Interactive Drama Authoring with Plot and Character: An Intelligent System that Fosters Creativity

Mei Si and **Stacy C. Marsella**
Information Sciences Institute
University of Southern California
{meisi, marsella}@isi.edu

Mark O. Riedl
School of Interactive Computing
College of Computing
Georgia Institute of Technology
riedl@cc.gatech.edu

Abstract

Computer-based systems for interactive dramas allow the user to participate actively in the unfolding of a story in a virtual world. Various approaches have been explored for facilitating the human author in creating computer-based interactive dramas. Most of these approaches can be categorized as either story-centric or character-centric designs. In this work, we present a new framework that integrates both character-centric and story-centric designs to support authoring of interactive dramas. This framework encourages the author to think in different levels of abstraction and different perspectives when designing interactive dramas. In addition, it works as a colleague to the author by suggesting ideas and critiquing the author's ideas. We explore the use of this new framework for fostering the author's creativity in designing interactive dramas. Preliminary examples of using this new framework to author an interactive drama are presented, followed by discussion and future work.

Introduction

Storytelling is an integral part of the human experience. Stories are told to exchange information about relevant events and occurrences, to entertain, and to educate. Dramas are artifacts that are deliberately created to achieve some desired effect on an audience. The process of creating dramas is one that requires a degree of skill, practical experience, and creativity.

Computer aided interactive drama allows the user to actively participate in the story, by playing a role or applying directorial control over the characters. The user's choices affect the unfolding of the story. The authoring process for interactive dramas is even more complex. Merging interactivity into narrative results in the need to handle a tremendous amount of contingencies. Authoring enough contingencies to create a rich environment for an engaging experience is often intractable (Riedl and Young 2006). Moreover, it is hypothesized that this labor intensive process often hurts the author's creativity.

Various computer aided authoring frameworks have been designed for dynamically generating interactive dramas and for facilitating human author's creation of interactive dramas. Many of these authoring frameworks adopt approaches inspired by theories of what makes a good story. In Poetics, Aristotle argued that character was subsidiary to action. A more contemporary view on character and action, as espoused by Lajos Egri (Egri 1949), suggests that plot unfolds based on the characters, that characters can essentially "plot their own story". Correspondingly, story-centric processes, e.g. (Mateas & Stern 2003; Szilas 2003; Braun 2003; Riedl, Saretto, & Young 2003; Young *et al.* 2004; Magerko 2005) for interactive drama focus on the structure of the overall story in terms of plot arc, and aim at providing automated approaches for arranging events to happen during the interaction to form a well-structured story. Character-centric processes, e.g. (Cavazza, Charles, & Mead 2001; Louchart & Aylett 2004; Traum *et al.* 2005; Si, Marsella, & Pynadath 2005), on the other hand, emphasize the development of individually plausible, autonomously motivated characters under the assumption that coherent narrative and dramatic effect should ideally emerge from interacting with these characters.

There are many ways in which a computer system can support creative activity. In Lubart's (Lubart 2005) work, four uses of computer systems are suggested:

1. Computer as *Nanny*: helps organizing the project and performs routine tasks for the user.
2. Computer as *Pen-pal*: facilitates collaboration among multiple users.
3. Computer as *Coach*: implements approaches to enhance creativity.
4. Computer as *Colleague*: contributes to idea generation.

Most of the existing authoring frameworks are not designed with a specific goal of fostering the author's creativity. Instead, they automate the generation of the interactive dramas based on the author's input, and therefore prevent the author's creativity from being hurt by onerous programming.

This paper presents a mixed initiative authoring framework that not only facilitates the author's design of interactive dramas, but also fosters the author's creativity. This new framework uses an original approach that integrates both story-centric and character-centric processes for interactive dramas. It allows the author to design interactive dramas at different levels of abstraction (plot level vs. moment-to-moment interaction level) and different perspectives (overall

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

story structure vs. motivated characters). The author receives feedback from the framework, which takes the form of explanations for how characters behave in guided simulations, and explanations for unexpected occurrences. In addition, the framework also gives creative suggestions on new actions to be included in the models of the story.

