Proactive Authoring for Interactive Drama: An Author's Assistant

Mei Si, Stacy C. Marsella, and David V. Pynadath

Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
meisi@isi.edu, marsella@isi.edu, pynadath@isi.edu

Abstract. Interactive drama allows people to participate actively in a dynamically unfolding story, by playing a character or by exerting directorial control. One of the central challenges faced in the design of interactive dramas is how to ensure that the author's goals for the user's narrative experience are achieved in the face of the user's actions in the story. This challenge is especially significant when a variety of users are expected. To address this challenge, we present an extension to Thespian, an authoring and simulating framework for interactive dramas. Each virtual character is controlled by a decision-theoretic goal driven agent. In our previous work on Thespian, we provided a semi-automated authoring approach that allows authors to configure virtual characters' goals through specifying story paths. In this work, we extend Thespian into a more proactive authoring framework to further reduce authoring effort. The approach works by simulating potential users' behaviors, generating corresponding story paths, filtering the generated paths to identify those that seem problematic and prompting the author to verify virtual characters' behaviors in them. The author can correct virtual characters' behaviors by modifying story paths. As new story paths are designed by the author, the system incrementally adjusts virtual characters' configurations to reflect the author's design ideas. Overall, this enables interactive testing and refinement of an interactive drama. The details of this approach will be presented in this paper, followed by preliminary results of applying it in authoring an interactive drama.

1 Introduction

Interactive drama allows people to participate actively in a dynamically unfolding story, by playing a character or by exerting directorial control. Because of its potential for providing interesting stories as well as allowing user interaction, it has been proposed for a wide range of training applications (e.g. [1–5]) as well as entertainment applications (e.g. [6–11]). A variety of approaches have been taken to build interactive dramas. These approaches can be roughly divided into two categories, character-centric and narrative-centric designs (when speaking from the agent's perspective, they are also referred as using autonomy and story-based agents for interactive dramas [12]). In character-centric approaches,

e.g. FearNot! [2], MRE [4], SASO [13], and Thespian [14, 3], the emphasis is on building plausible characters. The narrative ideally emerges from the characters' behaviors. Narrative-centric designs on the other hand focus more on the narrative structure of the overall story instead of the design of individual characters. Systems deploying this type of approach include Façade [15], Mimesis [16] and IDA [8]. These systems employ some representations of narrative structure that are used to control how the story unfolds.

One of the central challenges faced in the design of interactive dramas is how to reduce authoring effort resulting from the merge of narrative and interaction. This challenge is especially significant when a variety of users are expected. Different users may interact with the system in very different ways. To ensure that the desired (pedagogical and/or dramatic) effect will be experienced, or the experience is appropriately tailored for different types of users, the system needs to be repeatedly tested with different user behaviors. In most of the existing systems, this step is performed by hand. However, a thorough examination by hand is extremely time consuming and impossible in many cases. Consider a short scene consisting of 5 rounds of interaction between a user and virtual characters, if at each step the user has 10 reasonable moves, there are 10^5 possible user action sequences or paths through the story!

To address this problem, this work discussed here presents a proactive, automated approach to assist the author in evaluating and incrementally refining the performance of an interactive drama system. The key to our approach is modeling types of users in order to limit the number of story paths the author must evaluate.

The approach we take is based on extending Thespian [14, 3], a multi-agent system for authoring and simulating interactive dramas. The user is modeled as a decision-theoretic goal-driven agent. This allows the authoring procedure to restrict the generation of users' behaviors to ones that are *well-motivated* in the sense that they have consistent motivations throughout the story.

This selective generation based on the well-motivated criteria can greatly reduce the number of possible user action sequences because it spares the author from testing the system with user behaviors that are unlikely to happen. In addition, because it is often the case that most of the generated paths are consistent with the author's expectations and therefore do not need special attention, the authoring procedure can filter story paths using either default or author defined criteria that characterize types of stories worthy of inspection. Due to the selective generation combined with the post-generation filtering, the system can proactively prompt the author to pay attention to only a small set of paths that are most likely to be problematic. The author's changes to these paths can then be passed back to Thespian to incrementally refine the story's design.

