# Creating Story-Based Serious Games Using a Controlled Natural Language Domain Specific Modeling Language

De Troyer, Olga; Van Broeckhoven, Frederik; Vlieghe, Joachim

*Published in:*
Serious Games and Edutainment Applications

*DOI:*
10.1007/978-3-319-51645-5_25

*Publication date:*
2017

*Document Version:*
Accepted author manuscript

# Creating Story-Based Serious Games Using a Controlled Natural Language Domain Specific Modeling Language

Olga De Troyer, Frederik Van Broeckhoven and Joachim Vlieghe

**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 differences in terminology, background and the concerns of the people involved. As one step towards solving this problem, we developed a modeling language for authors of serious games to specify both the story and the pedagogical aspects of a narrative-based (i.e., story-based) serious game. The models created with the help of this language can then be processed in order to automatically generate (parts of) the game. The language is specifically designed to support the involvement of experts with a non-technical background. To achieve this, we employ a domain specific modeling language, i.e., a language specific for the domain of serious games and customizable to the terminology of the domain the serious game is dealing with. Furthermore, the language makes use of a Controlled Natural Language syntax and graphical notations. The combination of a domain specific vocabulary, a natural language syntax, and an easy to understand graphical notation allows different experts to be actively involved in the specification of the serious game, as such increasing consensus and enhancing quality. Furthermore, the model-based approach allows for a shortening of the development time of serious games (and therefore also their cost). As such, the approach tackles one of the major barriers for the development and widespread use of serious games. In this chapter,

Olga De Troyer
Vrije Universiteit Brussel, DINF - WISE, Pleinlaan 2, 1050 Brussel e-mail: olga.detroyer@vub.ac.be

Frederik Van Broeckhoven
Vrije Universiteit Brussel, DINF - WISE, Pleinlaan 2, 1050 Brussel e-mail: frederik.van.broeckhoven@vub.ac.be

Joachim Vlieghe
Vrije Universiteit Brussel, DINF - WISE, Pleinlaan 2, 1050 Brussel e-mail: joachim.vlieghe@vub.ac.be

we present a complete overview of the domain specific modeling language and the associated tools developed to support the model-based approach.

## 1 Introduction

Serious games are widely recognized for aiding the acquisition of knowledge and skills, or to induce behavior changes. Compared to learning or training in a classroom environment, serious games support knowledge, skills or performance development in a controlled and responsive environment without the barriers of time and space, while using game mechanics to make learning more fun.

However, the popularity of serious games has raised the need for dedicated development methodologies and tools that can help to reduce the development time and costs (Bellotti et al. 2010). In addition, serious games should be based on sound learning theories and instructional design principles to ensure efficient and successful training and learning. This calls for a multidisciplinary design approach and a team of experts that includes game developers, subject-matter experts, pedagogical experts, and narrative designers (Rooney et al. 2009). Unfortunately, such multidisciplinary teams often experience communication and collaboration problems due to the different terminologies, backgrounds and concerns of the people involved (De Troyer & Janssens 2014) as well as the lack of suitable design tools that allow the experts with a non-technical background to be actively involved in the design process. As stated by Djaouti et al. (2010) "people without professional game design skills, such as teachers, corporate trainers, therapists and advertising professionals, request tools that could allow them to create or modify such games".

ATTAC-L is a tool that assists multidisciplinary teams in the creation of story-based serious games. This tool and its underlying methodology are specifically designed to enable experts with a non-technical background to participate actively in the design and modeling process of story-based serious games. To achieve this, we employ a domain specific modeling language (DSML). In this case, a language that is specific for the domain of serious games and that is customizable to the terminology of the topic (i.e., domain) of the game. Furthermore, the DMSL is using a Controlled Natural Language syntax in combination with a graphical representation. The use of a Controlled Natural Language (CNL) provides an easy human-readable, yet extensible and expressive way to formulate stories and specify story elements. The combination of a domain-specific vocabulary, a controlled natural language, and easy to understand graphical notation allows different experts to be actively involved in the specification of the stories and related pedagogical issues. As such, the DSML supports collaboration during the design process and has the potential to increase consensus among the experts. In addition, it provides a means for improved monitoring of the serious game's quality by the different experts.

A story-based serious game should include a compelling narrative, but should also be based on empirically validated pedagogical methods. To accomodate for this, the modeling language provides the means to specify the story as well as the

links between the story models and the instructional design used in the serious game. In order to allow this, we introduced an annotation mechanism. Pedagogical aspects are specified as formal annotations on top of the story model. On the one hand, this allows and even gently urges designers to integrate proper pedagogical principles into the stories. On the other hand, this also helps to prevent that the specification of the learning and gaming aspects become too entangled inside the models. In other words, this way of working allows for the integration of pedagogical aspects into the story model while maintaining a clear distinction between the aspects and the narrative elements within the model. As such, different experts can concentrate on issues related to their own concerns (e.g., on the story, on how pedagogical objectives should be realized, etc.) without losing an integrated view.

The tool for designing story models with ATTAC-L is combined with others to form a model-driven authoring framework that facilitates the production of serious games at lower cost and with the active involvement of (non-IT schooled) domain experts (Van Hoecke et al. 2016). A model-driven authoring approach implies that the authors of the serious game create models, i.e., high-level conceptual specifications, which are then taken as input by tools to generate the actual game. To allow for early validation and testing of the story models, the authoring framework also provides a simulator. This is a kind of interpreter that executes the models directly, i.e., without code generation. The execution is performed, however, in a simple and predefined 3D environment with predefined Non-Player-Characters (NPCs) and predefined behaviors adapted to the topic of the serious game. In this way, the simulator can also be used as a fast prototyping tool. As such, the approach offered by the authoring framework has the potential of lowering some of the barriers that hinder the production of serious games, i.e., increasing the active involvement of experts without a technical background in IT and reducing costs.

The methodology and supporting tools for the authoring framework were developed within the Friendly ATTAC project (Friendly ATTAC 2012). This project aimed to develop a serious game for youngsters to help them deal with various cyber bullying issues. Cyber bullying, i.e., bullying via electronic communication tools, is a relatively recent phenomenon that occurs especially among early adolescents (12 to 15 year olds). Of course, the framework can also be used to develop serious games for numerous other purposes. Nonetheless, we will use fragments from the serious game developed in the context of the Friendly ATTAC project to illustrate the functionality and benefits of the DSML, i.e., ATTAC-L and the Simulator. By using the ATTAC-L tool the subject-matter experts in the Friendly ATTAC project (team members without a technical background) were able to be actively involved in the design of the serious game. It also allowed the team to make the development process much more iterative and at the same time shorten the overall development time.

The chapter is structured as follows. Section 2 provides an overview of related work. Section 3 explains the main principles of the ATTAC-L language. Section 4 discusses the different modeling concepts available in the language. In section 5, the controlled natural language syntax of ATTAC-L is provided and explained, and in section 6, we show how the syntax is mapped onto a graphical notation to

turn the language into a graphical modeling language. Section 7 demonstrates the use of the DSML for an example serious game. In section 8, we explain how the designed pedagogy can be explicitly linked with the narrative. Section 9 elaborates on the different tools developed to support the ATTAC-L language. In section 10, we present our experiences in using the ATTAC-L tool in the context of the development of a serious game against cyber bullying. Finally, section 11 presents conclusions and future work.

## 2 Related Work

In this section, we discuss related work that deals with the modeling and authoring of story-based serious games and compare these with our own work. Various authoring tools have been created for designing story-based serious games, such as interactive digital storytelling tools, e.g., StoryTech (Göbel et al. 2009), <e-Adventure> (Torrente et al. 2008), EDoS (Tran et al. 2010), and StoryBricks (StoryBricks 2014). In addition, several DSMLs have been developed and used for the same purpose, e.g., WEEV (Marchiori et al. 2003), Inform (Nelson 2006) and GLiSMo (Hirdes et al. 2012).

The 80Days project aims to establish a generic theoretical basis for immersive storytelling merged with cognitive, motivational and emotional aspects of learning processes. The StoryTec authoring tools (Göbel et al. 2009) were extended in the context of the 80Days project to enable the specification of adaptation and personalization aspects for targeted digital educational games. Nonetheless, the Story Editor tool is still using the same visual language that consists of story units (i.e., scene and complex scene visualized as rectangles) and transition between the different connected units that are visualized as arrows. There is also the possibility to create scenes that are not connected to each other. Such scenes will be selected during the runtime based on the adaptation mechanism. The author can define the expected time that the learner will stay in a scene. Furthermore, the author could identify the skill, tasks and goals to be achieved in the scene. StoryTec does allow to create relationships between different parts of the story and the associated learning objectives and goals, however direct links between high-level pedagogical strategies and low-level game mechanics cannot be established as we do in our approach. Also, in contrast to our DSML, support for code-generation out of story models is not provided.

<e-Adventure> (Moreno-Ger et al. 2008) is a platform for designing adventure games of the point-and-click style that are mostly used for educational purposes. The goals of the platform it is to enable people to create games without the necessity of possessing programming skills. For this purpose, <e-Adventure> provides an authoring tool (Torrente et al. 2008) which includes a mechanism for creating characters by importing photos of individual characters taken from various angles. Furthermore, the tool also includes mechanisms for creating items, conversations and cut scenes, as well as a mechanism for importing pictures that will represent the

scenes of the game. As such, the story of the game is basically represented as a sequence of scenes with the characters of the game positioned in them. The player of the game will interact with each scene by clicking on specific active parts defined by the designer to trigger actions. The relationships between the scenes are defined as connections represented by lines in the authoring tool. The story of the game is then narrated to the player through pieces of text or audio fragments. In contrast to our approach, <e-Adventure> is a user-oriented toolset for creating educational games, rather than a DSML. This makes the <e-Adventure> tool more dependent on its targeted game environment and limits the end-user to the creation of games of the point-and-click genre. This type of games only offers limited support for the kind of behavior change that we targeted in the Friendly-ATTAC project. Our language can be used with a broader range of game types and platforms for educational game platforms.

EDoS or Environment for the Design of Serious Games (Tran et al. 2010) is an interactive authoring environment for serious games that also aim to integrate educational strategies into the narrative by explicitly linking pedagogical design principles to particular elements of the narrative. Its purpose is similar to ours: to help an interdisciplinary team in designing a serious game by offering a number of standardized steps, starting with the formulation of pedagogical objectives and continuing all the way up to the point of elaborating a scenario and modeling user interactions. The outcome of following these standardized steps is "a structured scenario that will be automatically executed by an engine" (Tran et al. 2010, p. 393). EDoS focuses on the reusability of available components of different granularity and the creation of serious games for teaching engineering skills. The design process of EDoS builds on 3 models. The first one is a model of the targeted pedagogical objectives, e.g., professional competences for an engineer. The second model relates pedagogical objectives and pedagogical activities in order to construct pedagogical scenarios for serious games. These scenarios are created using an adapted version of the Instructional Management Systems – Learning Design language (IMS-LD) (Koper & Olivier 2004) which only describes the pedagogical content of the serious game. The third model helps to include the entertaining elements, i.e., the task model that describes the screens with which the users will interact. In contrast to our approach, EDoS relies on a specific learning design, namely IMS-LD and thus provides limited flexibility in this respect.

The StoryBricks framework is an interactive story design system that was discontinued in 2014 (StoryBricks 2014). It provides a visual language based on the visual programming language Scratch (Resnick et al. 2009) designed by the MIT lab. Without the need for programming skills. The designers do not need programming skills to edit the characters in the game and the artificial intelligence of the game that drives the characters. The designers can set up characters' inventory, needs and emotions by using so-called story bricks. The bricks can also be used to specify what is 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 interaction between the characters in the game without being programmed explicitly. The StoryBricks approach allows a

great deal of flexibility in defining the rules for the game logic, but a story cannot be modeled explicitly. A user experiment performed in the ATTAC-L project (see section 3) showed that an event-based approach like this would be less suitable for our target group (experts without programming knowledge). Our work has adopted the brick concept as basic building block for our language, but we require the designers to model the flow of the story explicitly. Moreover, we provide a mechanism to model the pedagogical aspects of games.