This new framework fosters the author's creativity by acting both as a *Coach* and a *Colleague* to the author. It allows and encourages the author to think of and design different aspects of the interactive experience; a common technique used for successfully designing traditional dramas. Moreover, this new framework stimulates the author's creativity by brainstorming with the author, and providing suggestions and feedbacks to the author's ideas.

The details of this new framework are provided in this paper. Preliminary examples of human author interacting with the framework to create an interactive drama are presented, followed by a discussion of implications for future work in intelligent creative authoring systems.

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 dares 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 hears 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 were eaten by it can escape.

Overview of the New Framework

When designing interactive dramas, the goal of the human author is to capture and encode his or her intentions for the interactive experience. We assume part of this intention is that the user's experience (a) constitutes a well-formed, story and (b) conforms to particular aesthetic, thematic, and/or pedagogical properties. For example, the author's intention may be that the user experiences a narrative that is reminiscent of the classic Little Red Riding Hood story (but not identical because the user has the ability to express his/her agency and explore alternative potential story paths) and itself makes up a coherent story. To support an authoring process with considerations of both character plausibility and well-formed story structure, this new framework integrates a partial order planner with the Thespian framework, which models virtual characters as autonomous agents, for interactive drama.

Partial order planners have often been used in story-centric processes for interactive drama creation because they can automatically generate sequences of the characters' actions - plans to reach story goals (e.g. Red is eaten by the wolf and then the wolf is killed by the hunter) and at the

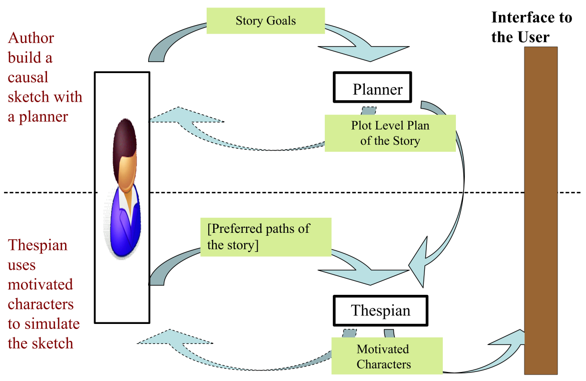
Author build a causal sketch with a planner
Story Goals
Interface to the User
Planner
Plot Level Plan of the Story
Thespian uses motivated characters to simulate the sketch
[Preferred paths of the story]
Thespian
Motivated Characters

Figure 1: Overview of the New Framework

same time ensure plausible causal relationship among events in the plan. However, such plans do not provide the author insight about the characters' motivations, and therefore can not avoid creating inconsistent character motivations during the interaction.

Thespian (Si, Marsella, & Pynadath 2005) is a multi-agent system for authoring and simulating interactive dramas. It adopts a character-centric approach, and uses decision-theoretic goal-driven agents to control each character, with the character's motivations encoded as agent goals with relative weights. Thespian provides an automated fitting procedure for authoring. This procedure can tune virtual characters' motivations to the story paths (sequences of the characters' actions) designed by the author. The resulting virtual characters will recreate the story paths when executed and the user's actions are the same as specified in the story paths. When the user deviates from the story paths, the characters will respond to the user using the motivations "learned" from the story path fitting process. However, each story path can only provide limited information about the characters' motivations. As a result, when the user's behavior dramatically differs from all the story paths designed by the author, the emergent interaction may not meet the author's expectations. To design an interactive drama, the author may need to design many story paths to prepare for different types of user interactions.

In this new framework, we integrate a partial order planner with the Thespian framework by using the planner to partially automate the story path designing process. The author works with the planner to construct plot outlines - represented as plans - and the plans are then used to provide guidelines for configuring Thespian agents, which are later used for moment-to-moment interactions with the users. To allow the author to design interactive dramas at different abstraction levels, this new framework provides a special integration procedure that can fit Thespian agents to plot level plans, which only contain major events (plot points) of the story. This procedure first fills in interactions between plot points, and then tunes Thespian agents' motivations to the final story path. During this process, feedback in terms of why the characters can or can not act according to the plot level plan are generated, including creative ideas on new actions to be included into the story.