The rest of this paper is structured as follows. Section 2 gives an overview of Thespian's proactive authoring procedure. Section 3 introduces the example domain. Section 4 and 5 describe the current architecture of the Thespian system and its extension to proactive authoring. Section 6 provides a detailed example

of applying the proactive authoring procedure. Finally, in section 7 the results are discussed and followed by proposed future work.

2 Overview

The approach we take to proactive authoring is based on extending Thespian's original authoring process. The new approach exploits two aspects of Thespian's design. First, each story character in Thespian is modeled as a decision-theoretic, goal-driven agent, with the character's personality/motivations encoded as goal preferences. Characters' behaviors in the story, their policies of actions, are automatically generated based on these goal preferences. Second, Thespian has a fitting algorithm that allows a character's goal preferences to be derived automatically from example story paths (in effect, sequences of character actions). Thus, an author can provide Thespian with example story paths and then Thespian can fit the behavior of the characters to perform according to those paths.

Figure 1 shows the overall structure of Thespian's new proactive authoring procedure. As in previous work on Thespian, the author provides story paths to the fitting procedure in order to fit the virtual characters. The system can then systematically simulate potential users' behaviors interacting with the virtual characters. Users are modeled as decision-theoretic agents, similar to other Thespian characters. More importantly, the simulation is restricted to consider only potential users whose behavior is well- motivated (See Section 5.2 for a discussion on how we use fitting to operationalize "well-motivated"). Next, the authoring procedure filters the story paths generated by this simulation in order to select those that are most likely to be problematic. The resulting paths are passed to the author. The author's feedback can then be passed back to Thespian's fitting procedure, in the form of modified story paths, to further refine the design of the story's characters. This overall process can proceed iteratively to design the interactive story.

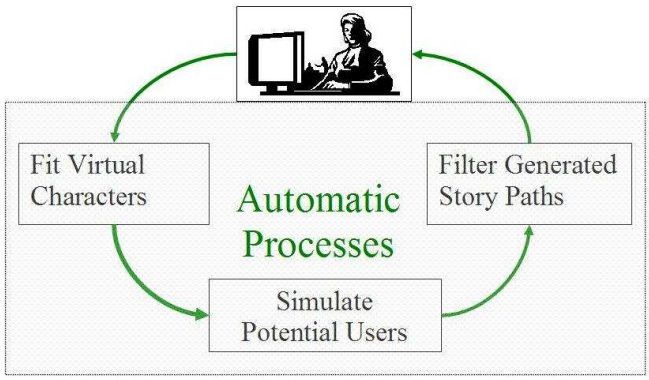

Fig. 1. Thespian's Proactive Authoring Procedure

3 Example Domain

The example domain of this work is a Grimms' fairy tale, "Little Red Riding Hood". The story contains four main characters, Little Red Riding Hood, Granny, the hunter and the wolf. The story starts as Little Red Riding Hood (Red) and the wolf meet each other on the outskirt of a wood while Red is on her way to Granny's house. The wolf has a mind to eat Red, but it dare not because there are some wood-cutters close by. At this point, they can either have a conversation or choose to walk away. The wolf will have a chance to eat Red at other locations where nobody is close by. Moreover, if the wolf heard about Granny from Red, it can even go eat her. Meanwhile, the hunter is searching the woods for the wolf to kill it. Once the wolf is killed, people who got eaten by it can escape. In our modeling of the story, the numbers of possible actions per move for Red, Granny, the hunter and the wolf are 14, 2, 4, and 10 respectively. The role of Red is assumed to be taken by the user. The user can interact with virtual characters through a text based interface.

4 Thespian's Current Architecture

Thespian is a multi-agent system for authoring and controlling virtual characters in an interactive drama. It is built upon PsychSim [17], a multi-agent system for social simulation based on Partially Observable Markov Decision Problems (POMDPs) [18].

4.1 Thespian Agent

Thespian's basic architecture uses POMDP based agents to control each character, with the character's personality and motivations encoded as agent goals. Each Thespian agent consists of five components, its state, action dynamics, goals, policies, and beliefs about self and others.

