Thespian:
Using Multi-Agent Fitting to Craft Interactive Drama

Mei Si, Stacy C. Marsella, and David V. Pynadath
Information Sciences Institute
University of Southern California
Los Angeles, CA 90292
{meisi,marsella,pynadath}@isi.edu

ABSTRACT

There has been a growing interest in designing multi-agent based interactive dramas. A key research challenge faced in the design of these systems is to support open-ended user interaction while ensuring dramatic user experiences and consistent character personalities. Autonomous agents with reactive and planning abilities are well suited for realizing characters that both adapt to user interactions and are consistent with their own goals. However, agents are often created manually and with extensive programming effort, that excludes authoring by non-technical authors. Thespian is a framework for realizing interactive drama that seeks to reduce programming effort. To start, an author provides linear scripts of the drama. An automated fitting algorithm then configures agents to behave according to the scripts via automated tuning of goal parameters. This capability allows authors to design in a familiar way by writing scripts. Thespian also supports reuse of characters and story elements. Given these advantages, new scenarios can be developed with less programming effort. We discuss the use of Thespian in fitting characters in the Tactical Language Training System and in a Grimms' fairy tale. We also present preliminary experiments on migrating characters between these stories.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms

Keywords

pedagogical agents, authoring tools

1. INTRODUCTION

There has been a growing research interest in the design of interactive dramas that allow a user to actively participate in a dynamically unfolding story (e.g. [4, 20, 9, 15, 2]). The user can play the role of a main character, can have a supporting role, or can have a more indirect, directorial influence on the story. Interactive dramas obviously have a long tradition in the commercial gaming world (e.g. role playing games) as a form of entertainment but also have been used as a pedagogical tool (e.g. interactive pedagogical drama [10]). In addition, there is now a considerable body of work in artificial intelligence and multi-agent systems addressing the many research challenges raised by such applications, including modeling engaging virtual characters [14] that have personality [16], that act emotionally [6], and that can interact with users using spoken natural language [19, 1].

A key research challenge to the designers of these systems arises from the desire to support open-ended user interaction and the consequent variability in how the story unfolds. The behavior of the characters in the story must be both responsive to the user and consistent with their own internal motivations. Further, the designer typically wants to enforce constraints on the ways the story can unfold (for example, that certain dramatic goals should be achieved). To create responsive and motivationally consistent characters, researchers have often used autonomous agents with reactive and planning abilities that adapt to user interactions while still being consistent with their own goals. However, this agent-based approach raises the new question of how to craft agents to a particular interactive experience or drama.

Although researchers have proposed a range of agent-based approaches to craft character and story, it has often turned out to be an undertaking that requires extensive programmer efforts. This tends to exclude non-technical authors. In the MRE system [18], a linear script was analyzed to inform the design of a team task model that motivated the agent-character's behavior and dialog with the user. Skilled programmers crafted this model by hand. In Façade [12], the story is organized around dramatic beats, realized as brief patterns of interactions between characters. The user influences how the beat progresses and how beats are strung together. Again, it was up to the programmers to design these beats and determine the user's influence on their progressions. In contrast, Gebhard [5] approaches the authoring process by trying to build authoring tools that lessen the need for programming skills.

The time-consuming and complex hand authoring process can be a serious design bottleneck. Furthermore, existing systems make writing scripts for interactive drama very different from writing standard stories. To account for interactivity, the author has to consider variations of user input and design reasonable responses for each of them. It is much harder for a human author to reason across all of the possible contingencies, rather than simply creating sequences of actions, as in traditional drama.

In this paper, we present Thespian, a framework for realizing

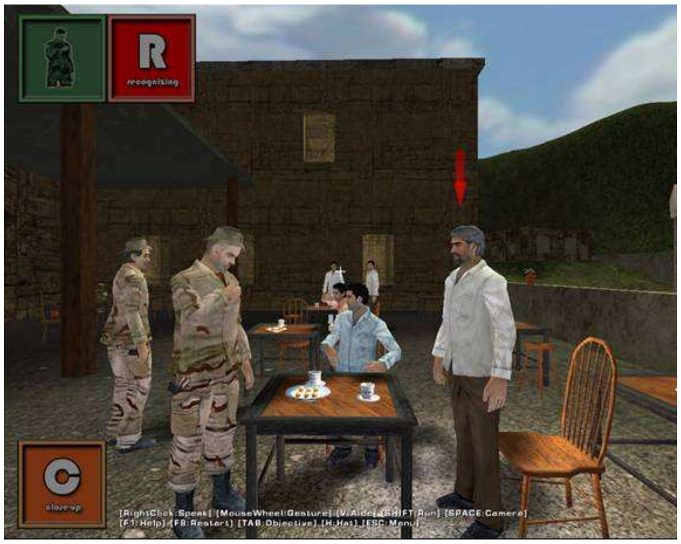
R
recognizing
C
close-up
[RightClick:Speak] [MouseWheel:Gesture] [V:Aide] [SHIFT:Run] [SPACE:Camera]
[F1:Help] [F8:Restart] [TAB:Objective] [H:Hat] [ESC:Menu]

