Modeling Parallel State Charts for Multithreaded Multimodal Dialogues

Gregor Mehlmann
Human Centered Multimedia
Augsburg University
Universitätsstrasse 6a
Augsburg, Germany
mehlmann@hcm-lab.de

Birgit Endrass
Human Centered Multimedia
Augsburg University
Universitätsstrasse 6a
Augsburg, Germany
endrass@hcm-lab.de

Elisabeth André
Human Centered Multimedia
Augsburg University
Universitätsstrasse 6a
Augsburg, Germany
andre@hcm-lab.de

ABSTRACT

In this paper, we present a modeling approach for the management of highly interactive, multithreaded and multimodal dialogues. Our approach enforces the separation of dialogue content and dialogue structure and is based on a statechart language enfolding concepts for hierarchy, concurrency, variable scoping and a detailed runtime history. These concepts facilitate the modeling of interactive dialogues with multiple virtual characters, autonomous and parallel behaviors, flexible interruption policies, context-sensitive interpretation of the user's discourse acts and coherent resumptions of dialogues. An interpreter allows the realtime visualization and modification of the model to allow a 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. We present a demonstrator illustrating our concepts in a social game scenario.

Categories and Subject Descriptors

D.1.7 [**Visual Programming**]

General Terms

Algorithms, Languages, Theory

Keywords

Dialogue Modeling, Multimodality, Multithreading

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 [11, 25]. 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 [19], creating believable facial expressions, gestures and body movements [16], expressive speech synthesis [31] and natural language recognition as well as dialogue and interaction management [35]. In this work, we address the tasks of modeling consistent, highly interactive multithreaded and multimodal 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 area of games, film or theater screenplay.

These systems are created to facilitate the modeling 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. These approaches usually generate parallel behavior of several agents, however they do not explicitly include support for scripting the behavior of multiple agents and their synchronization in a simple and intuitive way for an author. Examples for character-centric systems are *Improv* [27] or *Scream* [28], where an author defines an 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 the CSLU toolkit [23], *Cyranus* [12], *Scenejo* [33], *Deal* [1] and *Creactor* [13]. *Hybrid* approaches, as described in [7] or [8], 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 systems supports easily handled concepts for both, a dialogue and interaction his-

ICMI'11, November 14–18, 2011, Alicante, Spain.

tory and the modeling of concurrent processes for creating and synchronizing parallel behavior on an intuitive authoring level. However, this would facilitate the modeling task and reduce the complexity of the model. It would allow to follow a modular and compositional modeling paradigm, thus increasing clarity and reusability of the model. This 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 by presenting a hybrid approach to contribute on the user modeling level for the creation of multithreaded and multimodal interactive virtual character applications in a rapid-prototyping style.

2. DIALOGUE MANAGEMENT

Central concept of our modeling approach 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*, which is a statechart variant specifying the logic according to which scenes are played back and commands are executed. Sceneflows have concepts for the *hierarchical refinement* and the *parallel decomposition* of the model 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 [10, 37].

Sceneflows and scenescripts are created using a graphical modeling tool and executed by an *interpreter* software. This allows the realtime extension and modification of the model and the direct observation of the effects without the need for an intermediate translation step or even the need to pause the execution of a sceneflow. The realtime visualization of a sceneflow's execution and active scenes within the graphical user interface allows to test, simulate and debug the model. These features facilitate and accelerate the modeling process and, thus, allow the creation of interactive virtual character performances in a rapid-prototyping style.

2.1 Creating Multimodal Dialogue Content

As shown in Fig. 1, 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 (Fig. 1 ①), such as retrieved information from user interactions, sensor input or generated content from knowledge bases.

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 (Fig. 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.

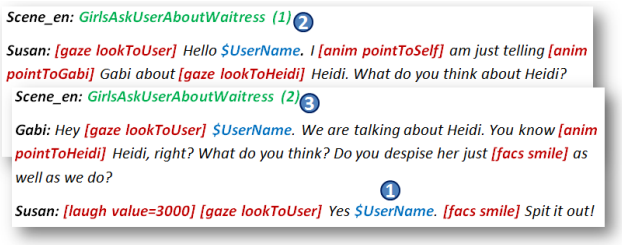

Figure 1: Parameterizable scenes of a scenegroup.

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, such as type- and variable definitions as well as variable assignments and function calls to predefined functions of the underlying implementation language (Fig. 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 (Fig. 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, external events or the change of environmental conditions.