WEEV or Writing Environment for Educational Video Games (Marchiori et al. 2011) 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> . In WEEV, 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 the overall complexity WEEV has language constructs that help to organize the structure. Whereas WEEV uses a state-transition approach, we use a flow-based approach. As already mentioned, this decision was informed by the results of a user experiment that we performed. Moreover, we impose a strict separation between the specification of the narrative and the pedagogical aspects, while both aspects are interwoven in WEEV.

The GLiSMo language or Serious Game Logic and Structure Modeling Language (Hirdes et al. 2012) 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, i.e., a block representing a single atomic step that can be executed in the context of an educational game-environment. This can be related either to a logical or a pedagogical functionality of the game. The bricks have input- and output ports. The overall game logic is modeled by linking several bricks through these ports. This interlinking defines a temporal relationship and data flow between the bricks, giving the model a data flow -based structure. An abstraction mechanism is provided in the form of a serious game composite which is used in the same way as a brick but encapsulates one or more interlinked bricks. As a consequence, the composite provides a way to organize more complex models. Our research, developed in parallel, uses similar principles. We have opted, however, for an explicit flow-based structure that only requires designers to define temporal relationship between game moves. Pedagogical aspects are expressed using annotations, which allows for a better separation of concerns (SoC) (Hürsch & Lopes 1995).

Inform (Nelson 2006) 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 setting (i.e., scene), character setup, and story flow. The DSML uses a CNL. In contrast to Inform, we opted for a graphical language as most DSMLs do. Also, our DSML does not allow designers to define aspects such as environment settings and low-level implementation aspects. Instead, it focuses on the specification of the narrative and the educational aspects, thereby reducing the complexity and increasing the understandability. In our approach, complementary tools specify these kinds of aspects.

# 3 Principles of the Language

## *3.1 Domain-Specific Modeling Language*

ATTAC-L is a Domain-Specific Modeling Language (DSML). A domain-specific language is usually a small language, dedicated and restricted to a particular domain (Deursen et al. 2000). It provides abstractions that make it easier and less time consuming to specify solutions for a particular class of problems in the domain. The final system is then generated from these high-level specifications (i.e., models) (Kelly & Tolvanen 2007).

By using suitable abstractions and building on the vocabulary of the problem domain, a domain-specific language enables domain experts to understand, validate and often even develop specifications autonomously. Luoma et al. (2004) showed that DSMLs require less modeling work and that this modeling work could often be carried out by persons with limited programming experience. They found a clear productivity increase.

## *3.2 Controlled Natural Language*

Various authors have presented arguments in favor of using a DSML for modeling (serious) games (e.g., Dobbe (2007), Furtado & Santos (2006), Guerreiro et al. (2010), Marchiori et al. (2011)). However, for most DSMLs, the gap between the user's mental model and the syntax of the DSML is still big. For instance, Marchiori et al. (2011) is using state transition diagram principles. The DSMLs proposed by Dobbe (2007), Guerreiro et al. (2010) and Furtado & Santos (2006) are still more oriented towards game developers than towards experts with a non-IT background. Contrary to these approaches, our DSML uses a Controlled Natural Language (CNL) (Wyner et al. 2010), which is a strict and controlled subset of natural language. Using a CNL syntax for our DMSL provides an easy and human-readable, flexible and expressive way to specify the story of the game. This makes it significantly easier for people without programming knowledge to understand as well as to create models. In this way, collaboration within multidisciplinary teams can be better supported. Compared to the use of natural language, a CNL approach offers the advantage that it still allows for fully automatic processing of the specified models needed for our model-driven approach.

## *3.3 Flow-Oriented Modeling*

ATTAC-L's CNL provides the necessary means to model the narrative of a serious game. The story is specified as a number of interactive scenarios. A scenario is

a flow-oriented specification of the different possible actions (called game moves) and choices in the narrative. The result is called a story model.

We have opted for a flow-oriented specification based on a small user experiment focused on determining which modeling approach was most convenient for people without a programming background. In this study, we compared three approaches: a state-based, a flow-based, and an event-based approach. The participants (7 males and 13 females between the age of 25 and 36, all without programming knowledge) were presented with scenarios using the three approaches. Inquiries were made about the ease of use, convenience, comprehensibility and the general preferences of the participants. The flow-based representation was clearly the most convenient one, followed by the rule-based approach, and the state chart, which was almost unanimously marked as the least favourite one.

### *3.4 Graphical Language*

A DSML is often graphical in nature, i.e., using visual notations. Well-designed visual notations are known to be more accessible for people without a technical background as they allow them to grasp large amounts of information more quickly than large listings of textual specifications (Moody 2009).

With this in mind, ATTAC-L also uses graphical notations. This means that the natural language sentences used to describe the narrative of a serious game (and expressed in the CNL syntax of ATTAC-L) are expressed using visual notations instead of plain text. The main graphical construct is the 'brick' concept, adopted from Storybricks. Bricks are used for a wide range of things, from modeling the stories by expressing narrative events or specifying the control of the interactivity, to specifying the overall story flow or even the pedagogical aspects of the serious game.

### *3.5 Open Narrative*

The CNL syntax used for ATTAC-L provides a mechanism for creating open narratives. This means that the CNL syntax does not required every involved game entity to be directly identified while the actions are specified. Instead, the syntax allows for the specification of interactions between highly generic described game entities, based on type, property, state or a combination of these. During gameplay, a specific instance of a narrative conforming to this general description is generated, resulting in a slightly different progression of the game narrative on each run. The advantage of this is that the game can be played several times without becoming too predictable.

# 4 Modeling Concepts

This section discusses the different modeling concepts available in the language for modeling a story-based serious game. We first describe the modeling constructs for specifying the game narrative. Next, we outline the mechanism for specifying and integrating other game aspects such as pedagogical aspects into the story model.

## *4.1 Modeling Concepts for the Game Narrative*

### 4.1.1 Game Moves

To model a narrative, we use the concept of a game move. Lindley (2005) defines a game move as a single step or turn taken by any player at any time during the execution of a game. However, in the context of ATTAC-L, a game move represents one individual step in the game narrative, performed either by the player or a non-playable character (NPC).

Modeling a narrative entails defining game moves and linking them to each other to denote their relative order in the flow of the story. Game moves are specified using the CNL syntax (see section 5 for its definition).

### 4.1.2 Bricks

Because we are using a CNL syntax, game moves are expressed as natural language sentences. But as ATTAC-L is a graphical modeling language, the game move sentences are composed in a graphical way by connecting bricks (Van Broeckhoven and De Troyer 2013) (see Fig. 1). In the context of a game move, a brick corresponds to a meaningful unit in the story: an act to be performed, a tangible object that can perform or undergo the act, a state, or a value. A brick is graphically represented as rectangle containing a word or word-group (see Fig. 2a). Bricks must be interconnected according to the rules of our controlled natural language (see section 5) (Van Broeckhoven et al. 2015*b*). The result is a construct that reads as a simple sentence and denotes a game play activity (see Fig. 1). The bricks used to construct the game moves are so-called game-move-bricks.

For composing game moves into a narrative, i.e., expressing their relative order, we provide the following common control structures: sequence, choice, and concurrency. Sequences are used to signify that game moves follow each other, thus resulting into storylines. Choice is used to express alternative storylines, i.e., branching. Concurrency is used to indicate that storylines should be performed in parallel. In addition, ATTAC-L provides an extra control structure to increase the language's expressiveness, called 'order independence'. This control structure allows modelers to specify that different storylines must all be performed regardless of the order. To avoid any link with programming constructs, the control structures are also visu-

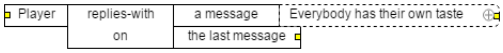
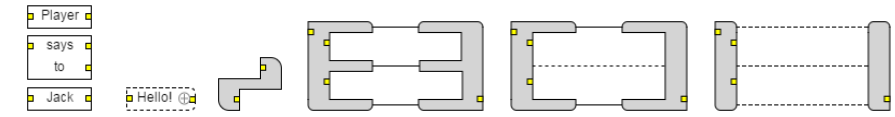



Fig. 1: A graphical representation of a game move. This representation reads as follows: “Player replies-with a message ‘Everybody has their own taste’ on the last message” and is composed of 5 bricks.

ally represented by bricks, called control-bricks. Figs. 2c, 2d, 2e and 2f show the graphical representations of the control bricks. Fig. 3 illustrates their use.

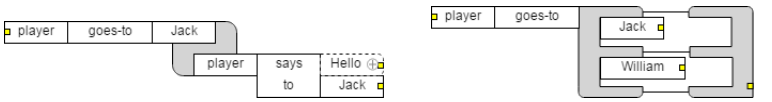



(a) Game-move (b) Value (c) Sequence (d) Choice (e) Order ind. (f) Concurrence

Fig. 2: Several types of bricks

A sequence of game moves and/or control bricks is called a story-flow. Note the difference between a storyline and a story-flow: the former only involves sequences of game moves, and thus literally represents a single linear ‘line’ of progression of the narrative, while the latter may also include control structures, i.e., choice, concurrency and/or order independence, and thus can express different alternatives of how a story could evolve during a performance.

Next to the regular game-move-bricks and control-bricks, there are also two other types of bricks: scenario-bricks and annotation-bricks associated respectively with the modeling concepts scenarios and annotations. These are explained in the next sections.




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

Fig. 3: Connecting bricks to form storylines and story-flows

#### 4.1.3 Scenarios

It is not uncommon that stories result in vast models with rich and complicated story-flows. To provide a structuring mechanism for such story-flows, the concept of scenarios was introduced. It allows for the decomposition of a story model into smaller logical units. The principle of using scenarios is analogous to the structuring of theater and film scripts whereby the story is divided into separate scenes. In contrast to scenes in theater and films scripts, scenarios in ATTAC-L can be reused in various places throughout the story model. To accommodate reusability, scenarios are separately defined and given a name that acts as a placeholder for the content of a scenario in the story-flow. The scenarios are used in a story-flow by means of a scenario reference-brick or shortly scenario-brick. Upon encountering a scenario-reference-brick during interpretation of a story model, the content of the referenced scenario is inserted as if it was an integral part of the story-flow. Note that scenarios can also be nested inside other scenarios.

An example of a simple story model is given in Fig. 4. The example depicts a scenario from the Friendly ATTAC serious game named 'Direct minor male Ugly and Stupid'. In this scenario, the player is expected to react adequately on a cyber bullying situation. The scenario is composed of two nested sub-scenarios: one in which the actual bullying situation occurs ('Ugly and Stupid' defined in the middle of the figure) and another in which the possible reactions of the player are listed ('Player chooses Ugly and Stupid' defined at the bottom of the figure). The actual scenario is defined by 'chaining' the two sub-scenarios. This is done by connecting the corresponding scenario-bricks with a sequence-brick (top of the figure). The two sub-scenarios are discussed in more detail in the section 7.

### *4.2 Non-Narrative Modeling Concepts: Annotations*

The concept of annotations enables the modelers to specify additional information to related parts of the story model, e.g., pedagogical relevant information and interventions, important gameplay aspects, or noteworthy visual and behavioral aspects of the environment (Van Broeckhoven et al. 2015*a*). The annotations add this information on top of the story model. This prevents that the information becomes too entangled with other aspects of the serious game. Pedagogical annotations, for example, enable the modelers to relate the pedagogical aspects of the serious game to the story-flow while maintaining a clear separation from the narrative content.

Annotations are represented graphically by means of small, square-like bricks called annotation-bricks. Annotation-bricks are "attached" to game moves or scenarios (depending on their type). The content of an annotation will pop up in the ATTAC-L tool when an annotation brick is clicked (described in more detail in the section Tool Support) (see Fig. 5 for an example).