Figure 1: A screen-shot from the Tactical Language Training System

Speaker	Addressee	Utterance
Student	Old man	*student-enquiry-to-oldman-about-name-of-leader* Who is the most important official in this town?
Young man	Old man	*youngman-slowdown-to-oldman* Slow down!
Young man	Student	*youngman-enquiry-to-student-about-identity* Who are you?
Old man	Student	(silence)
Student	Young man	*student-inform-to-youngman-about-identity* We are Americans.
Young man	Student	*youngman-enquiry-to-student-about-identity* CIA?
Student	Young man	*student-inform-to-youngman-about-identity* No, sir, we are from the American Army, Special Forces.

Figure 2: An excerpt from scene one

interactive drama that seeks to use automation to transform authoring from an onerous programming burden to a creative exercise. Thespian approaches interactive drama construction as a process of training agents to perform their roles. The author of an interactive drama starts by creating alternative linear scripts of the desired paths of the story, and specifying the story environment (how the agents' actions affect the status of the world and the agents' beliefs). Using the scripts as constraints on agents' behaviors, a fitting process tunes agents' parameters so that they behave according to the scripts. This capability allows the author to design interactive drama in a more familiar way by creating linear scripts or script fragments. Nevertheless, because the autonomous agents are self-motivated, and have personalities tuned to their specific roles in the story, they can be responsive to more open-ended interaction. The approach also supports reuse of characters (fitted agents) and story environment elements across stories. This reduces programmer effort and allows an author to creatively explore the effect of transposing characters between stories. We discuss several experiments, including fitting characters in stories of two different types, the stories used in the Tactical Language Training System (TactLang)[8], a military language training system, and the fisherman and his wife, a Grimms' fairy tale; and moving characters between these stories. Finally, we conclude with a discussion of various open issues.

2. EXAMPLE DOMAIN

The first interactive drama to incorporate Thespian is the Mission Environment of TactLang. TactLang is designed to teach the user a foreign language and cultural awareness. The user takes on the role of a male army sergeant who is assigned to conduct a civil affairs mission in a foreign (e.g., Lebanese, Iraqi) town. TactLang uses a 3D virtual world built on top of the Unreal Engine. The human user navigates in the virtual world and interacts with virtual characters using spoken Arabic and gestures. An automated speech recognizer identifies the utterance and converts it into a speech-act representation that Thespian takes as input. Output from Thespian consists of speech-acts and actions that control the virtual characters.

The story in TactLang consists of multiple scenes. We will use the first scene throughout the paper to illustrate how Thespian is used. The story begins in a village café. The user's aim in the scene is to find the senior official in the town to discuss providing aid. There are a variety of actions he can perform including moving around the town, greeting people, introducing himself, asking questions, using gestures etc. The user interacts with a range of characters in the scene, most notedly an old man and a young man. These two locals have different personalities. The old man is very cooperative. The young man worries mostly about the safety of the town, and may accuse the sergeant of being a CIA agent if the user does not establish trust. Figure 2 shows an excerpt from scene one.

3. THESPIAN

In Thespian, each character in a drama is controlled by a separate agent. A human user can replace any of the characters and interact with the others. Thespian is built upon PsychSim [11], a multi-agent system for social simulation based on Partially Observable Markov Decision Problems (POMDPs) [17]. Each agent reasons about how its actions affect its state, the state of other agents, and the environment. This reasoning occurs within the agent's subjective view of the world, including its beliefs about other agents and *their* subjective views of the world, a form of recursive agent modeling [7]. Each agent has its own goals, expressed as a reward function over the various state features it seeks to increase (e.g. safety). PsychSim agents have a policy of selecting actions according to either (1) reactive rules (e.g. "if people greet you, greet them back"), (2) bounded optimality, determined by limited lookahead (e.g. "answering the question will best achieve my goals in the near future"), or (3) a mixture of these reactive and deliberative policies.

