A visual language for the creation of narrative educational games

Eugenio J. Marchiori [1]*, Ángel del Blanco [1], Javier Torrente [1], Iván Martinez-Ortiz [1], Baltasar Fernández-Manjón [1,2]

[1] *Department of Software Engineering and Artificial Intelligence, Complutense University, C/ Profesor José García Santesmases S/N, 28040 Madrid, Spain*

[2] *Visiting Associate Professor, LCS-MGH, Harvard University, 50 Staniford St., 02114, Boston, Massachusetts, USA*

* Corresponding author:
email: emarchiori@fdi.ucm.es
telephone: (+34) 91 394 75 17
fax: (+34) 91 394 75 47
(emarchiori, angel.dba, jtorrente, imartinez, balta)@fdi.ucm.es

Abstract

This paper presents a DSVL that simplifies educational video game development for educators who do not have programming backgrounds. Other solutions that reduce the cost and complexity of educational video game development have been proposed, but simple to use approaches tailored to the specific needs of educators are still needed. We use a multidisciplinary approach based on visual language and narrative theory concepts to create an easy to understand and maintain description of games. This language specifically targets games of the adventure *point-and-click* genre. The resulting DVSL uses an explicit flow representation to help educational game authors (i.e. educators) to design the story-flow of adventure games, while providing specific features for the integration of educational characteristics (e.g. student assessment and content adaptation). These highly visual descriptions can then be automatically transformed into playable educational video games.

Keywords: Domain-Specific Visual Language, educational video games, serious games, educators, story-flow, game authoring

1. Introduction

The field of educational video games[1] is experiencing both increased acceptance and incorporation into current curricula as complementary content. This growth has been promoted by increasing investment in the field [2] and supported by research focused on overcoming known issues. Research in the field of Game-Based Learning (GBL) has tried to identify which factors in game design may have a deeper impact on games efficacy as learning tools [3-6]. Other works have evaluated the use of games in classroom settings [7-10], explored which kinds of games are more suitable for different educational scenarios [9, 11-13] and studied educational game evaluation methods [14]. Research in video game design is partly focused on the elements it shares with narrative theory. Some authors argue that both fields are tightly related [15, 16]. Firstly, narration

[1] It should be noted that in this paper the terms "video games", "computer games" or just "games" are used to refer to the whole field of video and computer games as well as interactive simulations, making no distinction on the specific device used to deliver the software and implying no particular implication about their entertaining nature. For a complete and accurate definition of these systems see [1].

is a feature that appears in many video games, and it is used to immerse players in the game world and as a powerful engagement factor that can improve the learning outcomes [17-19]. In addition, in both video game theory and in narrative theory, visual languages are presented to describe the flow of stories [16, 20, 21].

The definition and transformation of DSVL has been thoroughly researched [22, 23]. This research also establishes DSVLs as easy-to-use mechanisms to define the workings of systems with complex behaviors, especially when the target users are people with limited programming knowledge. However, even if other systems have applied DSVL to the educational video game domain (e.g. *Storytec* [24]), the approach has been limited to modeling tools. Development tools for educators are still needed.

The DSVL presented in this paper was designed to allow end-users (domain-experts and educators, in this case) to construct and modify interactive systems (i.e. educational games). For this reason, the end-users of the system are sometimes referred to as game developers in the context of this paper. The DSVL was created following End-User Development (EDU) [25, 26] guidelines, with a focus on creating a tool with a smooth learning curve. This is achieved by the use of an explicit story-flow representation, enhanced with specific elements to define complex user interactions and a wide range of educational situations.

The main aim of this work is to present a tool that helps educators to go directly from a game storyboard (or a textual description of a game) to a playable game. This is achieved by the use of a DSVL as an intermediate representation that both decreases the slope of the learning curve and shortens the development cycle. The proposed DSVL represents an abstraction of the story-flow in an educational game by explicitly presenting the potential actions of the player and their consequences. The basic elements of the language are represented as a Mealy-inspired state-transition diagram. The language also includes advanced features to define constructs that are common to game genres with a strong narrative (e.g. adventure games) but which usually require a substantial programming effort (e.g. out-of-order actions, such as when the player must accomplish two tasks in the game but the order in which they are done is not relevant).

An educational dialect was added to the DSVL by including constructs to address aspects that are complex to design in traditional systems, but relevant from a pedagogical perspective. These constructs focus on providing educators with mechanisms to maximize their control over the game-based learning experience, as this has been identified as an important shortcoming in educational video game development [27-29]. This includes support for tailoring the gaming experience for each individual student (i.e. adaptation) [30-32], which is important to achieve optimum learning experiences [33, 34], and tracking and evaluation of the learning outcomes (i.e. assessment) [35]. In addition, the DSVL also includes explicit support for the definition of the in-game guidance that will be available to the student.

The games defined using our DSVL can be directly and automatically transformed into playable educational video games. The resulting video games are adventure *point-and-click* games that run in the <e-Adventure> game engine [36]. "*Point-and-click* adventure games" (e.g. the *Monkey Island*™ saga) is the name usually used to describe 2D games in which the user interaction is mostly focused on the use of the mouse and the story (i.e. the adventure) plays a fundamental role. This has additional benefits, given that these games can be directly deployed in a wide range of VLE used in real educational contexts [37]. Besides, <e-Adventure> games have been previously used successfully in actual educational contexts [38]. We also considered other approaches for educational video game development, such as EMERGO [39].

This work is structured as follows: Section 2 presents the theoretical background for the development of the DSVL. In section 3, the basic elements of the underlying flow DSVL are detailed. Section 4 introduces the advanced elements in the language, while section 5 presents the different educational elements that are included. Later, in section 6, a usage example of the language is presented by describing a full educational video game. Finally, section 7 presents the conclusions of this work and the future work that we are conducting and expect to conduct in the related fields.

2. Theoretical background

The relationship between video games and narratives has been studied from different perspectives. A key concern involves whether games should be considered a narrative medium. However, even authors like Juul [1] who argue against this hypothesis make limited exceptions for some game genres in particular (i.e. adventure games). In contrast, other authors consider most games as a narrative medium and study the consequences of this assumption and the similarities between games and traditional media. For instance, Ryan [15] studies how games can be described as a specific kind of narrative structure that shares similarities with traditional narrative media (e.g. novels). Lindley [16] uses narrative theory to describe the different elements found in games and remarks on the similarities of the underlying structures used in games to describe the story with the ones used in traditional media such as novels or movies.

