PERGAMON

Computers & Graphics 26 (2002) 45–55

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Synergies between interactive training simulations and digital storytelling: a component-based framework

Ralf Dörner*, Paul Grimm, Daniel F. Abawi

Fraunhofer AGC, Varrentrappstrasse 40-42, 60486 Frankfurt am Main, Germany

Abstract

A vital requirement for a successful software framework for digital storytelling is that it takes the abilities and background of the story authors into account. Dedicated tools should support authors in expressing their stories within this framework at an adequate level and point out an according authoring process for digital stories. The software framework should provide communication interfaces between technology experts, storytelling experts and application domain-experts. These requirements are similar to the ones already encountered when setting up a framework for interactive training applications. We present a concept how component and framework methodologies from software engineering as well as concepts from artificial intelligence can foster the design of such a software framework. The software architecture of our proposed framework is discussed as well as the according authoring process and tools. An implementation of our concept is described and lessons learned during using this framework in the application domain of emergency training are addressed. Although the framework has been applied for training purposes in particular, it can be used as a basis for a digital storytelling framework in general.

1. Introduction

Digital storytelling differs in some fundamental aspects from conventional storytelling. In digital storytelling, users are more often viewed not only as listeners but also as people who can interact and shape the story. Another vital difference is that in digital storytelling computers and information technology is used as a medium for storytelling. As a consequence, the storyteller needs a dedicated software framework that allows her to describe the story she wants to tell and to present it to her audience, possibly in an interactive fashion. This dependency from technological issues is a problem, since conventional storytellers are seldom experts in mastering computer technology while computer experts seldom have the abilities to tell good stories.

Thus, a digital storytelling framework should at least free the authors from dealing with lower level technological issues like services concerning network distribution, communication infrastructure, provision of interaction mechanisms or animation presentation and rendering in real-time. Moreover, the framework should facilitate a seamless transition from story authoring to story presentation. But this is not enough. A crucial requirement for a successful digital storytelling software framework is that it takes the abilities and background of the story authors into account and supports "technology non-experts" in expressing their stories within this framework at an adequate level. In order to meet these requirements, dedicated authoring tools need to be conceived within the digital storytelling framework together with an authoring process that supports the communication between technology experts and storytelling experts.

In this paper, we want to show how component and framework methodologies from software engineering as well as concepts from artificial intelligence can foster the design of a digital storytelling software framework. We claim that the usage of these methodologies is not only

*Corresponding author. Fraunhofer Anwendungszentrum Computergraphik in Chemie und Pharmazie, Varrentrappstrasse 40-42, 60486 Frankfurt am Main, Germany. Tel.: +49-69-97-995-152; fax: +49-69-97-995-199
http://www.agc.fhg.de/.

E-mail address: doerner@agc.fhg.de (R. Dörner).

0097-8493/02/$ - see front matter
PII: S0097-8493(01)00177-7

beneficial for the internal structure of the software and promises a faster, more reliable and more cost-efficient implementation of applications that make use of digital storytelling techniques. But we will show that the concepts presented in this paper also have implications for the authoring process and improve the communication between the different groups of people involved in digital storytelling. Here, we distinguish not only between the computer technology group and the storytelling group with artistic or design background, but we also distinguish a group of so-called application authors. This third group is neither expert in storytelling nor expert in information technology but they have fundamental knowledge about the domain the story is about. Most software systems about digital storytelling have neglected this third group so far.

We use an implementation of our proposed framework for training teams how to cope with emergency scenarios. How does this relate to digital storytelling? One interesting training concept is to use methodologies from digital storytelling: The trainer describes a scenario (i.e. description of the location, the objects, and the participants) and a plot what is going to happen in the scenario (i.e. the story line). The trainees are engaged in this scenario, take over several roles and act accordingly, thereby influencing the outcome of the scenario. While the trainees are acting, the trainer on the other hand can modify the plot (e.g. making a scenario more crucial by introducing malfunctions) and may adapt the scenario to the performance of the trainees. Thus, we have a special case of digital storytelling: in this paper we regard the story as a plot of an emergency scenario, the storyteller as a trainer, and the listeners as trainees.

The paper is organized as follows. Our component-based concept for a digital storytelling software framework along with a proposal for its software architecture is described in Section 3. Section 4 discusses the according authoring process and how the storytelling looks from the users point of view. Here we will also present dedicated authoring tools. In the next sections we will deal with the implementation of our software framework and the lessons we learned when this framework was used in a real application scenario by application authors who want to tell an interactive story about a fire incident in a subway. Finally, we will summarize the paper, address unsolved problems and give an outlook on future work. But let us start in the next section with a review of the requirements we have for a digital storytelling software framework.

2. Requirements

What is the purpose of a digital storytelling system? Looking at the state-of-the-art in digital storytelling literature [1–6] one can find different answers to this question. Some see the support of human authors in writing stories (e.g. generation of plot ideas, planning of a book project, automated consistency checks in the story) as the main purpose while some systems want the computer itself as the storyteller. Others point out, that digital storytelling systems should provide a framework where users can interact with each other (like in multi-users dungeons) and experience an interactive story. Another possible answer is that digital storytelling systems transfer and integrate storytelling techniques in a software application, e.g. in an authoring system for developing software. In this paper, we want to take a view of a digital storytelling system that comprises these alternatives and is even broader.