Figure 1 lays out the overall structure of this new framework. This mixed initiative authoring framework lets the author interact with both the planner and the Thespian agents to create interactive dramas. The authoring process starts with the author using the planner to produce one or more plot level skeletons (plans) of the story. The author does so by providing specifications for what they would like to see in the plot. For example, the author might specify that Granny is eaten by the wolf. The planner responds by generating a plan such as: Red meets the wolf, Red tells the wolf about Granny and then the wolf eats Granny. It is important to note that this plan is merely a suggested set of plot points outlining some, but not all, of the events. The Thespian agents learn their motivations from the plans and give the author feedback, including the moment-to-moment interactions that need to be inserted between two plot points, e.g. Red and the wolf greet each other before Red tells the wolf about Granny, the motivations (goal weights) for the characters to behave following the plans, or new actions that need to be added to the models of the story, e.g. an action that can change Red's belief about the wolf from a bad character to a good character. Based on this feedback, the author may make modifications to the planner's model or Thespian's model of the story, and then generate new plans to fit the Thespian agents. The author can also directly interact with Thespian agents by specifying story paths to be included in fitting, in addition to those generated by the planner.

Implementation

In this section, we introduce the Thespian framework, and its integration with a partial order planner. A standard POP planner is assumed to be used in this framework, so no detail about the planner is included in this paper.

Thespian

Thespian is a multi-agent system for authoring and controlling virtual characters in an interactive drama. It is built upon PsychSim (Marsella, Pynadath, & Read 2004), a multi-agent system for social simulation based on Partially Observable Markov Decision Problems (POMDPs) (Smallwood & Sondik 1973). This section describes components in Thespian that are relevant to this new framework.

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 name location. The agent's **action dynamics** define how its state is affected by events (actions of characters) happen in the story. An agent's **goals** are expressed as a reward function over the various state features the agent seeks to maximize or minimize. For example, the wolf character may have goals of satisfying its hunger and keeping itself alive, with the latter one having higher importance. Agents have **recursive beliefs** about self and others, e.g. my belief about your belief about me. This forms a model of theory of mind. This model enables Thespian agents to reason about other characters' reactions when planning on their own behaviors. 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.

Fitting Procedure and Authoring Thespian's fitting procedure enables an author to define characters' roles in a story by creating alternative desired story paths (sequences of characters' actions) of the story. It judges if consistent character motivations can be inferred from them. If the answer is yes, it tunes the characters' motivations to the story paths.

