SceneMaker: Visual Authoring of Dialogue Processes

Gregor Mehlmann
Human Centered Multimedia
Augsburg University
Universitätsstrasse 6a
D-86159 Augsburg

Patrick Gebhard
DFKI GmbH
Campus D3 2
Stuhlsatzenhausweg 3
D-66123 Saarbrücken

Birgit Endrass
Human Centered Multimedia
Augsburg University
Universitätsstrasse 6a
D-86159 Augsburg

Elisabeth André
Human Centered Multimedia
Augsburg University
Universitätsstrasse 6a
D-86159 Augsburg

Abstract

In this paper, we present a visual authoring approach for the management of highly interactive, mixed-initiative, multi-party dialogues. Our approach enforces the separation of dialog-content and -logic and is based on a statechart language enfolding concepts for hierarchy, concurrency, variable scoping and runtime history. These concepts facilitate the modeling of dialogs for multiple virtual characters, autonomous and parallel behaviors, flexible interruption policies, context-sensitive interpretation of the user's discourse acts and coherent resumptions of dialogues. It allows the real-time visualization and modification of the model to allow rapid prototyping and easy debugging. Our approach has successfully been used in applications and research projects as well as evaluated in field tests with non-expert authors.

1 Introduction

Virtual characters in interactive applications can enrich the user's experience by showing engaging and consistent behavior. To what extent virtual characters contribute to measurable benefits is still fiercely discussed (Heidig and Clarebout 2010; Miksatko, Kipp, and Kipp 2010). Therefore, virtual characters need to be carefully crafted in cooperation with users, artists and programmers. The creation of interactive virtual characters with a consistent and believable dialogue behavior poses challenges such as modeling personality and emotion (Marsella and Gratch 2006), creating believable facial expressions, gestures and body movements (Kipp et al. 2007), expressive speech synthesis (Schröder 2008) and natural language recognition as well as dialog and interaction management (Traum et al. 2008). In this work, we address the tasks of modeling consistent highly interactive mixed-initiative multi-party dialogue behavior and realizing effective interaction management for dialogue situations with embodied conversational characters.

During the last years, several approaches for modeling interactive dialogue behavior of virtual characters have been researched. A variety of systems such as, frame-, plan-, rule- and finite state-based systems were presented. Most of these systems required a substantial degree of expert knowledge and programming skills, thus, being unserviceable for non-computer experts, such as artists and screenwriters that wanted to craft interactive applications with virtual characters. Therefore, as a next step, authoring systems were developed to exploit related expert knowledge in the areas of games, film or theater screenplay.

These systems are created to facilitate the authoring process and to allow non-computer experts to model believable natural behavior for virtual characters. They can be categorized by their conceptual and methodological approaches. On the one hand, character-centric approaches aim on creating autonomous agents for multi-agent systems, while they do not explicitly include support for scripting the behavior of multiple agents in a simple and intuitive way. Examples for character-centric systems are *Improv* (Perlin and Goldberg 1996) or *Scream* (Prendinger, Saeyor, and Ishizuka 2004), where an author defines the agents' initial goals, beliefs and attitudes. These mental states determine the agents' behavioral responses to received communicative acts. In author-centric approaches, on the other hand, a human author can communicate an artistic vision with the primary focus of scripting at the plot level. The user can contribute to the plot within the narrative boundaries defined by the author. Examples for author-centric systems include *Scenejo* (Spierling, Weiss, and Mueller 2006), *Deal* (Brusk et al. 2007) and *Creactor* (Iurgel et al. 2009). Hybrid approaches, as described in (McTear 1998; Gandhe et al. 2008) or (Gebhard et al. 2003), try to bridge the gap between the author-centric and character-centric approach by combining the advantages of both.

So far, none of the mentioned authoring systems supports concepts for dialogue and interaction history and concurrent process modeling for parallel behavior on the authoring level. However, this would facilitate the modeling task and reduce the complexity of the model. It would help to handle typical challenges in the creation of applications with interactive virtual characters, such as the modeling of reactive and deliberate behavior, the use of multiple virtual characters and their synchronization and the handling of user interaction. In this paper, we face these challenges using our new version of the authoring tool *Scenemaker* which pursues a hybrid approach to contribute on the user modeling level for the creation of interactive virtual character applications in a

Proceedings of the 7th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, pages 24–36,
Barcelona, July 2011

rapid-prototyping style.

2 Dialogue and Interaction Management

The central concept of our authoring approach with the Scenemaker authoring tool is the seperation of dialogue content and structure. Multimodal dialogue content is specified in a set of *scenes* that are organized in a *scenescript*. The narrative structure of an interactive performance and the interactive behavior of the virtual characters is controlled by a *sceneflow* - a statechart variant specifying the logic according to which scenes are played and commands are executed. Sceneflows have concepts for *hierarchical refinement* and the *parallel decomposition* as well as an exhaustive *runtime history* and multiple *interaction policies*. Thus, sceneflows adopt and extend concepts that can be found in similar statechart variants (Harel 1987; von der Beeck 1994).

Sceneflows and scenescripts are created using a graphical authoring tool and executed by an interpreter software. This allows the real-time extension and modification of the model and the direct observation of the effects without the need for an intermediate translation step. The real-time visualization of a sceneflow's execution and active scenes within the graphical user interface allows to test, simulate and debug the model.

2.1 Creating Multimodal Dialogue Content

A scene resembles the part of a movie script consisting of the virtual characters' utterances containing stage directions for controlling gestures, postures, gaze and facial expressions as well as control commands for arbitrary actions realizable by the respective character animation engine or by other external modules. Scenescript content can be created both manually by an author and automatically by external generation modules. The possibility to parameterize scenes may be exploited to create scenes in a hybrid way between fixed authored scene content and variable content (Figure 1 ①), such as retrieved information from user interactions, sensor input or generated content from knowledge bases. In Section 3.5 we present an application which makes extensive use of parameterized scenes and generated scene content from a domain knowledge module.

Figure 1: Parameterizable scenes of a scenegroup.

A scenescript may provide a number of variations for each scene that are subsumed in a *scenegroup*, consisting of the scenes sharing the same name or signature (Figure 1 ②,③). Different *blacklisting strategies* are used to choose one of the scenes from a scenegroup for execution. This mechanism increases dialogue variety and helps to avoid repetitive behavior of virtual characters, which would certainly impact the agents' believability.

2.2 Modeling Dialogue Logic and Context

A sceneflow is a hierarchical and concurrent statechart that consists of different types of *nodes* and *edges*. A *scenenode* can be linked to one or more scenegroup playback- or system commands and can be annotated with statements and expressions from a simple scripting language or function calls to predefined functions of the underlying implementation language (Figure 2 ①). A *supernode* extends the functionality of scenenodes by creating a hierarchical structure. A supernode may contain scenenodes and supernodes that constitute its subautomata. One of these subnodes has to be declared the *startnode* of that supernode (Figure 2 ②). The supernode hierarchy can be used for type and variable scoping. Type definitions and variable definitions are inherited to all subnodes of a supernode. The supernode hierarchy and the variable scoping mechanism imply a hierarchy of *local contexts* that can be used for context-sensitive reaction to user interactions.