PsychSim makes a natural basis for Thespian for a number of reasons. First, it supports multiple interacting agents with their own individual goals. Second, the subjective views that PsychSim agents have of the world, each other, and even of themselves, gives them a "theory of mind" that supports autonomous reasoning about social interactions. But the flexible and quantitative nature of POMDP-based representation can also make it hard to design agents. Fortunately, PsychSim provides us with a third benefit in the form of an automatic procedure for translating desired behavior into the goals needed for the agent to autonomously select that

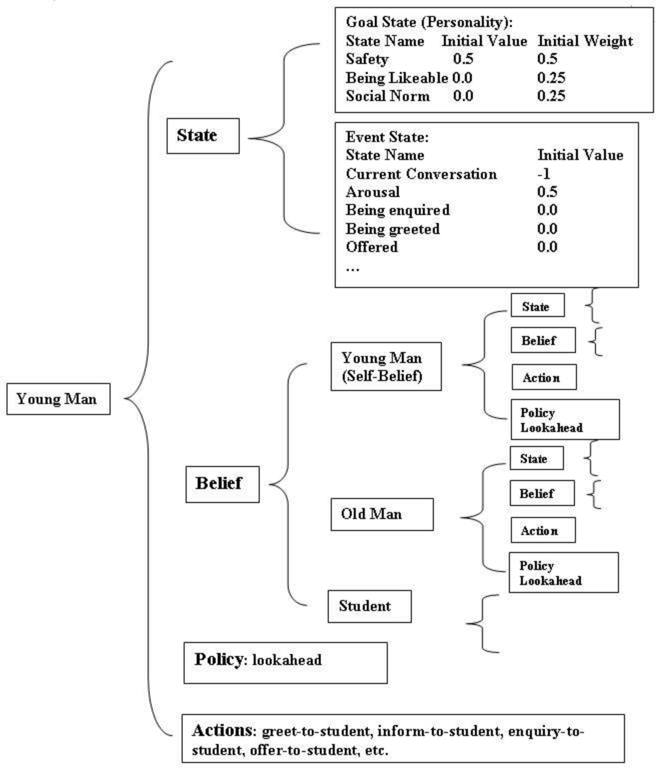

Figure 3: An example of agent structure

behavior [13]. This mechanism compiles the agent's policy of behavior into an invertible piecewise linear function of its goals. We can thus translate desirable behaviors (e.g. a partial script) into constraints on goals, potentially supporting the automatic configuration of characters.

Using PsychSim agents as our base architecture, we designed and implemented Thespian as a multi-agent framework to support authors in building interactive drama. Thespian extends and specializes PsychSim in several ways. It models multi-party conversational turn-taking as part of the interaction among characters. In addition, Thespian's architecture supports character models that are independent from any particular story and therefore suggests reusability of existing characters and story environments. Authors do not require programming skills to combine/modify existing characters and story environments, to explore new story paths and build new stories. Thespian agents are comprised of the following components: a set of goal state features, a set of event state features, policies, available actions, and recursive beliefs about itself and other characters appearing in the same story. Figure 3 shows an example of a typical agent, the young man character in the TactLang story. This example will be used through this section to illustrate the novel components of Thespian that enable its authoring capability.

3.1 Agent State

Thespian includes additional agent structure that supports some common distinctions made in story construction. We distinguish two kinds of state features: *goal states* and *event states*.

As shown in Figure 3, the young man character has three goal state features and several event state features. Goal states represent an agent's progress in achieving a particular goal, reaching an upper limit when it is satisfied. For example, the young man's initial level of safety is set to 0.5, while the upper limit for all goal state features is 1.0; therefore, the goal of safety is completely satisfied once its value reaches 1.0. Currently, in Thespian we have only one type of goal, which is to maximize or minimize a goal state variable. For example, the young man has three goals, which are to maximize its safety level, to maximize the level of being likable, and to maximize the level of following social norms; their initial weights are 0.5, 0.25, and 0.25 respectively.

Event states model environmental features that may not have any direct relation to the agent's goals, but which nevertheless are relevant to the agent's reasoning (e.g. by influencing the responses of other agents). For example, the state *being greeted* keeps track of whether the agent has not yet replied to a character's greeting. The young man character does not have a goal to maximize or minimize this state feature, but the value of this feature is used in the agent's reasoning process. Event states are often story-specific, so we do not consider them as part of the agent when reusing a character (as described in Section 4.3).

3.2 Acting Sequence

In dramas, as in real life, characters do not necessarily act or speak in turn and may also hold the turn over multiple steps. Therefore, we designed a flexible turn-taking mechanism based on arousal. The most aroused agent will get the next turn if it wants to speak. We currently keep track of each agent's arousal level in a state variable, and model the dynamics of it as a function of events; certain events can arouse the agent to seize the dialog turn. For example, the fact that the user tries to ask potentially sensitive information about the town without introducing himself properly drives the young man's arousal level high enough so that he takes the dialog turn from the old man and starts making accusations. We plan to improve this arousal model to take an agent's lookahead ability into account. Thus, an agent's projection into the future will affect its decision to seize the current turn.