These studies of the narrative elements in video games support a flow-like representation of the stories in games as a central element of the game description. Still, Lindley [16] openly argues against the practical use of an explicit visual representation of the game flow because the potential complexity of games described in this way. However, Lindley's argument does not consider different mechanisms (e.g. hierarchical representations) that can be used in visual languages to reduce complexity. Besides, narrowing the target type of video games to a specific genre within the serious game field (e.g. educational *point-and-click* adventure games) further reduces the potential complexity of the representation.

Studies that consider narrative aspects directly related to game development help to identify relevant elements. For instance, Dickey [40] identifies three main elements of any interactive design (e.g. games): a) setting; b) roles and characters; and c) actions, feedback and *affordances*. While the first two are the place, in a broad sense, where the action takes place and the different characters in the story respectively; the actions, feedback and *affordances* (i.e. "action possibilities" latent in the environment) are what make up the story in a video game. The actions drive the story forward; the feedback provides information to the player and describes the consequences of the actions; the *affordances* make the game interesting to the player by providing choice.

In the development of educational video games, a key challenge is how to integrate educators and other domain-experts (usually with limited technical knowledge) in the development process [41]. Several projects have aimed to bridge the gap between educators and game development by providing authoring tools that are specifically devised for non-technical users [41, 42]. However, even if such approaches are positive because they reduce the need for programming knowledge, they still use the wording, principles and development methods used in professional game development tools (e.g. variables and conditional logic). The EUD approach tries to reformulate the underlying concepts using vocabulary that is familiar to educators and providing a flow representation, which helps in the design process [43]. Additionally our approach benefits from the ease with which people can employ visual languages to convey information as many people think and remember things in terms of pictures [44].

Other systems have applied DSVL or Visual Programming Languages (VPL) to deal with specific domains. Most multimedia authoring tools present visual programming interfaces [45]. This includes other tools tailored to educators, such as those specific for learning design activities [46]. Other examples include, for instance, *Scratch* [47] which is an interactive software-authoring tool that uses a VPL to make programming easier for young children. *Scratch* is based on *OpenBlocks*[2], an "extendable framework for graphical block programming systems" developed at MIT. Google's *App Inventor*[3] is another system based on *OpenBlocks* for general-purpose mobile app development. *Thinking Worlds*[4], which is a system for the development of educational and serious video games, uses a visual language to describe the flow of the games. The approach taken by *Thinking Worlds*, however, is better identified with hybrid languages in which textual descriptions still have an important role in describing the content. Besides, LaMothe [48] proposed the use of a visual language based on state-transition diagrams to describe the artificial intelligence (AI) of game characters.
As for game development in general, Onder [20] has proposed the use of story-beat diagrams (a collection of ovals and arrows) to show the flow of computer games at a high level. Lewinsky [21] proposes the use of flow-charts to show the flow of the games, particularly of the story and cutscenes at a "mission" level and not in full detail. Rouse [49] proposes the use of storyboards to mock-up the action in the game, a proposal that could be considered complementary to the one proposed in this paper. Taylor, Gresty & Baskett [50] propose the use of a visual language based on UML to design individual game levels and help in the communication amongst different participants in the game creation process. All these proposals, although covering different level of details and being complementary to some extent, do not provide a direct path to implementation and were created mostly to improve documentation and communication within development teams.
In our case, to obtain a playable game from the DSVL game representation we have taken advantage of the research conducted in the field of visual languages. Guidelines to the creation and definition of the different kinds of visual languages are available, as well as different techniques that are proposed for verification and transformation of such languages [22, 44]. Different approaches to the generalization of the process of defining the syntax and semantics, as well as transformations of visual languages have been proposed. For example, systems such as Moses [23] use an approach similar to the one proposed in this paper. One important aspect of the DSVL we propose is that it can be automatically transformed into a playable educational video game.

3. Basic elements of the story-flow language

The basic description of the game story-flow is based on a state-transition diagram. The story-flow language redefines some assumptions made in other game development tools. For instance, in traditional game development frameworks actions are usually classified by their interactivity, where a passive action (e.g. watching a video) is defined in a different way than an interactive action (e.g. using a tool). However, from a narrative perspective both actions move the story forward and should therefore use the

[2] *OpenBlocks*: http://hdl.handle.net/1721.1/41550 (retrieved on 17/11/2010)

[3] *http://appinventor.googlelabs.com/about/* (retrieved on 17/11/2010)

[4] *http://www.thinkingworlds.com/* (retrieved on 17/11/2010)

same representational elements. Thus the representation of the story is closer to the narrative interpretation of the story than a *programmatic* interpretation of the system.
The basic elements in the language can be described as a Mealy finite-state machine. A Mealy machine describes a system in which an input (e.g. user action), which depends on a state (i.e. the point in the story), produces an output (e.g. text shown on the screen) and a change of state. Formally, a Mealy machine is described by a 6-tuple (S, S_o, Σ, Λ, T, G), where:

- S is a finite set of states. In this DSVL, the states represents points in the story or game states, defined as specific values for the internal (i.e. invisible to the player) variables and flags.
- S_o is the initial state, an element of S. This is the initial game state, the point where the story begins.
- Σ is the input alphabet. The input alphabet is made up of every action the player can perform in the game, both active and passive. For example, grabbing an object, talking to a Non-Player Character (NPC) and watching a video are all inputs in the language.
- Λ is the output alphabet. The output alphabet in games is made up mostly of feedback to the player. These outputs are sometimes referred to as effects in the video game lingo. For example, both displaying text on the screen and changing the appearance of an object are part of the output alphabet.
- T is the transition function. Formally defined as ($T: S \times \Sigma \rightarrow S$), this function describes how an input from the input alphabet (Σ) changes the current state. In the graphic representation the arcs or transitions describe it implicitly.
- G is the output function. Formally defined as ($G: S \times \Sigma \rightarrow \Lambda$), this function describes the output that a particular input will produce while the machine is at a certain state. In the graphic representation this function is implicit. The output is represented together with the input function, which in turn is represented by the transitions. The output function will add information to the transition that corresponds with the same $S \times \Sigma$ pair.

The basic elements of the game story-flow language are presented to the user in a graphic representation. In this representation the states are represented as circles (Figure 1, a). Transitions, which are an implicit representation of the transition function, are represented as directed arcs from one state to another (Figure 1, b). The effects or feedback, which correspond to an implicit representation of G (the output function), are represented as elements associated with the transitions (Figure 1, c).