Each annotation-brick contains an icon that indicates its type. The type of the annotation determines the structure of the content of an annotation. Currently, we dis-

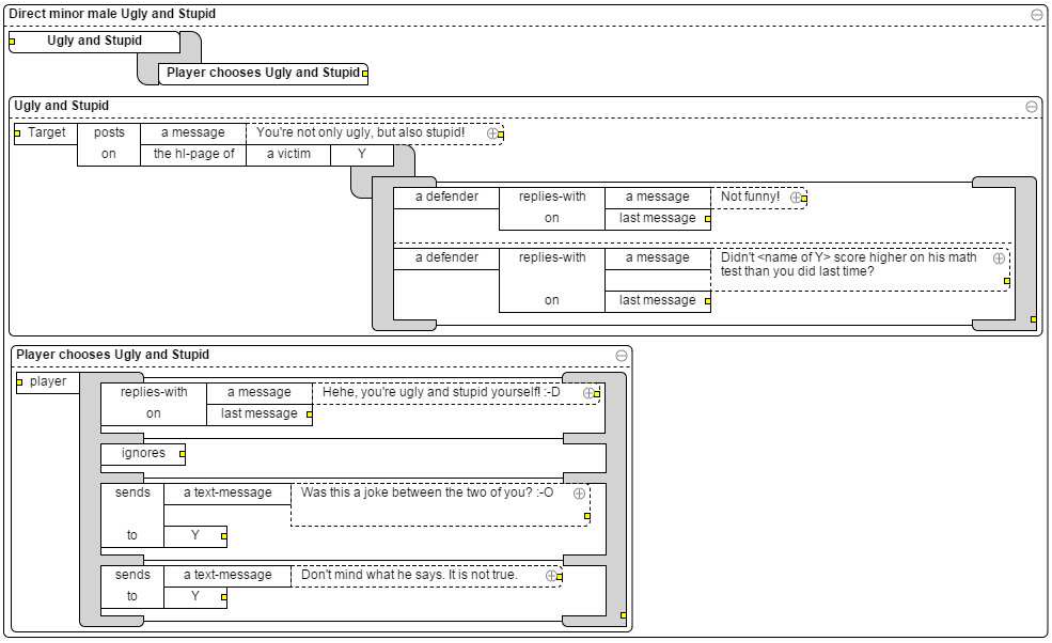



Fig. 4: Example storyline model divided into scenarios.

tinguish between two main types of annotations. The first type are the pedagogical annotations (PA) which are used for specifying pedagogical aspects of the serious game. The second type are the gameplay annotations which are used for specifying aspects related to gameplay, such as the game environment (e.g., for indicating a change of the scene) and NPCs (e.g., emotions that should be expressed or behavior that should be displayed). Note that this annotation system is extensible in the sense that other types of annotations can be defined and used when needed. For instance, the language can be extended with annotations to indicate specifications related to the mode(s) of interaction. An overview of the current annotation classification system is given in Fig. 6.

For the purpose of this chapter, we concentrate on the PAs because of their specific relation to the domain of serious games. PAs are divided into action PA, objective PA, pedagogical theory PA, and method PA:

- Action PAs are used to specify pedagogic actions that should be performed at particular moments in the story, for instance: providing additional information, giving assistance or feedback. In other words, action PAs are concrete pedagogical interventions. Action PAs are associated with a particular game move and

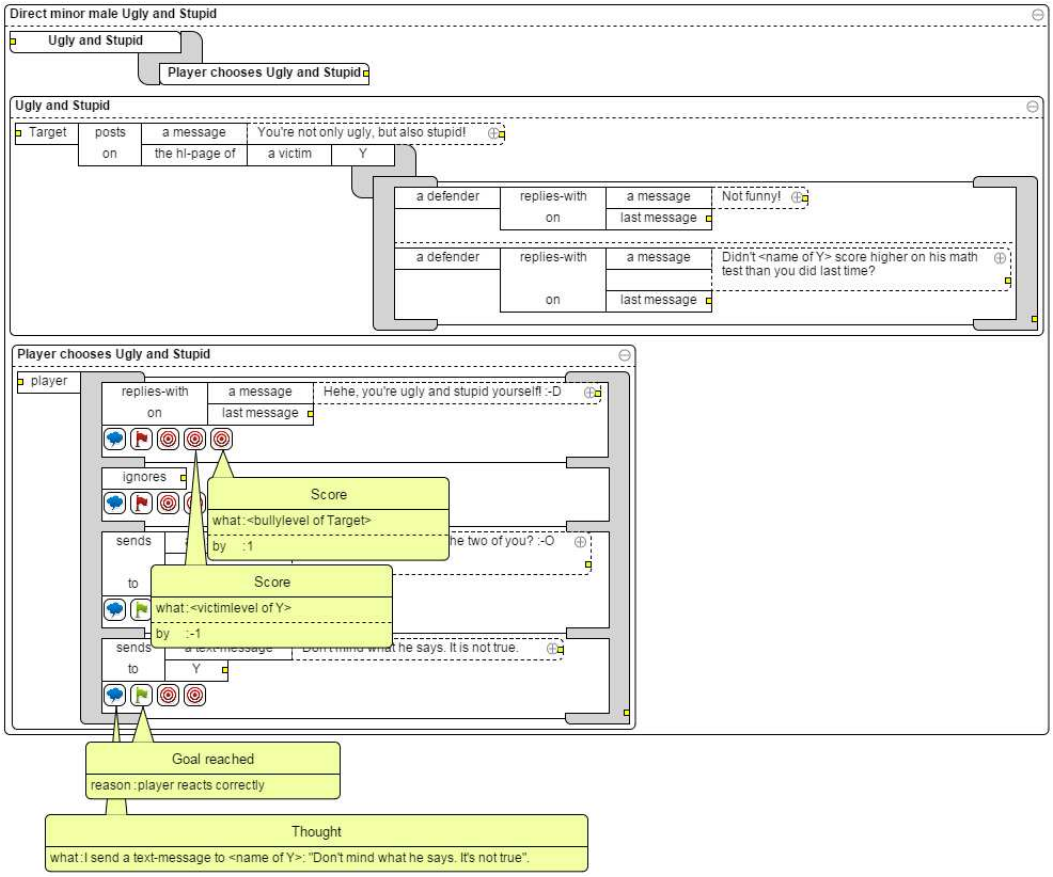



Fig. 5: Annotated storyline model from figure 4.

indicate that these interventions should be performed simultaneously with the game move. See Fig. 7a for an example of an assist PA.

- Objective PAs are used to explicitly relate pedagogical objectives, such as learning goals or behavioral change objectives, to scenarios or parts of the story-flow. An example of such an objective would be 'to know the multiplication tables of 1 to 10' or 'to understand the impact of cyber bullying'. They can only be associated with scenarios. See Fig. 7b for an example of an objective PA.
- Pedagogical theory PAs are used to specify the underlying pedagogical theory that is applied in order to achieve the pedagogical objectives (e.g., behaviorism, constructivism, ...). These PAs are attached to a complete story model or a scenario. See Fig. 7c for an example of a theory PA.

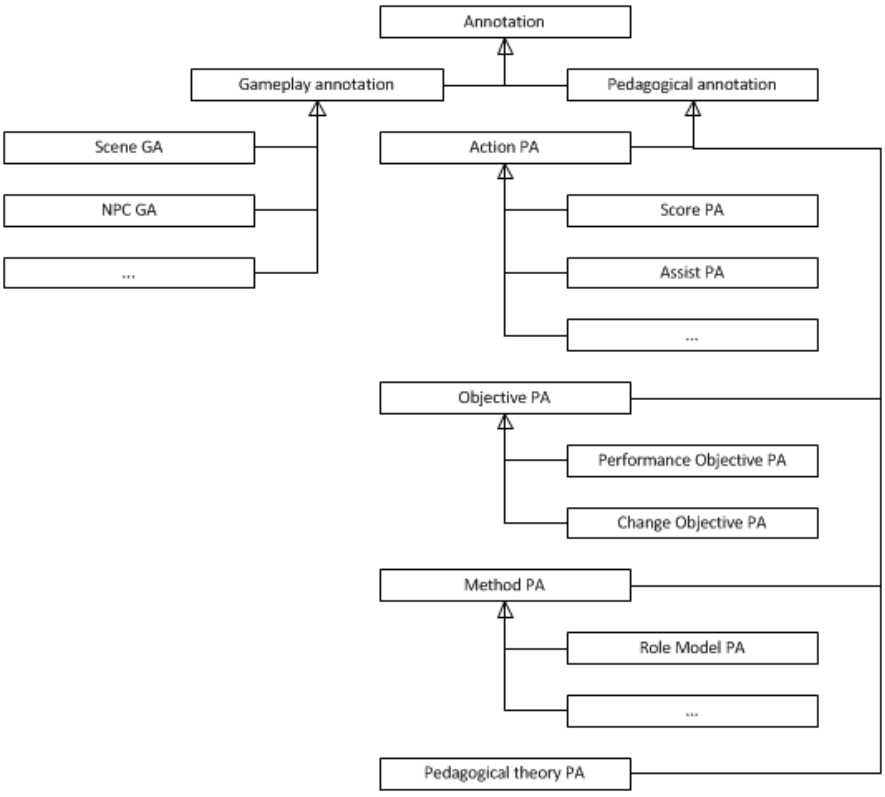
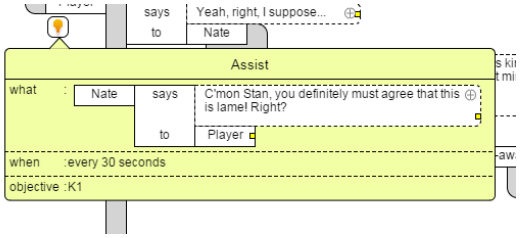
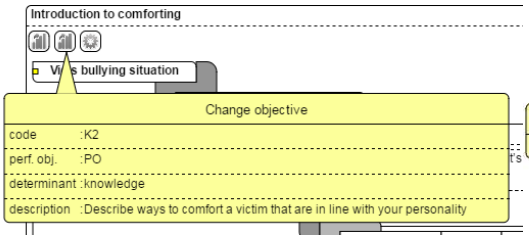



Fig. 6: The annotation classification.

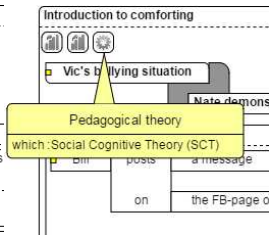
- Method PAs are used to specify the particular pedagogical methods used to reach particular pedagogical objectives in the story or the scenarios. Examples of such methods are drill & practice, problem-based learning, or learning-by-doing. Since many different methods are possible - each with different characteristics - this annotation type is an abstract one (cf. abstract class in UML), i.e., we cannot define all of its properties. Concrete subtypes can be defined for different methods. For instance, the role-modeling PA has been defined for the 'modeling' or 'observational learning' principle used in the Social Cognitive Theory (SCT) (Bandura 1991). Similar to Objective and Pedagogical PAs, Method PAs are also attached to scenarios.




(a) Pedagogical Action Annotation




(b) Pedagogical Objective Annotation




(c) Pedagogical Theory Annotation

Fig. 7: Several types of annotations (close-ups taken from Fig. 18)

## 5 ACE-Based Syntax

As motivated in the introduction, we use a CNL for the syntax of the game moves. The CNL used is a subset of Attempto Controlled English (ACE) (Fuchs et al. 1999). We opted for ACE because its expressiveness is well suited for our purpose and because it has a solid foundation, namely: first order logic. ACE expressions are formulated like plain English sentences written in the third person singular simple tense. They describe logical terms, predicates, formulas and quantification statements. Furthermore, ACE defines two word classes: function words (determiners, quantifiers, negation words, ...) and content words (nouns, verbs, adverbs and prepositions).