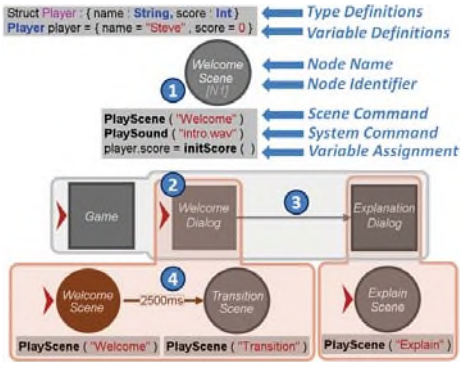

Figure 2: Node statements and supernode hierarchy.

Different *branching strategies* within the sceneflow, e.g. logical and temporal conditions or randomization, as well as different *interaction policies*, can be modeled by connecting nodes with different types of edges. An *epsilon edge* represents an unconditional transition (Figure 2 ③). They are used for the specification of the order in which computation steps are performed and scenes are played back. A *timeout edge* represents a timed or scheduled transition and is labeled with a *timeout value* (Figure 2 ④). Timeout edges are used to regulate the temporal flow of a sceneflow's execution and to schedule the playback of scenes and computation steps. A *probabilistic edge* represents a transition that is taken with a certain probability and is labeled with a *probability value* (Figure 5 ②). Probabilistic edges are used to create some degree of randomness and desired nondeterminism during the execution of a sceneflow. A *conditional edge* represents a conditional transition and is labeled with a *conditional expression*, as shown in Figure 3. Conditional edges are used to create a branching structure in

the sceneflow which describes different reactions to changes of environmental conditions, external events or user interactions. In Section 3.4 we present an application which makes extensive use of a hierarchy nested supernodes to refine the dialogue context for an adequate reaction to the user's interactions.

2.3 Continuous Real-Time Interaction Handling

User interactions as well as other internally or externally triggered events within the application environment can rise at any time during the execution of a model. Some of these events need to be processed as fast as possible to assert certain real-time requirements. There may, for example, be the need to contemporarily interrupt a currently running dialogue during a scene playback in order to give the user the impression of presence or impact. However, there can also exist events that may be processed at some later point in time allowing currently executed scenes or commands to be regularly terminated before reacting to the event. These two different interaction paradigms imply two different *interaction handling policies* that find their syntactical realization in two different types of interruptibility and inheritance of conditional edges:

- *Interruptive conditional edges* (Figure 3 ①,③) are inherited with an interruptive policy and are used for the handling of events and user interactions requiring a fast reaction. Whenever an interruptive conditional edge of a node can be taken, this node and all descendant nodes may not take any other edges or execute any further command. These semantics imply, that interruptive edges that are closer to the root have priority over interruptive edges farther from the root.
- *Non-interruptive conditional edges* (Figure 3 ②,④) are inherited with a non-interruptive policy, which means that a non-interruptive conditional edge of a certain node or supernode can be taken after the execution of the node's program and after all descendant nodes have terminated. This policy is implicitly giving higher priority to any conditional edge of nodes that are farther from the root.

Figure 3 shows a supernode hierarchy with different conditional edges. If the condition "stop" becomes true during the execution of the two innermost scene playback commands, then the scene within the supernodes with the non-interruptive conditions (Figure 3 ②,④) will be executed to its end. However, the scene within the supernodes with the interruptive conditions (Figure 3 ①,③) will be interrupted as fast as possible. In the non-interruptive case the execution of the sceneflow continues with the inner end node (Figure 3 ④) before the outer end node is executed (Figure 3 ②). In the interruptive case the execution of the sceneflow immediately continues with the outer end node (Figure 3 ①) because the outer interruptive edge has priority over the inner interruptive edge (Figure 3 ③).

2.4 Modeling Parallel Dialogue and Behavior

Sceneflows exploit the modeling principles of *modularity* and *compositionality* in the sense of a hierarchical and *parallel decomposition*. Multiple virtual characters and their behavior, as well as multiple control processes for event detection or interaction management, can be modeled as concurrent processes in parallel automata. For this purpose, sceneflows allow two syntactical instruments for the creation of concurrent processes: (1) By defining multiple startnodes for a supernode, as shown in Figure 4, each subautomaton which consists of all nodes reachable by a startnode, is executed by a separate process, (2) by defining *fork edges* (Figure 5 ①) an author can create multiple concurrent processes without the need for changing the level of the node hierarchy.

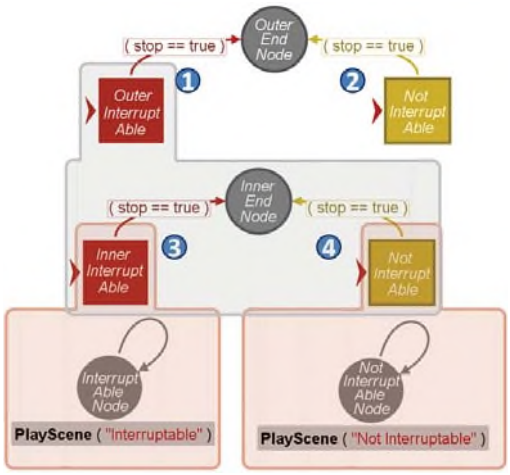

Figure 3: ①,② Interruptive conditional edges. ③,④ Simple non-interruptive conditional edges.

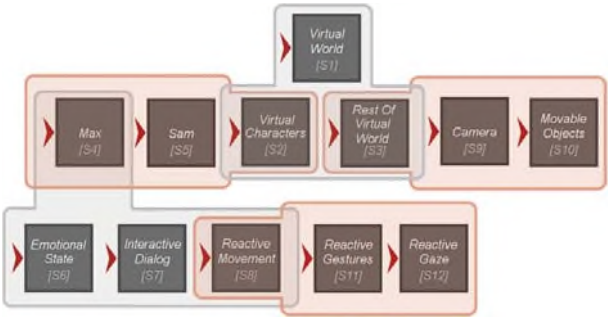

Figure 4: Hierarchical and parallel decomposition.

Following this modular approach, an author is able to separate the task of modeling the overall behavior of a virtual character into multiple tasks of modeling individual behavioral aspects, functions and modalities. Behavioral aspects can be modified in isolation without knowing details of the other aspects. In addition, previously modeled behavioral patterns can easily be reused and adopted. Furthermore, premodeled automata that are controlling the communication with external devices or interfaces can be added as plugin modules that are executed in a parallel process.

Individual behavioral functions and modalities that contribute to the behavior of a virtual character are usually not

Figure 5: ① Concurrent processes with fork edges. ② Randomization with multiple probability edges.

completely independent, but have to be synchronized with each other. For example, speech is usually highly synchronized with non-verbal behavioral modalities such as gestures and body postures. When modeling individual behavioral functions and modalities in seperate parallel automata, the processes that concurrently execute these automata have to be synchronized by the author in order to coordinate all behavioral aspects. This communication is realized by a *shared memory model* which allows an asynchronous non-blocking synchronization of concurrent processes.

Figure 6: Synchronization over configuration states.

Thereby, sceneflows enfold two different syntactic features for the synchronization of concurrent processes. First, they allow the synchronization over common *shared variables* defined in some supernode. The interleaving semantics of sceneflows prescribe a mutually exclusive access to those variables to avoid inconsistencies. Second, they enfold a *state query condition*, as shown in Figure 6, which represents a more intuitive mechanism for process synchronization. This condition allows to request weather a certain state is currently executed by the sceneflow interpreter during the execution of a sceneflow.