3.3 Agent Policy and Beliefs about Policy

In Thespian, all agents use a lookahead policy. When an agent selects its next action, it considers the response of other agents to each candidate action, and its own reaction to other agents' responses in the future. Since characters act one at a time, agents need to reason about turn-taking in their lookahead processes. Furthermore, in their subjective view of the world, they model other agents by using a lookahead policy to decide their next actions. Therefore, agents' behaviors are directly driven by their personalities; no story specific reactive rules need to be handcrafted by the author.

Theoretically each agent can perform lookahead for large enough number of steps until there is no gain for itself and other agents. For performance reasons, we limit the lookahead to a finite horizon that we determine to be sufficiently realistic without incurring too much computational overhead (e.g., for the examples in this paper, the horizon is the number of steps required for the agent doing the lookahead to get two more turns).

3.4 Character Personality and Fitting

Thespian also includes a special agent construct to represent the *personality profiles* of characters. We model a character's personality profile as its various goals and their relative importance (weight). One critical benefit of defining characters in terms of only goals is that we can now adapt the original PsychSim fitting algorithm to determine a character's personality from only its actions observed in the story.

PsychSim's fitting process simulates an agent's lookahead process, and automatically calculates constraints on goal weights. The constraints ensure that the desired action receives highest reward among all candidate actions. In the original PsychSim algorithm, a fixed order of turn-taking is assumed in agent's lookahead processes. In Thespian, we modified the original algorithm to reason about the dynamic, arousal-based action sequence. In our new algorithm, the agent's lookahead process reasons about all other agents' (including its own) reactions to its behavior only after first reasoning about which agent will take the next turn. In other words, we incorporate our model of the dynamics of arousal levels into the piecewise linear model of the overall lookahead process. In addition, our character fitting algorithm supports input in the form of at least one sequence of actions provided by the author to represent a "preferred" path of the story.

Algorithm 1 FIT-SEQUENCE(S_0, $char\_name$, seq)

```
 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: $C \leftarrow$ [] : constraint on goal weights
 5: $S \leftarrow S_0$
 6: for each action $A$ in $seq$ do
 7:    # update state
 8:    $S \leftarrow S \times$ Dynamics($action$)
 9:    if $A.actor$ = $char\_name$ then
10:       # adding constraints
11:       for each action $a$ in $possible\_actions$ do
12:          $new\_C \leftarrow$ Reward($A$,$S$) $\geq$ Reward($a$,$S$)
13:          $C$.Append($new\_C$)
14: return $C$
15: Reward(action,state) calculated similar to PsychSim, with
    modifications to take turn-taking into account
16: Dynamics(action) as defined in PsychSim
```

For each story path, Thespian proceeds iteratively, fitting the goals of one agent at a time, 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 6). If the current action is performed by this character (Step 9), constraints are imposed on the character's goal weights that ensure the action is preferred over alternative actions available to the character (Step 12). Thus each action in the story path effectively eliminates any goal weight values that would cause the agent to choose some other action instead.

When there are multiple story paths to fit, the constraints resulting from fitting each path can be merged into one common constraint set. Typically, there are multiple candidate goal weight values that are consistent with the preferred story path. Any one of these solutions guarantees that the characters will follow the preferred paths of the story. Thespian can pick one of these solutions according to its own heuristics, but we also give the author the option of manually selecting one from this constrained set, if desired. By allowing this potential interaction, Thespian can exploit the author's deeper understanding of the character, while still providing automated support in the form of constraining the set of goals left to choose from.

4. AUTHORING

The previous section describes Thespian's agent structure and the automatic fitting algorithm that enables agents to follow desired story paths. This section describes Thespian's authoring process.

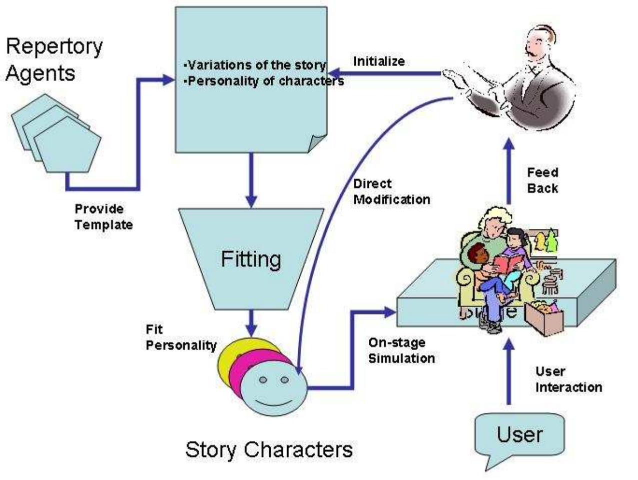

Figure 4: Thespian Authoring Process

Thespian helps to reduce authoring effort in two ways. First, its fitting process automates the authoring of characters. Second, Thespian has been designed so that the elements that comprise a story (its characters, how actions are defined, etc.) can be reused in new stories. This section discusses character authoring and reusability in Thespian.