To illustrate what we mean by this, let us consider an example. Imagine that the staff of a subway-company should be trained how to cope with different emergency scenarios (fire, terrorist attacks, etc.). Software can be written, where the trainees act together in a simulated virtual environment (like in a role-play). Non-linear storytelling techniques can be applied to this training environment when a security expert who trains the employees develops stories of possible emergency scenarios. As it is impractical that all the roles involved in an emergency (hundreds of passengers, policemen, fire-fighters) are taken over by trainees, some of the roles are filled by non-player characters (NPCs). The trainer would prepare a training session by developing a story outline for the scenario and the behavior description of the NPCs. During the training the trainer can intervene and steer the development of the story by issuing events or manipulating the NPCs online like a puppeteer.

What observations can we make in this example about digital storytelling software? We see that there are two main phases the system has to support: the preparation of the story and its characters (authoring phase) and the actual interactive storytelling (presentation phase). During the authoring phase the storyteller (in our example the trainer) needs help from different expert groups. First, she needs support from experts in information technology as the necessary software needs to be created. For example, the NPCs need to be programmed in some way. Second, she needs help from people with design/artistic or storytelling background. For example, she needs support for the timing of the story, for the modeling of the characters, or for the realization of geometric models and animations. Third, she needs help from experts of her own domain (in our example, people who are familiar with the procedures in a subway-emergency). These application experts can help her in describing the rules of the world that is modeled in the story or of typical characters, their motivations, and their usual behavior. It is unlikely that the storyteller possesses all the necessary skills and knowledge (story telling competencies, programming skills, domain knowledge, pedagogical knowledge), so

all the experts can be viewed as co-authors of the digital story. Note, that also all experts have complementary skills (for example, the expert in storytelling probably has no in-depth knowledge about emergency procedures in a subway). Fig. 1 illustrates these relationships.

As a consequence, there is not one single author but we need to decompose the storytelling task in several author roles. Each role should have a very limited set of prerequisites the author has to possess. An individual authoring tool that makes available the results of the authoring in a common framework should support each role. The roles should be organized as a hierarchy where each author role should be able to rely on the results that lower level authors provided, i.e. the roles should form an authoring pyramid (see Fig. 2). This hierarchy should force an authoring process with an increasing level of abstraction. Also means for communication between the different authoring roles should be provided. These are requirements the digital storytelling software framework needs to meet.

Note in the above example, that the outcome of the story is not only determined by the storyteller but also by the listeners. Also note, that in most digital storytelling systems the storytelling/artistic/design experts are identical with the application experts. This is only possible, if entertaining is the application domain and the experts have the control over the world model. This is especially true in a fantasy world where the listener can only check the world for consistency but not for realism (unlike in our emergency training example).

3. Component-based concept

In this section we will present our concept for a component-based framework for digital storytelling. Therefore we will explain what components and frameworks are and why they are appropriate for a digital storytelling system. We will show how behavior descriptions can be realized by the usage of agent methodologies. Finally, we will propose a system architecture.

3.1. Components and frameworks

In the last section we have motivated that we need to implement an authoring pyramid. Components show us a solution how this can be accomplished since generally we have a division of work associated with the usage of components: authors who *build* components and authors who *use* components. With a component, we mean a software component in the sense of the component theory [7]. A component is a building block of a software system with an explicitly defined interface designed for reusability. Reusing a component means to choose it from a given component library, adapt and customize it and insert it into a skeleton application. The skeleton application gives a pattern for arranging the components and consequently the author does not need to take care how they are expected to cooperate. In framework theory, this is called *inversion of control*, as the provided framework (and not the author) is in

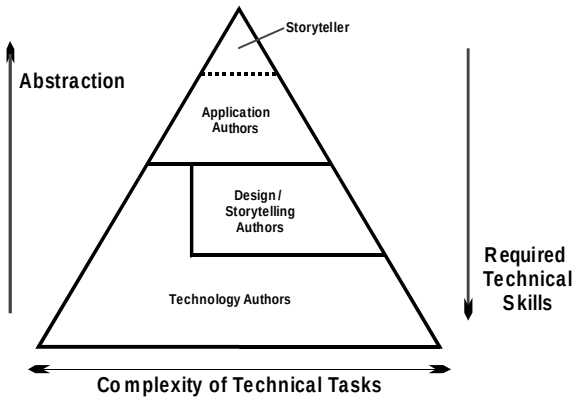

Fig. 2. Authoring pyramid for digital storytelling.

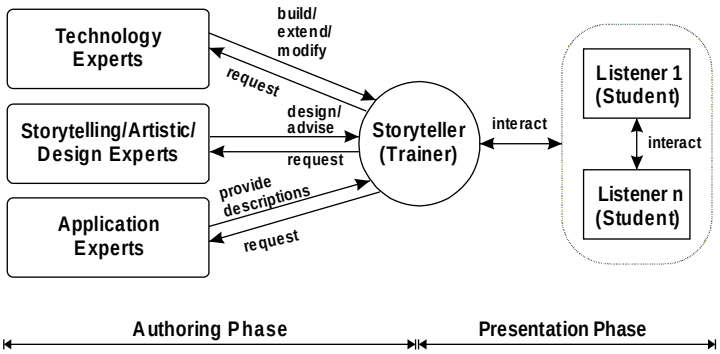

Fig. 1. Digital storytelling process.

charge of invoking and integrating the components' functionalities. Using object-oriented application frameworks [8] means that we have a generic skeleton application that can be filled with application-specific components by using dedicated authoring tools. As a result of this process it is possible to separate authors (esp. technical experts), who create the components, and authors (esp. application authors), who fill in the skeleton application. Since a skeleton application offers specific interfaces where a component can be plugged in automatically this insertion requires no programming skills and therefore it is possible to use even for non-technicians.