An agent's state is defined by a set of state features, such as its location and degree of hunger. The values of these state features change as events (actions of characters) happen in the story. The agent's action dynamics define how its state is affected by actions. An agent's goals are expressed as a reward function over the various state features the agent seeks to maximize or minimize. We model a character's personality profile as its various goals and their relative importance (weight). For example, the wolf character can have goals of satisfying its hunger (increasing the value of its state feature "full") and keeping itself alive, with the latter goal having higher importance. Thespian agents have a "Theory of Mind" that allows them to form mental models about other agents, including the user. An agent's belief (subjective view) of the world includes its belief about the state, action dynamics, goals and policies of itself and other agents and *their* subjective views of the world, a form of recursive agent modeling [19]. Policies tell the agent what to do given its current state. Currently, all agents use a bounded

lookahead policy. Following this policy, when an agent selects its next action it projects limited steps into the future to evaluate the effect of each option. The agent considers not just the immediate effect of an action, but also the expected responses of other characters and, in turn, the effects of those responses, and its reaction to those responses and so on. The agent evaluates the overall effect with respect to its goals and then chooses the action that has the highest expected value.

4.2 Fitting Procedure and Authoring

Thespian's fitting procedure enables an author to define characters' roles in a story by creating alternative desired paths (sequences of characters' actions) of the story. The fitting procedure [20, 14] can tune agents' goal weights so that they will re-create the paths if the user behaves following the paths. When the user deviates from the paths, the virtual characters will respond using the same motivations fitted from the story paths.

Algorithm 1 FIT-SEQUENCE(S_0, *char_name*, *seq*, $fixedgoals$)

```
1: S_0 : initial state set by author at initialization
2: char_name : character whose role is to be fitted
3: seq : time sequence of action - preferred path
4: fixedgoals : goals whose weights should not be changed in this process
5: C ← [ ] : constraint on goal weights
6: S ← S_0
7: for each action A in seq do
8:    if A.actor == char_name then
9:       # adding constraints
10:      for each action a in char_name.getOptions() do
11:         new_C ← Reward(A,S) ≥ Reward(a,S)
12:         C.Append(new_C)
13:   # update state
14:   S ← S × Dynamics(A)
15: Return AdjustGoalWeights(char_name, C, fixedgoals)
16:
17: Dynamics(action) as defined in PsychSim
18: Reward(action,state) calculated similar to PsychSim, with modifications to take
    turn-taking into account
19: AdjustGoalWeights(char_name, constraints, fixedgoals) returns if the char-
    acters' goal weights, except the weights of the fixedgoals, can be adjusted so that
    all the constraints are satisfied
```

In fitting, Thespian proceeds iteratively for each story path, fitting the goals of one agent at a time and holding all other agents' goals as fixed. Specifically, for each story path and each character, Algorithm 1 is invoked to fit that character so that it performs its actions in the story path. The algorithm proceeds down the

sequence of actions in the story path (Step 7). If the current action is performed by the agent that is currently being fitted (Step 8), the fitting process simulates the agent's lookahead process, and automatically calculates constraints on goal weights to ensure the desired action receives highest utility among all candidate actions (Step 11). By the end, the constraints resulting from fitting each path can be merged into one common constraint set. By default, in fitting, the weights of all of the agent's goals can be adjusted. The author can also specify a set of goals whose weights should not be adjusted, e.g. characters should always have a high goal weight on keeping themselves alive. Typically, there are multiple candidate goal weight values that are consistent with the story paths defined by the author. Any of these solutions guarantees that the characters will follow the preferred paths of the story. Thespian will pick the goal weights as close to the original ones as possible. When fitting results in no candidate goal weight values, it is not possible for the characters to be motivated to behave following all the story paths and the author has to exclude some of the paths.

5 Proactive Authoring Procedure

In this section, we will present our new proactive authoring procedure which can help the author to examine the interactive drama system's performance when facing a variety of users. This procedure can systematically simulate potential users' behaviors, filter correspondingly generated story paths, and proactively prompt authors to verify virtual characters' behaviors in paths that are most likely to be problematic. Furthermore, the author can specify the granularity of the exploration to control the number of story paths that will be simulated as well as criteria that determine which paths the system identifies for the author to inspect. In particular, the author can specify the achievement (or lack of achievement) of plot points as filtering criteria, where plot points can be defined in terms of events in the story, and characters' (including the user's) beliefs or their actual states in the story. Thus, the author can easily tell if the desired user experience is achieved with the current configuration of virtual characters. If it is not, the author can correct virtual characters' behaviors by specifying new story paths to be fitted.