4.1 Building Story Environment

A story environment consists of dynamics functions that define how characters' actions can affect the world. The process of building a new story environment can be broken down into three steps. The first step involves abstracting a list of possible actions each character can perform, and a list of relevant goals. Currently this step has to be done by hand. The second step is to check each pair of goal and action to determine whether Thespian has a dynamics function already defined. If the existing dynamics function is suitable for the new story, then the author can reuse it directly . If not, the third step of writing a new dynamics function (or editing the existing one) becomes necessary.

To aid authors in this process, an editor with action ontology is under development. In steps one and two, the editor should allow the author to choose from only a relevant list. For step three, the editor should prompt the author for parameters and generate the function automatically.

At present, building a story environment still requires some programmer effort. However, once a story environment is built, authoring can be done easily by a non-technical person.

4.2 Character Authoring

Figure 4 depicts Thespian's authoring process. In step one, the author sets the initial conditions for all of the characters. In particular, for each character, the author chooses, from a group of repertory agents, a repertory agent whose personality is closest to the desired character. The author then modifies these repertory agents by adjusting the goals and goal weights for each character[1] to make them closer to the characters' roles in the story. For example, as shown in Figure 3 an author can set the initial goal weights of the

[1]This step is not necessary for fitting the character itself since the fitting algorithms will set the character's goal preferences. But these initial values will be used to set the initial beliefs other characters have about this character.

young man to be 0.5, 0.25, and 0.25 respectively for safety, following social norms and being likable. By default, each character's belief about the world, its own state, and other characters' state is accurate. This belief can also be modified depending on the story.

In step two, the author defines desired paths for the story. This step is like writing one or more variations of the story, each of which follows a sequential path. Each path consists of a sequence of actions that should ideally happen in the story. Figure 2 shows an example of such a path with corresponding speech acts. We are currently developing a structured editing tool to facilitate mapping the dialog of a script into speech-acts, propositional content, and actions.

In step three, the automated fitting process uses these paths as guidelines for configuring characters' personalities. Specifically, the fitting process takes as input the initial values set in step one along with the paths defined in step two. The output is a set of suggestions on how to modify the initial goal weights (i.e. a refined personality). The fitting process may indicate the character could not be fitted (i.e. not all the constraints on its goal weights can be satisfied simultaneously). This can be due to flaws in the story or the script. Either way, the author will need to either adjust the story environment or alter the scripts. The final step involves repeating the third step with new changes to the characters' personalities taken into consideration. A character's goal weights after fitting may be very different from their initial values in step one. This difference can lead to discrepancies between a character's actual personalities and another character's mental model of that character. To synchronize the models, step three could be repeated with the other agent's belief set to the actual personalities. However, characters do not necessarily have to have accurate knowledge about other characters or themselves, in fact it can be dramatically interesting if they do not. In section 5, results of fitting various characters are presented, including fitting of the young man character.

Because the agents have personalities fitted to their roles in the story, unexpected user interaction (different from the given paths) will ideally be handled by the characters in a motivated fashion. However, if testing reveals a story path where the behaviors of the characters is undesirable, the author needs to modify the undesired paths into a desired one and add that path to the fitting process to refine further the characters' behaviors.

4.3 Reusability

Reuse of characters and story environment elements is an important feature of Thespian. The benefit of reusing characters is three-fold. First, it saves development time when creating similar characters in new stories. Second, by swapping new characters into a story, the author can creatively explore possible paths of a story. This feature is especially useful when the author knows the desired characters but is not fully clear about the storyline. Third, swapping characters as stand-ins for different types of users can test the robustness of the story. Reusing existing story environment elements can significantly save development time of setting up a new story environment. It especially helps non-technical authors to build new stories, as they can use elements from any existing story environment (if applicable) instead of writing their own. Reusing story environment elements is straightforward. This section will focus on how characters can be reused.

After fitting, an agent becomes a character with a certain personality. This character can be easily plugged into other stories to play a similar role. This is made possible because the character's personality is defined solely in terms of goals and therefore can be manipulated independent of story environment, and independent of possible actions the character can carry out.

4.3.1 Transferring Characters to New Environment

The main issue for transferring a character to a new story environment is that not all of this character's goals may be relevant in the new environment. If the new story environment does not have actions related to a goal, satisfying it is impossible.

We can address this by applying one of the following methods. One is to simply drop the goal that is not supported in the new story. The remaining goals have the same relative weights as in the original character, and therefore, the new character can potentially be perceived as having a very similar personality as the original character. The second method is to replace the goal with a similar one that is supported by the story. This method can also potentially preserve the character's original personality. [2] The third method is to include the necessary actions and dynamics functions in the new story for supporting the goal, which would tend to fundamentally transform the story.