3.2. Component-based 3D computer graphic

For the concept of our system it is important to integrate 3D graphics and behavior descriptions for the story (so-called story behavior) in components. In the last years some papers addressed the topic how 3D computer graphics can be used in component-based systems. The motivation of the authors for their work was different and therefore their perspective was not uniform. Zeleznik et al. introduced the idea of an encapsulation of geometry and behavior in so-called 3D Widgets for the usage in 3D user interfaces [9]. As examples a virtual sphere, a cone tree and other interaction components were presented. [10] extend this concept of 3D Widgets with regard to the inner structure of the 3D components. The authors introduced two kinds of directed acyclic graphs named geometry graph and behavior graph. A high level interface, which hides details of the implementation, is used to fulfill operation on the given graphs. So-called ports are used to connect 3D Widgets. A visual language for 3D Widgets allows an interactive construction of 3D applications. Our work is oriented on the work presented in [11]. This concept encapsulates geometry in 3D components in a way, that runtime and authoring framework use the same 3D components. More about component-based 3D graphics can be found in [12,13].

3.3. Story behavior

How can story behavior be divided into components? A possible solution is the usage of agent technology, since agents are per definition autonomous entities and fit well with component-based approaches. The term "agent" emerged in the 1970s and was since used for a variety of different systems and concepts. There is no exact definition for an agent, however there is some common ground for all such system [14,15]. The main features are:

- Autonomy: The agent is autonomous, i.e. the agent is not a tool waiting for commands but it is responsible for its own task, its state and its actions. This is an ability that is innate in all real world objects. Even a simple door has "control" of its state, it may comply with your attempt to open it, or it may be locked or stuck etc.
- Reactivity: The agent monitors the world it is in and acts as a reaction to changes in the world. The door mentioned above is typically reactive, i.e. someone pulls the handle, and the door opens.
- Proactivity: Even if there are no changes in the outside world, the agent may decide to take action. Proactivity is usually associated with living real world beings but even a non-living object can "act" according to some inner state without an immediate outside stimulus.
- Reasoning: Before the agent can take an action it has to decide to do so. This decision is part of the reasoning process in an agent. The reasoning may depend on built in knowledge, recollection or even experience. Learning might be an important aspect in the reasoning part of an agent [16].
- Communication: If communication is broadly defined, every interaction with the world can be interpreted as communication. Verbal or textual communication is especially important, either between agents or between an agent and a human user of the system.
- Personality: Personality is the difference between an agent and a stereotypical, linear system. This feature is important when the agent shall represent a human being. Individual behavior can make an agent believable rather than stereotypical.

In our storytelling system, agents can be used to specify the behavior of the characters, which are believable, autonomous and who interact proactive with a user. Agents can be realized based on a rule engine. Each rule consists of a condition part and an action part that is executed when the condition is met. The action part can be realized by rulesets or by action libraries that provide pre-defined actions. A rule-based behavior description is achieved simply by forming the union of single rulesets. This allows us to extract parts of the behavior description and reuse it in other components. We distinguish agents that model the behavior of different entities in the story (like the characters) from the specific agent, which models the story outline itself, the so-called scenario agent.

A special benefit of using agents is that there exist models for planning behavior, goal-oriented behavior, as well as team behavior, available in the literature on artificial intelligence [17]. In our implementation of the basic system, for example, we provide features from the shared mental model theory [18] in order to define cooperation behavior. Note that for authoring purposes, it is possible to conceive an intelligent assistant using these kinds of behavior components.

3.4. System architecture

In this section, we will present the architecture of our component-based framework for digital storytelling (see Fig. 3). Backbone of our system is a communication platform, which is implemented as a bus and which can be distributed over the Internet. Storyteller and listener as well as all parts of the system are connected by the usage of a communication component. We use an event-based communication layer, which is used to distribute all events and which is used for synchronization. To allow the usage of agents from different authors (who may use different ontologies) we define for all entities a set of passive and active vocabulary. In this model, communication between two entities of a story requires that the intersection of the active vocabulary of the sender and the passive vocabulary of the receiver is not empty. Since the vocabularies are explicitly defined it is possible to define mappings between different vocabulary entries.

4. Storytelling and authoring process

The authoring process is depicted in Fig. 4. It starts with the creation of underlying frameworks that can support the next production step. In our case the story editor and customization editors are created. Besides, the skeleton of 3D components is made and major subcomponents (like the agent part or the meta-part of the 3D component) are integrated so that we have an "empty" 3D component available. With this it is possible to model geometry and author the behavior bases (e.g. providing atomic animations). It is also possible to implement action libraries. These bases can be developed in parallel for application behavior and low-level behavior as they have a well-defined interface. The next step in the production process is to provide 3D components that match entities in the real world. This is done by filling the empty 3D components with geometry and low-level behavior as well as providing rules for application level behavior. Besides, it is possible to modify existing 3D components. Having a set of 3D components available the authoring of a story can start. This is done in iterations by putting 3D components in the story editor, customize their appearance and behavior (e.g. by writing new rules or introduce existing rulesets out of a library) and defining links between agents (like property-to-property connections or sending messages). If this step is finished the story is compiled for the presentation phase.

