Bat Cave: A Testing and Evaluation Platform for Digital Educational Games

Florian Mehm, Viktor Wendel, Stefan Göbel, Ralf Steinmetz
Multimedia Communications Lab, Technische Universität Darmstadt
Florian.Mehm@KOM.tu-darmstadt.de
Viktor.Wendel@KOM.tu-darmstadt.de
Stefan.Goebel@KOM.tu-darmstadt.de
Ralf.Steinmetz@KOM.tu-darmstadt.de

Abstract: Adaptive Digital Educational Games (DEGs) are based on complex processes and algorithms, which can be tested and evaluated in various fashions. Concerning evaluation, either the overall effects of the games (e.g. on learning efficacy) or the functionality of algorithms can be evaluated. For testing, it is desirable for game authors to try out the game they are working on early during development, in order to test the effects of changes to the games' structure or parameters of the adaptive algorithms.

The European research project 80Days has developed a DEG for teaching geography that is based on several adaptive technologies. Among them are a Story Engine and a Learning Engine, with the former controlling the execution of the game, balancing aspects of narrative, gaming and learning (using data by the latter).

Of the testing and evaluation possibilities described above, evaluations of the effects of the game have been carried out with an immersive game demonstrator at schools. However, for the other purposes outlined, a separate platform named "Bat Cave" is described in this paper. The approach of Bat Cave is to implement a game platform in a prototypical fashion, by creating simplified representations of situations found in the game. Details of the workings of the algorithms are visualized in several ways.

Using Bat Cave, evaluation and testing of the methods and parameters is possible. In this context, Bat Cave is used as a testbed for this novel approach of creating adaptive DEGs. On the other hand, Bat Cave, due to its simplified structure, can be used by game creators to rapidly create early versions of the game, which can be tested extensively without having to play the actual game. Therefore, both the design of the game's structure and content (narrative, learning content) as well as the impact of parameters for the adaptive algorithms on the game can be tested early, allowing a game creation process based on early prototypes.

Keywords: Testbed, Evaluation, Rapid Prototyping, Adaptivity

1. Introduction

Digital Educational Games (DEGs) have the goal of imparting knowledge to their players in fun and motivating ways by melding the educational content with exciting gameplay and stories (Kickmeier-Rust et al. 2008). For this purpose, it is desirable to fine-tune a given game to the specific background and current context of a player. This background could include aspects such as the "type" of player (e.g. is he/she very competitive or more oriented towards socializing in the game?) or the current level of knowledge of the player. The process of fine-tuning is carried out by adaptive systems which allow a game to be personalized to the player along different dimensions (including storytelling, gaming and learning) as well as on different levels of granularity (micro-level adaptation for small, localized changes to the game, macro-level adaptation for global changes such as a different storyline).

While such adaptations are very desirable for the player as they can increase both the efficacy of learning and the enjoyment of the game, they are also more difficult to handle during production. Factors that are included in this are increased cost, larger complexity and less authorial control over the game. Higher costs are due to the need to create more content or gameplay in order to allow the game the be adapted to different players, resulting in game variations that feature different paths and content that will only be shown to a part of the players. The increase in complexity and the reduction in authorial control over the game both stem from the observation that the producer/author of a DEG has less control over the exact unfolding of a game at runtime than in a non-adaptive game. This effect is commonly referred to as the Narrative Paradox (Louchart and Aylett 2003).

In an adaptive DEG, another layer of control is introduced into the system, where at runtime the continuation of the game also depends on choices the game's system makes depending on the

current context of the user. In this respect, it is important for an author creating such an adaptive DEG to acquire a feeling for the effects of input to the algorithms underlying this mechanism. This is added to the generally complex task of creating a non-linear, interactive game, which is a large departure from designing a non-interactive and non-linear experience such as a movie.

In order to support the design, production and run-time execution of games using these techniques, the authors have previously presented models and methods for both the authoring process and the execution of such games (Göbel et al. 2008), which have been used in the context of the European research project 80Days (Kickmeier-Rust et al. 2008). While the 80Days game, intended to be played by the target audience of younger students, is an example of an immersive 3D game, it does not allow to quickly test the underlying algorithms nor is it designed for rapid prototyping, where an early version of the game can be created using placeholder content etc. in a short timeframe. For this purpose, the "Bat Cave" application has been created with the intent of using it to test and validate the adaptive algorithms used in 80Days as well as providing a rapid prototyping toolkit. Since the base technology is shared between Bat Cave and the 80Days demonstrator games, results of tests using Bat Cave also apply to the other. In the following, a discussion of related work is followed by an architecture overview of the involved systems and a detailed discussion of Bat Cave.

2. Related Work

The results of an analysis of related work in the application fields for Bat Cave – a testbed for the developed adaptive DEG algorithms and a rapid prototyping tool during authoring – are presented here. For an overview of work related to the algorithms themselves, please see (Göbel et al. 2010).