5.1 Model Potential Users

Thespian models the user as a decision-theoretic goal-driven agent similar to other characters in the story. The users' different interaction styles are reflected by their different preferences over the set of goals they have and their different beliefs about other characters.

Similar to how other characters are designed, the user agent's initial values of state features, action dynamics and beliefs are based on the story being modeled. Most of this information is either common-sense knowledge, e.g. social norms, or conveyed to the user before the interaction through a back story, e.g. action

dynamics, beliefs about self and other characters. Therefore, this part of the model is fixed for simulating all types of users unless the author purposely leaves some information uncertain to the user, e.g. tells the user that there is 50% of the chance that the wolf is a good character and will not eat anybody.

Though we can usually assume the user has the goals of the character which he/she takes the role of, the user's goals are seldom limited to the role. People may have various personal goals, and habits to behave in certain ways, which can be cast as goals in Thespian. Currently we consider users' additional goals from two possible sources. Firstly, as a member of a society, people's behavior usually follows social norms. In Thespian, we cast these norm-following behaviors of a character as goals to maximize relevant state features that keep track of how consistent the norms and behaviors are. Secondly, people may have various personal goals during the interaction. Curiosity is a basic human need and a common goal in virtual environments; often a user wants to explore the environment more and converse more with the virtual characters. Table 1 lists some of the possible user goals we model in the "Little Red Riding Hood" story. In this example, the role of Red is assumed to be taken by the user.

Category	**Example Goals**
Character (Role) Goals	keep self alive give cake to Granny
Social Norm Goals[1]	complete adjacency pair keep appropriate conversational flow be succinct
Personal Goals	like to explore the environment like to converse

Table 1. An Example of User's Possible Goals

5.2 Simulate Interactions with Potential Users

This section presents Thespian's automated approaches for simulating interactions with potential users. The basic approach can exclusively generate all possible story paths that can be encountered by a well-motivated user, a user who has consistent motivations through out the interaction. A special case of applying this approach is to simulate "protypical" users who have fixed sets of goals as their prototypes. The prototypes are either designed by the author or are derived by discretizing the user's possible goal space. The basic approach assumes that the user only has one possible mental model of other characters. This can be extended to simulate users having alternative mental models.

[1] Note that only a subset of the goals in Thespian's model of social normative behaviors apply to this domain. For detailed description of this model, see [21].

Algorithm 2 GENERATE-ALL-PATHS(S_0, *user*, $fixedgoals$, *maxstep*, $existPath$)

```
1: S_0 : initial state set by author at initialization
2: user : the name of the user character
3: fixedgoals : goals whose weights should not be changed in this process
4: maxstep : maximum number of actions the user performs
5: existPath : path that has already happened
6: for each action a in user.getOptions() do
7:    pathnew ← existPath.append(a)
8:    res ← Fit-Sequence(S_0,user,pathnew,fixedgoals)
9:    # if this is a possible path
10:   if res == 1 then
11:      # simulate other characters' responses to the user's action
12:      while user does not have the next turn do
13:         other_character's_action ← getResponse()
14:         pathnew ← pathnew.append(other_character's_action)
15:      maxstep_new ← maxstep - 1
16:      if maxstep_new > 0 then
17:         Generate-All-Paths(S_0,user,fixedgoals,maxstep_new,pathnew)
18:      else
19:         # this is a path we want
20:         Output-to-File(pathnew)
```

Generate Story Paths for Well-Motivated Users Algorithm 2 shows the pseudo code of this process. This process moves in a stepwise fashion. First, we find all actions of the user that are well-motivated at the current state (well-motivated actions are determined by invoking the fitting procedure as described in section 4.2), then for each of these actions other characters' responses are simulated. When it is the user's turn to act again, the same process will be repeated until the length of story paths specified by the author is reached.

This process is implemented as a recursive function. When the function is called to generate story paths starting from the very beginning of the story, $existPath$ is empty. If the author wants to simulate possible story paths starting from the middle of the story, $existPath$ should contain all the actions that have already happened by that point.

