were asked to first sketch the story flow on paper and then model the scenario with our modeling tool. The participants had a comprehensive list of actions and entities (nouns, pronouns, verbs, adjectives) at their disposal. When needed, they could ask for assistance. After finishing the exercise, the participants were invited to report on their experience and the difficulties they encountered.

The results of this evaluation were positive. The participants were able to understand the example given. They described this as very intuitive and easy. Similarly, all of the participants reported that they understood the modeling assignment and were able to model the scenario. Early during the exercise, some participants were inclined to model story flows that were too complex (introducing more game moves than actually needed) or created a solution that was very specific (e.g., using specific pronouns instead of the more general common nouns). When they were made aware of this, the participants quickly reflected on this and adapted their solution accordingly. In relation to the construction

of game moves, the participants reported that they had no difficulties in complying to the natural syntax specification. They found the (syntax) rules very intuitive and natural. The participants noted that (the use of) this language shows analogies with methods for interactive narrative design that they had encountered before, i.e., using small simple sentences in combination with a flow graph structure.

Although this was a small-scale evaluation, it provides us an indication that our graphical CNL-based DSML will be able to satisfy our aims. The models specified in this language were easy to understand for people without programming background. Furthermore, we have an indication that this kind of people will actually also be able to actively model narratives. This would imply that narratives designers could specify their stories in a formal way, which can directly be processed. This will not only save the time to elaborate the narrative on paper and transferring it to the technically people, but it will also avoid misunderstanding and eliminates ambiguities. Whether the language could improve communication was not investigated in this pilot evaluation.

The DSML and its toolset are currently used in an interdisciplinary project [4] aiming at developing serious games against cyber-bullying. The development team is composed of people from the social domain, from healthcare, computer science people, game developers, and also a narrative designer. The specification of story-lines is still an ongoing process, but the feedback on the language and its tool is promising.

8 Conclusions and Future Work

We presented a graphical Controlled Natural Language (CNL) based Domain Specific Modeling Language (DSML) for specifying (i.e., modeling) story-lines for serious games. Our aim was to enhance the communication and collaboration in the interdisciplinary design teams that are needed to arrive at well-grounded and effective serious games. We explained how a DSML for specifying narratives is based on a CNL. For this, we have based the syntax on Attempto Controlled English, a strict subset of plain English. In addition, we have shown how this domain-specific CNL can be mapped on the graphical brick representation used before. Using a CNL has increased the flexibility and the expressive power of the language, while also enhancing its 'naturalness' (human-readability). We also briefly presented the tools developed to support the use of the language. A first pilot evaluation was conducted. The results were promising. Furthermore, the language and its toolset are used in an interdisciplinary project [4] to specify the story-lines of a serious game against cyber-bullying. The feedback received from the collaborators of the project is promising.

Currently, a code generator for a specific game engine is under development. Future work will concentrate on extending the simulator to allow easy customization for different domains, and on mechanisms to specify other aspects of a serious games, needed for full code generation, on top of the story-lines. Also a large-scaled evaluation is planned.

References

1. De Troyer, O., Janssens, E.: Supporting the requirement analysis phase for the development of serious games for children. International Journal of Child-Computer Interaction 2(2), 76–84 (May 2014)
2. Dobbe, J.: A Domain-Specific Language for Computer Games. Master's thesis, TU Delft (2004)
3. Dondlinger, M.: Educational Video Game Design: A Review of the Literature. Journal of Applied Educational Technology 1(4), 21–31 (2007)
4. Friendly ATTAC: http://www.friendlyattac.be/en/
5. Fuchs, N., Schwertel, U., Schwitter, R.: Attempto Controlled English - not just another logic specification language. Logic-based program synthesis and . . . (1999)
6. Furtado, A.W., Santos, A.L.: Using domain-specific modeling towards computer games development industrialization. In: The 6th OOPSLA Workshop on Domain-Specific Modeling (DSM06) (2006)
7. Guerreiro, R., Rosa, A., Sousa, V., Amaral, V., Correia, N.: Ubilang: Towards a domain specific modeling language for specification of ubiquitous games. In: INForum. pp. 449–460 (2010)
8. Kuhn, T.: A Survey and Classification of Controlled Natural Languages. Computational Linguistics 40(1), 121–170 (Mar 2014)
9. Lindley, C.A.: Story and narrative structures in computer games (2005)
10. Luoma, J., Kelly, S., Tolvanen, J.P.: Defining domain-specific modeling languages: Collected experiences. In: 4 th Workshop on Domain-Specific Modeling (2004)
11. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, M.: Scratch: a sneak preview. In: Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004. pp. 104–109. IEEE
12. Marchiori, E., Torrente, J., del Blanco, Á., Moreno-Ger, P., Fernández-Manjón, B.: A visual domain specific language for the creation of educational video games. IEEE Learning Technology Newsletter 12(1), 36–39 (2010)
13. Van Broeckhoven, F., De Troyer, O.: ATTAC-L: A Domain Specific Modeling Language for Defining Virtual Experience Scenarios in the Context of Cyber-bullying Prevention - Version 2. Tech. rep., Vrije Universiteit Brussel (2012)
14. Van Broeckhoven, F., De Troyer, O.: ATTAC-L: A Modeling Language for Educational Virtual Scenarios in the Context of Preventing Cyber Bullying. In: 2nd International Conference on Serious Games and Applications for Health. pp. 1–8. IEEE (May 2013)
15. Van Broeckhoven, F., De Troyer, O.: Specifying the Pedagogical Aspects of Narrative-Based Digital Learning Games Using Annotations. In: Proceedings of the 9th International Conference on the Foundations of Digital Games. Society for the Advancement of the Science of Digital Games (2014)
16. Van Broeckhoven, F., Vlieghe, J., De Troyer, O.: Mapping between Pedagogical Design Strategies and Serious Game Narratives. In: Proceedings of the 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games). IEEE (2015)
17. Van Deursen, A., Klint, P., Visser, J.: Domain-specific languages: An annotated bibliography. Sigplan Notices 35(6), 26–36 (2000)