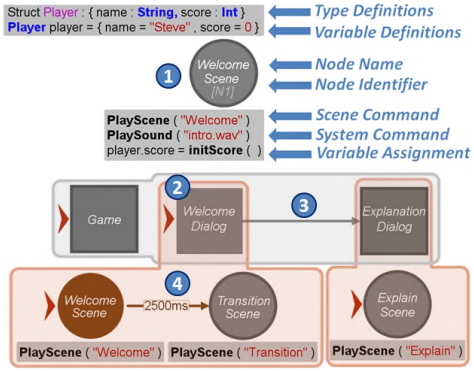

Figure 2: Node statements and supernode hierarchy.

Different *branching strategies* within the sceneflow, such as 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 (Fig. 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* (Fig. 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* (Fig. 5 ②). Probabilistic edges are used to create some degree of randomness and desired non-determinism during the execution of a sceneflow. A *conditional edge* represents a conditional transition and is labeled with a *conditional expression*, as shown in Fig. 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.

2.3 Interaction Handling Policies

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 realtime 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* or inheritance of conditional edges:

- *Interruptive conditional edges* (Fig. 3 ①,③) are inherited with an interruptive policy and are used for 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* (Fig. 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.

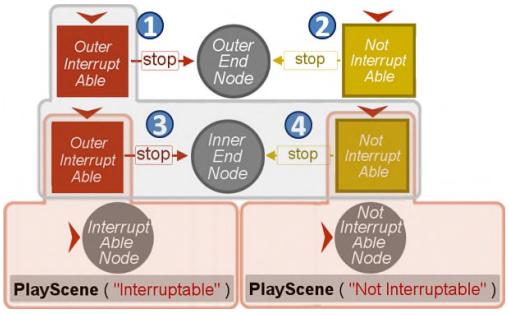

Figure 3: ①,③ Interruptive conditional edges and ②,④ simple non-interruptive conditional edges.

Fig. 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 (Fig. 3 ②,④) will be executed to its end. However, the scene within the supernodes with the interruptive conditions (Fig. 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 (Fig. 3 ④) before the outer end node is executed (Fig. 3 ②). In the interruptive case the execution of the sceneflow immediately continues with the outer end node (Fig. 3 ①) because the outer interruptive edge has priority over the inner interruptive edge (Fig. 3 ③).

2.4 Modeling Multihreaded Dialogue

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. By defining multiple startnodes for a supernode (Fig. 4), each subautomaton which consists of all nodes reachable by a startnode, is executed by a separate process. By defining *fork edges* (Fig. 5 ①), an author can create multiple concurrent processes without the need for changing the level of the node hierarchy.

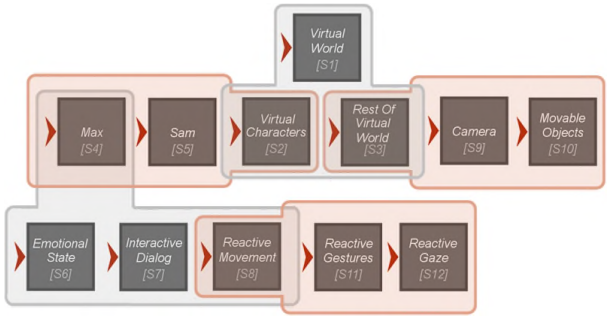

Figure 4: Hierarchical and parallel decomposition.

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 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 be-

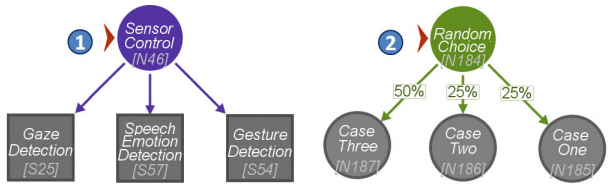

Figure 5: ① Concurrent processes with fork edges and ② randomization with three probability edges.

havioral aspects. This communication is realized by a *shared memory model* which allows an asynchronous non-blocking synchronization of concurrent processes.

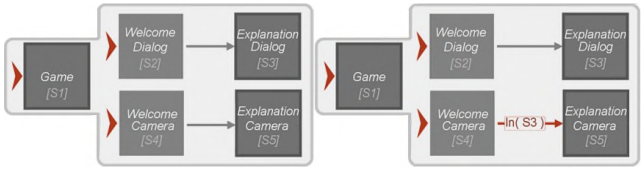

Figure 6: Synchronization over configuration states.

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* (Fig. 6) which represents a more intuitive mechanism for process synchronization. This condition allows to request weather a certain parallel automaton's state is currently executed by the sceneflow interpreter.