The production process is characterized by concurrency and the division of the tasks of the author in several author roles. As the components used form a hierarchy the author roles can be organized in the form of an authoring pyramid as it was demanded. Examples

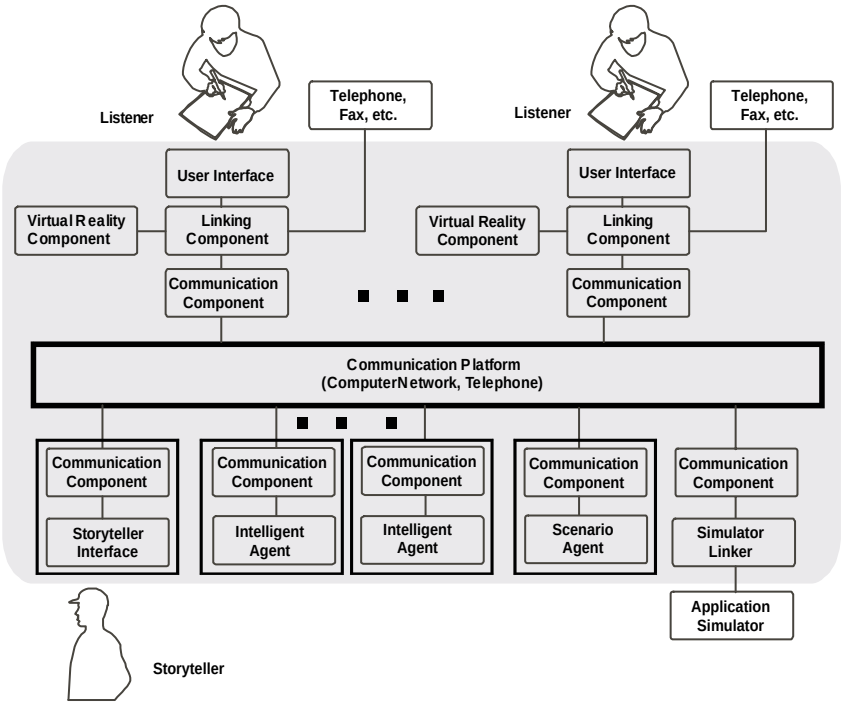

Fig. 3. Architecture of a component-based framework for digital storytelling.

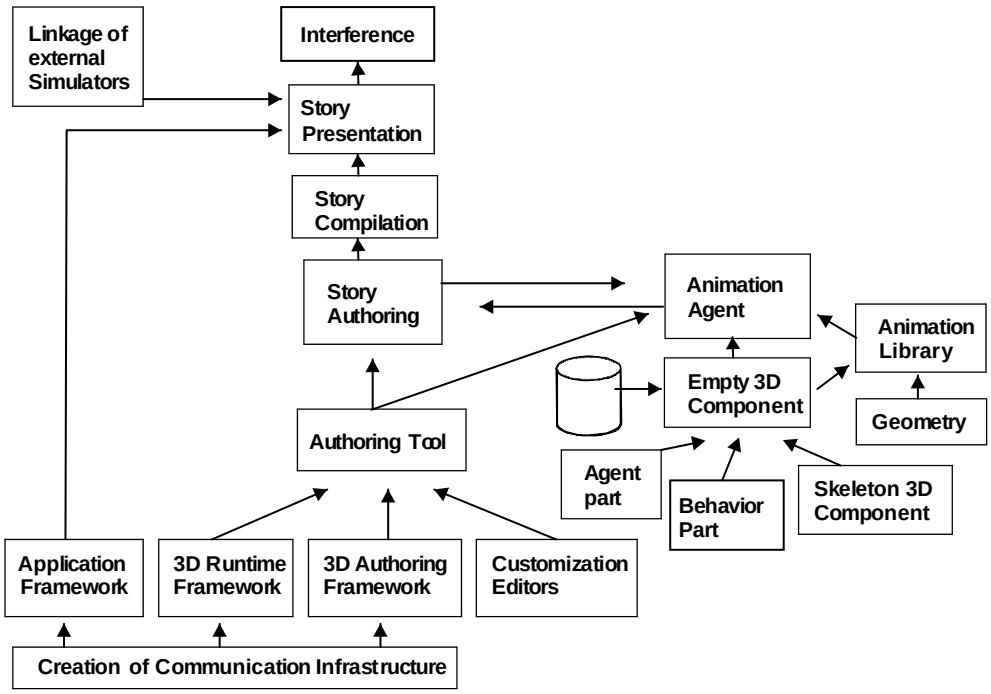

Fig. 4. Authoring process.

for authoring roles are authors for geometric modeling, authors for behavior components, authors for authoring frameworks, authors for combining 3D frameworks or authors for modifying existing applications.

For instance, let us take the point of view of an author who wants to describe the application behavior of a 3D component describing a train driver. Other authors with more knowledge in agent specifications, 3D animation and programming have written components that this author may use. On the one hand there is a library of actions, e.g. actions like "moving from one place to an other" or "telephoning". The author does not have to take care about implementation details, for example how the telephone is actually addressed or how a movement behavior is animated. The actions may have side effects like altering the status of goals pursued. Using these pre-defined actions an author writes rules that describe which action the train driver component takes when certain conditions are met. Beside the actions, the author is provided with rulesets and behavior patterns that he can add to the rules he writes. Or the author can modify existing rulesets by substituting actions. For example, a ruleset may describe the behavior of trying to reach someone via telephone. To support the author, tools are provided that allow him to browse the provided components, write the rules, organize behavior descriptions at different level of abstraction and view the results of his authoring efforts as the train driver component can be directly executed in the story-editing environment. The storyteller may choose the train driver component as a whole, integrate it in a story and modify it with a customize editor. Instead of examining the rules of the train driver component in order to find out about its behavior, the storyteller can be provided with small scenes that were written by the component author and characterize the component's behavior. This is an example, how communication between different authors (here: the storyteller and a component author, i.e. an application expert) can be designed at an adequate level.