4.3.2 Creating New Characters

To make characters more universally applicable, we can combine characters that do not have conflicting personalities into one single character. This helps to build characters with richer capabilities and personalities. There are two ways to combine existing characters. One is to directly put goals from different characters together. This method is simple and leaves the relative goal weights of the original characters unchanged. Furthermore, while combining the goals, we can control which original character's personality will dominate. For example, we can create a new character by summing the old man's goals weighted at 0.7 and the fish's (from the Grimms' fairy tale) goals at 0.3. Section 5.3 gives the details of this example. The problem with this method is that it is unclear how to weigh any goals shared by the existing characters, but given different weight by the two. Alternatively, we can do incremental fitting of an agent over multiple stories. This process combines the constraints generated by the separate stories, and finds a solution that satisfies all of them. Characters in these stories must possess non-conflicting personalities and a common subset of goals, because the purpose of incremental fitting is to decide the relative weights of these goals. Because the intermediate results of fitting usually give a set of solutions instead of one, this method has a higher likelihood of combining characters. However, unlike the previous method, the resulting character cannot be manipulated by explicit weighting personalities of original characters. Therefore, its personality is not as predictable as using the first method.

5. EXPERIMENTS

To date, Thespian has been used to model the first three scenes from the pedagogical drama used in TactLang and a Grimms' fairy tale, "The Fisherman and his Wife" (the Fish Story). The TactLang stories and the Fish story are chosen because they are of different types. One is realism, and the other is fantasy. Furthermore, they share few speech acts. Having successfully fit these stories, we also conducted preliminary experiments with Thespian to explore reusability of characters. These experiments are also examples for how non-technical authors can easily modify existing stories and characters. The fitting and reuse experiments are presented below.

[2] In our examples, we pick characters' goals only from the goal taxonomy hierarchy created by Chulef et al.[3]. This hierarchy is based on a summary of common human goals. By determining a distance metric on these goals, the taxonomy organized the goals into a seven-level hierarchical structure. Judging whether two goals are similar is trivial if all goals are drawn from this hierarchy.

5.1 Fitting the first scene of the TactLang Story

We had two variations of scene one of the TactLang story. In one version, the user is appropriately polite and respectful in his behavior, such as, greeting local people in a proper manner and introducing himself before asking a question. As a consequence, the local people, the young man and the old man, are helpful. The other variation involves the user failing to do one of the acts, for example, not introducing himself properly, and resulting in the young man making accusations against the user.

We identified that both the old man and the young man have the following goals: being likable, following social norm, and safety. The fitting process results in the following goal weights for the old man: [following social norm: 0.5; being likable: 0.1; safety: 0.5]. In other words, he is a cooperative person who is willing to assist the user in his investigations by answering questions posed to him. The young man on the other hand, ends up with high goal weights on safety (1.0), and low goal weights on the other two (both 0.05). As a result, he is guarded in his approach. He does not trust the user and suspects that the user is a possible enemy.

The story has been tested under a range of variations of possible user behavior, including changing from very rude behaviors to over-polite. In these tests, characters behave consistently with their personalities. The consequence of the fitting is an interactive story that generalizes the original story and variations whereby any key failures to be polite will result in non-cooperation of local people.

After fitting characters in the first scene, we used the same procedure to fit characters in scene two and three. The story environments of these two scenes are very similar to that of scene one. Therefore, the authoring processes of these two scenes was trivial. We were able to use most of the dynamics functions defined in scene one to build their story environments. We just added a few new dynamics functions to support new actions appearing in these two scenes, specified the desired story path for these two scenes, and finally run the fitting process.

5.2 Fitting the Grimms' fairy tale

This fable has three characters, the fisherman, his wife, and an enchanted fish. The story begins with the fisherman catching the fish but letting it go, without asking for anything in return. But his greedy wife keeps asking for more and more from the fish who out of gratitude to the fisherman keeps granting the wife's wishes. It ends when the wife wishes to be God and the fish takes back all that was given.

Fitting for this story leads to the following weights:
Fish: [safety: 0.8; seeking fairness: 0.2]
Fisherman: [better physical condition: 0.05; safety: 0.05; belongingness and love: 0.8; esteem: 0.2; being better than others: 0.05]
Wife: [better physical condition: 0.8; safety: 0.8; belongingness and love: 0.001; esteem: 0.001; being better than others: 0.8]
This setting of personalities recreates the original story. Testing of robustness is currently ongoing.