The use of ACE for the syntax of game moves implies that game moves are expressed in 3$^{rd}$ person simple tense. This implies that narratives are modeled as if they were told from a narrator's point of view. Similar to ACE, the ATTAC-L syntax distinguishes between function words and content words. Our set of content words consists of nouns, verbs, and adjectives. These correspond respectively to entities, acts, and states in the story-flow. Our set of function words consists of determiners, negation words, pronouns, and the copula 'to be'. Unless otherwise specified, words are always written in lower case letters. Also in accordance with ACE, 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 called a noun phrase, while the latter describes the action itself and is expressed as a verb phrase.

The overall syntax for game moves in Extended Backus Naur Form (EBNF) (ISO 1996) is as follows:

```
game-move       : subject predicate;
subject         : noun-phrase;
predicate       : verb-phrase;
```

In the following sections we describe the formal syntax for the different types of phrases. In the EBNF excerpts, the formal definitions of some general expressions have been omitted for simplicity. This includes rewrite-rules for the most basic elements that are part of a game move expressions, such as those for noun, pronoun, verb, value etc. They amount to regular expressions for words, numbers, quoted strings, and more. The following listing summarizes those elementary parts and describes how they should be interpreted:

- **noun**: a lowercase word. Semantically, this can be any *singular common noun* as determined by the set of content words.
- **nouns**: a lowercase word. Semantically, this can be any *plural common noun* as determined by the set of content words.
- **proper**: a lower or uppercase word, but always starting with an uppercase character. It may contain dashes ('-'). Semantically, this *may* refer to predefined proper nouns contained in the set of content words.
- **adjective**: a lowercase word. Semantically, this can be any *adjective* contained in the set of content words.
- **verb**: a lowercase word. Semantically, this can be any *3rd person singular verb* as determined by the set of content words.
- **passive-verb**: a lowercase word. Semantically, this can be any *past participle form of a verb* as determined by the set of content words.
- **value**: any of:
  - quoted string
  - number
  - sequence of values separated by comma (','), enclosed by brackets ('[]')
  - sequence of key-value assignments separated by comma (','), enclosed by braces ('{}')
- **determiner**: the set of all supported determiners as defined by the set of function words ('a', 'an', 'one', '1', ...)
- **quantifier**: the set of all supported quantifiers as defined by the set of function words ('some', 'all', 'two', '2', ...)
- **copula**: 'is' or 'are'
- **preposition**: the set of all supported quantifiers as defined by the set of function words ('in', 'to', 'out', 'until', ...)

## *5.1 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 the player is controlling.

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

Variables are a way to assign a proper name to a countable common noun phrase, for instance 'a person Mr-X'. In this case, the proper name can be used to refer to the same entity in subsequent game moves. This corresponds to an inline and implicit declaration of a variable in programming languages. For people without programming knowledge, this approach is more natural and likely easier to grasp than the use of explicit variable declarations.

Adjectives can be used to make a countable common noun phrase more specific. As such, the set of entities out of which a specific noun (i.e., game entity) is selected at run-time can be narrowed down. Adjectives can be positive, superlative or conjoined (e.g., 'the last person', 'two highest trees', 'a sad and angry person').

Genitives 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 ', for example 'the back-pack of Jack' or '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 have refrained from introducing extra prepositions in order to keep the syntax of ATTAC-L as simple as possible.

Values (text, numbers, lists and parameter-value tuples) 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 directly by prepending a common noun (e.g., 'a message "Hello!" ').

The formal syntax for the noun phrases is as follows:

```
noun-phrase          : proper
                     | value-noun-phrase
                     | common-noun-phrase
                     | common-noun-phrase , 'of' ,
```

```
                    sing-common-noun-phrase;
value-noun-phrase   : value
                    | determiner noun value;
common-noun-phrase  : sing-common-noun-phrase
                    | mult-common-noun-phrase;
sing-common-noun-phrase
                    : determiner, [ adjective ], noun,
                      [ proper ];
mult-common-noun-phrase
                    : quantifier, [ adjective ], nouns,
                      [ proper ];
```

## *5.2 Verb Phrases*

A verb phrase corresponds to the predicate of a sentence and describes the actual activity performed in a game move. Generally, a verb phrase starts with the conjugated verb. Depending on the type of verb 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 transitive verb involves only one direct passive object (e.g., 'to see something '), whereas a di-transitive verb involves both a direct and indirect passive object (e.g., 'to give something to somebody '). Some verbs can also be associated with a preposition. In this case, the combination verb-preposition is considered as a whole and is written as a hyphenated combination (e.g., 'X walks-to Y' or 'X looks-at Y').

The formal syntax for the verb phrases is as follows:

```
verb-phrase         : verb
                    | verb, noun-phrase,
                      [ preposition, noun-phrase ];
```

## *5.3 Adjective Phrases*

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 copular verb 'to be' serves a special purpose. 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, for example 'X is happy' or 'X is angry-with Y'.

The formal syntax for the adjective phrases is as follows:

```
adjective-prase      : adjective, [ noun-phrase ];
```

The formal syntax for the verb phrases, also taking adjective phrases into account is as follows:

```
verb-phrase          : verb
                     | verb, noun-phrase,
                       [ preposition, noun-phrase ]
                     | copula, [ 'not' ], adjective-prase;
```

## *5.4 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’ or ‘there are two persons The-Johnson-Brothers’). The modeler can express any sentence containing a transitive or di-transitive 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. 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 where the character that performs the action is unknown, i.e., when no subject would be present in the active equivalent of the sentence, for instance ‘an item of Player is stolen’. The formal syntax for the special sentences is as follows:

```
passive-verb-phrase : copula, passive-verb,
                        [ 'by', noun-phrase ]
                    | copula passive-verb noun-phrase,
                        [ 'by', noun-phrase ];
definition          : 'there', 'is', determiner, noun, value,
                        proper
                    | 'there', 'are', quantifier, nouns, value,
                        proper;
```

## 6 Mapping the Syntax to Bricks

The CNL-based game moves are expressed by means of our graphical language constructs, i.e., bricks. The game-move-bricks contain words or word-groups and are connected to each other in accordance with the syntax rules of the language.

A game-move-brick containing a noun phrase is called a noun-brick and refers to a tangible entity, i.e., an object, an NPC or a player (see Fig. 8a for examples).

Adjectives used for noun-phrases are contained in the noun-brick because they are part of the noun-phrase (see Fig. 8b for an example). 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. 8c).

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. 8d). This allows the variable to be used independently in subsequent game moves.

Values are represented by value-bricks (see Fig. 8e) and can only be connected to the end of a game move. Visually, they are delineated by a dotted line, whereas regular game-move-bricks have a solid border. As values can be text and numbers, but also lists and parameter-value tuples, value-bricks should support the specification of complex data. Therefore, value bricks can contain multiple lines or key-value combination (see second example in Fig. 8e).

A game-move-brick containing a verb is called a verb-brick. It connects to noun-bricks and/or adjective-bricks. In the case of a di-transitive verb, the associated preposition serves as an extra connecting point for the indirect passive object (see Fig. 9 for illustrations of the most common cases).

As explained, game moves containing a transitive or a di-transitive can also be expressed in a passive form (Fig. 9f represents the passive equivalent of Fig. 9e). In passive sentences, the placements of the indirect passive object and the direct passive object (i.e., the original subject) can be interchanged (Fig. 9f and 9g can be used interchangeably to represent the same game move). In some cases, the 'by' part of the sentence can be omitted (see Fig. 9h). The subject of the game move is then assumed to be any NPC.

## 7 Overall Example

To illustrate the use of the language for a more elaborated narrative, we provide fragments from the serious game developed in the context of the Friendly ATTAC project. The overall story of the game is composed out of several sub-stories, which are all modeled as separate scenarios. Annotations are used to associate several performance metrics to the scenarios that are tracked as the player progresses through the game. Based on the player's progression, the overall game narrative is adapted by the runtime environment to the player's performance.

The first example is a scenario in which the player is expected to react adequately on a cyber bullying situation (introduced earlier in section 4.1.3 see Fig. 4). The

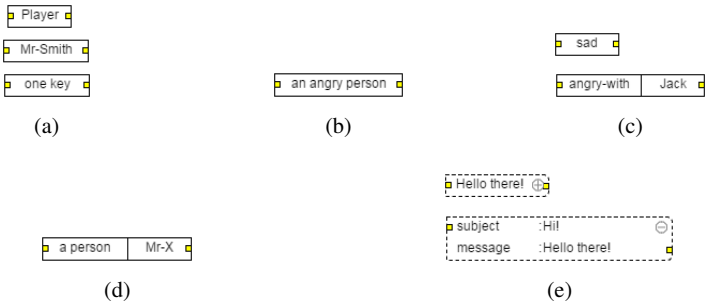



Fig. 8: Noun and adjective phrases

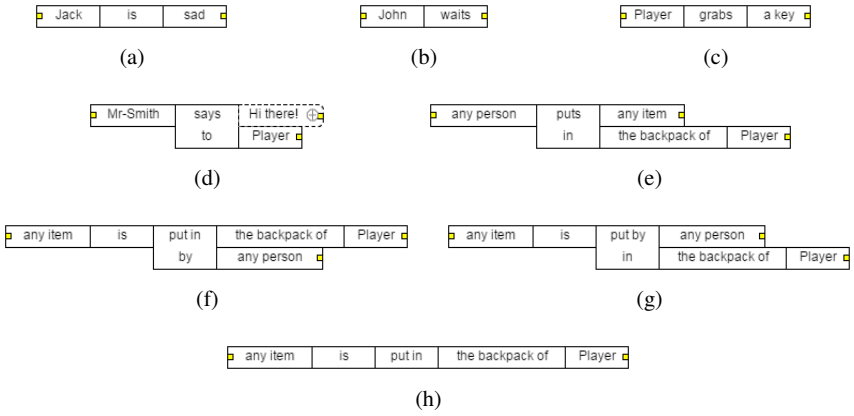



Fig. 9: Verbs connecting noun and adjective phrases, forming game moves

scenario is composed of two nested scenarios: one in which the actual bullying situation occurs (‘Ugly and Stupid’), and another in which the possible reactions of the player are enlisted (‘Player chooses Ugly and Stupid’). Both sub-scenarios are ‘chained’ by connecting the corresponding scenario-bricks with a sequence-brick (at the top of Fig. 4).

The ‘Ugly and Stupid’ scenario starts with a bully (referred as ‘Target’) posting an insulting message on the in-game social profile page of some NPC, referred as ‘a victim Y’. The noun ‘a victim’ indicates that any in-game character that fits the role of a victim could be selected (at runtime) to fulfill the role of the targeted NPC. The pronoun ‘Y’ is used to name this selection. This allows the modelers to refer to the selected character further on in the scenario simply by means of ‘Y’. The use of this construct (‘a victim Y’) allows the game to select different NPCs in each run. This will reduce predictability and increase the replay value of the game.

Next, two other characters (defenders) express their disapproval of the message that has been posted. The noun 'a defender' has a similar purpose as the victim noun, but in this case, at runtime the game will select NPCs that are inclined to react against bullying situations. The adjective 'last' in 'the last message' allows the modelers to refer to the last specified entity indicated by the noun, here: the message that was posted by Target. Both game moves are combined with an order-independence structure, indicating that both should occur, but the order is unspecified because it is irrelevant.

The 'Player chooses Ugly and Stupid' scenario specifies the choices that will be presented to the player. The use of 'Player' as subject indicates that for this game move, interaction by the player is required. In this case, the player has the choice between 4 options: insulting the bully, ignoring the situation, or two ways of comforting the victim.

## 8 Linking Narrative and Pedagogy

During the design of a serious game, designers are not only faced with the challenge of creating a compelling narrative, but also with the additional challenge of incorporating suitable learning theories and pedagogical methods into their narratives in order to ensure that the game can achieve the defined objectives. To create the link between the story model and the pedagogical methods developed for the serious game, we have introduced the concept of annotations, in particular: a collection of PAs (for the overview of the PA types, see section 4.2).