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.

Fig. 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

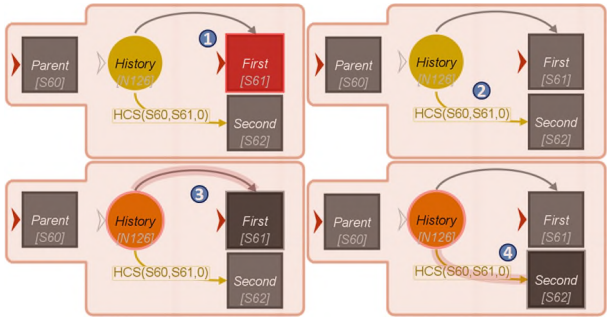

Figure 7: History node and condition.

`First` (Fig. 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 (Fig. 7 ②) to find out if the supernode 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` (Fig. 7 ③) is executed or the node `Second` (Fig. 7 ④) is started over the history node.

3. *SOAP:* MULTI-PARTY DIALOGUES IN INTERACTIVE STORYTELLING

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 [20, 32, 34] as well as in entertainment and art [21, 30, 3]. 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. Thereby, there arise challenges such as the continuous realtime 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. We address these challenges in the social game scenario *Soap* by using the concepts of our approach presented in Section 2. We realized our ideas of dialogue-based interactive entertainment in a demonstrator, which is explained in the following.

3.1 Concept Overview

To this end, we choose a social game located in our *Virtual Beergarden* scenario (Fig. 8). 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 (Fig. 8 ①), meets a group of girls (Fig. 8 ②) and a group of guys (Fig. 8 ③) as well as a waitress (Fig. 8 ④). The user can approach the focus groups, listen to their conversations

or contribute to the story. In case the user does not focus a certain group, the characters show idle dialogue behavior and talk gibberish. Through dialogue interactions, the user can advise the characters and, thus, influence the progress and outcome of the story.

Figure 8: The setting in the virtual beergarden.

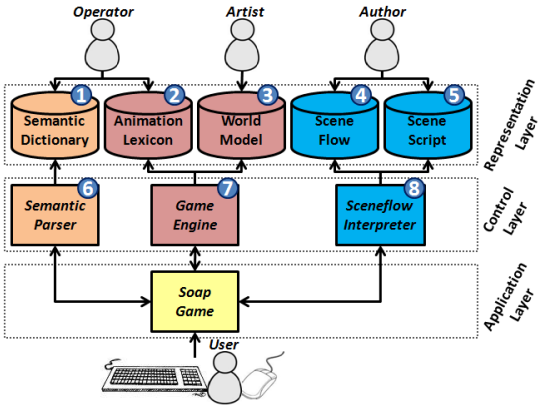

Figure 9: Component-based system architecture.

The different components of the system architecture are embedded in three independent layers: The representation layer contains knowledge bases and models specifying the scenario content (Fig. 9 ①-⑤). The control layer manages the processing of user input and the computation of system output (Fig. 9 ⑦-⑧). Finally, the application layer enfolds the user interface. Vertically, the components can be categorized into dialogue and interaction management (Fig. 9 ④,⑤,⑧), natural language interpretation (Fig. 9 ①,⑥) and autonomous behavior control (Fig. 9 ②,③,⑦).

3.2 Dialogue and Interaction Management

The behavior modeling as well as the dialogue and interaction management of the virtual characters is realized with our modeling tool. An author can specify dialogue- and behavior content in a *scenescript* (Fig. 9 ⑤) and model the logic of behavior and dialogue with a *sceneflow* (Fig. 9 ④). An interpreter software executes the model and is, thus, controlling the virtual characters in the game (Fig. 9 ⑧).

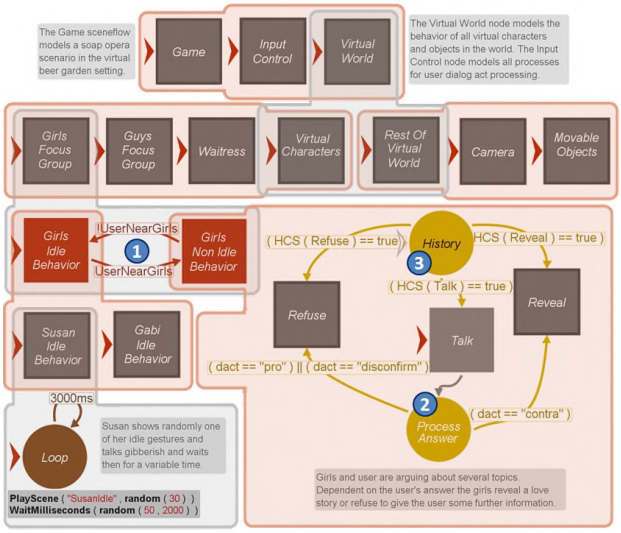

Figure 10: Part of the sceneflow from Soap.