Another example for the segmentation of a behavior-specification between two authors and its formalization is given in Fig. 5.

Author A describes a behavior at a high level, she just specifies that the agent should be inattentive under some circumstances. The tool which is used for this authoring could range from a simple text editor up to a graphical front-end (see Fig. 6). For the result of this authoring step, the if-then-clause in Fig. 5 is predictive enough. The concrete characterization of the behavior "be_inattentive()" could be provided by author B, who has to work out the formal specification on the abstract level of author A into a formal specification of the single tasks, which are necessary to implement the behavior. Author B has the possibility to choose between three alternatives to describe the behavior. Alternative 1 could

be, that the inattention is a state of the character, which influences her activities and their impact. The different state could result in a higher probability for the incidence of specific events (i.e. the state of being inattentive increases the probability, that a traindriver provokes an accident). Alternative 2 occurs, when the appearance of inattention results in an action, which itself could be composed by a set of sub-actions. In comparison to alternative 1, an action could include a change of a state but particularly it has the possibility to realize *actively* the execution of tasks. In alternative 3, author B could define a ruleset, which should be processed. Within the scope of a ruleset authors are able to describe actions which should be performed under specific conditions.

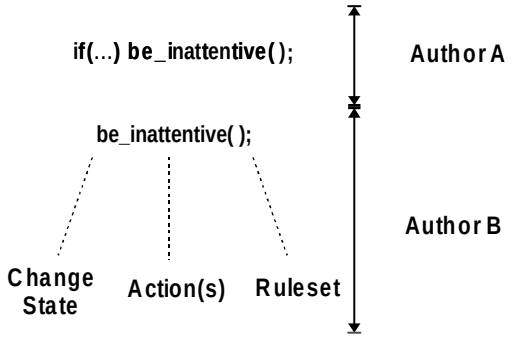

Fig. 5. Segmentation of a behavior between authors.

To support an author in the production process our digital storytelling system has to provide several editors:

- A *behavior editor* to specify the agent behavior. Our system supports the non-technical experts with a visual behavior editor where an author can specify behavior as well as reuse it (see Fig. 6).
- A *VR editor* for modeling the virtual reality as well as its geometries and animations. In most cases this editor is a professional modeling tool like Maya or 3D Studio Max.
- A *GUI editor* to define the user interface. Filling the skeleton application with components is done with this editor. Therefore the editor provides support for selection, customization and insertion of components.

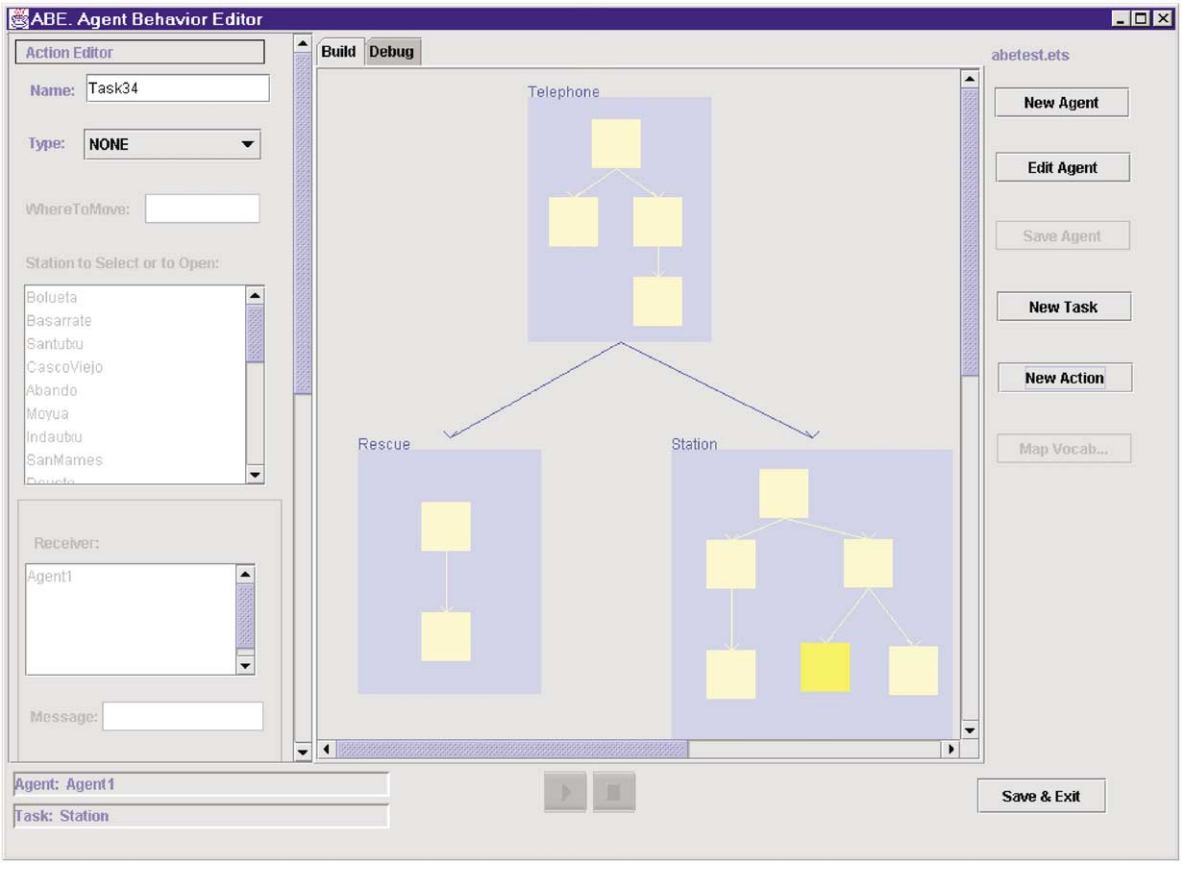

Fig. 6. Behavior editor to specify the agent behavior.