Algorithm 1 Fit_Sequence(S_0, $charName$, seq, $fixedGoals$)

```
1:  $S_0$ : initial state set by author at initialization
2:  $charName$ : character whose role is to be fitted
3:  $seq$ : story path
4:  $fixedGoals$ : goals whose weights should not be
    changed in this process
5:  $C \leftarrow [\ ]$ : constraint on goal weights
6:  $S \leftarrow S_0$
7:  for each action $A$ in $seq$ do
8:      if $A.actor == charName$ then
9:          # adding constraints
10:         for each action $B$ in $charName$.getOptions() do
11:             $newC \leftarrow$ Reward($A$,$S$) $\geq$ Reward($B$,$S$)
12:             $C$.Append($newC$)
13:     # update state
14:     $S \leftarrow S \times$ Dynamics($A$)
15: return GoalWeights($charName$, $C$, $fixedGoals$)
16:
17: GoalWeights($charName$, $constraints$, $fixedGoals$)
    returns if $charName$'s goal weights 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. Typically, there are multiple candidate goal weight values that are consistent with the story paths defined by the author. 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 character to be motivated to behave following all the story paths and the author should exclude or modify some of the story paths.

Suggest Changes to Character's Beliefs "Suggest" is a function provided by PsychSim. Similar to fitting, it can make an agent perform an action that was not chosen previously. "Suggest" achieves this functionality almost the opposite way as fitting. In fitting, the relative goal weights of the agents are adjusted, and the agent's beliefs and state is untouched. The "suggest" function suggests changes to the agent's beliefs. These changes can lead the agent to choose the author's desired action without affecting the agent's current goals. For example, we want Red to perform the action of talking about Granny without being asked. Using fitting, the result may be that Red's goal of being talkative has an extremely high weight. On the other hand, the "suggest" function will suggest a change to Red's beliefs that Red thinks she is asked for the question. Finally, unlike fitting the "suggest" function currently cannot ensure that the agent performs a sequence of actions as specified. It only works with the agent's immediate next action.

Use Plot Level Plans to Train Thespian Agents

Using a planner, the author sets some goals for the planner to achieve - a goal can be a final state or intermediary states - and the planner automatically searches the action spaces of virtual characters in the story and constructs a plan (story path) to reach the final state. For example, in the Red Riding Hood domain, given the author's goal of the story as both Red and Granny being eaten by the wolf, the following plan may be generated:

1. Red and the wolf meet on the road.
2. Wolf: where are you going?
3. Red: I am going to Granny's house to give her this cake.
4. ...

In this framework, we replace the hand authored story paths in Thespian's authoring procedure with story paths generated by a planner, and use plans to provide guidance for virtual characters' behaviors. It is straightforward to tune Thespian agents' motivations to plans generated at the same detail level as that used by Thespian's model; the fitting procedure can be directly called. However, Thespian agents may also need to be trained to behave according to plot level plans, because this new framework is designed to allow and encourage the author to think of interactive dramas at different abstraction levels. To train Thespian agents in this case, each plot level plan is first expanded into a full story path. The Thespian agents are then fitted to these story paths and any additional ones designed by the author. This section presents the algorithms for expanding a plot level plan into a full story path.

Model Story at Different Abstraction levels To enable the integration of the plot level model of the story and detailed model of the story, the two models have to be compatible. Currently in this framework we require actions modeled in the planner and in Thespian share the same names and effects (action dynamics). When modeling a story at a more abstract level, the planner utilizes a smaller set of actions that does not include small talk or moment-to-moment interactions - only major events are kept. For example, the planner may model the Red Riding Hood story as only consisting of seven major events: Red tells the wolf where Granny lives, the wolf eats Granny, the wolf eats Red, the hunter kills the wolf, Red and Granny escape from the wolf and Red gives the cake to Granny. On the other hand, Thespian will have a detailed model that not only includes these events, but also includes the characters' conversations, and other actions such as moving around.

Identify Gaps in a Plan When a plan is passed to Thespian, no special tag is needed to indicate whether it is a plot level plan. Instead, the system uses automated procedures to find out if there are moment-to-moment interactions missing from the plan. In other words, whether there is a "gap" in the plan. If the answer is yes, the next step is to determine what actions should be inserted into the plan and where they should be inserted. This information is returned to the author as feedback. These steps may need to be repeated multiple times until all the gaps in the plan are filled. Algorithm 2 is used for locating the first gap in a plan.

Algorithm 2 Identify_Gap($plan$)

```
1: if Fit($plan$[1:len($plan$)]) then
2:     return -1
3: else
4:     for $i$=1:len($plan$) do
5:         if Fit($plan$[1:$i$]) then
6:             break
7: return $i$
```

Algorithm 3 Fit(seq,$fixedgoals$=[ ])