It is not possible to provide a single recipe for integrating an instructional design strategy into a serious game as each learning theory and pedagogical method has its own unique principles. In (Van Broeckhoven et al. 2016), a general description of the process of linking a game's narrative with pedagogical theories and methodological design strategies has been elaborated and illustrated with two specific cases, namely: Social Cognitive Theory (SCT) (Bandura 1991) and the Intervention Mapping Protocol (IMP) (Eldredge et al. 2011).

In this chapter, we illustrate how narrative and pedagogical aspects have been linked in the serious game developed for the Friendly ATTAC project. To come to a well-grounded and effective game, the Friendly ATTAC team used the Intervention Mapping Protocol (IMP). IMP has been developed to aid in the systematic planning and design of behavioral change programs focused on health issues. IMP investigates the behaviors that can help to reduce the targeted problem. These behaviors are set forth as performance objectives. Furthermore, change objectives must be established. These express what needs to be changed in order to achieve the performance objectives. The change objectives present the basis for the development of the actual pedagogical interventions in the serious game.

An example of a performance objective for the cyber bullying program developed in the Friendly ATTAC project is: 'always comfort the victim'. For this performance

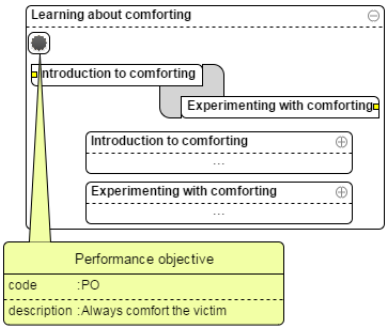



Fig. 10: A scenario with a performance objective 'always comfort the victim' and two sub-scenarios (both detailed in Fig. 16 and Fig. 17).

objective, the following change objective was defined: 'Recognize that by comforting the victim, you are making the victim feel better'.

To illustrate how the linking is achieved, we use the story model given in Fig. 10. The scenario that we use here focuses on the performance objective 'always comfort the victim'.

To specify which part of a story model is dealing with a particular performance objective, we use a special objective PA called a performance objective PA. This type of annotation is associated to a scenario and has a parameter to state the performance objective. Fig. 10 shows an example of a performance objective PA. The performance objective is 'always comfort the victim'. In this example, the scenario is divided into two sub-scenarios, each dealing with one or more change objectives.

To indicate with which change objectives the sub-scenarios are dealing, the scenarios are annotated with another kind of objective PA called change objective PA. This type of annotation has parameters to refer to the related change objective, to its encompassing performance objective, and to the determinants it is expected to change. The parameters also include a description (for documentation purposes). The sub-scenarios of the scenario 'Learning about comforting' used in Fig. 10 are 'Introduction to comforting' (elaborated in Fig. 16, fully annotated in Fig. 18) and 'Experimenting with comforting' (elaborated in Fig. 17, fully annotated in Fig. 19). The first one is annotated with change objectives related to the determinant 'knowledge'. This implies that the scenario should contain intervention methods specifically tailored to help increase the player's knowledge about comforting a victim. The second sub-scenario focuses on the determinants 'outcome expectations' and 'perceived social norms' and has two change objectives. This sub-scenario should thus include intervention methods that affect these determinants with the aim of achieving the change objectives.

Intervention methods identified with the help of IMP can be embedded in a story-flow in two ways: (1) as game mechanics that are complementary to the story-flow,

or (2) as events expressed directly in the story. In order to express that an intervention method is operationalized through game mechanics, action PAs are used. To express the link with the change or performance objective, an action PA includes an argument that denotes the targeted objective. In Fig. 19, a score PA (special kind of action PA) is used to express that a scoring mechanism is used as an intervention method for achieving the change objective 'Sn1: Recognize that your friends expect you to comfort or provide advice to the victim'. Similarly, the same figure uses an assist PA (special kind of action PA) to help achieve the change objective 'Oe1: Expect that by comforting the victim, he/she will feel better'. This annotation expresses that at this point in the story-flow some assistance must be provided when the player takes too long to make a choice. As illustrated by the example, this can be done in the form of an NPC that makes the player aware that a victim will feel better when he comforts him, thus 'directing' the player to making the right choice.

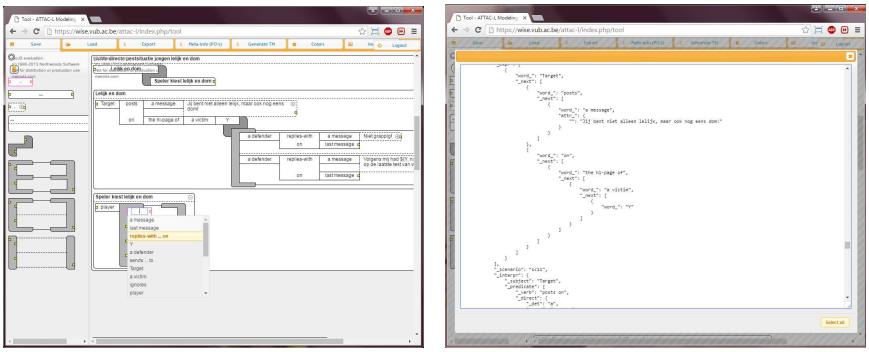
## 9 Tool Support

Tool-support is provided for the creation of educational game scenarios using the ATTAC-L language. The developed toolset was used in the context of the Friendly ATTAC-project. The toolset consists of three major parts: a web-based graphical modeling tool for specifying educational game narratives, an export module for translating the model to executable code for the targeted game environment and a simulator for direct interpreting and fast-prototyping of the ATTAC-L models.

### *9.1 ATTAC-L Editor*

The *ATTAC-L editor* (see Fig. 11) allows modelers to specify a story model by means of a drag-and-drop functionality. Extra assistance is provided by means of automatic layout management and an auto-complete suggestion mechanism for the construction of game moves. Given the fact that the use of colors can help to increase the legibility of graphical models, the possibility for colorization of the graphical models is included as well. Note that a fixed color scheme for bricks is not imposed by the ATTAC-L language specification. Instead, the modelers can define their own color scheme or choose from a set of predefined ones.

The editor is implemented as a web-based tool, which inherently allows it to operate on a broad range of platforms while avoiding the requirement for installation on a local machine. Currently, the tool runs in major browsers like Google's Chrome, Mozilla's Firefox, Microsoft's Internet Explorer and derivatives. Modelers have the option to store and load their created models online to and from 'the cloud', or to export them from or import them to local storage facilities. The tool generates a JSON-based, machine interpretable data-structure representing the modeled story



(a) The editor pane.

(b) Export to JSON.

Fig. 11: The ATTAC-L editor

that serves as input for the tool's export module explained in the next section (see Fig. 11b).

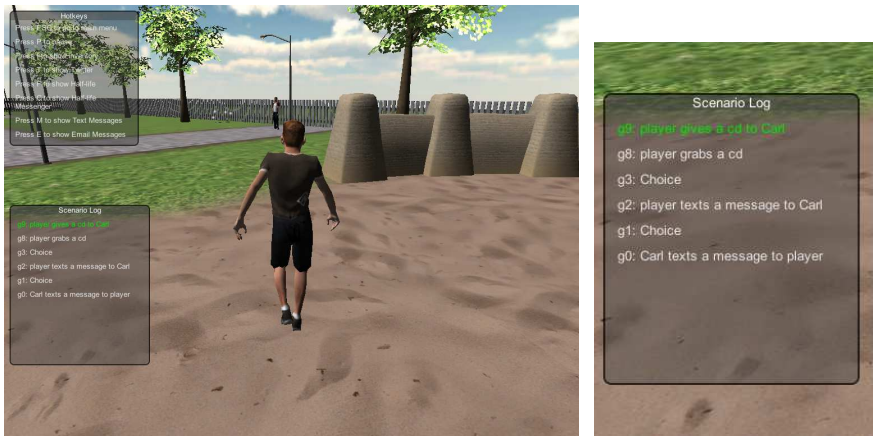
## *9.2 Export Module*

The *export module* generates executable code or scripts from a story model for a specific targeted game environment. The basis for this translation is the interpretable JSON data-structure that is exported by the ATTAC-L editor.

As part of this module, an *Event Script Generator* (ESG) has been developed, which translates a (JSON encoded) model into Event Scripts. These scripts are files in XML-format and are specific for the target game environment. They contain a high-level description of the event driven specifications of the narrative. Currently, the scripts are specific for the game environment used for the ATTAC-L project. However, other game environments can be supported by developing new specific event script generators for these environments.

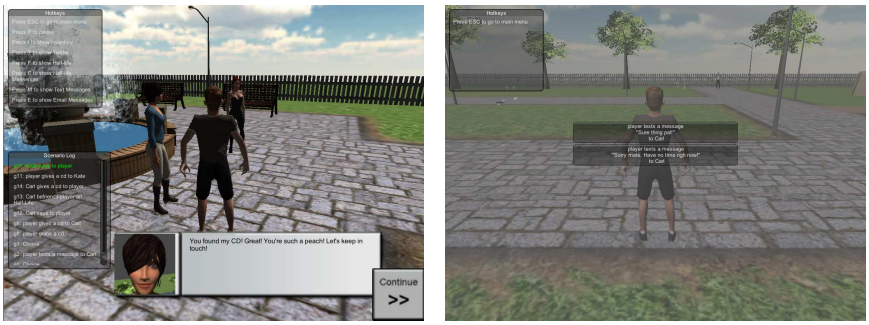
## *9.3 Simulator*

The *simulator* is a separate module that interprets an ATTAC-L model directly and executes the story 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. 12a shows a screenshot from the simulation of the example from Fig. 15).




(a) The simulator in action. (b) Detail of the scenario log.

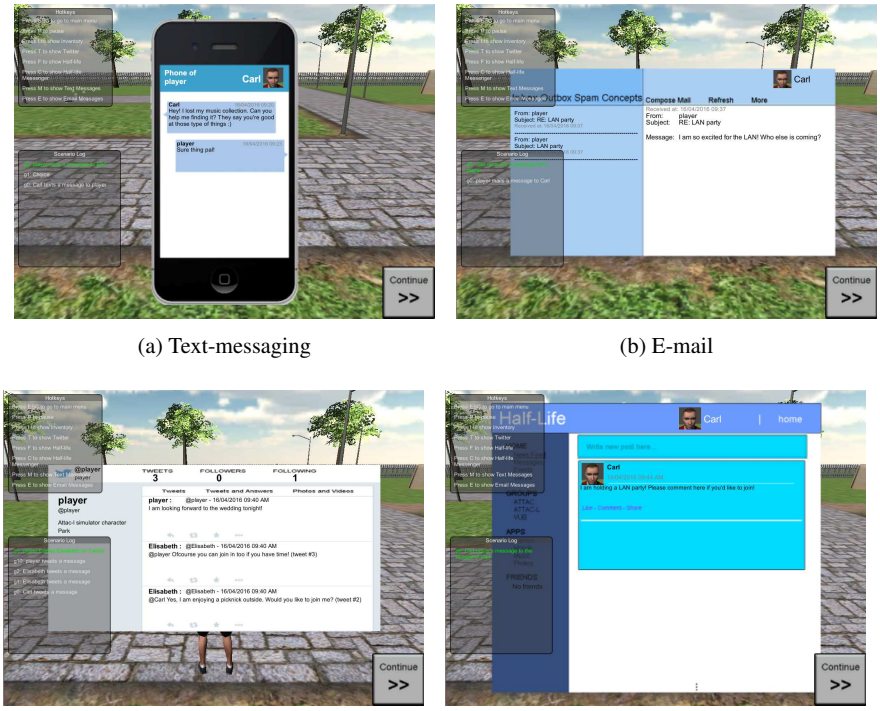
Fig. 12: Simulating the scenario from Fig. 15.




(a) Talking with an NPC. (b) The player is presented with a choice.

Fig. 13: Interactions within the simulator.