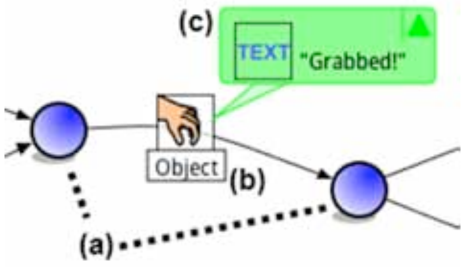

Figure 1. The story-flow uses basic elements to describe changes in the story based on the actions of the player. The states (a) represent points in the story, transitions (b) represent possible actions by the player and the outputs of a transition (c) represent additional feedback of the transition.

These basic elements allow for the explicit representation of a wide range of game interactions and stories. However, as the game grows in size, the representation grows in complexity. To address this problem advanced macro-like elements were introduced into the language, which represent repeating patterns in the DSVL representation.

4. Advanced elements in the story-flow representation

The DSVL here presented is defined as an extensible platform to meet the requirements of the fast changing field of serious games, the varying needs of different educational requirements and the complexity of game development. This allows for the creation and incorporation into the language of new representational elements. In particular, this section explores some advanced elements, defined as part of the language, which helps in the story-flow definition process. The goal is to reduce the complexity of the descriptions and provide reusable structures to address recurring problems in game development. Other advanced features are included in the DSVL to cover specific aspects of adventure games. For instance, a specific component to define in-game conversations and an element to define time sensitive behavior (i.e. timers) are included.

4.1 Hierarchical representation

Video games represented using only basic graphs can become too complex to be easily understood and modified. This problem, identified by Lindley [16] can make an explicit graph-like representation of games useless due to practical reasons. However, visual language theory proposes the use of hierarchical representations, similar to the use of procedures in general-purpose programming languages, to limit the complexity of the whole description. To address this problem in our language, a story-section element is introduced to allow for the hierarchical representation of games.
The definition of a story-section is divided into two phases. First, a new kind of state is added to the representation, the story-section. The transitions from this node, instead of representing actions by the player, represent different consequences of the player following the story-section flow (e.g. succeeded or failed) (Figure 2, a).
The second part of the definition involves the actual story-section flow. Each story-section flow is defined as a sub-diagram, using the full set of elements in the DSVL, and it includes one starting node (i.e. the point in the story the player is at when reaching the story-section node) and N ending nodes (i.e. one for each consequence of the story-section) (Figure 2, b). Based on the previous definition of the language (i.e. the Mealy machine), the story-section element implies the modification of the state definition to allow for the inclusion of sub-diagrams as states. Besides, the values in the input alphabet are extended to include the tags for the different consequences in of visiting the story-section. In particular:

- S is now a finite set of states and sub-diagrams. Sub-diagrams are defined as 8-tuples (S', S_o', Σ, Λ, T', G', E, E_S), extending the definition described in section 3 with two new elements.
- E is a finite set of ending situations. The situations are identified by their names (e.g. "Succeeded" and "Failed" in Figure 2) and will be achieved (i.e. selected in the enclosing flow) when the story-section flow reaches the corresponding end state.

- E_S is the set of ending states or states tagged with an ending situation. Formally defined as (Es: E -> S), it indicates the states in S' that represent a final state of the sub-diagram or story-section flow.
- Σ, the input alphabet, is extended with all the elements in E. This allows defining transitions from the sub-diagram for each different ending in the story-section flow.

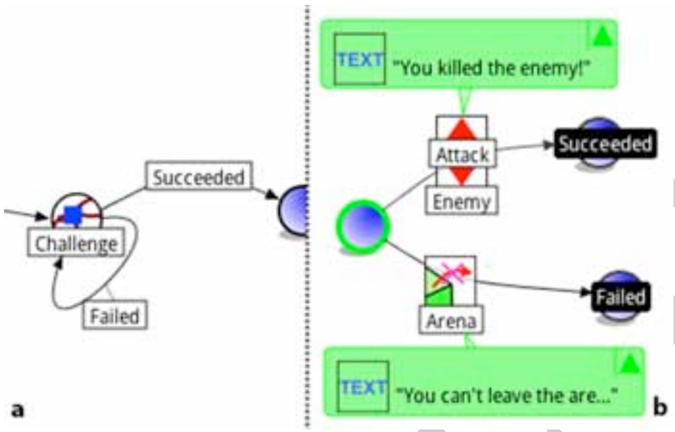
Succeeded
Challenge
Failed
a
TEXT "You killed the enemy!"
Attack
Enemy
Succeeded
Arena
Failed
TEXT "You can't leave the are..."
b

Figure 2. Story-section representation elements allow for a hierarchical definition of the story-flow of the game. This allows for both the reduction in complexity (i.e. by hiding unnecessarily complex interactions) and the reuse of story-sections (i.e. by adding the same story-sections several times in the same game). The story-section node is added to the representation (a) and the sub-diagram (the story-section flow) is defined on its own (b).

4.2 Recurring patterns

One of the central aspects of games highly regarded for its educational potential is the fact that they foster exploration and self-guided discovery of solutions [51]. Games usually achieve this by giving multiple choices to the user at all times. The language here presented allows for the creation of games with a high level of choice by the definition of alternative paths and branching at different points in the story. This can add significant complexity to game designs. For instance, introducing out-of-order actions (sequences of actions that have to be performed but in no particular order to push the story forward) becomes overly complicated even if these are central to provide choice and interactivity to the games. An example of an out-of-order action in a game is where at some point the player needs to grab a key and talk to another character to move forward, while the outcome does not depend on which task is accomplished first. The complication arises from the fact that the number of branches necessary would increase exponentially with the number of actions (e.g. two actions, 1 and 2, could be performed as 1-2 or 2-1, while three actions, 1, 2 and 3, could be performed in the orders 1-2-3, 1-3-2, 2-1-3, … and so on). This would result in the repetition of many transitions, adding unnecessary complexity to the representation. To address this concern a "multi-interaction node" element is defined in the language.
The "multi-interaction node" element eases the definition of out-of-order actions, removing the need to repeat transitions, bringing down the exponential branching and using a straightforward representation (Figure 3, left). This element, represented as a special node has different parts represented in the figure:

- (Figure 3, a) An arrival node, or the node that the story-flow must reach for the out-of-order actions to be available to the player.

- (Figure 3, b) N_1 starting nodes for the different out-of-order action sequences. Once the player reaches the multi-interaction node, the story will branch N_1 times and be at all starting points simultaneously.
- (Figure 3, c) N_2 ending nodes. The player might reach as many as *max*(N_1, N_2) ending nodes at the same time, depending on the sequences followed from the starting nodes.
- (Figure 3, d) N_3 groups of nodes to define interesting combinations (i.e. combinations for which user-actions become available). When the player reaches one of the ending nodes contained by a group of nodes the actions defined for the group became available to the player. Groups of nodes will usually define actions that lead outside of the multi-interaction node.