```
1: for each $character$ in $story$ do
2:     if Fit_Sequence($S_0$, $character$,$seq$,$fixedgoals$) ==
       $false$ then
3:         return $false$
4: return $true$
```

Algorithm 2 first tries to treat the entire plan as a regular story path. If virtual characters can be successfully fitted to the plan, it is suggested that the planner models the story at the same level as Thespian, and no further actions need to be taken. Otherwise, this algorithm progresses stepwise to find the first gap in the plan. Starting with i equals to 1, it fits the virtual characters to the first i actions in the plan. If it succeeds, it fits the virtual characters to the first $i + 1$ actions in the plan. When fitting fails, we know that there is a gap between the i^{th} action and the $i + 1^{th}$ action in the plan, and additional actions need to be inserted either between these two actions or before the i^{th} action. The **Fit** function used in Algorithm 2 is defined in Algorithm 3. It fits all the characters in the story to a sequence of actions, and only returns true if all the characters can be successfully fitted.

Fill Gap with Actions Existing in the Model When a gap occurs in the plan and needs to be filled with moment-to-moment interactions, usually the solution is not limited to a unique one. For example, the plan may specify a plot level description of the story as: Red meets the wolf in the wood, and then Red tells the wolf that she is on her way to Granny's house ... When fitting Thespian agents to this plot level plan, a gap is found between the two actions; because Red and the wolf are strangers when they met, Red will not tell the wolf so much information about herself. The simplest way to fill the gap is to add small talk between Red and the wolf. The small talk will gradually build rapport between the two characters. Alternatively, more complex stories can be made, such as a wood-cutter happens to pass by and he convinces Red the wolf is a good friend. Different ways of filling the gap shapes the resulting story path and the Thespian agents, which will be used to interact with the user, differently.

Upon the identification of a gap, the system automatically divides virtual characters' actions into three categories based on how much the action can potentially change the story when used for filling the gap. The first category contains small talk actions that involve only the characters related to the gap. The characters related to the gap are those who act right before or after the gap, e.g. Red and the wolf in the previous example. The second category contains these characters' other actions, such as talking to other characters and moving around. The third category contains other characters' actions, such as the wood-cutter's actions. Adding small talk actions between related characters affect the story least, because small talk is most likely to be omitted when modeling a story at an abstract level; and it is assumed that unrelated characters' actions can potentially change the story most dramatically. The author is given freedom to specify which categories of actions to use for filling the gap, and their priorities. In addition to what actions can be inserted, the location of the inserted interactions, e.g. within the gap or right before the gap vs. several steps before the gap, and the length of the inserted interactions also affect the difference between the resulting story path and the original plot level plan. Algorithm 4 & 5 give the pseudo code for filling gaps based on the parameters set by the author.

As shown in Algorithm 4, the plot level plan is first cut into three parts. $Path0$ contains the sequence of actions before the gap. $Islands$ contains the two actions around the gap and the n actions that immediately precede the gap. The basic idea is to replace $islands$ with detailed moment-to-moment interactions; all the actions in islands need to be included in the final story path with their original order kept. This way, moment-to-moment interactions are inserted between the events in $islands$. Finally, the third part of the plan is the sequence of actions that happen after the gap. Currently they do not affect the gap filling process.

The author can indicate the sets of actions to be

Algorithm 4 Fill_Gap($plan$, i, $allActionSets$, n, $maxL$)

```
1: $i$: location of the gap in plan
2: $allActionSets$: sets of actions to be used. The sets are
   ordered in descending priorities.
3: $n$: the starting location for filling
4: $maxL$: maximum length of interaction allowed for fill-
   ing the gap
5:
6: $res \leftarrow false$
7: $path0 \leftarrow plan[0, i-n]$
8: $islands \leftarrow plan[i-n, i+2]$
9: $actionSet \leftarrow allActionSets[0]$
10:
11: $res \leftarrow$ RI $(path0, islands, actionSet, maxL)$
12: if $res$ == $false$ then
13:     for $newActions$ in $allActionSets[1:]$ do
14:         $actionSet \leftarrow actionSet + newActions$
15:         $res \leftarrow$ RI $(path0, islands, actionSet, maxL)$
16:         if $res$ == $true$ then
17:             break
18: return $res$
```

Algorithm 5 RI($path0, islands, actionSet, maxL$)