Testbeds are a common way of carrying out functional evaluation of algorithms. In the area of adaptive game and storytelling technology, a testbed is intended to evaluate how a given algorithm performs under various inputs. (Peinado et al. 2008) describe the Remote-Controlled Environments Interface (RCEI) protocol, which defines a set of commands for connecting an interactive storytelling system to a virtual environment, with the intent of allowing the virtual environment to be used as a testbed for the storytelling system. A testbed to be used in the evaluation of algorithms for dynamic difficulty adjustment, an adaptive technology with a similar aim as the one described in this paper, is presented in (Bailey and Katchabaw 2005).

Whereas the intention of a testbed is the evaluation of algorithms during their development, rapid prototyping is used in the game design and development process to test ideas and implementation possibilities at an early stage, when changes can be made easily. For an account of the role of rapid prototyping in the game development process, see (Fullerton et al. 2004, chapter 7). (Coiana et al. 2008) describe a system for play-centric design by means of rapid prototyping of games on mobile devices.

Concerning the technical foundation of Bat Cave as described in section 3, an analysis of related work shows that the Story Engine component is an instance of an architectural pattern underlying many digital storytelling applications and storytelling-based games. Similar components found in other systems are Drama Managers or Story Directors, which are used to effect control over narrative parameters of a story. An example is found in (Magerko et al. 2004). Utilizing this pattern in a similar way to our approach, (BinSubaih and Maddock 2008) describe a system in which the so-termed "G-factor" ("game state, object model" and "game logic") is separated from an actual engine executing the game by the introduction of a layer of abstraction. A related pattern is found in systems that are controlled by the interactions of virtual characters as described in (Cavazza et al. 2002), which are referred to as character-driven storytelling systems.

Since the Story Engine pattern introduces a layer of abstraction between the game description as created in an authoring tool or editor and the actual runtime execution environment (e.g. a game engine) of the game, this game description and its execution can be seen as an instance of model-driven software development. The models in the game description include the control structures in the description language and the object models found in the game. Other instances where these properties of the story sngine pattern are used include (Montero-Reyno and Carsí-Cubel 2009) and (Marchiori et al. 2010), which is also related to the approach of the story editor of the StoryTec authoring system.

3. Background and Architecture

In this section, an overview of the architecture for modelling, authoring and executing adaptive DEG that is utilized by Bat Cave and the 80Days game is described.
Figure 1 is provided as an overview of the architecture; elements of the diagram are ordered as layers, with basic layers at the bottom and an application such as Bat Cave at the top. In general, the Story Engine and associated system together form a reusable component for the execution of DEGs.