Note that unlike the fitting procedure in the authoring process, in which actions with equal utility as others are regarded as not motivated enough, the fitting procedure in this function will treat the actions as well-motivated. The reason for this difference is that here we want to simulate all actions the user can possibly perform instead of ensuring certain actions will be picked by an agent. In fact, the author can make this simulation process more tolerant to inaccurate user modeling by passing an epsilon value to the fitting procedure to relax the **AdjustGoalWeights** function (see Algorithm 1 for definition). This way, actions will be treated as well-motivated even though their utilities are slightly lower than the maximum utility the agent might achieve via some other action(s).

Generate Story Paths for Protypical Users It is often sufficient and interesting to test a system's behaviors with protypical users. The prototypes can be designed by the author. Alternatively, Thespian models protypical users by systematically varying the user agent's preference over its goals. The granularity preference set by the author will define the fineness of the discretization. For each set of user's goal weights, the **Generate-All-Paths** procedure will be called to generate all possible story paths with these goal weights. While calling this process, the $fixedgoals$ parameter needs to be set to all of the agent's goals. This way the **AdjustGoalWeights** function will only perform a test on whether the action is motivated by the agent's current goal weights, and will not try to adjust the agent's goal weights.

Generate Story Paths for Users with Alternative Mental Models To simulate potential users with different goals and different mental models about other characters, the author needs to supply users' alternative mental models in terms of beliefs about other characters' goals and initial states. The author can either design the alternative mental models by hand or derive them using automated means (see [22] for a possible approach). Thespian will then set the user agent's belief according to each of the models (as shown in [22], the number of distinct models in a story environment is limited), and call the **Generate-All-Paths** procedure to generate all story paths that can be encountered by users with that mental model. Finally, story paths generated by simulating users with different mental models will be merged.

5.3 Filter Generated Story Paths

The number of story paths resulting from simulating well-motivated users may still approach a large number. In addition, it is often the case that most of the generated paths are consistent with the author's expectations and therefore do not need special attention. So Thespian has a special mechanism for selecting story paths to present to the author.

Thespian can use both its default heuristics and author specified criteria for filtering story paths. Currently, Thespian provides two default heuristics. The first one picks story paths in which the virtual characters repeat the same behaviors more than a certain percentage, e.g. 75% of the time. The second heuristic selects story paths in which the virtual characters repeat the same behavior continuously more than a certain number of times, e.g. 3 times. These two heuristics are designed based on the observation that it is usually not a good interactive experience if the virtual characters always respond to the user with the same actions. Both heuristics are configurable by the author. The author can specify the names of the characters to be watched for (by default all the virtual characters' behaviors will be included in the analysis), the thresholds for reporting, e.g. instead of 75% the threshold can be lowered to 60% for the first heuristics, and the actions to be watched for, e.g. paths will be selected only if the wolf repeats "do nothing" all the time. In addition to the default heuristics,

the author can specify the achievement (or lack of achievement) of plot points as additional filtering criteria. Plot points can be defined in terms of events in the story, e.g. pulling out story paths in which the wolf eats Granny and is not killed by the end, and characters' (including the user's) beliefs or their actual states in the story, e.g. pulling out story paths in which Red believes she has a close social distance with the wolf by the end. The paths selected by different criteria often overlap with each other. As the last step, the filtering procedure will merge paths selected by all the criteria and present the distinct paths to the author.

To complete the authoring process, the author needs to review the paths being selected, and make modifications to them (correct virtual characters' behaviors) if the paths are different from his/her expectations. For example, in one of the paths generated in the examples given in Section 6, the hunter arrives at the place where Red and the wolf are. The hunter would have killed the wolf, but Red keeps on talking to him. While the hunter is entrapped in the conversation with Red, the wolf runs away. The author may want to design a new story path that forces the hunter to kill the wolf instead of talking to Red in that situation. Next, the newly corrected story paths together with the originally designed story paths are passed to the **Fit-Sequence** function to reconfigure the virtual characters. When necessary, additionally rounds of evaluation (by simulating potential user's behaviors) and reconfiguration of the virtual characters can be iteratively performed.

6 Example Authoring Process

This section provides an example of using the proactive authoring approach to facilitate the design of an interactive drama, the story of Red Riding Hood.

Step 1: Tune characters' goals to story paths specified by the author. The following story path is passed to the **Fit-Sequence** function. Sentences in parentheses describe the status of the story that can not be seen easily from the characters' actions.