- A *story editor* to arrange specified components and agents to a story. The author has to describe, which roles exist and how they are filled. This tool allows the specification of a storybook as well as alternatives to it (see Fig. 7).
- A *component browser*, which allows managing and administrating component.
- An *interference editor*, that enables the storyteller to interfere with the story telling process.

All these editors have to be adapted for the individual authoring role. Thus, tools for authors with artistic background use metaphors from their domain, for example ''stage'', ''storyboard'', ''directing''. In contrast to this, tools for the application experts use metaphors from the application domain, e.g. in our application example metaphors like ''emergency plan'', ''schedule'', ''signal'' would be appropriate.

5. Implementation

Our concept was implemented in the context of the ETOILE (Environment for Team, Organizational and Individual Learning in Emergencies) project (cf. Fig. 8), partly funded by the European Commission. The realization of the framework and the behavior editors as well as the implementation of the other tools, which were described according to the concepts above, were done with the object-oriented programming language Java and its component-model named JavaBeans [19]. To use Beans they have to be connected or used in conjunction with a framework. The framework makes the beans executable by accessing the meta-informations of the beans. This brings an explorative testing and evaluation of beans forward. The component-model of Java was enhanced by the use of so-called 3D-Beans [11]. These are regular Java components, which have in addition to their behavior a 3D-geometry, on the basis of the Java3D-API.

For the distribution of the application to a number of computers, the Remote Method Invocation (RMI) was used. RMI makes it possible, to distribute messages synchronized between different computers. To realize the communications between different components, we used the communication-bus Infobus [20]. The Java based agent system and rule engine we used, is Blaze Advisor [21] from HNC Software. In our developed system ETOILE, we were able to build scenes, which contains up to 25 different agents. The bigger part of the agents was developed by end-users (employees of a Spanish power company, respectively, a Spanish subway-company), whereby the agents had between 150 and 300 rules each. Further on our scenarios had around 35,000–50,000 polygons. If we combine these values about performance, we have to put out, that all our scenarios were capable to be executed on a standard PC (i.e. requirements like Intel Pentium IV, 256 MB memory, nVidia GeForce III graphics card or an equivalent computer).

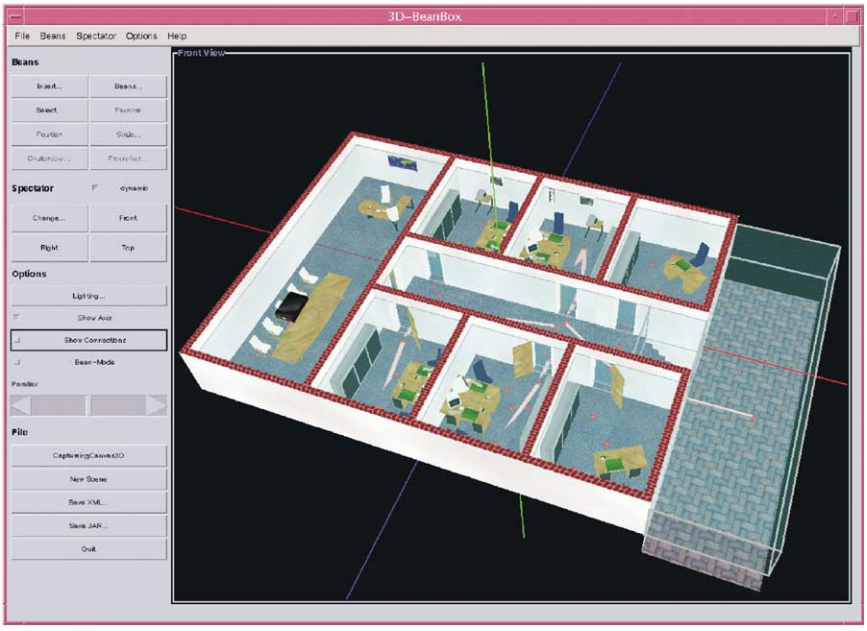

Fig. 7. Insertion of a 3D-Beans within the story editor.