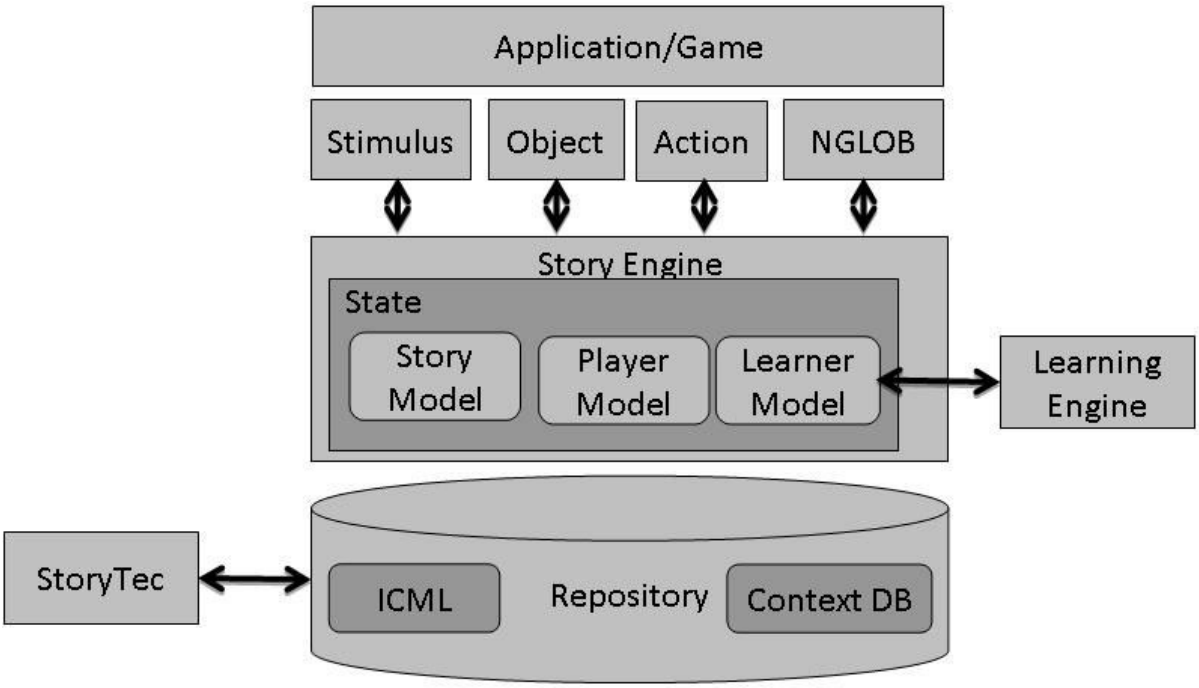

Figure 1: An architectural overview of applications built using the Story Engine (see section 3.2)

3.1 Narrative Game-Based Learning Objects

In order to combine the aspects of learning, gaming and storytelling for DEGs, the concept of Narrative Game-Based Learning Objects (NGLOBs) was introduced in (Göbel et al. 2009b) and extended in (Göbel et al. 2010). An NGLOB is an attributing element which is added to a scene of a game containing information about the scene's learning context, gaming context and storytelling function. The contained information is used to determine how the game should proceed from a certain point on (scene) respecting the player's learning background/performance and his/her gaming preferences as well as the meaningfullness of the story path.

Concerning the learning context, each scene is attributed with a set of prerequisite skills P, i.e. skills which should already be achieved by the player in order to play this scene and a set of associated skills A, the set of skills which will be taught by playing this scene. Possible next scenes are evaluated by comparing the scenes' prerequisite skills with the player's already learned skills.

Concerning the gaming context, an NGLOB contains information about the player types it is suited for. Therefore, based on the used player model, a value between 0 and 1 is assigned for each player type indicating how well the scene is suited for the respective player type. Additionally, the player's decisions are evaluated and according to his/her actions the model is updated. Possible next scenes' player model attributes are compared to the player model using a Least-Means-Square algorithm.

Concerning the narrative context, the main challenge is to overcome the Narrative Paradox (Louchart and Aylett 2003), which means to provide a suspenseful and motivating story and to give the player the feeling of influencing the course of the game. Using the Hero's Journey (Campbell 2008, Vogler 1998) as underlying story model, an NGLOB is attributed with a set of story steps indicating for which parts inside the story the scene could be used. Depending on the current story step this provides a selection of possible next scenes.

3.2 Story Engine

A reusable component for loading and executing games described using the xml-based ICML language was created in the form of the “Story Engine” (Göbel et al. 2008), which is attached to other applications (such as a game engine) and acts as a high-level control instance.

Concerning the concept of NGLOBs, the Story Engine was enhanced to include support for their handling. This includes a set of new functionality, including the parsing of NGLOB-related annotations of ICML scenes, the inclusion of model representations for player, learner and story models, as well as an enhanced method of choosing the next scene, thereby implementing the concept of NGLOBs as described in the last section.

For communication with the player application, the Story Engine uses several constructs. The player application acts as an observer and is informed whenever certain events in the Story Engine have taken place. The following information is distributed to the player application as a means of control:

- Scenes/NGLOBs as the basic organizational unit of ICML
- Objects such as characters or props
- Variables
- Actions which define events that should happen in the game world

Input into the engine from the player application is done mostly by means of stimuli. A stimulus in ICML is comparable to a configurable event handler that includes actions that are executed whenever the stimulus is triggered in the Story Engine.

Since these control structures are only defined as categories, actual implementations are specific to the game to be created and must be described and implemented both for the authoring process and for the execution in a player. For example, for the Bat Cave player as described below, one type of object are rectangular hotspots, which are defined to be overlayed onto images and to trigger a stimulus once they are clicked.