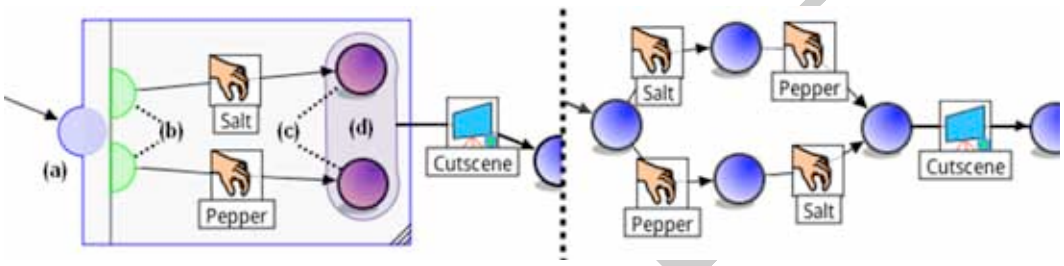

Figure 3. A "multi-interaction node" describes a series of actions that have to be performed but their order is not important. This example shows that adding the salt or the pepper in any order does not affect the result, by using a "multi-interaction node" (left) and by using the basic elements in the system (right).

It must be noted that the "multi-interaction node" representation cannot be expressed formally within the state-diagram definition, as the player will simultaneously be at several nodes within the "multi-interaction node" at once. However it is possible to define an algorithm to convert any "multi-interaction node" into a set of basic representation elements as shown in the example (Figure 3, right).

4.3 Structures to reduce complexity

Complex games, with high number of actions available to the player at any given point, can enhance player engagement, as choice and challenge are two of the main aspects of flow [52]. Therefore our language favors games with many choices making explicit when these choices have consequences (which is favored in game design). However, one way to easily add greater choice in traditional game development tools is to add actions that have loose conditions (i.e. that can be performed at any given time). This cannot be directly achieved by the DSVL presented here, given that such actions would need to be defined for every possible state in the story-flow. This problem is tackled in two different ways, by the use of "parallel-story nodes" and "virtual nodes", which addresses two different aspects of actions widely available throughout the game.
The "parallel-story node" element allows for the definition of actions that, while available at different points in the story, trigger a sequence of actions that do not affect the story itself. For instance, one of these elements could be used to define a sequence of actions needed to read some background information (fire protocol book, in this example), while allowing the player to continue along the story without completing the sequence (Figure 4).

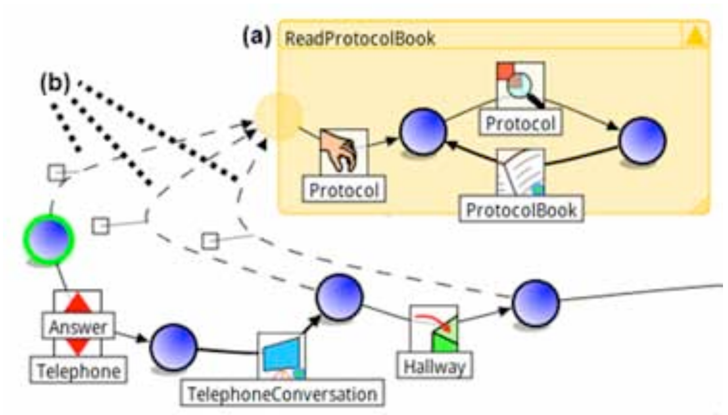

Figure 4. A "parallel-story node" (a) allows the definition of actions that can be performed at different points in the story (b), potentially describing a parallel story, but that have no effect on the game story-flow. In this example, the player can grab a "manual" and read it at different points in the story.

The "virtual node" element, in contrast, allows for the definition of actions that can change the story-flow. These elements can be used, for example, to define actions with "catastrophic" consequences (e.g. the game ends), which "ejects" the player from the story-flow (e.g. performing a dangerous action in a simulation game that ends the game with a warning and/or forces the player to restart) (Figure 5, left). Any action defined for a "virtual node" is available to any state associated with the virtual node (i.e. virtual nodes behave in a similar way to abstract classes in an object-oriented programming language).

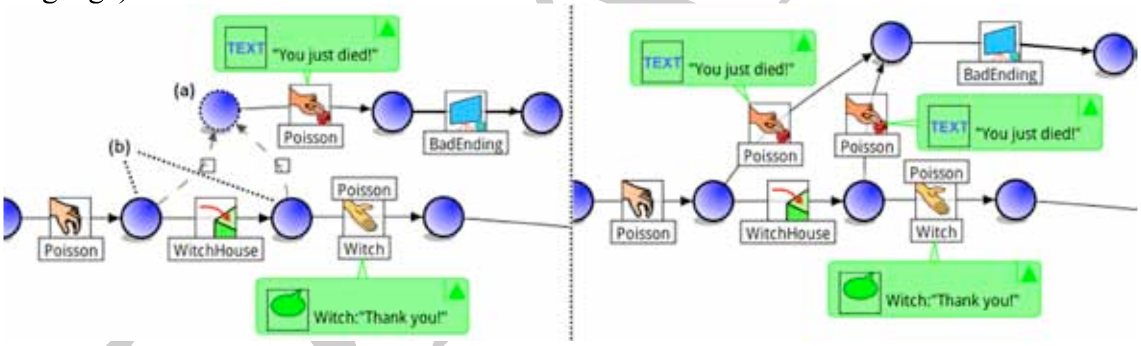

Figure 5. A "virtual node" (a) allows the definition of actions that can be performed at different points in the story (b) and that have the same effect on the flow of the main story. In this example, using the poison will cause the character to die. To the right, the same behavior is shown without using virtual nodes and, it is only possible to represent it by repeating states and effects.

5. Educational elements

Educational elements in the language cover different aspects of educational video games. We focus on three characteristics of educational video games that contribute to learning: assessment, guidance and adaptation [30, 35, 53]. Each of these aspects is covered by one or more specific elements of the DSVL, in an attempt to make the introduction of these characteristics in the game as straightforward as possible. This follows the central premise that specific language features might be needed to fit different requirements (e.g. inexperienced or advanced users), cover different standards (e.g. the IMS Question & Test Interoperability [54]) or create games for specific fields (e.g. medicine or math). Another aim of the use of explicit educational constructs is to increase the visibility of the educational features to the developer and, as a consequence, increase the chance that they will be added to the games.