In light of the Friendly-ATTAC project, the current simulator targets stories related to cyber bulling. The stories can be executed in three different but predefined environments: a park, a house, and an office. Each environment has a number of predefined locations. For instance, the park has a playground, a fountain, a football field and a sitting area; whereas the house has different rooms. A couple of options are available to further customize the environment, e.g., day/night and weather conditions. There are also nine predefined NPCs, a fixed set of items and an inventory for the player. As the domain is cyber bulling, there is explicit support for social media such as an e-mail application interface, a Twitter interface, a Facebook-like




(a) Text-messaging

(b) E-mail

(c) Twitter

(d) Half-life, a Facebook-like social network.

Fig. 14: The social media interfaces.

interface, and a text-messaging interface for mobile phones (for examples, see Fig. 14). Those interfaces are shown as an overlay when a related action is performed in the story. However, the user can always inspect the social media interface using function keys.

To be able to execute the game moves defined with ATTAC-L, the verbs used in the game moves are associated with specific behaviors for the NPCs. Some general behaviors have been predefined for verbs, such as ‘walks-to’, ‘goes-to’, ‘says-to’, ‘gives-to’, ‘picks-up’, and ‘grabs’. In addition, behaviors have been defined for the common verbs used in cyber bullying, for example for ‘chats-to’, ‘tweets’, ‘re-tweets’, ‘follows-on’, ‘emails-to’, ‘posts-on’, ‘comments-on’, and ‘likes’. Conversations among NPCs and between NPCs and the player, which are specified in ATTAC-L with the verb ‘says-to’, are simulated using pop-up dialog boxes (see Fig. 13a). The NPC that is talking is shown at the left side of the dialog box.

The simulator keeps track of the steps followed and displays them in a scenario log (see Fig. 12b, a detail of the scenario log of Fig. 12a)). The last performed step is shown at the top of the list and is marked in green. When the player has to

make a choice, the possible options are showed on the screen. Once the user of the simulator selects an option, the story will continue in accordance with that choice (see Fig. 13b). This allows to try out different paths in the simulated game.

The simulation can be done in two modes. In the 'step-by-step' mode, the simulator performs the game moves one by one. The mode requires the user of the simulator to explicitly instruct the Simulator to continue after each step. In the 'continuous' mode, the Simulator will stop only on occasions when the player needs to make a choice or when the user presses the 'pause' button.

The simulator takes the JSON data structure of a story model as input and is implemented using Unity[1]. Unity's built-in path finding system (Nav Mesh)[2] is used to simulate the movement of the characters. JSONObject[3] is used for parsing the JSON data structure. All scripts are written in C#.

The current simulator is tailored towards the cyber bullying domain but by providing other predefined NPCs and behaviors, the simulator can be adapted for use in other domains (see also future work).

## 10 Evaluation

The DSML and its toolset were used in the interdisciplinary Friendly ATTAC project (Friendly ATTAC 2012) for developing a serious game against cyber-bullying. The team was composed of game developers, narrative designers, and people from computer, social and healthcare science. The team applied IMP (Eldredge et al. 2011) to identify intervention methods that could be incorporated in the serious game. The social scientists in the team developed the story models using the ATTAC-L editor with some support by the developers of ATTAC-L. In total, 7 story models were developed consisting of 91 separate situations: 40 'neutral' situations, 26 minor and 25 severe bullying-situations. Using the pedagogical annotations of ATTAC-L, the selected pedagogical strategies were documented and the intervention methods were integrated into the story models. The story models were translated into game-engine specific event scripts by the Event Script Generator. Development of the game environment and scenes was also done in a model-based manner by means of a Sandbox Editor (Van Hoecke et al. 2016). The sandbox exports this environment model in the form of XML files.

In parallel with the social scientists, the game developers worked on the implementation. This included the translation of the domain vocabulary into implementation concepts, i.e., the domain actions and behaviors. For the cyber bullying domain, for instance, they implemented behaviors such as 'send-to' and 'post-on'. Furthermore, they created those parts of the game that would not be generated, like the user interface, levels and gameplay code. Finally, the developed code was used in com-

[1] https://unity3d.com/

[2] https://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/navmeshes

[3] https://msdn.microsoft.com/en-us/library/system.json.jsonobject(v=vs.95).aspx

bination with an interpreter to create a running game from the event scripts and the environment model (Janssens et al. 2014).

In addition to the use of ATTAC-L in the Friendly-ATTAC project, we also conducted a first small-scale evaluation to test whether our modeling language and toolset satisfies our objective. In other words, we investigated whether the language and the toolset suit the needs of people without programming knowledge and skills. Four people participated in the evaluation, each with a background in game narrative design but without formal schooling in programming. None of the participants had prior experience in modeling stories with a DSML.

The evaluation took place as follows. First, the participants received a small introduction on the purpose of ATTAC-L and how to use it. They also received an example story model and a written outline of the syntax rules. They were then asked to model a small, prescribed story provided in textual form. For this purpose, we used a scenario about a player who has to escape from a locked house and having three possible ways of doing so. The participants were then asked to sketch the story-flow on paper. Next, they were invited to model the story with the help of our modeling tool. The participants were provided with a comprehensive list of actions and entities (nouns, pronouns, verbs, adjectives). They could ask for assistance at any given time. After completing the exercise, the participants were prompted to reflect on their experience and the difficulties they encountered. The results of this evaluation were generally positive. The participants reported that they understood the example, as well as the modeling assignment and were fully able to model the scenario. Furthermore, the participants reported that they found the syntax rules very intuitive and natural and thus had no difficulties in complying with them. It is worth nothing that early during the exercise, some participants were inclined to model story-flows that were too complex (e.g., by introducing more game moves than actually needed) or to create highly specific solutions (e.g., using specific pronouns instead of the more general common nouns). When they were made aware of this, the participants quickly adapted their solution accordingly.

Together with our experiences in the Friendly ATTAC project, this small-scale evaluation provides an indication that our graphical CNL DSML will be able to meet its goal. People who used ATTAC-L agreed that the models specified in this language were easy to understand, even without a programming background. Furthermore, in both instances, we saw indications that people without a background in programming are able to actively model narratives and pedagogical interventions using ATTAC-L. As such, ATTAC-L enables narrative designers to specify their stories in a formal format that can be processed directly, and enables pedagogical designers to add the pedagogical interventions. This way of working has two main advantages. Firstly, it helps to save time that is otherwise spend on elaborating the narratives and pedagogical interventions on paper and transferring it to the technically staff of the development team. Secondly, it supports better communication practices as it helps to eliminate ambiguities and prevent misunderstandings.

## 11 Conclusions and Future Work

In this chapter, we have presented the domain-specific modeling language ATTAC-L developed to enable a multidisciplinary approach for the development of narrative-based serious games. It provides the necessary means to designers with and without a background in IT to model the narrative and pedagogical methods for the serious game in a clear and structured manner. A designer can be anybody involved in the design process: a subject-matter expert, a pedagogical expert, a storywriter, a game developer, even an end-user. An important characteristic that allows all of these designers to work with and communicate through ATTAC-L is the absence of programming concepts, as well as technical and implementation aspects. Furthermore, the associated tools are carefully designed to be intuitive and easy to use by people without a background in IT. The narrative is expressed using a strict form of natural language and organized in a schematic representation to provide the flow of the story. An unobtrusive, yet versatile annotation system is provided to integrate the pedagogical aspects and layer them on top of this story-flow.

We have also discussed how to explicitly link the narrative with the pedagogical intervention methods. This linking approach has two main advantages. First, it allows modelers to incorporate the designed pedagogical methods into the narrative in a rigorous way and without both becoming entangled and inseparable from one another. The clear separation between the narrative and pedagogical aspects of the serious game enables modelers to concentrate on particular aspects of the game in accordance with their own expertise, while maintaining an overview of the relationships to all others aspects of the game. As such, ATTAC-L stimulates and enhances multidisciplinary collaboration. Secondly, the annotations provide detailed documentation of the relationship between the story and the corresponding pedagogical principles allowing for verification of whether or not the serious game conforms to these principles.

We have shown that in the context of the Friendly-ATTAC project (Friendly-ATTAC, 2012) social and health scientist were able to actively contribute to the development of a serious game against cyber-bullying by creating narrative models. These models were then used to generate parts of the implementation of the serious game. This helped to speed up the development process and facilitated an iterative development approach. We can thus conclude that the work presented in this chapter represents an important step towards (1) allowing people with expertise in various domains other than computer programming or game development to become actively involved in the creation of serious games. Furthermore, the work also provides a means for (2) enhancing serious game development because it helps to reduce the development time of serious games and therefore also their costs. After all, time and money are two of the major barriers that hinder development of serious games and inhibit their widespread use.

In future work we aim to extend the current toolset to allow for the easy customization of the modeling language and the tools towards different serious game topics. We will do this by using an explicit and replaceable topic vocabulary. We also plan to extend the annotation system to allow for the specification of gameplay

related aspects. In addition, we will provide a means for automatic code generation for these gameplay aspects. Finally, we plan to investigate how we can formally assess whether the serious game conforms to its intended pedagogical design.

**Acknowledgements** This research was funded by IWT (Institute for Science and Technology) under the Friendly ATTAC project. We also like to thank our colleagues of the Friendly ATTAC project for the fruitful cooperation resulting in a lot of improvements for ATTAC-L.

## References

Bandura, A. (1991), 'Social cognitive theory of self-regulation', *Organizational behavior and human decision processes* **50**(2), 248–287.

Bellotti, F., Berta, R. & De Gloria, A. (2010), 'Designing Effective Serious Games: Opportunities and Challenges for Research', *International Journal of Emerging Technologies in Learning (iJET)* **5**(SI3), 22–35.

De Troyer, O. & Janssens, E. (2014), 'Supporting the requirement analysis phase for the development of serious games for children', *International Journal of Child-Computer Interaction* **2**(2), 76–84.

Deursen, A. V., Klint, P. & Visser, J. (2000), 'Domain-specific languages: an annotated bibliography', *ACM Sigplan Notices* **35**(6), 26–36.

Djaouti, D., Alvarez, J. & Jessel, J. (2010), Can Gaming 2.0 help design Serious Games?: a comparative study, *in* '5th SIGGRAPH Symposium on Video Games', pp. 11–18.

Dobbe, J. (2007), 'A domain-specific language for computer games', *Delft University of Technology, Delft, the Netherlands* .

Eldredge, L. K. B., Parcel, G. S., Kok, G. & Gottlieb, N. H. (2011), *Planning health promotion programs: an intervention mapping approach*, John Wiley & Sons.

Friendly ATTAC (2012), 'http://www.friendlyattac.be/en/'. Accessed: 2016-04-18.

Fuchs, N., Schwertel, U. & Schwitter, R. (1999), 'Attempto Controlled English Not Just Another Logic Specification Language', *Logic-based program synthesis and transformation* **1559**, 1–20.

Furtado, A. W. B. & Santos, A. L. M. (2006), 'Using domain-specific modeling towards computer games development industrialization', *6th OOPSLA Workshop on DomainSpecific Modeling DSM'06* p. 1.

Göbel, S., Mehm, F., Radke, S. & Steinmetz, R. (2009), 80Days: Adaptive digital storytelling for digital educational games, *in* 'CEUR Workshop Proceedings', Vol. 498.

Guerreiro, R., Rosa, A., Sousa, V., Amaral, V. & Correia, N. (2010), UbiLang: Towards a Domain Specific Modeling Language for Specification of Ubiquitous Games, *in* 'INForum', pp. 449–460.

Hirdes, E. M., Thillainathan, N. & Leimeister, J. M. (2012), Towards modeling educational objectives in serious games, *in* 'CEUR Workshop Proceedings', Vol. 898, pp. 11–14.

Hürsch, W. L. & Lopes, C. V. (1995), 'Separation of concerns'.

ISO (1996), Information technology—Syntactic metalanguage—Extended BNF, ISO 14977:1996(E), International Organization for Standardization, Geneva, Switzerland.

Janssens, O., Samyny, K., Van de Walle, R. & Van Hoecke, S. (2014), Educational virtual game scenario generation for serious games, *in* 'Serious Games and Applications for Health (SeGAH), 2014 IEEE 3rd International Conference on', IEEE, pp. 1–8.