3.3 StoryTec Authoring Tool

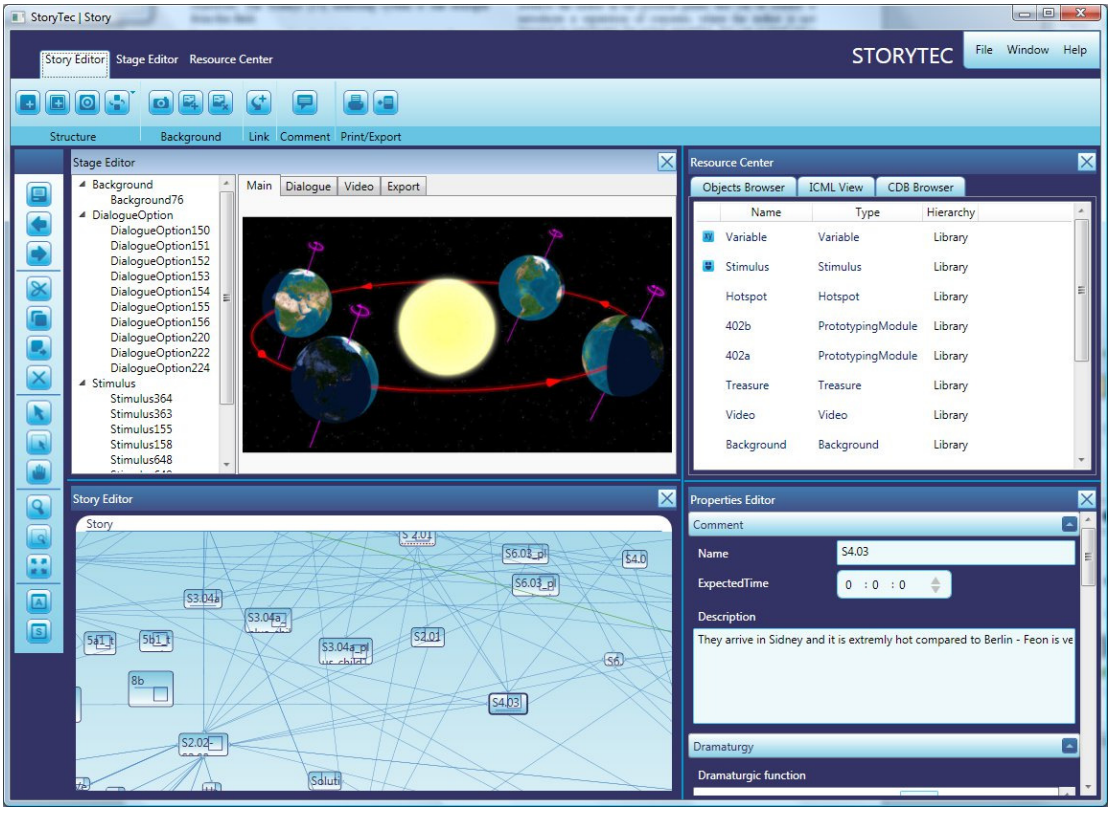

Figure 2: The StoryTec authoring system

In order to allow the creation of extensive game scenarios and aid in the authoring process of such scenarios, an authoring system which targets the ICML language and the Story Engine has been created. This authoring system, StoryTec (Göbel et al. 2008, Mehm et al. 2009), is based on a modular framework which can be adapted for different target player applications which are built upon the Story Engine.

StoryTec (shown in Figure 2) allows entering scenarios which use the full potential of the Story Engine, including annotations for NGLOBs and the creation of skill dependency graphs for the learning aspect and the Adaptive Learning Engine (see next section). Stimuli and the actions to be executed once they are triggered are set up in a visual fashion, allowing authors without knowledge of a programming language to set them up.

3.4 Adaptive Learning Engine

For handling the current knowledge state of a user (c.f. section 3.1), the Adaptive Learning Engine (Conlan et al. 2009) found in the 80Days games is also included in the Bat Cave player application. Besides managing the current state of the learner model, it interacts with the remainder of the application in the form of game evidence and micro-adaptive interventions. Game evidence is information about the current state of the game which is mapped to a discrete set of states and shared with the Learning Engine. Using this evidence from the game, the Learning Engine updates the learner model as appropriate, and may additionally make immediate short-term changes to the game in the form of adaptive interventions. These interventions are handled in the Story Engine using the mechanism of stimuli, and result in useful feedback for the player by accenting correct results, giving hints in case of incorrect decisions or motivating the player to try again.

3.5 Basic Prototyping Facilities

Instead of developing a full-fledged player application, it can be worthwhile to first test the envisioned design of the player application before it is implemented. For example, early tests could give valuable feedback that can be incorporated into the final implementation of the player and the game design.

Since the control and interaction mechanisms of the Story Engine are all defined on high levels of abstraction, a very basic initial prototype of a player application can be created semi-automatically. In the simplest case, all output of the Story Engine, such as the notification of scene changes or the execution of actions, could simply be logged by an application. For handling input to the Story Engine, the mechanism of stimuli can be used, since they are the default way of translating player actions into events to be handled by the Story Engine.

The prototyping facilities described here represent a baseline for a test and prototyping environment for games using the NGLOB-based approach and the Story Engine described in section 3.2. Their advantage is that, provided a basic framework, they are applicable without additional implementation work. However, they only allow a very restricted impression of how the game will be experienced by a user later on, which results in the need for prototyping and testing facilities mimicking the actual game experience better. To this end, it is possible to implement lightweight players which can simulate and visualize the actual gameplay better. One such player is the Bat Cave platform, as described in the next section.

4. Bat Cave

The Bat Cave platform is built upon the Story Engine as shown in Figure 1. It is composed of a generic part which is used for visualizing various useful sets of data that are accumulated during play and a part for simulation of gameplay. Both act as observers of the Story Engine, with the visualization parts passively visualizing the incoming information and the gameplay part also converting user input into Story Engine commands afterwards.