5.1 Explicit representation of student assessment

Educational games have great potential for tracking and assessing students' learning outcomes compared with less interactive educational media. In-game student assessment allows educators to highlight and score relevant actions. These can then be used to generate feedback and help identify and rectify inaccurate assumptions, which is essential in educational games [3].
In our DSVL, student assessment is part of the output language (Figure 6, b). This output element, however, has no immediate consequences for the player. Instead, it modifies the overall score and writes a line in a human-readable assessment report. As the assessment element is part of the language it is part of the story-flow representation (Figure 6, a) and helps make evaluation an integral part of the game.

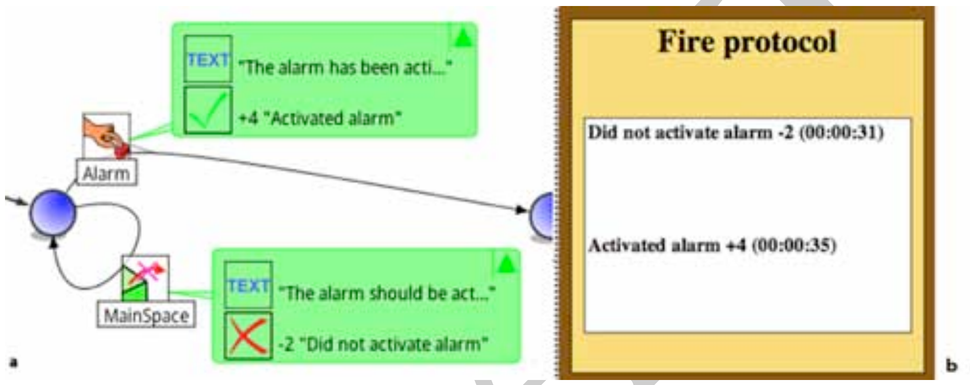
TEXT "The alarm has been acti..."
+4 "Activated alarm"
Alarm
MainSpace
TEXT "The alarm should be act..."
-2 "Did not activate alarm"
a
Fire protocol
Did not activate alarm -2 (00:00:31)
Activated alarm +4 (00:00:35)
b

Figure 6. The use of assessment as part of the output language allows the easy definition of relevant actions. In this case, the activation state of the fire alarm, influences the player's overall score (a). (b) shows an assessment report where the student didn't activate the alarm at first but later did.

The DSVL allows for the tracking of any in-game interaction. Nevertheless some educators might find it hard to define their own interactions to evaluate students. For instance, a construct familiar to educators is the multiple-choice question, but the definition of such elements in an educational game can become a series of complicated interrelations between variables and conditions. In an effort to present a clear and intuitive representation of multiple-choice questions in games, a specific element is included in the language (Figure 7). This element, like the story-section node previously described, adds a new set of states and transitions to the state-diagram. The language could be extended further, for example, by adding a visual representation of mathematical questions with numerical answers.

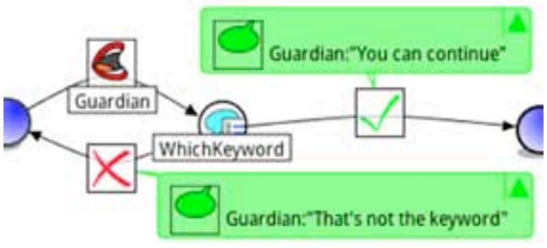
Guardian:"You can continue"
Guardian
WhichKeyword
Guardian:"That's not the keyword"

Figure 7. Ready to use components such as multiple-choice questions simplify the development process and encourage in-game student assessment. In this case the player must answer a multiple-choice question before continuing.

5.2 In-game hints for student guidance

In-game help allows players to make progress regardless of knowledge level or background. One way to provide such direction is through an in-game hint system, a solution also used in commercial games such as the latest "remake" of the *Monkey Island*™ saga. These hints provide guidance to players who need it, but do not disturb players who can do without. This broadens the audience, by providing breadcrumbs in the game for players that would have found it too difficult otherwise.

Taking this into consideration, the DSVL includes an element aimed to provide in-game tips to the student. These hints are defined as properties of nodes (the point in the story in which the hint is relevant) (Figure 8). As the excessive use of hints can reduce the exploratory nature of games, the hints are delivered when requested and at a cost to the player (i.e. reduction of the global score) established by the designer. Some hints can be provided at no cost, such as those that help users deal with the interface. However, an adaptive pedagogical agent (a.k.a. mentor or tutor) such as the one found in the *Prime Club/Clime* game could also use the information to provide hints to the player when needed [55]. Two approaches can be used to create such a mentor: explicitly create a new character in the story with which the player can talk at the different times and who provides the information; or, with modifications to the system, use the hints as conversation lines for a new character in the game that takes the role of the mentor. However, decisions about how to present the mentor or the hints to the student can have different consequences in the educational outcome of games (for instance, proactive or on-demand hints can affect learning in different ways for different students [56]) and thus must be left for the educator to decide.

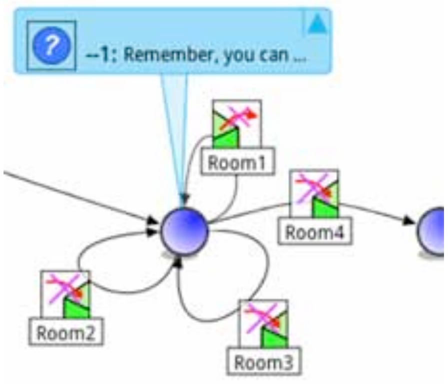

Figure 8. In-game guidance or hints are defined as properties of nodes. The hints define their cost to the player (-1 in this case) and the text that will be shown.

5.3 Adaptation of game content

Adaptation in video games is the act of changing something in the game to suit the needs of a particular individual or group of individuals. The most basic kind of adaptation found in games is based on changing difficulty based on a finite set of stereotypes (e.g. "novice", "intermediate" and "expert"). This helps a player find the golden mean between boredom and frustration. In this case it is selected by the player at the beginning of the game (i.e. "static adaptation"), while in other cases it is automatically inferred according to the player's past performance (i.e. "dynamic adaptation") [57]. Other adaptations can also be performed in a similar manner (e.g. presenting the game differently the first time it is played). However, as adaptation usually modifies only a part of the story, story-section elements are used as a part of the adaptation mechanism in this case.

In our DSVL a story-section flow can have more than one initial state (Figure 9). The decision of which initial state is used in a particular session is left to the game engine, introducing welcome variation in the game from the players' perspective. The initial nodes of the story-section flow are associated with different adaptation profiles in the game, for instance, "easy", "medium" and "hard" difficulty levels. Which profile is actually used can be left to the player or configured by the Virtual Learning Environment (VLE) in games played within those systems.