Kelly, S. & Tolvanen, J. P. (2007), *Domain-Specific Modeling: Enabling Full Code Generation*, John Wiley & Sons, Inc.

Koper, R. & Olivier, B. (2004), 'Representing the learning design of units of learning'.

Lindley, C. a. (2005), 'Story and narrative structures in computer games', *Bushoff, Brunhild. ed* (January), 1–27.

Luoma, J., Kelly, S. & Tolvanen, J.-P. (2004), 'Defining domain-specific modeling languages: Collected experiences', *4th Workshop on Domain-Specific Modeling* (June 2015), 198–209.

Marchiori, E. J., del Blanco, n., Torrente, J., Martinez-Ortiz, I. & Fernández-Manjón, B. (2011), 'A visual language for the creation of narrative educational games', *Journal of Visual Languages and Computing* **22**(6), 443–452.

Marchiori, E. J., Torrente, J., Del Blanco, Á., Moreno-Ger, P. & Fernández-Manjón, B. (2003), 'A Visual Domain Specific Language for the Creation of Educational Video Games', *IEEE Learning Technology Newsletter* **30**(1), 11–11.

Moody, D. (2009), 'The physics of notations: Toward a scientific basis for constructing visual notations in software engineering', *IEEE Transactions on Software Engineering* **35**(6), 756–779.

Moreno-Ger, P., Martínez-Ortiz, I., Sierra, J. L. & Fernández-Manjón, B. (2008), 'A content-centric development process model', *Computer* **41**(3), 24–30.

Nelson, G. (2006), 'Natural language, semantic analysis, and interactive fiction', *IF Theory Reader* **141**.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. a. Y., Silverman, B. & Kafai, Y. (2009), 'Scratch: Programming for All.', *Communications of the ACM* **52**, 60–67.

Rooney, P., O'Rourke, K. C., Burke, G., Mac Namee, B. & Igbrude, C. (2009), Cross-disciplinary approaches for developing serious games in Higher Education: Frameworks for Food Safety and Environmental Health Education, *in* 'Proceedings of the 2009 Conference in Games and Virtual Worlds for Serious Applications, VS-GAMES 2009', pp. 161–165.

StoryBricks (2014), 'http://www.storybricks.com'. Accessed: 2016-04-18.

Torrente, J., Moreno-Ger, P., Fern??ndez-Manj??n, B. & Sierra, J. L. (2008), Instructor-oriented authoring tools for educational videogames, *in* 'Proceedings - The 8th IEEE International Conference on Advanced Learning Technologies, ICALT 2008', pp. 516–518.

Tran, C. D., George, S. & Marfisi-Schottman, I. (2010), EDoS: An authoring environment for serious games design based on three models, *in* '4th European Conference on Games Based Learning 2010, ECGBL 2010', pp. 393–402.

Van Broeckhoven, F., Vlieghe, J. & De Troyer, O. (2015*a*), 'Mapping between Pedagogical Design Strategies and Serious Game Narratives', *Proceedings of the 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)* pp. 1–8.

Van Broeckhoven, F., Vlieghe, J. & De Troyer, O. (2015*b*), 'Using a Controlled Natural Language for Specifying the Narratives of Serious Games', *Interactive Storytelling: ICIDS 2015, Lecture Notes in Computer Science 9445* pp. 1–12.

Van Broeckhoven, F., Vlieghe, J. & De Troyer, O. (2016), 'Linking serious game narratives with pedagogical theories and pedagogical design strategies'. Submitted.

Van Hoecke, S., Samyn, K., Deglorie, G., Janssens, O., Lambert, P. & Van de Walle, R. (2016), Enabling control of 3d visuals, scenarios and non-linear gameplay in serious game development through model-driven authoring, *in* 'Serious Games, Interaction, and Simulation', Springer, pp. 103–110.

Wyner, A., Angelov, K., Barzdins, G., Damljanovic, D., Davis, B., Fuchs, N., Hoefler, S., Jones, K., Kaljurand, K., Kuhn, T., Luts, M., Pool, J., Rosner, M., Schwitter, R. & Sowa, J. (2010), On controlled natural languages: Properties and prospects, *in* 'Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)', Vol. 5972 LNAI, pp. 281–289.

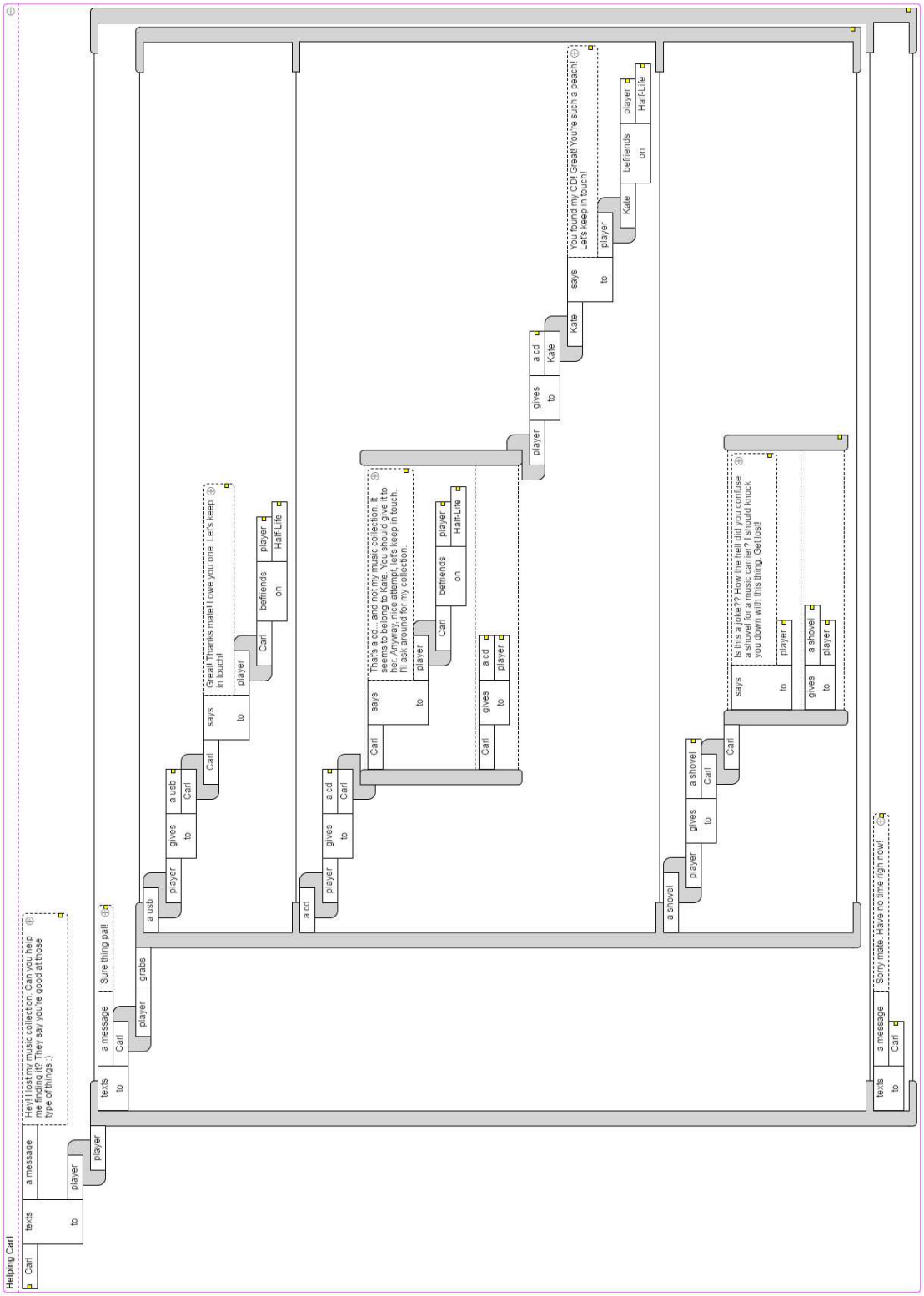



Fig. 15: An example scenario ‘Helping Carl’, running in the simulator in Fig. 12a.

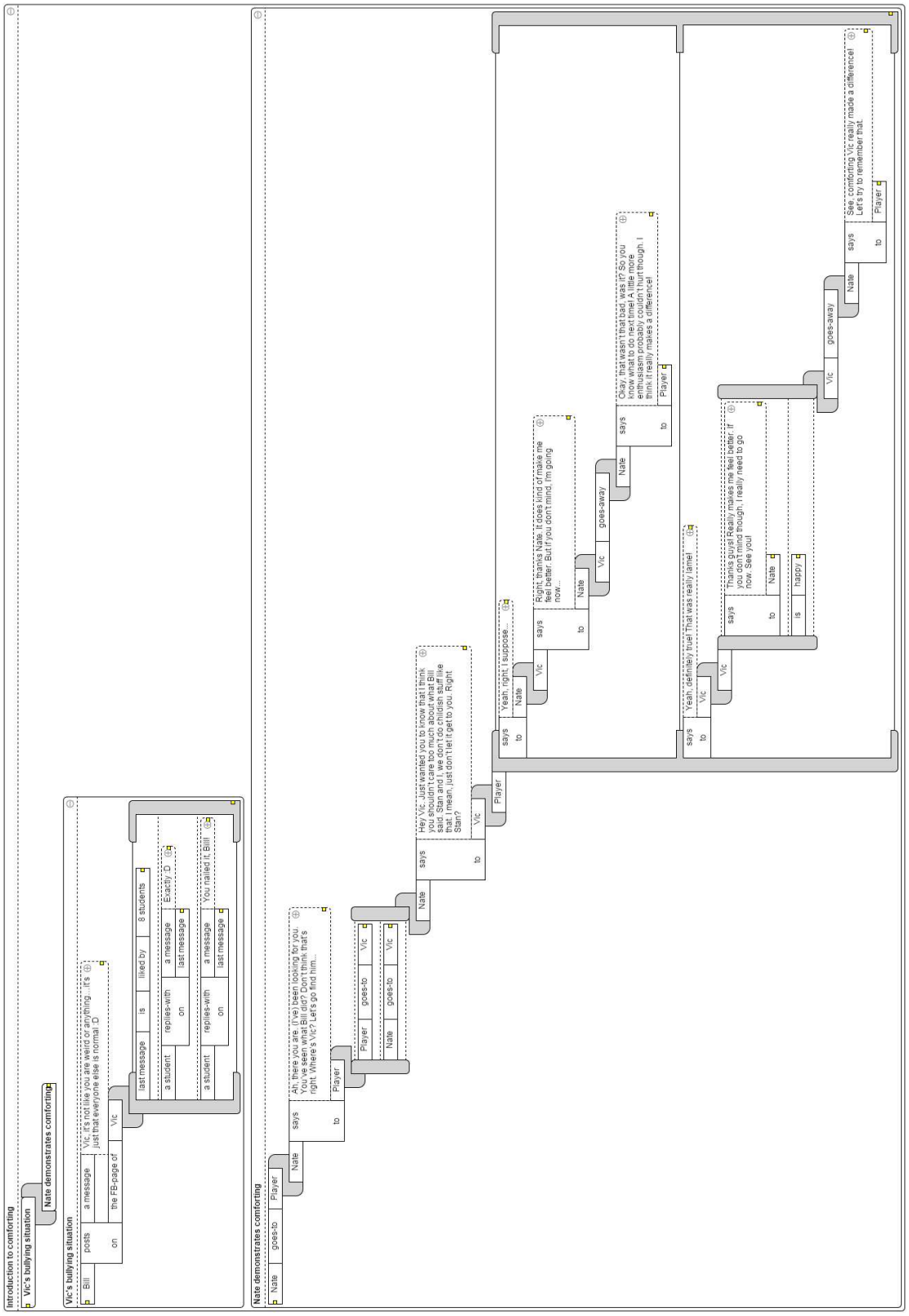



Fig. 16: Example scenario ‘Introduction to Comforting’.