Concerning content, a complete Game Design Document was created in a similar fashion and level of detail as that used for the 80Days game, using a similar story of an interstellar treasure seeker who meets two children on Earth and embarks on an adventure with them. The design also included the definition of 132 skills and the prerequisite relations between them. The design was entered

completely using the StoryTec authoring tool as described in section 3.3. The following sections give an overview of the visualization part of Bat Cave, shown in Figure 3.

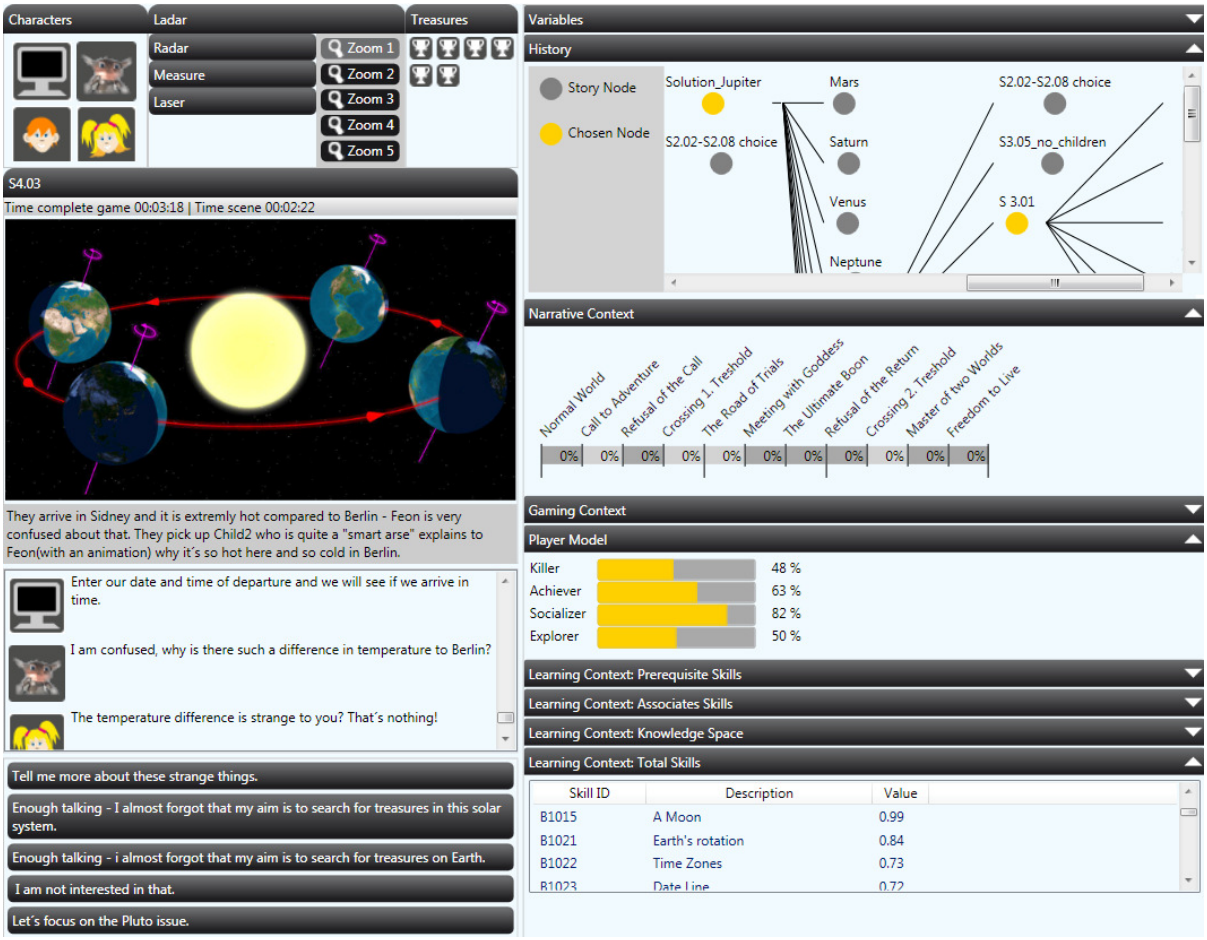

Figure 3: A complete overview of the Bat Cave application.

4.1 Gameplay

In order to generate user input and to visualize results of computations inside the Story Engine, the left part of the Bat Cave application is dedicated to a prototypical gameplay implementation. As noted above, using the abstraction layer provided by the architecture of the Story Engine, it would be possible to automatically generate a user interface for visualizing the current state of the game and generate inputs to the Story Engine. This, however, would result in a very abstract representation of the gameplay, and can be improved upon by providing new layers of more concrete implementations of the gameplay.