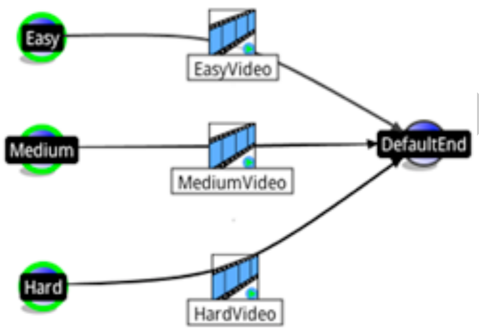

Figure 9. Adaptation of the story-flow is defined at a story-section level in the DSVL. This allows different part of the game to behave differently in each game run, depending on the needs of the player.

6. Usage example

This usage example describes the full story-flow of a simple, but representative, educational video game using our DSVL. This example video game can be downloaded[5] or replicated by transforming a visual description such as the one presented here into an <e-Adventure> game. In this game, the player must carry out an evacuation protocol during a fire. According to this protocol, there is one person on each floor who is responsible of investigating any sign of fire on that floor. If a fire is confirmed (and can not be extinguished) he/she should proceed with the evacuation. In this game the player plays the role of such a person.

The game starts when the player receives a phone call reporting a potential fire in one of the offices on his/her floor (Figure 10, a). After the situation is explained to the player, the office in question must be checked to verify both the existence and assess the intensity of the fire (Figure 10, b). Later, the fire alarm must be activated (Figure 10, c) and all the occupants of the building evacuated (Figure 10, d). This last step is detailed

[5] *http://e-adventure.e-ucm.es/course/view.php?id=18&topic=2* (retrieved on 17/11/2010)

using a story-section to hide complexity (Figure 11). Finally, the game ends (successfully) when the player leaves the building using the stairs (Figure 10, e). If the player does not complete the whole procedure in less than 4 minutes, is unable to provide the staff with appropriate instructors to evacuate the building calmly, panics, or tries to use the elevator the game arrives at the "failure" end state.

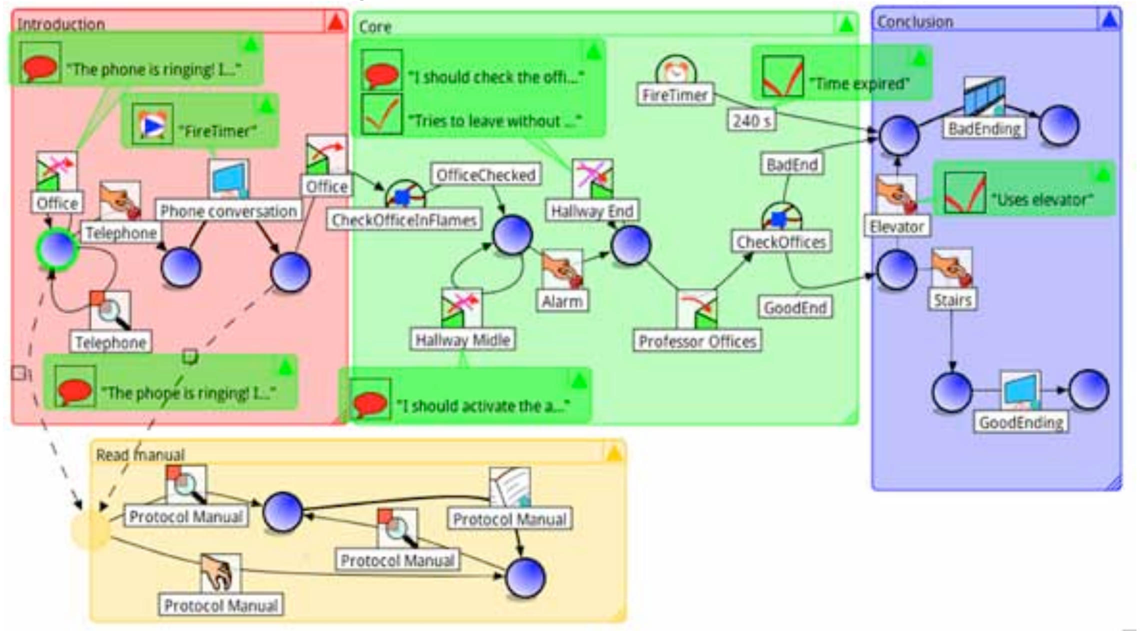

Figure 10. Fire protocol game description. This story-flow represents the full story of the fire protocol game, including student assessment, using an implementation of the DSVL described in this paper.

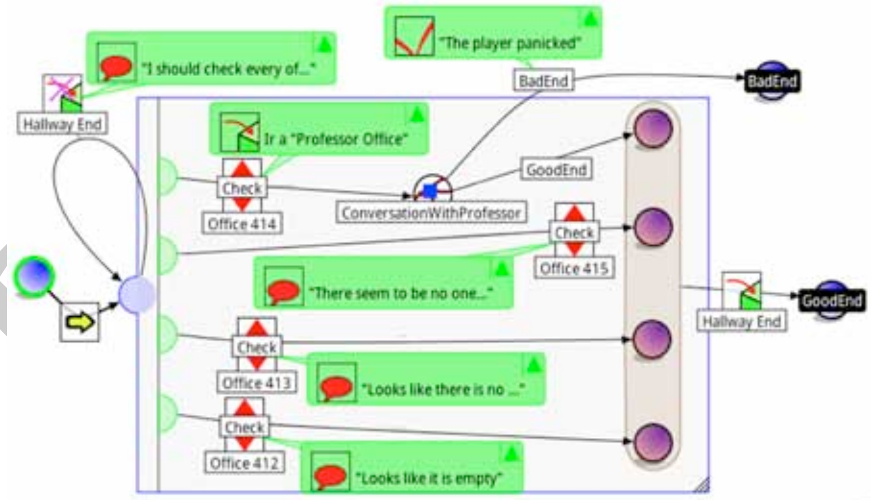

Figure 11. *Check offices* story-section in the fire protocol game. This story-flow represents the part of the Fire protocol game in which the player must check every office in a floor to make sure no one is left behind.