Red greets the wolf. − > the wolf greets back. − > Red tells the wolf that she is on her way to visit Granny. − > the wolf says bye to Red. − > Red says bye to the wolf. − > Both Red and the wolf walk away. − > (Red and the wolf meet outside of Granny's house) the wolf eats Granny. − > (the hunter arrives) the hunter kills the wolf. − > Granny escapes from the wolf.

Step 2: Simulate potential users' behaviors. To demonstrate simulating users with alternative mental models of other characters, we assume the users are informed that in this story the wolf might be a good character and will never eat people. These users might have this mental model or the default mental model of the wolf being evil. Thespian generated all possible story paths that can be encountered by users with 5 rounds of interaction. In addition, Thespian simulated

protypical users' behaviors. Due to space limitation, only the results of simulating the following two prototypes are listed in Table 3: the user has a dominant goal of giving the cake to Granny and the user has a dominant goal of NOT giving the cake to Granny (This is the case that the user does not adapt the character's goals). When simulating each of these prototypes, the user agent's other goal weights are set to the results of fitting to the story path described above.

Step 3: Filter the generated story paths and present to the author. When filtering the paths to present to the author, the wolf is selected as the only character to be examined because in this story both the hunter and Granny are expected to have repeated actions, e.g. the hunter moves around searching for the wolf all the time. In addition to the default filtering heuristics, we define the achievement of the following plot point as a filtering criterion: the wolf eats Granny and is not killed by the end. The summaries of results are given in Table 2 and Table 3. Note that by letting the user play RED, the total number of paths in the story is 14^5!

Mental model	**Possible paths**	**Selected by heuristics**	**Selected by plot point**
the wolf is good	2456	181	72
the wolf is evil	1837	95	56

Table 2. Results of simulating users with alternative mental models

Mental model	**Want Granny to have cake**	**Possible paths**	**Selected by heuristics**	**Selected by plot point**
the wolf is good	Yes	1688	98	68
the wolf is good	Opposite	2318	175	72
the wolf is evil	Yes	1513	86	56
the wolf is evil	Opposite	1729	92	56

Table 3. Results of simulating protypical users

7 Discussion and Future Work

Though we have not formally evaluated this proactive authoring procedure, the numbers reported in Section 6 are very informative. In this domain, simulating a 5-round interaction with a well-motivated user results in thousands of possible story paths. Though these numbers are already dramatically smaller than the total number of paths (14^5), it is still impossible for a human author to exam each of these paths. Fortunately, the filtering process further cuts the numbers

of paths that need to be reviewed into about 1/10 of all the paths that have been generated, and allows the author to specify his/her own filtering criteria using plot points. The numbers being reported here are story dependent and filtering criteria dependent. Nevertheless we feel this proactive authoring procedure is potentially a very helpful tool for evaluating and refining the design of an interactive drama. It enables automatic testing of the performance of the interactive drama, and the author only needs to pay attention to a limited set of story paths that are likely to be problematic. It is also worth mentioning that by extensively testing the virtual characters' behaviors, the modeling deficits in the virtual characters are also more easily revealed.

On the other hand, this authoring procedure is still at its early stage. We currently foresee three parts in future work. Firstly, we plan to build richer syntax which allows the author to specify more complex filtering criteria, e.g. a partial order of events is violated. Secondly, we seek to find more efficient automatic and domain independent algorithms for identifying potentially problematic story paths and assigning priorities to paths that need to be reviewed. These algorithms can help authors who are not familiar with a domain to start authoring. Finally, the current fitting procedure in Thespian can only fit characters' motivations to sequences of actions. As plot points can be specified in terms of not only actions (events), but also characters' beliefs, and actual states, we plan to extend the fitting procedure to enable automated character configuration using this information.

8 Conclusion

In this paper, we made our initial movement towards building a proactive authoring procedure within the Thespian framework. The proactive authoring procedure presented in this paper can automatically generate story paths that can be encountered by a well-motivated user, who may have alternative mental models about other characters. The authoring procedure can also simulate protypical users' behaviors. Further, the story paths generated by simulating potential users are filtered before being presented to the author, so that the author only needs to pay attention to paths identified as problematic. Two default, but configurable filtering heuristics are provided by Thespian. In addition, the author can specify the achievement (or lack of achievement) of plot points as filtering criteria. Finally, the author's feedback will be used to fine tune the virtual characters. This approach can reduce authoring effort resulting from open-ended user interaction. Especially, it can help the author design interactive dramas for a wide range of users for either ensuring the same desired effect will be experienced in all types of users, or tailoring the experience for different types of users.