```
1: $res \leftarrow false$
2: for $action$ in $actionSet$ do
3:     $path \leftarrow$ copy($path0$)
4:     $path \leftarrow path + action$
5:     if checkOrder($path, islands$) then
6:         if checkComplete($path, islands$) then
7:             $res \leftarrow$ Fit ($path$)
8:             if $res$ == $true$ then
9:                 return $true$
10:        else
11:            $maxL \leftarrow maxL$ -1
12:            if $maxLength \geq 0$ then
13:                $res \leftarrow$ RI $(path, islands, actionSet, maxL)$
14:                if $res$ == $true$ then
15:                    return $res$
16: return $res$
17:
18: checkOrder($path, islands$): returns if the order of ac-
    tions in $islands$ is retained in $path$.
19: checkComplete($path, islands$): returns if each action
    in $islands$ is included in $path$
```

considered for filling the gap with priorities using the $allActionSets$ parameter. Initially only the set with highest priority is used (line 9 in Algorithm 4). If it fails to fill the gap, the set of actions with next highest priority will be added for consideration (line 14-15 in Algorithm 4).

Algorithm 5 illustrates how actions are taken from the allowed action sets ($actionSets$) and combined together to replace $islands$ in the plan. As actions are appended one by one to the end of $Path0$ (line 4 in Algorithm 5), the function keeps on checking if the story path satisfies the basic requirements for replacement - all actions in $islands$ are included in the story path with their original order (line 5-6 in Algorithm 5). If the story path passes this checking, the function will try to fit virtual characters to the story path (line 7 in Algorithm 5). If fitting succeeds, the gap is successfully filled with moment-to-moment interactions. In the worst case, this recursive function will try all the combinations of actions from $actionSets$ where the length of the sequence is equal to or less than $maxL$.

Fill Gap by Suggesting New Actions to be Created It is possible that Algorithm 4 fails to fill the gap using the parameters specified by the author. In this case, just using actions that already exist in Thespian's model is not enough to recreate the story laid out by the planner. This is not bad news for the author though. The "suggest" function can now be used to identify necessary belief changes to enable the Thespian agents to follow the plot level plan. This process often generates many interesting and creative ideas. A few examples are given in the next section.

Example

In this section, two examples of using this new framework to author the Red Riding Hood story are provided. We omit demonstrating the step of generating plot level plans, since it is a standard procedure to use a POP planner to generate such plans. Both examples start with the plot level plan and demonstrate how the system interacts with the author to configure Thespian agents.

Example 1

This example demonstrates how small talk actions can be used to turn a plot level plan into a complete story path.

In this example, the planner produces the following plan and hands it to Thespian.

1. the wolf comes to Red (on the road)
2. Wolf: where are you going?
3. Red: I am going to Granny's house to give her this cake.
4. ...

The system applied Algorithm 2 and found a gap between the wolf's enquiry and Red's reply. Next, the system tried to fill the gap using only small talk actions and succeed. The following story path was generated and returned to the author as feedback. The Thespian agents' motivations were also turned to this story path.

1. the wolf comes to Red (on the road)
2. Wolf: hello!
3. Red: hello!
4. Wolf: how are you?
5. Red: I am doing good.
6. Wolf: where are you going?
7. Red: I am going to Granny's house to give her this cake.

If the author approves the story path is reasonable, the system will proceed to identify the next gap in the plan. Otherwise, the author can always ask the system for more solutions, design new story paths, or even change the planner's model or Thespian's model of the story.

Example 2

This example demonstrates how belief changes in characters are suggested for linking plot points in the plan; and how these belief changes can be tuned into creative ideas for new actions in the story.

The plot level plan indicates that the following scenario should happen after the wolf eats Granny in the cottage.

1. Red comes to the door.
2. Red enters the cottage.
3. the wolf eats Red.

The system applied Algorithm 2 and found there is a gap between "Red enters the cottage" and the previous step if Red knows the wolf is inside. The system then tried to fill the gap using all the actions that exist in Thespian's model and found only one solution: the wolf leaves the cottage before Red enters. However, the author disapproves this solution because they foresee that it will make the later action - the wolf eats Red (in the cottage) - impossible to happen. Next, the system called the "suggest" function, which returns the following results:

- Red believes that it is Granny who is inside
- believes that the wolf is somewhere else
- Red believe that the wolf is full
- Red believes that the wolf thinks she is dead (the logic behind is that the wolf will not eat a dead person)