7. Conclusion

This paper presents a new DSVL that aims to simplify the creation and maintenance of educational video games. The main goal is to facilitate the transition from the game storyboard to a playable game by means of an explicit representation of the game story-flow. This language was built following EUD guidelines, with educators and domain-

experts as target users. It is intended to fill the needs of educators, with limited or no technical background, who are developing new games or modifying existing ones.
The language is defined as an extensible framework, allowing new constructs to be included to fit different needs. For instance, it includes advanced expressive capabilities to represent both the most usual game situations and to easily include the educational game characteristics (e.g. assessment) decreasing the required effort to build the final educational game.
This DSVL has been applied in a simple yet relevant usage example to showcase its capabilities and advantages. This shows how the hierarchical representation allows for a compact but comprehensive visual representation of an actual game's story-flow.
A beta release of the language can be found in the WEEV (Writing Environment for Educational Video games) system, part of the <e-Adventure> project, which is freely available for download[6]. This system facilitates the rapid prototyping of games to allow for early client and user evaluation, as it provides mechanism for the direct transformation of the games visual descriptions into playable games. We are currently working on creating a final release of the system.
Future work includes the creation of more components in the DSVL, including the development of some language constructs that already exist in other E-Learning specification such as IMS QTI for representing questions and tests to be presented to the user. Besides, end-user evaluations such as those presented for other languages [58] are needed to confirm the usability of the proposed DSVL and evaluations of the resulting games need to be performed to asses their usability, playability and the interest they can arise in students. Other efforts are underway focused on identifying and implementing specific elements useful in different fields, such as the medical field where educational games using <e-Adventure> have proven to be useful given the tight constraints of real environments [38], and the use of the hint mechanism to complement the adaptation mechanism by providing different hints for users with different needs.

Acknowledgments

The Spanish Committee of Science and Technology (TIN2010-21735-C02-02) and the Ministry of Industry (grants TSI-020110-2009-170, TSI-020312-2009-27) have partially supported this work, as well as the Complutense University of Madrid and the Regional Government of Madrid (research group 921340 and project e-Madrid S2009/TIC-1650), and the PROACTIVE EU project (505469-2009-LLP-ES-KA3-KA3MP) and the GALA EU Network of Excellence in serious games.
Thanks to Carl Blesius, LCS-MGH, Harvard University for his proof reading, insightful comments and language assistance in the writing of this paper.

References

[1] J. Juul, Half-Real: Video Games between Real Rules and Fictional Worlds, MIT Press, Cambridge, Massachusetts, 2005.
[2] S. Wexler, K. Corti, A. Derryberry, C. Quinn, A. V. Barneveld, 360 Report on immersive learning simulations, The eLearning Guild, 2008.
[3] R. Garris, R. Ahlers, J. E. Driskell, Games, motivation and learning: A research and practice model, Simul. & Gaming, 33 (2002) 441-467.
[4] M. J. Dondlinger, Educational video game design: A review of the literature, J. of Appl. Educ. Technol., 4 (2007) 21-31.

[6] *http://sourceforge.net/projects/e-adventure/files/WEEV/* (retrieved on 17/11/2010)

[5] P. Sancho, R. Fuentes, P. P. Gómez-Martín, B. Fernández-Manjón, Applying multiplayer role based learning in engineering education: Three case studies to analyze the impact on students' performance, Int. J. of Eng. Educ., 25 (2009) 665-670.
[6] M. D. Dickey, World of Warcraft and the impact of game culture and play in an undergraduate game design course, Comput. & Educ., 1 (2010) 200-209.
[7] R. Blunt, Does game-based learning work? Results from three recent studies, Proceedings of Interserv./Ind. Train., Simul. & Educ. Conf. (I/ITSEC) Orlando, Florida, USA, (2007) 945-954.
[8] A. McFarlane, A. Sparrowhawk, Y. Heald, Report on the educational use of games, TEEM: Teach. Eval. Educ. Multimedia, 2002.
[9] M. Pivec, P. Pivec, Games in schools, ISFE-EUN Partnership, 2008.
[10] R. Sandford, M. Ulicsak, K. Facer, T. Rudd, Teaching with games: Using commercial off-the-shelf computer games in formal education, Futurelab 2006.
[11] P. Felicia, Digital games in schools: A handbook for teachers, European Schoolnet, 2009.
[12] A. Amory, K. Naicker, J. Vincent, C. Adams, The use of computer games as an educational tool: Identification of appropriate game types and game elements, Br. J. of Educ. Technol., 30 (1999) 311-321.
[13] M. D. Dickey, Game design narrative for learning: Appropriating adventure game design narrative devices and techniques for the design of interactive learning environments, Educ. Technol. Res. and Dev., 54 (2006) 245-263.
[14] S. De Freitas, M. Oliver, How can exploratory learning with games and simulations within the curriculum be most effectively evaluated?, Comput. & Educ., 46 (2006) 249-264.
[15] M. L. Ryan, Avatars of story, University of Minnesota Press, 2006.
[16] C. A. Lindley, Story and narrative structures in computer games, Developing interactive narrative content, in B. Bushoff (Ed.), Sagas/Sagasnet reader. Munich, 2005.
[17] S. W. McQuiggan, J. P. Rowe, S. Lee, J. C. Lester, Story-based learning: The impact of narrative on learning experiences and outcomes, Lecture Notes in Comput. Sci., 5091 (2008) 530-539.
[18] S. M. Fisch, Making education computer games "educational", Proceedings of 2005 Conf. on Interact. Des. and Child., Boulder, CO, 2005.
[19] A. Waraich, Using narrative as a motivating device to teach binary arithmetic and logic gates, Proceedings of 9th annual SIGCSE Conf. on Innov. and Technol. in Comput. Sci. Educ., Leeds, United Kingdom, 2004.
[20] B. Onder, Writing the adventure game, Game Design Perspectives, F. Laramee (Ed.), Charles River Media, Hingham, MA, 2002.
[21] J. Lewinski, Developer's Guide to Computer Game Design, Plano, TX, Wordware Publishing, 2000.
[22] A. Van Deursen, P. Klint, J. Visser, Domain-specific languages: An annotated bibliography. ACM SIGPLAN Notices, 35 (2000) 26-36.
[23] R. Esser, J. W. Janneck, A framework for defining domain-specific visual languages, Workshop of Domain Specif. Vis. Lang., in conjunction with ACM Conf. on Object-Oriented Programming, Syst., Lang. and Appl. OOPSLA-2001, Tampa Bay, Florida, USA, 2001.
[24] S. Göbel, L. Salvatore, R. A. Konrad, F. Mehm, StoryTec: A digital storytelling platform for the authoring and experiencing of interactive and non-linear stories, Int. Conf. on Interact. Digit. Storytelling (ICIDS 2008), (2008) 325-328.