2.5 Consistent Resumption of Dialogue

Our concept of an exhaustive *runtime history* facilitates modeling reopening strategies and recapitulation phases of dialogues by falling back on automatically gathered information on past states of an interaction. During the execution of a sceneflow, the system automatically maintains a *history memory* to record the runtimes of nodes, the values of local variables, executed system commands and scenes that were played back. It additionally records the last executed substates of a supernode at the time of its termination or interruption. The automatical maintainance of this history memory releases the author of the manual collection of such runtime data, thus efficiently reducing the modeling effort while increasing the clarity of the model and providing the author with rich information about previous interactions and states of execution.

The scripting language of sceneflows provides a variety of built-in *history expressions* and conditions to request the information deposited in the history memory or to delete it. The history concept is syntactically represented in form of a special *history node* which is an implicit child node of each supernode. When reexecuting a supernode, the supernode starts at the history node instead of its default startnodes. Thus, the history node serves as a starting point for the author to model reopening strategies or recapitulation phases.

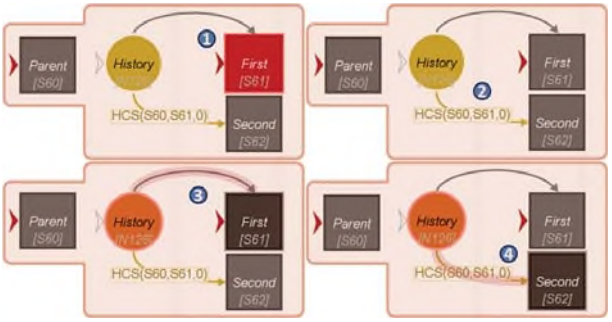

Figure 7: History node and condition.

Figure 7 shows a simple exemplary use of a supernode's history node and a history condition. At the first execution of the supernode "Parent", the supernode starts at its startnode "First" (Figure 7 ①). If the supernode "Parent" is interrupted or terminated at some time and reexecuted afterwards, it starts at the history node "History". The history memory is requested (Figure 7 ②) to find out if the supernode "Parent" had been interrupted or terminated in the node "First" or the node "Second". As the snaphot of the visualized execution shows, depending on the result, the either the node "First" (Figure 7 ③) is executed or the node "Second" (Figure 7 ④) is started over the history node.

3 Applications

The new Scenemaker authoring tool supports the creation of applications with interactive virtual characters on various levels such as the modeling of reactive and deliberate behavior, the use of multiple virtual characters and their synchronization, and the advanced handling of user interactions. In the following, we describe several applications and research projects in which the new Scenemaker tool was used. We present specific aspects of the models created in these applications in order to illustrate the use of certain modeling features of Scenemaker introduced in the previous section.

3.1 IGaze - Modeling Reactive Gaze Behavior

Gaze as an interaction modality has many functions, such like signaling attention, regulating turn-taking or deictic reference (Kipp and Gebhard 2008). An absence of gaze in a virtual character's behavior would be recognized directly by a human interlocutor. Therefore gaze is highly relevant for such characters, especially in human-computer interaction (e.g. COGAIN[1]).

[1] http://www.cogain.org

In the IGaze project, we have modeled a virtual character's gaze behavior as *dominant* or *submissive*. A fine-grained control of gaze can help to improve the overall believability of a virtual character. In one of the first projects that uses the new SceneMaker the gaze behavior for the two characters Sam and Max is represented by a concurrent Sceneflow (see Fig. 8).

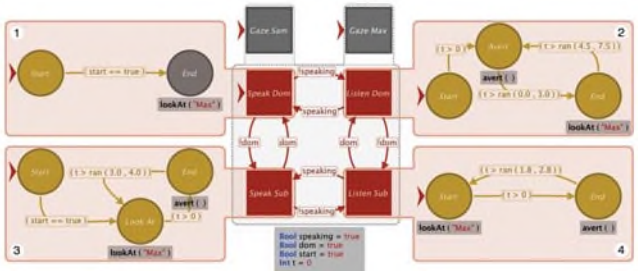

Figure 8: Hierarchical concurrent Supernodes model gaze behavior (*Gaze Sam* and *Gaze Max*)

A separate gaze Supernode for each character holds commands to control its gaze (head) behavior on an abstract level (e.g. **lookat(character)**, **avert**). Those commands define the interface to a characters movement control. In general, the Sceneflow model represents the following gaze behavior:

- Dominant (Dom): High status, according to Johnstone, is gained by outstaring the interlocutor and if a person breaks eye contact and does not look back (Johnstone 1979). The dominant gaze behavior consists of maintaining eye contact while speaking and randomly changing from gazing to averting while listening. More precisely, the character establishes and holds eye contact when speaking (see Fig. 8, ①), and after speaking, immediately looks away. When listening, the character establishes eye contact after 0-3 sec., then holds it for 4.5-7.5 (see Fig. 8, ②).
- Submissive (Sub): Low status, according to Johnstone, means being outstared by the interlocutor or by breaking eye contact and looking back. The submissive gaze behavior makes a character only look briefly every now and then and immediately avert the gaze again. In the submissive gaze mode, a character establishes eye contact when starting to talk but averts his gaze immediately after eye contact. His gaze remains averted for 3-4 sec (see Fig. 8, ③). He then establishes eye contact again and looks away immediately. During listening, the pattern is the same with the difference that the character holds eye contact for 1.8-2.8 sec (see Fig. 8, ④). The submissive avert behavior consists of a head movement away from the user (5° while speaking, 8° while listening) and 15° downward.

A major improvement provided by the Visual SceneMaker is the possibility to verify and alter the timing specifications directly during run-time. This enabled us to carefully adjust the time of a specific gaze aspect in order to achieve an overall compelling result.

On a conceptual level, reactive behavior patterns can be realized as global concurrent Sceneflows or as local concurrent Supernodes that are executed by fork edges at a specific location of a master Sceneflow. Such Supernodes can easily be reused.

3.2 AI Poker - Playing Poker with two Virtual Characters

In the AI Poker, we investigate how modern ECA technologies can help to improve the process of creating computer games with interactive expressive virtual characters. Based on the experience of a computer game company, we identified four main challenges in creating computer games:

- *Fast creation of game demonstrators*. In order to compete with other game companies the implementation of demonstrators has to be fast and reliable.
- *Localization of game content*. To sell games in other countries content has to be translated into the respective language. The more dialogs a game contains, the higher the costs for the translation.
- *Intuitive interaction*. The success of a game is tremendously related to an easy interaction concept.
- *Consistent quality*. The quality of audio and visual presentation should be consistent for the whole game. Every exception lowers its acceptance.

The AI Poker application reuses the gaze control supernodes from the IGaze project. These are extended by two concurrent supernodes, one for automatic camera pan and another for the game interaction control that also controls the two 3d Virtual Characters Sam and Max (see Fig 11), which are in the role of two poker teammates. Sam is a cartoon-like character, whereas Max is a mean, terminator-like robot character. A human user acts as the card dealer and also participates as a regular player.