Among the output mechanisms used in Bat Cave are background images, characters and their speech acts, “treasures” that can be collected as well as descriptive texts.

For the Bat Cave scenario, the following elements are used as input mechanisms for the Story Engine:

- Hotspots: Rectangular hotspots can be overlayed onto the background image for a range of different effects. They can be directly linked to stimuli which are injected into the Story Engine when they are clicked.
- Multiple-Choice dialogues: In the lower part of the player, dialogue choices for the player’s avatar are offered, which again result in stimuli being injected.
- Adaptive Interventions: Adaptive Interventions as issued by the Learning Engine are internally transformed into stimuli and injected into the Story Engine.

- Specialized prototyping modules: Software modules which, when loaded, take control of the player and the communication with the Story Engine. They are used to provide more specialized prototype gameplay than is possible with the regular Bat Cave system.

Internally, all output mechanisms are either directly associated with Story Engine objects (which are linked to an output game element, e.g. an image) or an action (such as the speech act of a character).

4.2 Visualization

Wheras the left part of the Bat Cave user interface is dedicated to the representation of gameplay and is linked to a certain degree to the used game objects, the right side of the user interface is composed of several general-purpose and reusable components for visualization of the current state of the underlying system. Relevant information is continuously updated in the visualization interface and simultaneously written into a log for later use. The following sections describe the individual visualization components in more detail.

4.2.1 Variables

The first visualization found in Bat Cave is used to display the active variables found inside the current game session. ICML variables can either be defined locally for a certain scene or globally for the whole game. They can be used in several traditional ways such as flags (e.g. to indicate that a certain, important point in the game's story has been passed or that a certain event has happened) or as counters (e.g. for counting the number of times a player visited a certain location or the number of failed tries for a given challenge).

4.2.2 History

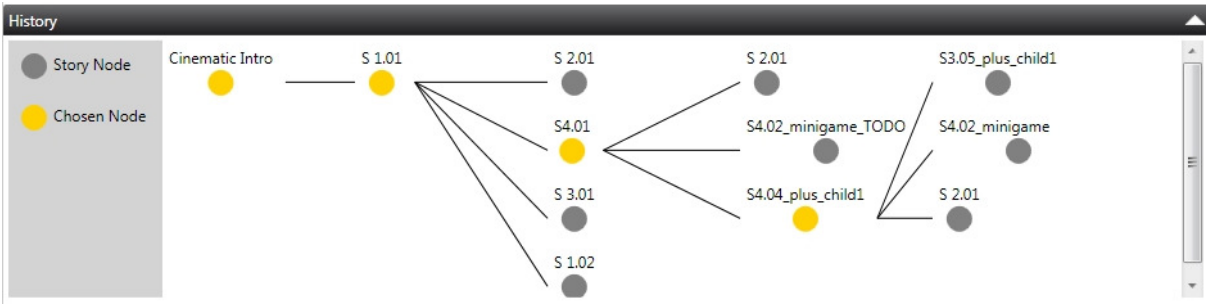

Figure 4: The History visualization component. Yellow nodes indicate the current and past active scenes.

This component, shown in Figure 4, fulfills a multitude of visualization and assistance functions during a test session. It displays scenes as dots, similar in abstraction to the way they are displayed in the Story Editor of StoryTec. Scenes are differentiated by colors, with yellow indicating the scenes that have been visited previously as well as the currently active scene. Whenever branches are possible (as inferred by inspecting the transitions defined from a certain scene), they are visualized inside a new column to the right of the currently active scene.

Additionally, when the mechanism of NGLOBs is used in the form of "free transitions", i.e. transitions between scenes that are influenced by the adaptive algorithms in the system, a context menu offers detailed information about the algorithmic parameters that lead to a certain choice.

4.2.3 Narrative Context

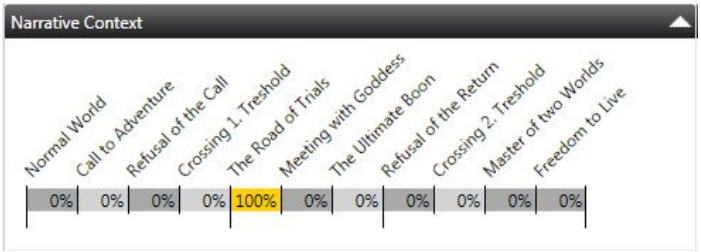

Figure 5: The Narrative Context visualization component

The visualization for the Narrative Context, as shown in Figure 5, displays the individual steps of the story model that is used as the basis for the game's story. For each step, a percentage shows the appropriateness values as entered by the author. For easy visual reference, all steps for which the current appropriateness value exceeds a certain threshold are marked in yellow. Furthermore, the background color of the cells for Story Model steps that have previously been visited in the course of the game session are marked in a brighter colour, to allow the tester to see how far along in the narrative he/she currently is and what steps have already been visited.

