Proactive Mediation in Plan-Based Narrative Environments

Justin Harris and R. Michael Young, *Senior Member, IEEE*

***Abstract*—In interactive plan-based narrative environments, users' actions must be monitored to ensure that conditions necessary for the execution of narrative plans are not compromised. In the Zocalo system, management of user actions has been performed on a reactionary basis by a process called mediation. In this paper, we describe an extension to this approach, *proactive mediation*, which calculates responses to user input in an anticipatory manner. A proactive mediation module accepts as input a plan describing the actions being performed by the user (generated by a plan recognition system) and identifies portions of that plan that jeopardize the causal structure of the overall narrative. Once these portions are identified, proactive mediation generates modifications to the narrative plan structure that avoid the unwanted interaction between user and story. This extension to the original mediation algorithm provides more responses to a user's actions and generates responses that are tailored to the user's activity.**

***Index Terms*—Games, intelligent systems, interactive computing.**

I. Introduction

RECENTLY, there has been an increase in the development of interactive applications, including computer games, training simulations, and intelligent tutoring software that involve a human user interacting with one or more embedded agents acting in a virtual environment. These applications often require the agents, in concert with the user, to perform coordinated sequences of novel actions structured as an unfolding story or narrative. One approach used to address the coordination of the actions within these story-based systems is the use of a centralized planning system, in which a single planner defines the actions of all agents in a narrative plan [1], [2].

In these plan-based systems, it is typical for the system to create a detailed plan for all the action within a story. This plan details the complex causal and temporal relations between all actions intended to be performed as the story plays out. Planning systems go to great lengths to reason about all possible interactions between actions in the plans they produce, guaranteeing that the plans they create (and thus the stories that they drive) will execute correctly in the story world. If the user in such plan-based systems is allowed a significant amount of autonomy, careful attention must be paid to guarantee that she does not alter the environment such that those actions specified by the planning system cannot be performed. A previously defined process called reactive mediation [2] addresses this issue by predetermining responses to destructive user behavior. Reactive mediation, described in more detail below, is a process in which plan data structures are analyzed for points where a user's unexpected or unplanned-for actions could break the causal dependencies in a story plan. The mediation process then computes possible responses to the occurrence of such actions that may serve to preserve the coherence of the plan's unfolding story structure.

One noteworthy limitation of reactive mediation is that user behavior is examined on a per action basis. That is, mediation responses are taken only at the point where the harmful action is performed. While preserving the validity of the plan's causal structure, this approach fails to take into account the larger context of the user's actions. Often, a user performs a sequence of actions leading to some desired result, in which one or more of those actions may be harmful to the global plan.

In this paper, an extension to reactive mediation, a process called proactive mediation, is described. Rather than examining single user actions, the proactive mediation module examines a proposed plan (provided by an external plan recognition component) that the user is performing in the context of a larger story. Having knowledge about hypothetical future actions that the user may execute allows the proactive mediation module to generate a wider variety of responses to potential harmful user activity, as well as to shape those responses to better integrate with the overall course of the narrative.

II. Background

A. Interaction in Narrative

One approach to the management of an interactive narrative was developed in Weyhrauch's dissertation work on the Moe architecture [3]. Moe allows the user to experience an interactive drama (ID), acting freely within the story environment. The user's actions, called user moves, are combined with actions that the system takes, called Moe moves, to form a history describing the story thus far. A modified adversarial search is used to generate possible future storylines, each of which is evaluated using several features, including logical connectivity, the user's excitement, and the user's sense of freedom. These future storylines are used to select Moe moves that can control characters or manipulate the environment to subtly guide the plot toward the "better" future scenario.

Lamstein and Mateas [4] revive many of the ideas presented in Moe in their search-based drama manager (SBDM). SBDMs

Manuscript received July 23, 2009; revised October 05, 2009; accepted October 10, 2009. First published October 30, 2009; current version published December 01, 2009. This work was supported in part by the U.S. National Science Foundation under CAREER Award #0092586 and by Microsoft Research's University Grants Program. Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

J. Harris is with Red Hat, Raleigh, NC 27695 USA (e-mail: jharris@redhat.com).

R. M. Young is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695 USA (e-mail: young@csc.ncsu.edu).

Digital Object Identifier 10.1109/TCIAIG.2009.2035610

are targeted at more open-ended simulations in which a player is subtly guided so that the experience fits a favorable story arc. Like Moe, the SBDM searches possible future storylines upon the recognition of a player move (synonymous with USER MOVE in Weyhrauch's work), which allows the SBDM to consider the player move's impact on the entire remainder of the story. However, player moves only impact the narrative as they occur; no consideration is given to determine which player moves are more likely when evaluating future story direction.