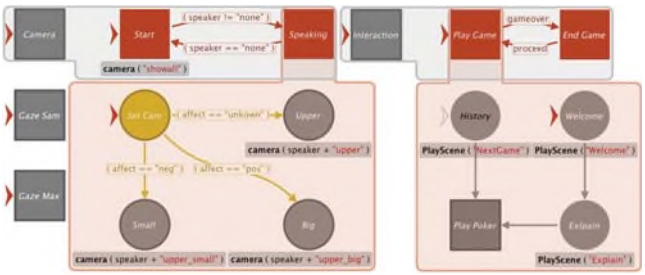

Figure 9: AI Poker's Game Model using hierarchical concurrent supernodes

By using real poker cards with unique RFID tags, a user can play draw poker against Sam and Max. The characters rely on the MARY expressive speech synthesizer using HMM-based and unit-selection-based speech synthesis approaches and the ALMA model for the simulation of affect (Gebhard et al. 2008). It simulates three affect types (emotions, moods, and personality) as they occur in human beings. Based on game events, the affect of each character

is computed in real-time and expressed through speech and body. Both characters are rendered by a 3d visualization engine based on Horde3D (Augsburg University). In order to support Sam's and Max's individual character style, different poker algorithms (realized as separate software modules) are used. Sam relies on a rule based algorithm, whereas Max relies on a brute-force algorithm that estimates a value for each of the 2.58 million possible combinations of five poker cards. The Visual SceneMaker as a central component allows to control of all these techniques and enables the characters to show a consistent emotional expressive behavior that enhances the naturalness of interaction in the game.

Similar to the gaze behavior in the IGaze project, we realized an automatic camera pan that takes into account the affective state (see Fig 11, left side). While a character speaks, the camera shows its upper body. If no character speaks, both characters are shown. Generally, the camera angle is tilted according to the speaker's affective state. If the character is in a positive affective state, the camera shows the upper body with an ascending angle giving the impression that the character appears slightly bigger. In negative affective states, the camera shows the upper body with an descending angle giving the impression that the character appears slightly smaller.

When a user initiates a game, Sam and Max let the user welcome and explain the game setup and as well the general rules (see Fig 10, right side) before the poker game emerges. The use of History nodes at several positions in the *Interaction* supernode has reduced the complexity of the scene graph by reducing the amount of nodes and edges. As the example shows, the scene `"NextGame"` is played, if the *PlayGame* supernode is executed again, skipping the Welcome and Explain nodes and the connected scenes.

Figure 10: The AI Poker demonstrator at the CeBit exhibition.

The use of previously created gaze supernodes and the basic use of history nodes allowed us to create the AIPoker game in 3 months in total. Technically, the content of the poker game consists of 335 scenes organized in 73 groups. The final demonstrator application has been exposed at the CeBit exhibition and was extremely well attended by the visitors of the exhibition, as shown in Figure 10.

3.3 INTAKT - Multiple Interactive Virtual Characters as Shopping Assistants

This project investigates for an future grocery store approach the use of Virtual Characters in two different roles: 1) personal shopping assistant and 2) expert consultant (Kröner et al. 2009). The latter resides in a special display at every shelf and freezer. These characters' purpose is to explain details of food and provide navigation hints for a faster product localisation. Personal shopping assistants resides in a shopping cart display. They guide a user through the grocery store helping her/him to gather all goods of the provided shopping list.

The used dialog and interaction Sceneflow model reuses several Supernodes from the AI Poker Sceneflow model. Necessary was a slight revision of the camera control Supernodes due to the fact that the Virtual Shopping Assistants do not have any affect. However, the camera zooms at the speaking character showing his upper body.

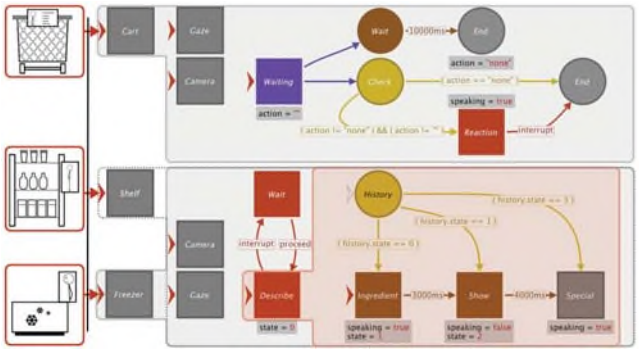

Figure 11: Reuse and extension of hierarchical concurrent Supernodes to model interaction with Virtual Shopping Assistants

Carts and shelves/freezers are equipped with a display showing (different) Virtual Characters that communicate via natural language and natural conversational behaviour with a user and between each other. In addition, the cart display shows a user's shopping list. The characters react every time a product is taken or placed.

The role of the cart character is to guide the user through the shopping list by making suggestions about products to buy (relying on the personal profile, e.g. user prefers ecological products). It serves as a personal advisor that checks every product that is placed in the card against individual needs and individual interests. Therefore, the content of the DPM of each product is used to reason about conflicts with the personality profile. Emerging conflicts are addressed via natural language by the cart character in a low voice - respecting the privacy. Additional information is presented on a display that is attached at the cart. In addition, the cart character may ask the shelf character with a loud (public) voice for help, e.g., if there is a product alternative.

The shelf character provides help by giving in shelf navigation hints for a faster product localisation. In addition, the character provides general information (like price, producer ...) in a natural conversational style.

The user becomes part of this dialog between the characters. Knowledge retrieved from the DPM of the involved products helps to create the illusion that Virtual Characters reacting intelligent to the consumer's interaction with the product.

3.4 SOAP: Modeling Multi-Party Dialogues for an Interactive Storytelling Application

The development of *interactive digital storytelling* systems has been a growing topic of research over the past years. They have been applied for applications in education and training (Marsella, Johnson, and Labore 2003; Si, Marsella, and Pynadath 2005; Swartout et al. 2006) as well as in entertainment and art (Mateas and Stern 2003; Riedl, Saretto, and Young 2003; Cavazza, Charles, and Mead 2001). While some of these systems explore user interaction by putting the user into the role of an observer that can change the world as the story progresses, the majority of them pursues a *dialogue-based* interaction approach. Such systems focus on creating a dramatic experience by offering a selection of dialogue situations in which the user is able to influence the progress and the outcome of the story through interactions.

For the development of interactive storytelling applications it is indispensable to provide authoring software that can be used by non-experts such as artists and screenwriters in order to create highly interactive and consistent multi-party dialogues with the virtual actors. These authoring tools need to have concepts to face challenges such as the continuous real-time processing and context-sensitive interpretation of user interactions, an adequate contemporary reactions to the user's discourse acts and the resumption and revision of dialogue content after unexpected interruptions.

Figure 12: The social game setting in the Virtual Beergarden.

We address these challenges in the social game scenario *SOAP* by using Scenemaker for the dialogue- and interaction management. These ideas have been realized in a demonstrator located in a *Virtual Beergarden* scenario, shown in Figure 12. In the soap-like story, the user and the virtual characters are involved in a romantic conflict. The user, who is represented by an avatar (Figure 12 ①), meets a group of girls (Figure 12 ②) and a group of guys (Figure 12 ③) as well as a waitress (Figure 12 ④). The user can approach the focus groups, listen to their conversations and contribute to the story and thus, influence the progress and outcome of the story.

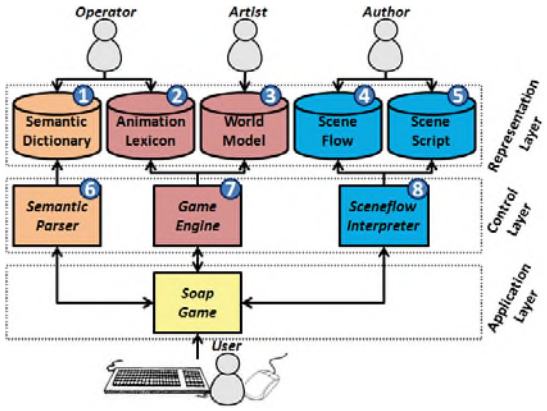

Figure 13: SOAP's component-based system architecture.