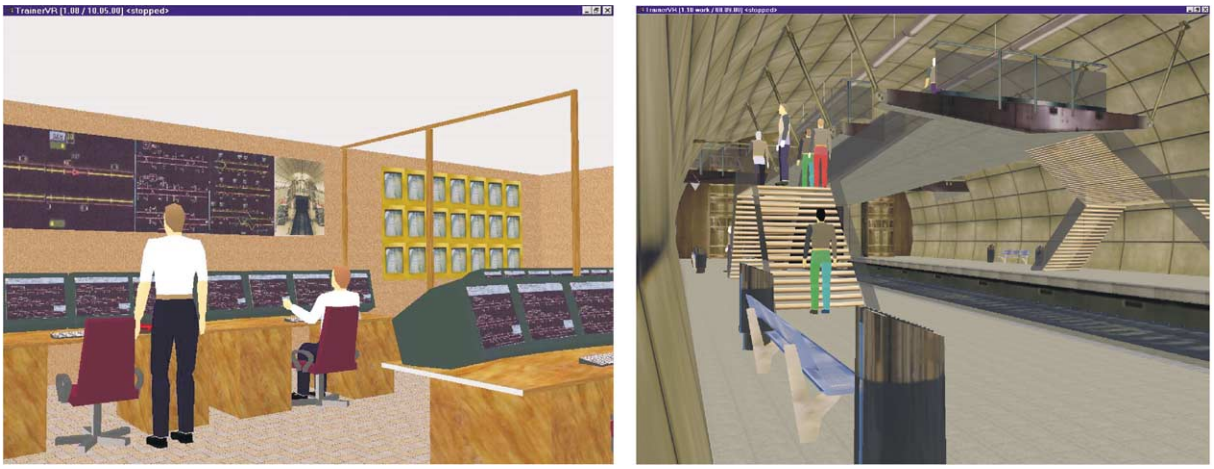

Fig. 8. Snapshots of an ETOILE scenario.

6. Lessons learned

In the ETOILE project were different person groups of end-users, who utilize the application, which was developed according to the above-mentioned concepts, to build training scenarios.

First of all, there were the authors with a consolidated knowledge in information technology. However is was not their task to develop the tools, but to build application-specific components, like the geometrical structure of a subway, behavior elements of a subway-engine driver or elements of scenario-sequence. The challenge for this group of authors was, that they had an empty skeleton-application with no existing components in it. Even they understood the underlying concepts quickly. However, users experienced difficulties that we constitute in the mightiness of the system. Within the scope of creating simple behavior elements for the agents were difficulties because of the lack of support tools. We found out, that tools for consistency checks, structuring and validating would be helpful.

The components that were produced by the first group, were used by another group to describe a concrete story (scenario). Contrary to the experiences of the first group, the second group lamented the absence of flexibility of the components and the restricted possibility to change a training session during the storytelling (training). Again, some components had a too extensive and deterministic functionality, which were better divided in separate elements. In this context it is obvious, that it is difficult to find an optimal granularity for the components. Our component-based approach makes high demands on the authors of the components.

To master the situation between the two extremes of the functionality of the resulting components, it is useful to establish an iterative process during the development of components. Through the advantage of the exchangeability of existing components over different application fields this loss wanes increasingly with the number of developed components. In our ETOILE project, authors of a nuclear power plant were able to use components from the subway-company (i.e. components that specify a fire in a subway-tunnel could also describe a overheating in the power plant). The exchange of components was straightforward, because of the well-defined interfaces between components and framework.

Another experience was, that the possibility of reviewing the story (scenario) was frequently used by the authors. This is constituted in the matter of fact, that an author just describes the initial state of a story, instead of predefining the whole sequence. A forecast, how a presentation looks, is in the ETOILE system only possible by simulating the sequence. Thereby we note a change of how the authors use the system. While in the beginning the scenarios were created with time-based constraints (i.e. 15 s after action 1 action 2 happens) or orientation on milestones, authors with increased experience changed to an event-based description of actions (i.e. action 2 happens after action 1 under condition x). In our opinion this behavior correlates in the scale authors are using the simulation of scenarios. By iterative simulation scenarios the authors detect incomplete, inconsistent or wrong behavior of their agents [4].

In ETOILE there is the special application that the listeners of a story (the learners) are involved in their flow. Since in an ETOILE scenario roles are only defined, without determining in the run-up if the role is taken over from an agent or a human, it is possible for a storyteller (trainer) to simulate a

whole training as all roles are engaged by agents. In our opinion it is meaningful to support the simulation further, e.g. by a check of the time parameter (i.e. fast motion tools) and by the introduction of several levels of detail, which enables a sketching, which can be refined later gradually. A further support by tools could consist by the observation of a certain behavior during the simulation and the direct referring to the appropriate behavior in order to enable an efficient manipulation of agents. Depending upon the function the author needs suitable visualizations and interaction possibilities, in order to be able to produce a scenario. Therefore the author's view of the scenario should be situation-dependent. Thus for the setting up of the virtual environment other metaphors are needed than during the structuring of behaviors of objects or setting up the relations between the objects. For example the metaphor of the flow diagrams can be used for emergency situations. Besides a visualization of further characteristics, like relations structures, motivations of the characters, emotions of the characters or the dramaturgy-oriented view can be helpful for the author.

As an important characteristic of intelligent agents in digital storytelling systems the believability of the characters was called. Believability is thereby a necessary requisition for the dramaturgy of a story, without influencing it however. By adding dramaturgy-oriented special rules to an agent at first sight the reliability may reduce, but the possibilities of the author to influence the dramaturgy however improved.

The direct access of the trainer felt at run-time as too restrictively, because every direct behavior pattern of the agents had to be switched individually. Remedy creates here a bundling and a grouping of the instruction into an abstract hierarchy. This should be clarified by the example of a subway-engine driver. A subway-engine driver could have different malpractices, which can be released by the trainer, as for instance "disregard the next signal!" or "ignore further messages!". These concrete malpractices can be summarized under a abstract instruction like "be inattentive!". In this way the trainer can release easily an abundance of single actions, without having to call everyone explicitly. Although it was shown that agents could be exchanged and reused quite simply between different scenarios, the necessity is clear to manage agents, scenarios and characters effectively. For this purpose agents etc. can be equipped with meta-informations. This makes it possible in an effective way, to give a rough course of action for scenarios, and special abilities, weaknesses etc. for agents or roles. With a database, which operates on these meta-informations, a large number of combinations and designs can be administered.