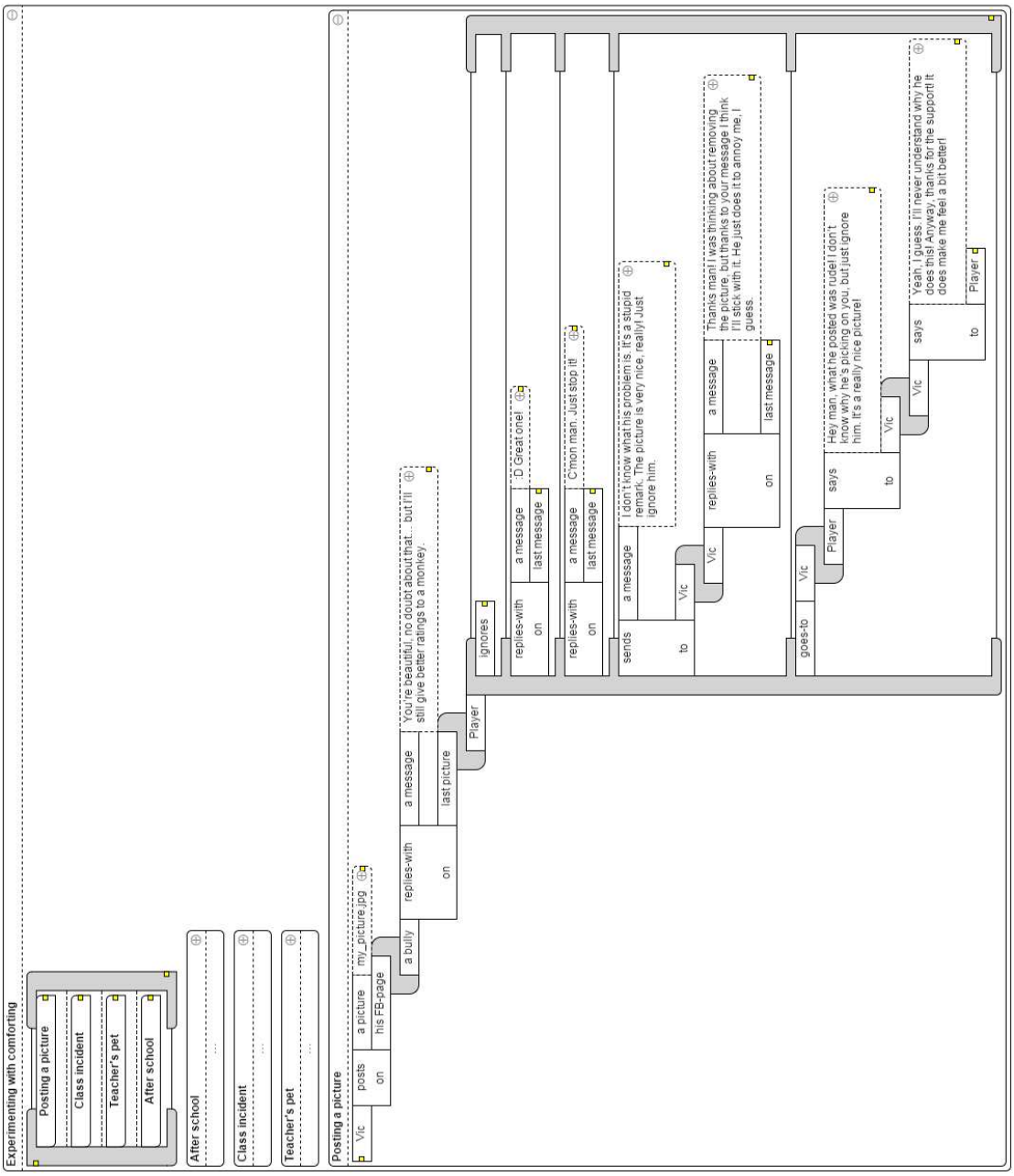



Fig. 17: Example scenario ‘Experimenting with comforting’.

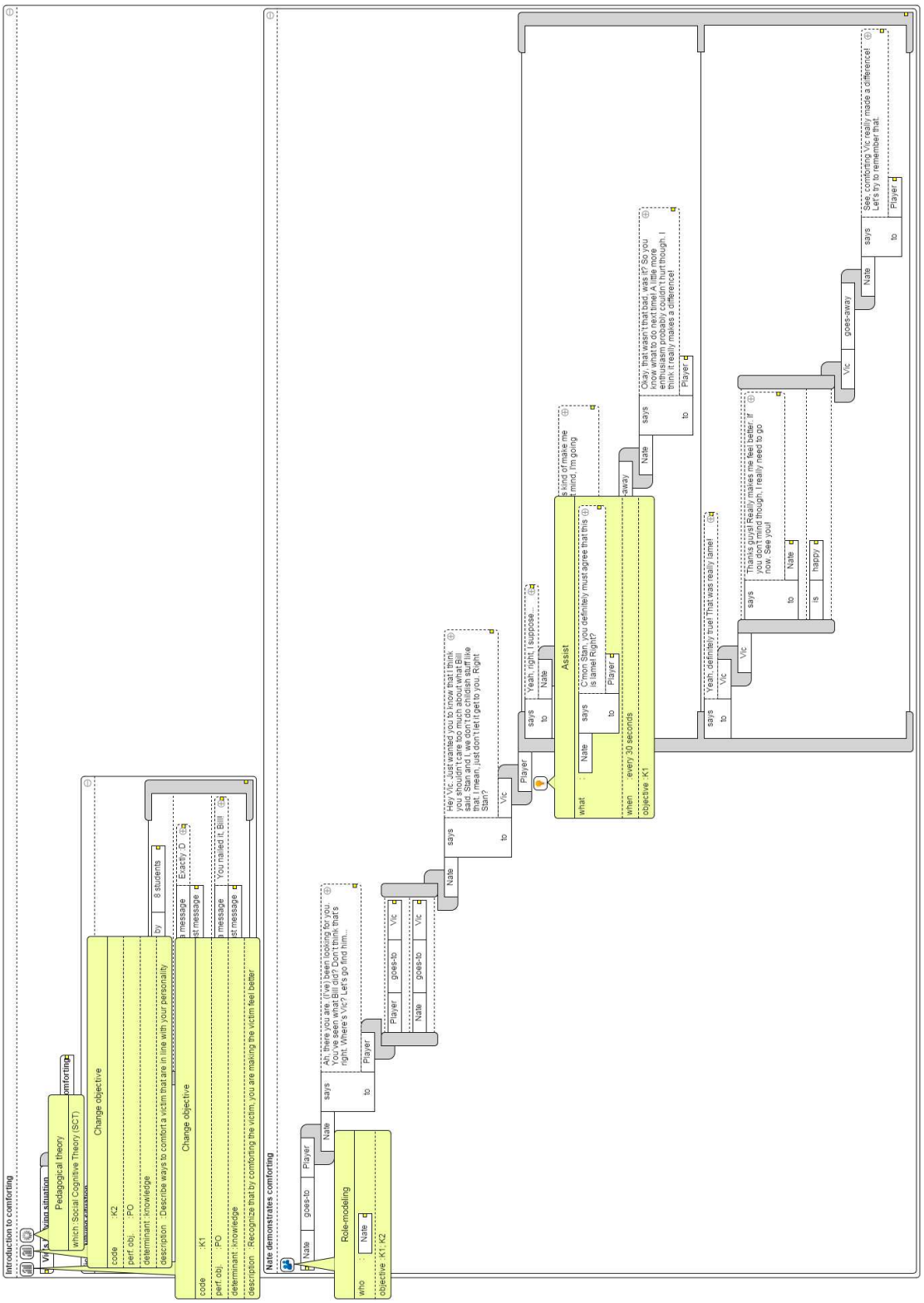



Fig. 18: A scenario (Fig. 16) annotated with change objective PAs for determinant 'knowledge', a method PA for 'role-modeling' and a theory PA for SCT.

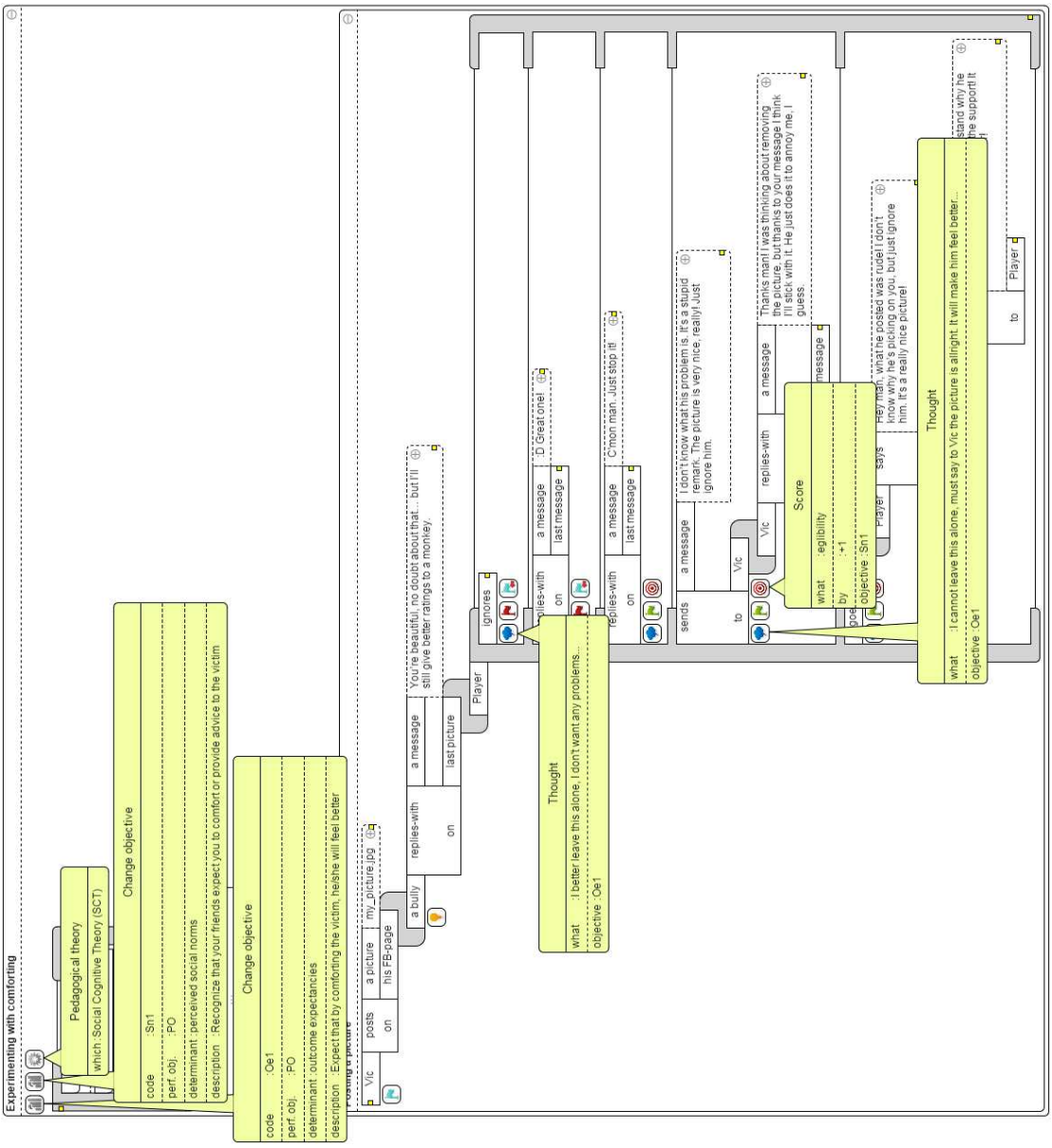



Fig. 19: A scenario with change objective PAs for determinants ‘outcome expectations’ and ‘perceived social norms’ and action PAs for ‘scoring’, ‘assistance’, ‘thought’, and ‘role-model’.