The system architecture can be found in Figure 13. The different components are embedded in three independent layers: (1) A representation layer, containing knowledge base and models specifying the scenario content. (2) The control layer, handling the processing of user input and the computation of system output and (3) the application layer enfolding the user interface. Vertically, the components can be categorized into (1) dialogue and interaction management, (2) natural language interpretation and (3) autonomous behavior control, described in the following:

Dialogue and Interaction Management: The behavior modeling as well as the dialog and interaction management of the virtual characters is realized with our modeling tool. An author can specify dialog- and behavior content in a *scenescript* (Figure 13 ⑤) and model the logic of behavior and dialog with a *sceneflow* (Figure 13 ④). An interpreter software executes the model and is, thus, controlling the virtual characters in the game (Figure 13 ⑧).

Figure 14 shows a part of the modeled sceneflow. Each focus group, the user avatar and other game objects are modeled in separate concurrent automata. We also recursively make use of parallel automata in order to model the behavior of individual characters and their behavioral aspects. This procedure reduces the modeling effort and increases the clarity of the model because it prevents the state explosion of the model, which could be observed if we modeled the whole scenario with a simple flat statechart. Furthermore, it allows us to change the behavior of individual focus groups or characters in isolation.

A major requirement in this application was to allow the user to change the focus group, or initiate and terminate a conversation respectivly, at any time. Therefore, each dialogue situation had to be contemporarily interruptible. To create a coherent storytelling experience an interrupted di-

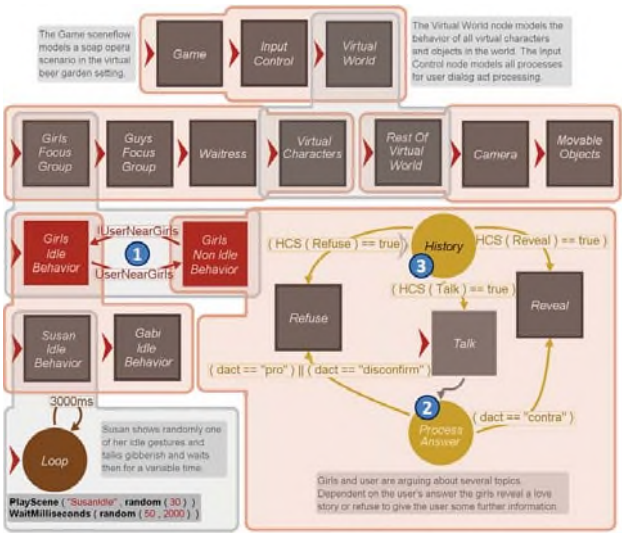

Figure 14: Part of the sceneflow from Soap.

alogue situation had to be consistently resumed after reentering the target group. For these reasons, a highly interactive dialogue structure was modeled and the runtime history was used in order to keep track of previous interactions and the progress of the dialogue. Ongoing dialogues are interrupted whenever the user leaves a focus group and resumed whenever the user reenters the focus group (Figure 14 ①). Consistent resumption or reopening of a previous dialogue is guaranteed by a recursive use of the runtime interaction history (Figure 14 ③). Context-sensitive reaction to the user's interaction is modeled by branching the dialogue structure dependent on the current state of the dialogue and the user's dialogue act provided by the NLU pipeline (Figure 14 ②).

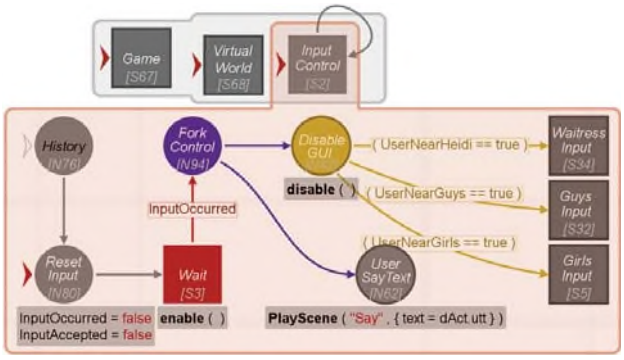

Figure 15: User input processing and control.

To factor out the logic for the detection and the processing of user interactions, we modeled a separate parallel automaton, as shown in Figure 15. This reduces the effort of modeling such logics within the automata for the individual dialog situations to a minimal amount, again effectively increasing the clarity of the model.

Natural Language Interpretation: Our natural language recognition and interpretation pipeline includes a spell checker and the semantic parser *Spin* (Engel 2005) (Figure 13 ⑥) which translates the user's typed-text input into abstract *dialogue-acts* based on the *DAMSL* coding scheme (Core and Allen 1997). The underlying semantic rules are specified in a set of dictionaries (Figure 13 ①) specifying knowledge about the dialogue content. Figure 16 exemplifies a set of rules, syntactic and semantic categories as well as preprocessing steps. The example rule (Figure 16 ③) states that if the user's input contains one of the words "how", "do" or "what" in correlation with the word "you" and any word belonging to the semantic category *location*, the abstract speech act *ask-location* is triggered. The semantic category *location* (Figure 16 ④) contains the words "location", "place", "beergarden", "here" and "party". Thus, different user utterances (Figure 16 ⑤) are parsed into the same dialogue-act. In addition, word stems (Figure 16 ②) and other pre-processing steps can be defined (Figure 16 ①) such as summarizing negations.

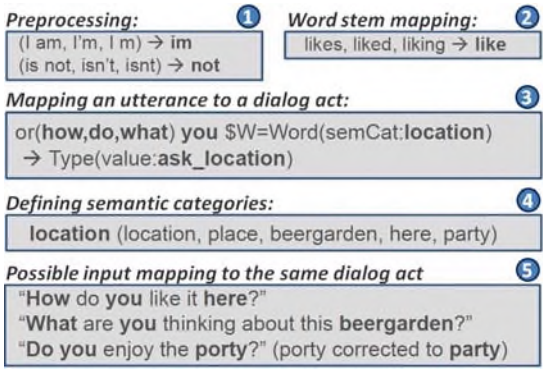

Figure 16: Knowledge defined for the semantic parser.

Autonomous Behavior Control: The scene displayed in the Virtual Beergarden is described by a world model created by an artist (Figure 13 ③). While high-level behaviors of the virtual characters such as speech and gestures are specified using the Scenemaker tool, low-level behaviors such as positioning, agent orientation and proximity or inter-agent gazing are handled automatically by the Virtual Beergarden application. In that manner the author does not need to take care of them. Animations for virtual characters are specified in an *animation lexicon* (Figure 13 ②) including over 40 different gestures and postures for each agent. Following (McNeill 1992), we divide every animation into preparation, stroke and retraction phases, which are used for gesture customization. A Bayesian network can be set for the agents in order to define aspects such as personality or emotional state that influence the manner in which nonverbal behaviors are executed. This has been exemplified for the phenomena of culture-related differences in behavior (Rehm et al. 2007).