Each of these belief changes can make Red enter the cottage. The author needs to design new actions that can reach any of these effects. The system's suggestions ideally spur the author's creativity. In our case, we designed several new actions, such as "the wolf disguises as Granny", "the wolf pretends he is leaving" and "the wolf eats food in Granny's kitchen". Following is one possible final story path the system generated after adding all these new actions to Thespian's model of the story:

1. the wolf *disguises* as Granny.
2. Red comes to the door.
3. Red enters the cottage.
4. the wolf eats Red.

Discussion and Future Work

In this paper, we present a new framework that integrates both story-centric and character-centric designs to facilitate authoring interactive dramas. With this new framework, the author starts by using a planner to sketch a plot level plan of the story. The system then works with the author to expand it into a full story path, as shown in the examples. The system first tries to fill in moment-to-moment interactions between plot points. If this is not feasible or the author does not like the suggested solutions, the system will propose new actions to be modeled in the story. The author needs to respond by inventing some of the actions or changing the design (the plot level plan) of the story. There may be several rounds of interactions between the author and the system. Finally, Thespian agents "learn" their motivations from the final story path and any other paths designed previously by the author or the system, and interact with the users.

This framework is designed to foster the author's creativity in two ways. First, it acts as a *coach* to the author. It encourages the author to think about the story at different levels of abstraction (plot level vs. moment-to-moment interaction level) and different perspectives (overall story structure vs. motivated characters). Secondly, this new framework also works as a *colleague* and designs the story together with the author. It turns a skeleton of the story designed by the author into a story path with full details, gives feedback to the author's designs, and suggests new character actions to be included in the model of the story.

We are currently planning on future work in two directions. One direction is to perform formal evaluations for this framework to compare the authors' subjective experience and the actual qualities of the interactive drama when using our framework to other authoring approaches, e.g. using a planner alone. The knowledge we gain in these studies will give us a better understanding of people's cognitive process in creative works, and help us to refine the framework in the future.

We will also improve the algorithms used for integrating a partial order planner with the Thespian framework. Currently, we treat each gap in the plan independently. When filling a later gap, we assume all the gaps that happen earlier have already been filled and a normal story path exists before the gap. However, when a plan contains multiple gaps, certain solutions to an earlier gap may make the filling of a later gap infeasible. Similarly, in a larger scale where multiple plot level plans are provided, solutions for gaps in one plan can affect the feasibility of solutions for gaps in another plan. We will address this problem by developing a heuristic for updating the priorities of solutions for filling each gap based on their influences on successfully filling other gaps.

References

Braun, N. 2003. Storytelling in collaborative augmented reality environments. In *Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision.*

Cavazza, M.; Charles, F.; and Mead, S. J. 2001. Agents' interaction in virtual storytelling. In *Proceedings of the International WorkShop on Intelligent Virtual Agents*, 156–170.

Egri, L. 1949. *The Art of Dramatic Writing.* New York: Simon & Schuster.

Louchart, S., and Aylett, R. 2004. The emergent narrative theoretical investigation. In *the 2004 Conference on Narrative and Interactive Learning Environments.*

Lubart, T. 2005. How can computers be partners in the creative process: Classification and commentary. *International Journal of Human-Computer Studies* 63(The Special Issue).

Magerko, B. 2005. Story representation and interactive drama. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE).*

Marsella, S. C.; Pynadath, D. V.; and Read, S. J. 2004. PsychSim: Agent-based modeling of social interactions and influence. In *Proceedings of the International Conference on Cognitive Modeling*, 243–248.

Mateas, M., and Stern, A. 2003. Integrating plot, character and natural language processing in the interactive drama Façade. In *the International Conference on Technologies for Interactive Digital Storytelling and Entertainment.*

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

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

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21:1071–1088.

Szilas, N. 2003. IDtension: a narrative engine for interactive drama. In *the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment.*

Traum, D. R.; Swartout, W.; Marsella, S. C.; and Gratch, J. 2005. Fight, flight, or negotiate: Believable strategies for conversing under crisis. In *IVA.*

Young, R. M.; Riedl, M. O.; Branly, M.; Jhala, A. H.; Martin, R. J.; and Saretto, C. J. 2004. An architecture for integrating plan-based behavior generation with interactive game environments. In *Journal of Game Development.*