5.3 Reuse Experiment 1: Creating a new character from existing characters

In this experiment, we explored the possibility of combining two non-conflicting characters together by directly putting their goals and dynamics functions together. The Fish story has high-level dialog with little in the way of normal social interaction, such as greeting and thanking. In contrast, the dialog in TactLang story has these social interactions. To equip the fish character with basic social skills, we added two social-norm related goals, defined in the old man character. To support these social-norm related goals, relevant dynamics functions, state variables are also imported from the TactLang story. These two goals will drive the fish to initialize these social behaviors and respond to other characters' social contact properly. We tried to weigh the fish character's old goals and the newly added goals with different relative importance to create different behaviors. With one set of relative weights, we got an overly polite fish, who greets and introduces itself before asking the fisherman to release it, even though being out of water can endanger its life. And after being released, the fish politely thanks and bids goodbye to the fisherman. With another set of weights, in which the new goals receive lower weight, the fish asks to be released first.

5.4 Reuse Experiment 2: Moving a character to a new environment with all its goals supported

Using the same method as described in experiment 1, we first made all the characters in the Fish story capable of basic social interactions. Now, we let the old man character in TactLang story take the role of the fish in the Fish story. The result of this operation is that the fish satisfies all requests from the fisherman. This is expected, as the new fish's behavior is consistent with the old man's personality of being cooperative.

5.5 Reuse Experiment 3: Moving a character to a new environment without all its goals supported

Next, we proceed to a more complex situation: the new environment does not support all the goals of the character to be added. We applied the simplest approach, which is to drop the extra goals of the new character. We let the fisherman character take the role of young man in the TactLang story, and dropped some of the fisherman's goals that are not supported in the TactLang story. The result is the young man now does not question the sergeant's identity even when the sergeant is not completely polite. This result is expected, as the fisherman does not consider safety as an important goal.

6. DISCUSSION

We have demonstrated that fitting, Thespian's approach for configuring characters' personalities, is effective. The resulting characters can generalize the original script and respond to events that occur in a different order from the script. The reusing character experiments show that it is easy to manipulate a character's personalities at a high level. Authors can reconstruct characters without touching story details, and the resulting characters' personalities are easily controllable by the author.

There are two more methods, incremental fitting and replacing mismatched goals, suggested in the reusing characters section, but we found they are not applicable to the characters in these two stories. Because of the huge difference between the two stories, their characters' goals are highly diverged. Therefore, we could not find replacements for mismatched goals. Incremental fitting is also not applicable because it requires shared goals. We do not have experiments for demonstrating reusing story elements. But as we built the stories, we observed that reuse of dynamics functions is almost always possible. The three scenes in TactLang have similar background, and therefore share a lot of actions and goals, and we largely could use the same dynamics functions. Also, in experiment one, we could build a polite fish by directly importing dynamics functions from the TactLang story. This shows that dynamics functions can be reused even between two very different types of stories.

Transfer of characters will sometimes lead to surprising results. We once directly replaced the wife's goals with that of the fisherman. Quite unexpectedly, the new wife still kept on wishing for more stuff. On a closer examination, we realized what was happening. Being loved was a very important goal for the fisherman and when that goal was transferred to the wife, she also wanted to be loved. Getting requests accepted increases a character's sense that it is loved. Additionally, the wife modeled the fisherman as always accepting her requests. Therefore, by making requests, she expected an increased sense of being loved. This explains why the wife, with goals replaced, kept on requesting even though the fisherman himself does not make requests in the first place. All that the new wife cares about is her requests being accepted by the fisherman, not the actual granting of the reward from the fish. To prevent the wife from keeping on requesting, we adjusted the dynamics functions so that when the fisherman made requests of the fish at his wife's behest, his wife's self-esteem would be hurt. Thus, the fisherman's action to the fish would now affect the beliefs his wife has about herself, a connection that initially we did not think was necessary.

In Thespian, the author exerts control over the direction the drama can take by providing story variants to the fitting process. In contrast to this design-time control, one can also try to provide runtime management of the story, for example, to achieve a certain dramatic curve. There are two ways to achieve this functionality. One is to add a centralized director agent to monitor and moderate story tension. To enable the direction of the characters, a goal of obeying the director needs to be added. Alternatively, we could give the agents dramatic goals. This approach controls the story in a distributed fashion with no need for a separate director agent.

7. CONCLUSION

There are many forms of interactive drama and many ways to construct them. In Thespian, we are taking an approach that enlists automation to facilitate the design process. Thespian transforms and simplifies the authoring process in several ways. Instead of manually designing agents to follow some scripts, the agents' personalities, their goals, are fitted so that they are motivated to perform according to the scripts. Ideally, this will open the authoring process to non-technical designers. To facilitate this automation, the agents are motivated solely by their goals. They do not have handcrafted policies, plans or rules that guide their behavior. Instead, they use lookahead search in a decision-theoretic framework to determine how best to achieve their goals. Thus, their goals become the key determinant of their behavior and the agents ideally respond to unexpected user interaction in ways consistent with their motivations. And if they do not, the misbehavior too can be fed into the fitting process. This helps to address one of the hardest challenges facing the designer, the handling of variability in user interactions in appropriate ways. Without fitting, it can consume considerable programming effort.