Fig. 10 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 respectively, at any time. Therefore, each dialogue situation had to be contemporarily interruptible. To create a coherent storytelling experience, an interrupted dialogue 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 (Fig. 10 ①). Consistent resumption or reopening of a previous dialogue is guaranteed by a recursive use of the runtime interaction history (Fig. 10 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 (Fig. 10 ②).

To factor out the logic for the detection and the processing of user interactions, we modeled a separate parallel automaton, as shown in Fig. 11. This reduces the effort of modeling such logics within the automata for the individual dialogue situations to a minimal amount, again effectively increasing the clarity of the model.

3.3 Natural Language Interpretation

This component allows the natural language communication between the user and the focus groups. Our natural language recognition and interpretation pipeline relies the

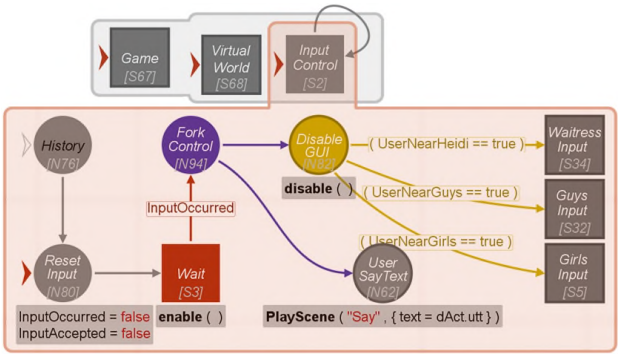

Figure 11: User input processing and control.

semantic parser Spin [6] (Fig. 9 ⑥), allowing text input to be processed by semantic rules mapping the users utterances into abstract *dialogue acts* based on the *DAMSL* coding scheme [4]. These rules are specified in a set of dictionaries that are created by an operator or the author (Fig. 9 ①). The dialogue acts are processed by the sceneflow interpreter according to the rules defined by the sceneflow model to ensure a contemporary, adequate and context-sensitive reaction to the user's utterances.

Fig. 12 exemplifies a set of rules, syntactic and semantic categories as well as preprocessing steps. The example rule (Fig. 12 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* (Fig. 12 ④) contains the words "location", "place", "beergarden", "here" and "party". Thus, different user utterances (Fig. 12 ⑤) are parsed into the same dialogue-act. In addition, word stems (Fig. 12 ②) and other pre-processing steps can be defined (Fig. 12 ①) such as summarizing negations. In addition, we integrated a *spell checker* into our language recognition component to be able to cope with faulty or incomplete text input. In interactive entertainment applications this is of special interest, since typos or incomplete sentences occur rather often.

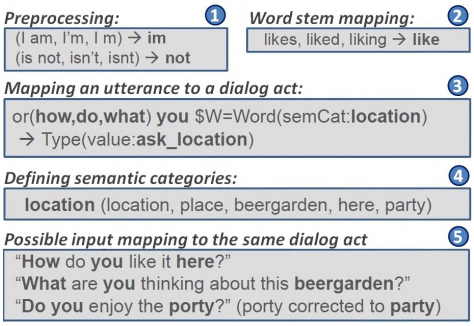

Figure 12: Knowledge for the semantic parser.

The language recognition engine also allows the dynamic exchange of dictionaries from the sceneflow model. In that manner, authors can adapt the dictionary rules to the local dialogue context. User interactions could, for example, be interpreted differently in different contexts. With a set of dictionaries, these are less complex, clearly arranged and exchangeable. In addition, the parsing of sentences can be a lot easier in limited domains. Dictionaries that do not deliver interpretable results in a certain context, can be dynamically substituted with more general dictionaries.

3.4 Autonomous Behavior Control

The game component is responsible for the graphical representation of our scenario. The scene displayed in the Virtual Beergarden is described by a world model created by an artist (Fig. 9 ③). Core concept of this component is the automatic animation selection for autonomous low-level behaviors. While high-level behaviors of the virtual characters such as speech and gestures are specified using the modeling tool, low-level behaviors such as positioning, agent orientation and proximity or inter-agent gazing can be handled automatically by the application. They are triggered by the system automatically, accompanying the behavior specified by the author, so that the author does not need to take care of them. If, for example an author specifies that an agent should move through the scenario and approach another agent, the virtual characters will automatically detect each other when entering their social distance zones and react by e.g. turning their bodies in a way that they are facing each other or turning their gaze towards the other characters face.