4.2.4 Player Model and Gaming Context

Figure 6: The Player Model and Gaming Context visualizations.

Two different visualizations are provided in Bat Cave which are associated with the current state of the player model and the gaming context of the current scene (see Figure 6). Each player attribute is represented by a bar which shows the percentage to which this attribute is set, as well as the exact value as a percentage.

4.2.5 Skills, Knowledge Space

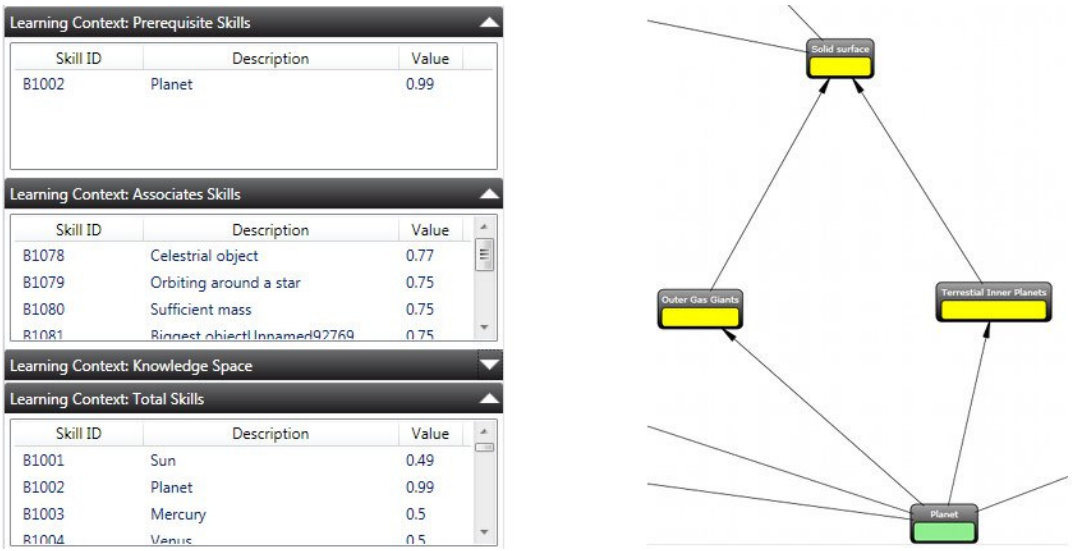

Figure 7: Visualization of associated, prerequisite and total skills (left) and an extract of the visualization of the player's skill state (right).

The visualizations described here (see also Figure 7) are all related to the learning context of NGLOBs. The associated and prerequisite skills as specified for the currently active scene are each listed together with their current values as determined by the Learning Engine. The Total Skills visualization similarly lists all skills defined in the skill structure for the active game together with their current values.

As an alternative to the Total Skills visualization, the structure defined by the skills and the dependency relations between them is visualized in the Skill Tree visualization component. This component features a graphical representation of the skill structure, with skills being shown as rectangles and the dependencies between them as arrows. For each skill, its current probability value is used to determine the color of the node representing it, interpolated from red for 0 to green for 1. This allows to quickly see areas of the skill structure that have been covered well in a certain

playthrough and others that have been less covered during play, especially when skill structures get larger.

5. Conclusion

In this paper, the challenges of providing a framework for the creation and execution of DEGs, the functional evaluation of such a framework and the provision of rapid prototyping facilities during the DEG creation process have been addressed. The Bat Cave player as well as the underlying architecture, including the Story Engine implementing the concept of Narrative Game-Based Learning Objects, the Adaptive Learning Engine for learner modelling and the StoryTec authoring tool have been presented as our approach towards solving these challenges.

Both for functional evaluation of the adaptive mechanisms described in this paper as well as to demonstrate the usefulness of Bat Cave as a tool for rapid prototyping, a comprehensive game design document, intended to be of a similar level of detail as the original 80Days design document, was created. Using the general-purpose prototyping facilities of Bat Cave as described in section 4.1, it was possible to implement most of the scenes found in the game design and map the interactions found in the design onto the prototypical interaction types in the player application. Furthermore, after initial instructions and working on smaller tutorial scenarios, the author tasked with entering the game did not encounter serious limitations during the process and was able to enter the structure and content of the design in a fraction of the time that was required for the implementation of the full 80Days demonstrator games. Also, the basic nature of Bat Cave made it possible to quickly try out newer versions of the authored game by being able to rapidly play to the scenes that are currently being worked on. Therefore, the feasibility of the approach for this purpose has been demonstrated.