3.5 DynaLearn: Modeling Educational Roles for Teaching Assistants

Embodied conversational agents are widely used in educational applications such as virtual learning and training environments (Johnson, Rickel, and Lester 2000). Beside possible negative effects of virtual characters (Rickenberg and Reeves 2000), there is empirical evidence that virtual pedagogical agents and learning companions can lead to an improved perception of the learning task and increase the learners' commitment to the virtual learning experience (Mulken, André, and Müller 1998). They can promote the learners' motivation and self-confidence, help to prevent or overcome negative affective states and minimize undesirable associations with the learning task, such as frustration, boredom or fear of failure. Teams of pedagogical agents can help the learners to classify the conveyed knowledge and allow for a continuous reinforcement of beliefs (André et al. 2000).

Modeling Different Educational Roles In the framework of the *DynaLearn* project, we developed an interactive learning environment in which learners can express their conceptual knowledge through qualitative reasoning models (Bredeweg et al. 2009) and enriched the learning experience with a cast of virtual characters, aiming at increasing learners' motivation and learning success. We considered a variety of teaching methods, learning strategies and ways of knowledge conveyance and verification. These strategies were realized by modeling different virtual hamsters that can play different educational roles (Mehlmann et al. 2010; Bühling et al. 2010). Beside several teachable agents, we modeled a teacher character and a quizmaster character and employed them in various teaching sessions. Figure 17 shows the example of a quizmaster (Figure 17 ①) and two teachable agents (Figure 17 ②) from an educational quiz session as well as an entity diagram representing conceptual system knowledge (Figure 17 ③) from which the reasoning module generates the questions asked by the quizmaster during the quiz session.

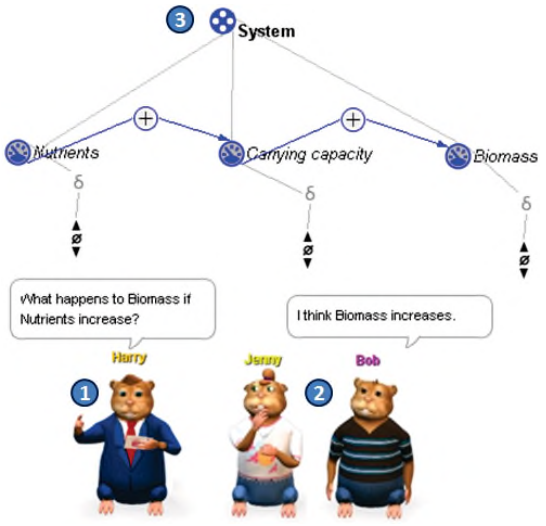

Figure 17: A quizmaster and teachable agents in a quiz

Creating Scenes with Generated Content The logic and the dialogue structure of the different teaching methods were modeled with the Scenemaker authoring tool. Therefore, we integrated the Scenemaker authoring suite with the knowledge reasoning engine by providing a set of functions callable from within the sceneflow model that directly accesses the application interface of the knowledge reasoning module. The possibility to parameterize scenes with arguments from within the sceneflow model allowed the creation of dialogue content consisting both of prescripted content and generated content retrieved from the knowledge reasoning module. Figure 18 shows two examplary scenes (Figure 18 ①, ②) containing generated content that was beforehand retrieved from the knowledge reasoning module over one of the application interface functions (Figure 18 ③).

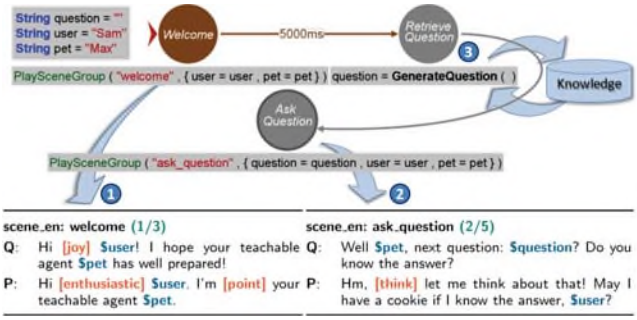

Figure 18: Hybrid scene creation from generated knowledge

4 Field Tests

Success in building different interactive applications with virtual characters for entertainment (Gebhard et al. 2008), education (Mehlmann et al. 2010; Kipp and Gebhard 2008) and commerce (Kröner et al. 2009) permits very promising conclusions with respect to the suitability of our approach. However, mainly computer-experts have been involved in the development of these applications. For this reason, we conducted several field tests and practical workshops with students of different age groups and genders to determine in how far the approach is suited for non-experts. The participating students from various educational levels brought no specific background skills or previous knowledge.

4.1 Nano Camp 2009: School Students Creating Flirting Embodied Conversational Agents

The Scenemaker authoring tool was exposed to a challenging field test in June 2009 at the German Research Center for Artificial Intelligence (DFKI) with secondary and grammar school students (age 12-17). The European broadcasting company *3sat* had invited students from all over Germany to a one-week science camp for hands-on experience with scientific topics. In this context, 12 students were to try out Scenemaker to create an interactive scenario in only 1.5 hrs without prior knowledge or experience. To make this possible, we created a sample scenario where two agents are engaged in a flirt dialogue. Interactivity was given by being able to change the one agent's "flirting strategy" (careful

vs. aggressive). The 12 students were grouped into 6 teams of two people each. After a short introduction (10 mins) including a sample dialogue, the students had some time for brainstorming and sketching the dialogue (40-60 mins). Afterwards, they implemented the sceneflow with minimal assistance (20-40 mins). Every team had to be finished in 1.5 hrs maximum. All 6 teams finished and gave positive feedback about the authoring experience. The resulting scenarios were viewed in the whole group.

4.2 Girls' Day 2010: Teenage Girls Creating a Family Sitcom with Virtual Hamster Characters

A second field test was conducted in the context of Germany's nationwide Girls' Day program (Endrass et al. 2010). This initiative is geared exclusively to female middle school students (age 12-15) and aims at encouraging the students to pursue a career in the natural sciences. Augsburg University invited 9 middle school students to their computer science institute. The students used the Scenemaker tool to create a social game scenario with virtual hamster characters. The 9 students were grouped into 3 teams of three people each and all decided to create some kind of family sitcom episode of about 5 minutes. After a short introduction (20 mins) into the concepts of Scenemaker's modeling approach and the handling of the graphical user interface, the students had some time for brainstorming and sketching the dialogue (40 mins). Afterwards, they modelled the sceneflow with minimal assistance (40 mins), as shown in Figure 19. All 3 teams finished in time and the remarkable resulting scenarios were viewed by the whole group. The students were asked to fill out an evaluation sheet in which they gave positive feedback about the authoring experience.

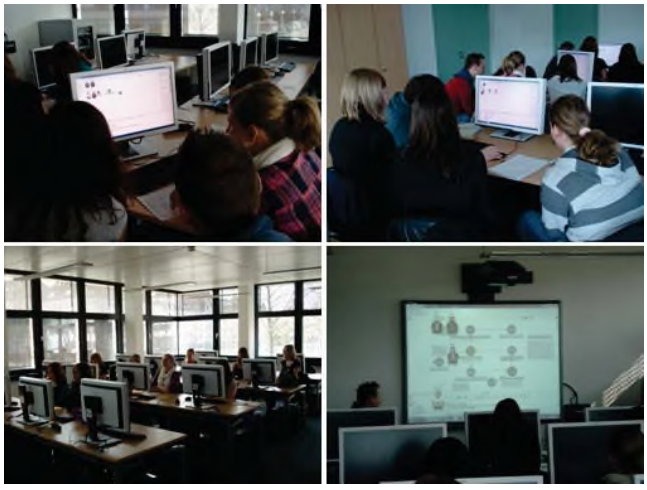

Figure 19: Pictures from the Girls' Day Authoring Session.