As shown in Fig. 9, animations are specified in a nonverbal knowledge base, an *animation lexicon* (Fig. 9 ②). In the current version of the system, each agent can perform over 40 different gestures and postures. Following [22], we divide every animation into the following phases: preparation, stroke and retraction. In the preparation phase, the hands are brought into the gesture space, the stroke phase carries the content of the gesture, while in the retraction phase, the hands are finally brought back into a resting position. These phases are used as a basis for gesture customization, e.g. to match the agent's social or personal background. In particular, the stroke phase of a gesture can be varied in order to show different gestural expressivities (see [26]), such as speed or repetition. In the Virtual Beergarden application the parameter repetition, can be varied by playing the stroke phase several times, while it can be played faster or slower to customize the speed parameter. A Bayesisian 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 [29]. In order to create different characters in an interactive story, the Bayesisian network can easily be replaced, e.g. to focus on other aspects such as personality or emotional state. Depending on the values set for a virtual character, animations are customized accordingly. In that manner, the author of a story needs to define a character only once, e.g. as being extroverted. In case the author wants to change the gestural style for a characters, e.g. because of a different emotion, this can easily be done at runtime.

4. FIELD TESTS AND DISCUSSION

Success in building different interactive applications with virtual characters for entertainment [9], education [24, 14] and commerce [18] 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 [5]. The participating students from various educational levels brought no specific background skills or previous knowledge. Nevertheless, they 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, 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 dialogue structure modelled by the students was mostly linear or a tree-like branching structure. They only occasionally had the idea to model dialogue 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. The field tests all had the same schedule. After a short introduction (20 mins) into the concepts of the 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). 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 for multithreaded and multimodal dialogues with virtual characters. Our approach 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. An interaction history allows an author to model reopening strategies for dialogues so that they can consistently be resumed and previous dialogue topics can be revised. Hierarchical and 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. Autonomous behavior can be specified without the need for an author to explicitly model it. We realized our ideas of dialogue-based interactive entertainment in a demonstrator application.

Our future work refers on the one hand to technical improvements of the modeling tool based on the user feedback we received so far and refinements of our modeling approach, on the other hand to additional user studies that explore further issues, such as the quality of the scenarios generated with our approach. 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 [36, 15, 17]. Furthermore, we want to integrate a dialogue domain knowledge component to the authoring framework, which allows authors to easily define domain knowledge and rules that can map utterances to abstract dialogue acts dependent on the dialogue domain. This implies the use of an ISO standard for dialogue act annotation [2]. Feedback from users in different field tests and projects has shown that one strength of the presented modeling approach 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 statechart models, implementing behavioral patterns that can be reused in several projects and can easily be adjusted to the respective context.

Regarding the experiences with the presented approach in many applications and the feedback from several field tests, we conclude that our approach is suitable for modeling complex multithreaded and multimodal dialogue behavior and interaction management for multiple virtual characters in a rapid-prototyping style and that our modeling tool can be used as an educational device.

6. ACKNOWLEDGMENTS

The work described in this paper was supported by the European Commission within the 7th Framework Programme in the project IRIS (project no. 231824).

7. REFERENCES

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

[2] H. Bunt, J. Alexandersson, J. Carletta, J.-W. Choe, A. C. Fang, K. Hasida, K. Lee, V. Petukhova, A. Popescu-Belis, L. Romary, C. Soria, and D. R. Traum. Towards an ISO standard for dialogue act annotation. In *7th International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta, May 19-21, 2010 2010.

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

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

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

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

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

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

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

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

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

[12] I. Iurgel. Cyranus : An authoring tool for interactive edutainment applications. In *Technologies for E-Learning and Digital Entertainment*, Hangzhou, China, April 16-19 2006. Springer, Berlin.

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

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

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

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

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

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

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

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

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

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

[23] M. F. Mctear. Using the cslu toolkit for practicals in spoken dialogue technology. In *University College London*, pages 1–7, 1999.

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

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

[26] C. Pelachaud. Multimodal expressive embodied conversational agents. In *ACM international conference on Multimedia*, pages 683–689, 2005.

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

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

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

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

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

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

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

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

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

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

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