Functional evaluation using Bat Cave showed some possible areas of improvement for the implementation of the adaptive algorithms for handling NGLOBs, especially the need for better calibration of the trade-off between the different axes, e.g. by defining weighting factors for setting priorities for learning, gaming and storytelling, which will be incorporated in the future work on the system.

6. Acknowledgements

The research and development introduced in this work is funded by the European Commission under the seventh framework program in the ICT research priority, contract number 215918 (80Days, www.eightydays.eu).

References

Bailey, C. and Katchabaw, M. J. (2005) “An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games”, *Proceedings of the 2005 GameOn North America Conference*, Montreal, Canada, August 2005, pp. 18-22.

BinSubaih, A. and Maddock, S. (2008) “Game Portability Using a Service-Oriented Approach”, *International Journal of Computer Games Technology*, Vol. 2008, Article ID 378485.

Cavazza, M., Charles, F. and Mead, S. (2002) “Character-Based Interactive Storytelling”, *IEEE Intelligent Systems*, Vol. 17, Issue 4, pp. 17-24.

Coiana, M., Conconi, A., Nigay, L. and Ortega, M. (2008) “Test-Bed for Multimodal Games on Mobile Devices”, *Fun and Games*, LNCS Volume 5294/2008, Springer Berlin / Heidelberg, pp. 75-87.

Campbell, J. (2008) *The Hero with a Tousand Faces*. 3rd edn. New World Library.

Conlan, O., Hampson, C., Peirce, N. and Kickmeier-Rust, M. (2009) "Realtime Knowledge Space Skill Assessment for Personalized Digital Educational Games", *Proceedings of Ninth IEEE International Conference on Advanced Learning Technologies*, pp. 538-542.

Fullerton, T., Swain, C. and Hoffman, S. (2004) *Game Design Workshop: Designing, Prototyping, & Playtesting Games*, Focal Press.

Göbel, S., Salvatore, L., Konrad, R. and Mehm, F. (2008) “StoryTec: A Digital Storytelling Platform for the Authoring and Experiencing of Interactive and Non-linear Stories”, *Interactive Storytelling*, LNCS 5334, Springer, Berlin / Heidelberg, pp. 325-328.

Göbel, S., Mehm, F., Radke, S. and Steinmetz, R. (2009) “80Days: Adaptive Digital Storytelling for Digital Educational Games”, *Proceedings of the 2nd International Workshop on Story-Telling and Educational Games (STEG'09)*, no. 498, CEUR Workshop Proceedings.

Göbel, S., de Carvalho Rodrigues, A., Mehm, F. and Steinmetz, R. (2009) “Narrative Game-based Learning Objects for Story-based Digital Educational Games”, *Proceedings of the 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games*, pp. 43-53.
Göbel, S., Wendel, V., Ritter, C. and Steinmetz, R. (2010) “Personalized, Adaptive Digital Educational Games using Narrative Game-based Learning Objects”, accepted to *Edutainment 2010.*
Kickmeier-Rust, M., Göbel, S. and Albert, D. (2008) “80Days: Melding Adaptive Educational Technology and Adaptive and Interactive Storytelling in Digital Educational Games”, *Proceedings of the First International Workshop on Story-Telling and Educational Games (STEG'08)*, CEUR Workshop Proceedings
Louchart, S. and Aylett, R. (2003) “Solving the narrative paradox in VWs - Lessons from RPGs”, *IVA*, LNCS 2792, Springer, Berlin / Heidelberg, pp. 244-248.
Magerko, B., Laird, J., Assanie, M., Kerfoot, A. and Stokes, D. (2004) “AI characters and directors for interactive computer games”, *Proceedings of the 16th conference on Innovative applications of artifical intelligence*, AAAI Press / The MIT Press, pp. 877-883.
Marchiori, E., Torrente, J., del Blanco, Á., Moreno-Ger, P. and Fernández-Manjón, B. (2010) “A Visual Domain Specific Language for the Creation of Educational Video Games”, *IEEE Learning Technology Newsletter*, Vol. 12, No. 1, January 2010, pp. 36-39.
Montero-Reyno, E. and Carsí-Cubel, J. (2009) “A Platform-Independent Model for Videogame Gameplay Specification”, *Breaking New Ground: Innovation in Games, Play, Practice and Theory*, Brunel University, London.
Mehm, F., Göbel, S., Radke, S. and Steinmetz, R. (2009) “Authoring Environment for Story-based Digital Educational Games”, *Proceedings of the 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games*, pp. 113-124.
Peinado, F., Navarro, Á. and Gervás, P. (2008) “A Testbed Environment for Interactive Storytellers”, *International Conference on Intelligent Technologies for Interactive Entertainment*, ACM Digital Library, New York.
Vogler, C. (1998) *The Writer's Journey. Mythic Structure for Storytellers and Screenwriters*. Michael Wiese Productions.