This goal-centric agent model also allows Thespian to cleanly separate the agent model from story-specific knowledge. This supports reuse of both agents and story-specific knowledge across stories. Reuse has several benefits. It promises to further alleviate design effort, especially for non-technical designers. It can also be used to facilitate creative exploration; swapping characters between stories leads to often interesting variations on the story. Finally, it provides a way to do automated testing. Swapping different characters into a story to play the role of the user's character provides a way to evaluate the interactive drama's behavior.

Although Thespian's fitting and reuse capabilities can facilitate authoring, good authoring tools that exploit these capabilities are still in development. These tools will be necessary before Thespian can be easily used by non-technical users. Once these tools have been developed, a key focus in our work will be to fully evaluate Thespian's ease of authoring.

8. ACKNOWLEDGMENTS

We thank our colleagues on the Tactical Language Training Project, especially Lewis Johnson and Hannes Vilhjálmsson for their support and thoughtful discussions. This project is part of the DARWARS initiative sponsored by the US Defense Advanced Research Projects Agency (DARPA).

9. REFERENCES

[1] J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, editors. *Embodied Conversational Agents*. MIT Press, Cambridge, MA, 2000.

[2] M. Cavazza, F. Charles, and J.S. Mead. Agents' interaction in virtual storytelling. *Lecture Notes in Computer Science*, 2190:156–170, 2001.

[3] A. Chulef, S.J. Read, and D.A. Walsh. A hierarchical taxonomy of human goals. *Motivation and Emotion*, 25:191–232, 2001.

[4] T. Galyean. *Narrative Guidance of Interactivity*. PhD thesis, Media Arts and Sciences, MIT, 1995.

[5] P. Gebhard, M. Kipp, M. Klesen, and T. Rist. Authoring scenes for adaptive, interactive performances. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pp. 725–732, 2003.

[6] J. Gratch and S. Marsella. A domain-independent framework for modeling emotion. *Journal of Cognitive Systems Research*, 5:269–306, 2004.

[7] P. Gmytrasiewicz and E. Durfee. A rigorous, operational formalization of recursive modeling. In *Proceedings of the International Conference on Multi Agent Systems*, pp. 125–132, 1995.

[8] W.L. Johnson, C. Beal, A. Fowles-Winkler, S. Narayanan, D. Papachristou, S. Marsella, and H. Vilhjálmsson. Tactical Language Training System: An interim report. In *Proceedings of the International Conference on Intelligent Tutoring Sys.*, pp. 336–345, 2004.

[9] M.T. Kelso, P. Weyhrauch, and J. Bates. Dramatic presence. *Teleoperators and Virtual Environments*, 2(1), 1993.

[10] S. Marsella, W.L. Johnson, and C. Labore. Interactive pedagogical drama. In *Proceeding of the International Conference on Autonomous Agents*, pp. 301–308, 2000.

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

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

[13] D.V. Pynadath and S.C. Marsella. Fitting and compilation of multiagent models through piecewise linear functions. In *Proceedings of the International Joint Conference on*

Autonomous Agents and Multi Agent Systems, pp. 1197–1204, 2004.

[14] W.S. Reilly and J. Bates. Building emotional agents. Carnegie Mellon University, 1992. Technical Report CMU-CS-92-143.

[15] M.O. Riedl, C.J. Saretto, and R.M. Young. Managing interaction between users and agents in a multi-agent storytelling environment. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pp. 741–748, 2003.

[16] D. Rousseau and B. Hayes-Roth. Improvisational synthetic actors with flexible personalities. Knowledge Systems Laboratory, Stanford University, 1997. Technical Report KSL-97-10.

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

[18] W. Swartout, R. Hill, J. Gratch, W.L. Johnson, C. Kyriakakis, C. LaBore, R. Lindheim, S. Marsella, D. Miraglia, B. Moore, J. Morie, J. Rickel, M. Thiébaux, L. Tuch, R. Whitney, and J. Douglas. Toward the holodeck: Integrating graphics, sound, character and story. In *Proceedings of the International Conference on Autonomous Agents*, pp. 409–416, 2001.

[19] D. Traum, J. Rickel, J. Gratch., and S. Marsella. Negotiation over tasks in hybrid human-agent teams for simulation-based training. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.

[20] P. Weyhrauch. *Guiding Interactive Drama*. PhD thesis, Carnegie Mellon University, 1993. Technical Report CMU-CS-97-109.