4.3 MMP 2011: College Students Creating an Interactive Game with a Virtual Opponent

A third field test was conducted in the context of the multimedia project workshop at the computer science institute of Augsburg University. College students (5th semester) had to model the logic of an interactive battleship game with a virtual opponent (see Figure 20) and the narrative structure of dialogues with the Scenemaker authoring tool. The 9 students were grouped into 2 teams of four people each. The students had been introduced to the modeling concepts of Scenemaker's modeling approach and the handling of the graphical user interface in one lecture session. Most remarkable was, that these students made extensive use of parallel automata for the specification of the virtual opponent's behavior. Each group modeled an automaton simulating the emotional state of the virtual opponent dependend on its success in the game. They synchronized the emotional state with several parallel automata specifying the expressive behavior of the virtual opponent, e.g. for facial expressions and gestures. This showed that the students completely understood and applied Scenemaker's concepts of modularity and compositionality. Already having some previous knowledge in programming, the students claimed that they would have needed much more time and effort to implement the game logic and the agents behavior in a higher programming language, such as Java. They especially praised the intuitive way of creating and synchronizing several concurrent processes, compared with the difficulty they would have had with multi-threading models.

Figure 20: Pictures from the Battleship Game.

4.4 Conclusion

The participants of the field tests were able to quickly pick up most of our concepts for modeling interactive narrative with statecharts and writing scenescripts was promptly and completely understood. This comprehension was directly transfered into the creation of vivid interactive scenarios with virtual characters.

During the field tests, it has been noticeable that the students, with the exception of the college students, in contrast to the computer experts, occasionally had difficulties to apply the more complex concepts of our approach. While the concept of hierarchical and parallel decomposition was

fully understood, the students mostly used concurrent processes to model completely independent parallel behaviors while they rarely utilised the synchronization measures of our language. Furthermore, the history concept was rarely used, because the dialog structure modelled by the students was mostly linear or a tree-like branching structure. They only occasionally had the idea to model dialog situations that could be resumed or reopened after an interruption by using the history concept of our approach.

These observations could be be explained with the short amount of practice time for the students and the schedule of the workshops. We believe that the more complex modeling concepts of our approach will also be fully understood by non-experts after more intensive practice. In the future we plan to do more field test over a longer period of time in order to prove our assumption. This would also allow us to evaluate the quality of the stories and dialogues modeled by the students.

5 Conclusion and Future Work

In this paper, we described a modeling approach to mixed-initiative multi-party dialogues with virtual characters. We presented the integrated authoring tool Scenemaker which allows an author to model complex dialogue behavior and interaction management for multiple virtual characters in a rapid-prototyping style. The statechart language provides different interaction handling policies for an author to handle continuous real-time interaction. The user can interrupt a dialogue at any time and expect a contemporary response. An interaction history allows the author to model reopening strategies for dialogues. After a dialogue was interrupted, it can consistently be resumed and previous dialogue topics can be revised. Our statecharts can be hierarchically refined to create contexts for the interpretation of user input. Parallel decomposition allows to model different behavioral aspects, functions and modalities in isolation. This modular approach reduces the complexity of the model while improving extensibility and reusability. Dialogue content can be authored manually or generated automatically. Blacklisting strategies allow an easy way to provide variability by avoiding repetitive behavior. Autonomous behavior can be specified without the need for an author to explicitly model it.

Success in building different interactive applications with virtual characters for entertainment (Gebhard et al. 2008), education (Mehlmann et al. 2010) and commerce (Kröner et al. 2009) as well as promising feedback from several field tests (Endrass et al. 2010; Kipp and Gebhard 2008) validates the usefulness of our approach. In field tests, students were able to quickly pick up the visual concepts or our UI. In addition, the concept of finite state based modeling of interactive narrative with statecharts as well as the writing scenescripts were promptly and completely understood by the students. This comprehension was directly transfered into the creation of vivid interactive scenarios with virtual characters. Regarding the results of the field tests, we conclude that the Scenemaker software is suitable for rapid prototyping, even for beginners and may be used as an educational device.

Our future work refers on the one hand to technical improvements of the authoring tool based on the user feedback we received so far, such as refinements of our modeling approach, and on the other hand to additional user studies that explore further issues, such as the quality of the scenarios generated with Scenemaker.

For the future we plan to integrate our system with other components, as for example emotion simulation and emotional speech synthesis, nonverbal behavior generation as well as speech recognition. This implies the use of standard languages such as FML, BML and EmotionML (Vilhjalmsson et al. 2007; Kipp et al. 2010; Kopp et al. 2006). Furthermore, we want to integrate a dialog domain knowledge component to the authoring framework, which allows authors to easily define domain knowledge and rules that can map utterances to abstract dialog acts dependent on the dialog domain. This implies the use of an ISO standard for dialogue act annotation (Bunt et al. 2010).

Feedback from users in different field tests and projects has shown that one strength of the modeling approach with Scenemaker is the reusability of already modeled behavioral patterns in the form of sub-models, because this can drastically reduce the modeling effort and complexity. We plan to factor a library of reactive behavior patterns that can be reused for an easy creation of different behavioral aspects for multiple virtual characters. Therefore, one of our main purposes is to identify abstract universal behavioral patterns that appear in multi-party dialogues with multiple virtual characters. We want to provide the author with a library of predefined and parameterizable state chart models, implementing behavioral patterns that can be reused in several projects and can easily be adjusted to the respective context.

6 Acknowledgments

The work described in this paper work was supported by the European Commission within the 7th Framework Programme in the projects IRIS (project no. 231824) and DynaLearn (project no. 231526). In addition, the research was funded in part by the German Federal Ministry of Education and Research under grant number 01 IA 08002 (project SemProM) and under grant number 01 IS 08025B (project INTAKT). We are grateful for the Virtual Character technology and the extensive support provided by the Charamel GmbH.

References

André, E.; Rist, T.; van Mulken, S.; Klesen, M.; and Baldes, S. 2000. The automated design of believable dialogues for animated presentation teams. In *Embodied conversational agents*. Cambridge, MA, USA: MIT Press. 220–255.

Augsburg University. http://mm-werkstatt.informatik. uni-augsburg.de/projects/gameengine.

Bredeweg, B.; Linnebank, F.; Bouwer, A.; and Liem, J. 2009. Garp3 - Workbench for qualitative modelling and simulation. *Ecological Informatics* 4(5-6):263–281.

Brusk, J.; Lager, T.; Hjalmarsson, A.; and Wik, P. 2007. Deal: Dialogue management in scxml for believable game characters. In *ACM Future Play*, 137–144. ACM.

Bühling, R.; Wissner, M.; Häring, M.; Mehlmann, G.; and André, E. 2010. Design Decisions for Virtual Characters in the DynaLearn Interactive Learning Environment. In *Book of Abstracts of the 7th International Conference on Ecological Informatics*, 144–145. Ghent University, Belgium.