[25] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, N. Mehandjiev, Meta-Design: A manifesto for end-user development, Commun. of the ACM, 47 (2004) 33-37, 2004.
[26] M. F. Costabile, D. Fogli, P. Mussio, A. Piccinno, Visual interactive systems for end-user development: A model-based design methodology, IEEE Trans. on Syst., Man, and Cybern., 37 (2007) 1029-1046.
[27] P. Moreno-Ger, D. Burgos, J. L. Sierra, B. Fernández-Manjón, Educational game design for online education, Comput. in Hum. Behav., 24 (2008) 2530-2540.
[28] S. Egenfeldt-Nielsen, J. H. Smith, and S. P. Tosca, Understanding video games: the essential introduction. Taylor & Francis, New York, 2008.
[29] J. W. Rice, New Media Resistance: Barriers to Implementation of Computer Video Games in the Classroom, J. of Educ. Multimed. and Hypermedia, 16 (2007) 249-261.
[30] N. Peirce, O. Conlan, and V. Wade, Adaptive Educational Games: Providing Non-invasive Personalised Learning Experiences, in proceedings of Second IEEE Int. Conf. on Digit. Games and Intell. Toys Based Educ., Banff, Canada, IEEE Computer Society Washington, (2008) 28-35.
[31] R. Carro, A. Breda, G. Castillo, A. Bajuelos, A methodology for developing adaptive educational-game environments, Proceedings of 2nd Int. Conf. on Adap. Hypermedia and Adap. Web-Based Syst. (AH 2002), (2002) 90.
[32] M. D. Kickmeier-Rust, N. Peirce, O. Conlan, D. Schwarz, D. Verpoorten, D. Albert, Immersive digital games: The interfaces for next-generation e-Learning, Universal Access in Human-Computer Interaction, Applications and Services, 4556 (2007).
[33] K. Salen, E. Zimmerman, Rules of play: Game design fundamentals: MIT Press, 2003.
[34] D. J. Shernoff, M. Csikszentmihalyi, B. Schneider, E. S. Shernoff, Student engagement in high school classrooms from the perspective of flow theory, Sch. of Psychol. Q., 18 (2003) 158-176.
[35] V. J. Shute, M. Ventura, M. I. Bauer, D. Zapata-Rivera, Melding the power of serious games and embedded assessment to monitor and foster learning: Flow and grow, in Serious games: Mechanims and effects, U. Ritterfeld, M. Cody, P. Vorderer (Eds.), Mahwah, NK, Routledge, Taylor and Francis, 2009, 295-321.
[36] J. Torrente, Á. del Blanco, E. J. Marchiori, P. Moreno-Ger, B. Fernández-Manjón, <e-Adventure>: Introducing educational games in the learning process, Proceedings of *IEEE EDUCON 2010* Madrid, Spain, 2010.
[37] A. Del Blanco, J. Torrente, P. Moreno-Ger, B. Fernández-Manjón, A general architecture for the integration of educational videogames in standards-compliant virtual learning environments, Proceedings of 9th IEEE Int. Conf. on Adv. Learn. Technol. (ICALT 2009) Riga, Latvia, IEEE Computer Society, 2009.
[38] P. Moreno-Ger, J. Torrente, J. Bustamante, C. Fernández-Galaz, B. F. Manjón, M. D. Comas-Rengifo, Application of a low-cost web-based simulation to improve students' practical skills in medical education, Int. J. of Med. Inf., 79 (2010) 459-467.
[39] R. J. Nadolski, H. G. K. Hummel, H. J. V. D. Brink, R. E. Hoefakker, A. Slootmaker, H. J. Kurvers, J. Storm, EMERGO: A methodology and tookit for developing serious games in higher education, Simul. & Gaming, 39 (2008) 338-352.

[40] M. D. Dickey, Engaging by design: How engagement strategies in popular computer and video games can inform instructional design, Educ. Technol. Res. and Dev., 53 (2005) 67-83.
[41] J. Torrente, P. Moreno-Ger, B. Fernández-Manjón, J. L. Sierra, Instructor-oriented authoring tools for educational videogames, Proceedings of 8th Int. Conf. on Adv. Learn. Technol. (ICALT 2008), Santander, Spain, (2008) 516-518.
[42] M. Overmars, Teaching computer science through game design, IEEE Comput., 37 (2004) 81-83.
[43] V. Grigoreanu, R. Fernandez, K. Inkpen, G. Robertson, What designers want: Needs of interactive application designers, Proceedings of IEEE Symposium on Vis. Lan. and Hum.-Centric Comput. (VL/HCC), Corvallis, Oregon, USA, (2009) 139-146.
[44] M. Boshernitsan, M. Downes, Visual Programming Languages: A Survey, University of California, Berkley, California UCB/CSD-04-1368, 2004.
[45] T. Arndt, E. Katz, Visual software tools for multimedia authoring, J. of Vis. Lang. and Comput., 21 (2010) 184-191.
[46] J.M. Dodero, Á. Martínez del Val, and J. Torres, An extensible approach to visually editing adaptive learning activities and designs based on services, J. of Vis. Lang. and Comput., 21 (2010) 332-346.
[47] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai, Scratch: programming for all, Commun. of the ACM, 52 (2009) 60-67.
[48] A. LaMothe, Tricks of the Windows Game Programming Gurus, second ed., Sams, Indianapolis, IN, 2002.
[49] R. Rouse, Game Design, Theory and Practice, Plano, TX, Wordware Publishing, 2001.
[50] M. J. Taylor, D. Gresty, M. Baskett, Computer Game-Flow Design, ACM Comput. in Entertain., 4 (2006).
[51] T. Malone, Toward a Theory of Intrinsically Motivating Instruction, Cogn. Sci., 5 (1981) 333-369.
[52] J. Chen, Flow in games (and everything else), Commun. of the ACM, 50 (2007) 31-34.
[53] D. Leutner, Guided discovery learning with computer-based simulation games: Effects of adaptive and non-adaptive instructional support, Learn. and Instr., 3 (1993) 113-132.
[54] IMS Global Consortium, IMS Question & Test Interoperability Specification, Version 2.0 Final Specification, 2005.
[55] C. Conati, Probabilistic assessment of user's emotions on educational games, Appl. Artif. Intell., 16 (2002) 555-575.
[56] L. Razzaq, N. Heffernan, Hints: Is It Better to Give or Wait to Be Asked?, in: V. Aleven, J. Kay, J. Mostow (Eds.), Intelligent Tutoring Systems, Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, 6094 (2010) 349-358.
[57] R. Hunicke, The case for dynamic difficulty adjustment in games, Proceedings of the 2005 ACM SIGCHI Int. Conf. on Advances in Comput. Entertain. Technol., Valencia, Spain, 2005.
[58] G. Costagliola, A. De Lucia, F. Ferrucci, C. Gravino, G. Scanniello, Assessing the usability of a visual tool for the definition of e-learning processes, J. of Vis. Lang. and Comput., 19 (2008) 721-737.