7. Summary and future work

We described a concept for a component-based framework that is used for an interactive training simulation, but may also be used for digital storytelling in general. We pointed out that the claim for a component-based framework is not only advantageous for the internal structure, but rather it brings benefit for the application field of digital storytelling.

Through the spreading of incidental task while the authoring process, we provide a classification corresponding to the abilities and background of story authors, domain-experts as well as technology experts. Because of the importance of the communication between experts of the same, respectively, of different authoring groups, we described what kind of communication is necessary and which interfaces are essential. In progression of the ETOILE project we had the opportunity to evaluate our concept.

While the acceptance of our concept was satisfying on the whole, there are still issues to be improved and some additional tools could be conceived that would foster the authoring process (e.g. an assistant tool). Right now the user-interfaces of our system could distinguish more far reaching between the membership of a user to the different groups of experts or authors that we mentioned. Through an adapted user-interface, which matches the terminology of the specific user more sophisticated, the habituation to the system could be reduced. By offering a more intuitive system, the dissemination of such digital storytelling frameworks would be higher. While in our concept the focal point is the authoring of the behavior of single characters, which are part of the story, we neglected so far to give an automated approach, how to determine a specific dramaturgy for the story. Within the scope of our concept, the final sequence and therefore also the dramaturgy of the story was dependent of some random elements. For future work, a tool to guarantee a specific progression (or a progression within specified ranges) for dramaturgy would be useful.

References

[1] Davenport G. Seeking dynamic: adaptive story environments visions and views. IEEE Multimedia 1994;1(3):9–13.

[2] Mateas M, Stern A. Towards integrating plot and character for interactive drama. http://home.netcom.com/~apstern/interactivestory.net/papers/MateasSternAAAIFS00.pdf.

[3] Murray JH. Hamlet on the holodeck: the future of narrative in cyberspace. Cambridge, MA: MIT Press, 1997.

[4] Steiner KE, Moher TG. Graphic StoryWriter: an interactive environment for emergent storytelling. In: Con-

ference Proceedings on Human Factors in Computing Systems, Monterey, CA, 1992. p. 357–364.
[5] Trappl R, Petta P. Creating personalities for synthetic actors: towards autonomous personality agents. Berlin: Springer, 1997.
[6] Umaschi M, Cassell J. Storytelling systems: constructing the innerface of the interface. In: Proceedings of the Second International Conference on Cognitive Technology. Los Alamitos, CA: IEEE Computer Society, 1997. p. 98–108.
[7] Sametinger J. Software engineering with reusable components. Berlin: Springer, 1997.
[8] Fayad ME, Schmidt D, Johnson R. Building application frameworks: object-oriented foundations of framework design. New York: Wiley, 1999.
[9] Conner DB, Snibbe SS, Herndon KP, Robbins DC, Zeleznik RC, van Dam A. Three-dimensional widgets. In: Computer Graphics Symposium for Interactive 3D Graphics, vol. 25(2). New York: ACM Press, 1992. p. 183–188.
[10] Döllner J, Hinrichs K. Interactive, animated 3D widgets. In: Proceedings of the 1998 IEEE International Conference of Computer Graphics International (CGI), Hanover, Germany, 1998. p. 278–286.
[11] Dörner R, Grimm P. Three-dimensional beans—creating web content using 3D components in an 3D authoring environment. In: Proceedings of the Web3D-VRML 2000, Fifth Symposium on Virtual Reality Modeling Language, Monterey, CA, 2000. p. 69–74.
[12] Schönhage B, van Ballegooij A, Eliens A. 3D gadgets for business process visualization. In: Proceedings of Web3D—VRML 2000. Monterey, USA, 2000. p. 131–138.
[13] Miller T, Zeleznik R. The design of 3D Haptic widgets. In: 1999 Symposium on Interactive 3D Graphics, Atlanta, GAUSA. New York: ACM Press.
[14] Wooldridge M, Jennings NR, editors. Agent theories, architectures, languages: a survey. In: Intelligent agents, lecture notes in artificial intelligence research, vol. 7. Los Altos, CA: Morgan Kaufmann Publishers, 1997. p. 83–124.
[15] Wooldridge M, Jennings NR. Pitfalls of agent-oriented development. In: Proceedings of the Second International Conference on Autonomous Agents, Minneapolis, MN, 1998. p. 385–391.
[16] Fung J, Tu X, Terzopoulos D. Cognitive modelling: knowledge, reasoning and planning for intelligent characters. In: Proceedings of SIGGRAPH, Los Angeles, CA, 1999.
[17] Tambe M. Towards flexible teamwork. Journal of Artificial Intelligence Research 1998;7:83–124.
[18] Leitch RR, Sime JA. A specification methodology for intelligent training systems. In: Proceedings of Sixth International PEG Conference on Knowledge Based Environments for Teaching and Learning (PEG'91). Rapallo, 1991. p. 331–342.
[19] Java Homepage of Sun Microsystems Inc. http://java.sun.com.
[20] Infobus Homepage of Sun Microsystems Inc. http://java.sun.com/products/javabeans/infobus/
[21] Blaze Advisor Homepage, HNC Inc. http://www.blazesoft.com