Bunt, H.; Alexandersson, J.; Carletta, J.; Choe, J.-W.; Fang, A. C.; Hasida, K.; Lee, K.; Petukhova, V.; Popescu-Belis, A.; Romary, L.; Soria, C.; and Traum, D. R. 2010. Towards an ISO standard for dialogue act annotation. In *7th International Conference on Language Resources and Evaluation (LREC)*.

Cavazza, M.; Charles, F.; and Mead, S.-J. 2001. Agents' Interaction in Virtual Storytelling. In *IVA 2001*, 156–170.

Core, M., and Allen, J. 1997. Coding Dialogs with the DAMSL Annotation Scheme. In *Working Notes of AAAI Fall Symposium on Communicative Action in Humans and Machines*.

Endrass, B.; Wissner, M.; Mehlmann, G.; Buehling, R.; Haering, M.; and André, E. 2010. Teenage Girls as Authors for Digital Storytelling - A Practical Experience Report. In *Workshop on Education in Interactive Digital Storytelling on ICIDS*.

Engel, R. 2005. Robust and efficient semantic parsing of freeword order languages in spoken dialogue systems. In *Interspeech 2005*.

Gandhe, S.; Whitman, N.; Traum, D.; and Artstein, R. 2008. An integrated authoring tool for tactical questioning dialogue systems.

Gebhard, P.; Kipp, M.; Klesen, M.; and Rist, T. 2003. Authoring scenes for adaptive, interactive performances. In *AAMAS 2003*. ACM. 725–732.

Gebhard, P.; Schröder, M.; Charfuelan, M.; Endres, C.; Kipp, M.; Pammi, S.; Rumpler, M.; and Türk, O. 2008. IDEAS4Games: Building Expressive Virtual Characters for Computer Games. In *IVA 2008*, 426–440.

Harel, D. 1987. Statecharts: A visual formalism for complex systems. In *Science of Computer Programming*, volume 8, 231–274. Elsevier.

Heidig, S., and Clarebout, G. 2010. Do pedagogical agents make a difference to student motivation and learning? A review of empirical research. *Educational Research Review*.

Iurgel, I. A.; da Silva, R. E.; Ribeiro, P. R.; Soares, A. B.; and dos Santos, M. F. 2009. CREACTOR - An Authoring Framework for Virtual Actors. In *IVA 2009*, 562–563. Springer.

Johnson, W. L.; Rickel, J. W.; and Lester, J. C. 2000. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. *International Journal of Artificial Intelligence in Education* 11:47–78.

Johnstone, K. 1979. *Impro. Improvisation and the Theatre*. New York: Routledge/Theatre Arts Books.

Kipp, M., and Gebhard, P. 2008. IGaze: Studying reactive gaze behavior in semi-immersive human-avatar interactions. In *IVA 2008*, 191–199.

Kipp, M.; Neff, M.; Kipp, K. H.; and Albrecht, I. 2007. Toward natural gesture synthesis: Evaluating gesture units in a data-driven approach. In *IVA 2007*, LNAI 4722, 15–28.

Kipp, M.; Heloir, A.; Gebhard, P.; and Schröder, M. 2010. Realizing multimodal behavior: Closing the gap between behavior planning and embodied agent presentation. In *IVA 2010*. Springer.

Kopp, S.; Krenn, B.; Marsella, S.; Marshall, A.-N.; Pelachaud, C.; Pirker, H.; Thórisson, K.-R.; and Vilhjálmsson, H. 2006. Towards a common framework for multimodal generation: The behavior markup language. In *IVA 2006*.

Kröner, A.; Gebhard, P.; Spassova, L.; Kahl, G.; and Schmitz, M. 2009. Informing customers by means of digital product memories. In *1st international Workshop on Digital Object Memories*.

Marsella, S., and Gratch, J. 2006. Ema: A computational model of appraisal dynamics. In *Agent Construction and Emotions*.

Marsella, S.; Johnson, W.-L.; and Labore, C.-M. 2003. Interactive pedagogical drama for health interventions. In *Artificial Intelligence in Education*, 341–348. IOS Press.

Mateas, M., and Stern, A. 2003. Facade: An experiment in building a fully-realized interactive drama. In *Game Developer's Conference: Game Design Track*.

McNeill, D. 1992. *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press.

McTear, M. F. 1998. Modelling spoken dialogues with state transition diagrams: experiences with the cslu toolkit. In *ICSLP 1998*.

Mehlmann, G.; Häring, M.; Bühling, R.; Wissner, M.; and André, E. 2010. Multiple agent roles in an adaptive virtual classroom environment. In Albeck, J.; Badler, N.; Bickmore, T.; Pelachaud, C.; and Safonova, A., eds., *IVA 2010*, 250–256. Springer.

Miksatko, J.; Kipp, K.-H.; and Kipp, M. 2010. The persona zero-effect: Evaluating virtual character benefits ona learning task. In *IVA 2010*. Springer.

Mulken, S. V.; André, E.; and Müller, J. 1998. The persona effect: How substantial is it? In *Proceedings of HCI on People and Computers XIII*, 53–66. London, UK: Springer-Verlag.

Perlin, K., and Goldberg, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *Computer Graphics (SIGGRAPH)*, 205–216. ACM.

Prendinger, H.; Saeyor, S.; and Ishizuka, M. 2004. MPML and SCREAM: Scripting the Bodies and Minds of Life-like Characters. In *Life-like Characters – Tools, Affective Functions, and Applications*. Springer. 213–242.

Rehm, M.; Bee, N.; Endrass, B.; Wissner, M.; and André, E. 2007. Too close for comfort? Adapting to the user's cultural background. In *Workshop on Human-Centered Multimedia*.

Rickenberg, R., and Reeves, B. 2000. The effects of animated characters on anxiety, task performance, and evaluations of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, 49–56. New York, NY, USA: ACM.

Riedl, M.; Saretto, C.-J.; and Young, R.-M. 2003. Managing interaction between users and agents in a multi-agent storytelling environment. In *AAMAS 2003*, 741–748. ACM.

Schröder, M. 2008. *Emotions in the Human Voice, Culture and Perception*, volume 3. Pural. chapter Approaches to emotional expressivity in synthetic speech, 307–321.

Si, M.; Marsella, S.; and Pynadath, D. 2005. Thespian: An architecture for interactive pedagogical drama. In *AIED 2005*.

Spierling, U.; Weiss, S.-A.; and Mueller, W. 2006. Towards accessible authoring tools for interactive storytelling. In *Technologies for Interactive Digital Storytelling and Entertainment*. Springer.

Swartout, W.; Gratch, J.; Hill, R.; Hovy, E.; Marsella, S.; Rickel, J.; and Traum, D. 2006. Toward virtual humans. *AI Magazine* 27(2):96–108.

Traum, D.; Leuski, A.; Roque, A.; Gandhe, S.; DeVault, D.; Gerten, J.; Robinson, S.; and Martinovski, B. 2008. Natural language dialogue architectures for tactical questioning characters. In *Army Science Conference*.

Vilhjalmsson, H.; Cantelmo, N.; Cassell, J.; Chafai, N.-E.; Kipp, M.; Kopp, S.; Mancini, M.; Marsella, S.; Marshall, A.-N.; Pelachaud, C.; Ruttkay, Z.; Thórisson, K.-R.; van Welbergen, H.; and van der Werf, R.-J. 2007. The behavior markup language: Recent developments and challenges. In *IVA 2007*.

von der Beeck, M. 1994. A comparison of statecharts variants. In *ProCoS 1994*, 128–148. Springer.