Acknowledgments

We thank our colleagues, especially John Gratch, Mark Riedl, Bill Swartout for their support and thoughtful discussions.

References

1. Marsella, S.C., Johnson, W.L., Labore, C.: Interactive pedagogical drama for health interventions. In: AIED. (2003)
2. Paiva, A., Dias, J., Sobral, D., Aylett, R., Sobreperez, P., Woods, S., Zoll, C.: Caring for agents and agents that care: Building empathic relations with synthetic agents. In: AAMAS. (2004) 194–201
3. Si, M., Marsella, S.C., Pynadath, D.V.: Thespian: An architecture for interactive pedagogical drama. In: AIED. (2005)
4. Swartout, W., Hill, R., Gratch, J., Johnson, W., Kyriakakis, C., LaBore, C., Lindheim, R., Marsella, S.C., Miraglia, D., Moore, B., Morie, J., Rickel, J., Thiúbaux, M., Tuch, L., Whitney, R., Douglas, J.: Toward the holodeck: Integrating graphics, sound, character and story. In: Agents. (2001) 409–416
5. Riedl, M.O., Andrew, S.: Believable agents and intelligent scenario direction for social and cultural leadership training. In: the 15th Conference on Behavior Representation in Modeling and Simulation, Baltimore, Maryland (2006)
6. Riedl, M.O., Saretto, C.J., Young, R.M.: Managing interaction between users and agents in a multi-agent storytelling environment. In: AAMAS. (2003) 741–748
7. Cavazza, M., Charles, F., Mead, S.J.: Agents' interaction in virtual storytelling. In: Proceedings of the International WorkShop on Intelligent Virtual Agents. (2001) 156–170
8. Magerko, B.: Story representation and interactive drama. In: Artificial Intelligence and Interactive Digital Entertainment (AIIDE), Marina del Rey, CA (2005)
9. Louchart, S., Aylett, R.: The emergent narrative theoretical investigation. In: the 2004 Conference on Narrative and Interactive Learning Environments. (2004)
10. Szilas, N.: IDtension: a narrative engine for interactive drama. In: the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt Germany (2003)
11. Braun, N.: Storytelling in collaborative augmented reality environments. In: Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. (2003)
12. Mateas, M., Stern, A.: Towards integrating plot and character for interactive drama. In: Working notes of the Social Intelligent Agents: The Human in the Loop Symposium. AAAI Fall Symposium Series. (2000)
13. Traum, D.R., Swartout, W., Marsella, S.C., Gratch, J.: Fight, flight, or negotiate: Believable strategies for conversing under crisis. In: IVA. (2005)
14. Si, M., Marsella, S.C., Pynadath, D.V.: Thespian: Using multi-agent fitting to craft interactive drama. In: AAMAS. (2005) 21–28
15. Mateas, M., Stern, A.: Integrating plot, character and natural language processing in the interactive drama Façade. In: the International Conference on Technologies for Interactive Digital Storytelling and Entertainment. (2003)
16. Young, R.M., Riedl, M.O., Branly, M., Jhala, A.H., Martin, R.J., Saretto, C.J.: An architecture for integrating plan-based behavior generation with interactive game environments. In: Journal of Game Development. (2004)

17. Marsella, S.C., Pynadath, D.V., Read, S.J.: PsychSim: Agent-based modeling of social interactions and influence. In: Proceedings of the International Conference on Cognitive Modeling. (2004) 243–248
18. Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable Markov processes over a finite horizon. Operations Research **21** (1973) 1071–1088
19. Gmytrasiewicz, P., Durfee, E.: A rigorous, operational formalization of recursive modeling. In: ICMAS. (1995) 125–132
20. Pynadath, D.V., Marsella, S.C.: Fitting and compilation of multiagent models through piecewise linear functions. In: AAMAS. (2004) 1197–1204
21. Si, M., Marsella, S.C., Pynadath, D.V.: Thespian: Modeling socially normative behavior in a decision-theoretic framework. In: IVA. (2006)
22. Pynadath, D.V., Marsella, S.C.: Minimal mental models. In: AAAI. (2007)