Management of user activity along a more structured storyline is proposed by Gordon and Iuppa [5]. Their storyline adaptation strategies define how a story, and the player's experience of the story, reacts in response to unanticipated user action at certain choice points. Logical formalizations of commonsense psychology are used to maintain lists of the previous and future storylines, as well as previous user activity. When the user deviates from the intended storyline, these lists are examined in order to pick an appropriate adaptation strategy to modify the narrative. However, these adaptation strategies are primarily used to guide the user back onto a fixed or slightly branching story. A user's influence over the story's direction is more pronounced in the facade system [6]. Facade uses the concept of a beat to define characters' behavior as the story progresses. The story plays out as a series of beats dynamically selected to fit a desired story arc. Each beat contains information about how characters should react to various user activity in addition to how that activity affects the likelihood of future beats.

Alternatively, a plan-based approach to interactive narrative is presented by Cavazza *et al.* [7] in which a character's behavior is represented by a hierarchical task network (HTN). The user's role in their system is that of a spectator, not of a character, and the degree of user interaction reflects that role; they can verbalize commands to the characters or manipulate objects in the virtual environment. No framework is put in place to explicitly account for user activity. Rather, agents replan their current course of action if any plan step fails, as a result of activity by the user or another character.

B. Anticipating Future State

While the above approaches to managing user activity can alter the narrative state to account for unexpected and harmful user activity, none attempt to make any guesses about future activity that may be harmful. Laird's work on the Quakebot [8] incorporates anticipation abilities into an agent playing *Quake*, a commercial video game, using the Soar cognitive architecture [9]. While this particular environment is not especially narrative oriented, it does provide evidence pertaining to the use of prediction in video games, which are often used in the production of interactive narratives [7], [10]. Thue *et al.* [11] describe techniques for the online classification of player actions that exploit expectations about a player's goal-directed behavior to improve the accuracy of its predictions. This approach specifically leverages expectations that players take action at any point in game play to move the world closer towards their goals. It does this without an explicit model of the plan space for its domain; integration with the plan-based models described here or used by other researchers could yield additional interesting results.

Anticipation of future story direction has been examined by Laaksolahti and Boman [12]. They present an interactive narrative framework in which autonomous character agents guide the narrative through a finite automaton over story states. A story manager examines each agent's current goals, plans, and intentions to anticipate future states of the system. If those states are "undesirable," the story manager can perform certain actions to subtly manipulate the narrative direction, such as changing aspects of the agents' internal model or modifying objects in the story world. However, the story manager only takes the synthetic agents' models into account and makes no predictions of any human users' actions.

Work by Magerko [13] does incorporate hypothesized future user behavior in the interactive drama architecture (IDA) system. IDA uses a Monte Carlo simulation to predict world state changes between predefined plot points in a narrative. Their model is used to determine if a user's expected actions are likely to satisfy the preconditions of any plot points and to adapt their execution environment accordingly to further advance the story. The system requires an author to explicitly enumerate potential responses for potentially harmful actions. This information, encoded by the author at design time, allows IDA to select responses in a manner that is tailored to factors such as subtlety and effectiveness, but requires an author to provide numeric values for each response's contributions to these factors.

In contrast, the process we define below uses an explicit plan representation to describe hypothesized user behavior. This representation not only allows for planning responses to user actions, but also identifies specific harmful actions and the conditions they require for execution. The resulting system can proactively alter the world state and the actions the system will execute in order to prevent the user from performing any harmful actions.

C. Plan Recognition

In order to reason about the interactions between future story events and future user activity, a plan describing that activity must be submitted to the proactive mediation module. While the specifics of inferring a user's plan lie beyond the scope of this work, a brief overview of relevant work in plan recognition is given (for further discussion, see the survey provided in [14]).

Plan recognition involves observing an agent's actions in order to infer their goals, intentions, or future actions, and can be partitioned into two broad categories. The first is keyhole recognition, which passively observes the agent whose plan is being inferred. In keyhole recognition, the agent being observed is assumed either to be unaware of the recognition process, or to not be attempting to influence it in any way. The second category is intended recognition, in which the agent is aware of being observed and is understood to affect the inference. For the purpose of this paper, it is assumed that the former method is used in providing inputs to the proactive mediation component.

The general task of plan recognition begins with a set of goals that the agent may pursue and a set of plans, called a plan library, which describes how the agent can achieve those goals. As the agent's actions are observed, hypotheses for plans are formed based on the action history and possible goals the agent may be

trying to achieve [14]. There are many variations on this process, some involving the use of machine learning to infer goals [15] or to adapt plan libraries to a specific agent's preferences [16].

Some work in plan recognition has applied machine learning to infer a player's plan in video games [17], [18]. As stated previously, video games have been incorporated into the architecture of various interactive narrative systems. While the focus of this paper is on interactive narrative and not on interaction within games, the potential for planning to find use in other narrative-focused game contexts (e.g., quest generation in role-playing games) suggests that game-based plan recognition methods may be effective in combination with proactive mediation in those specific contexts. It should also be stated, however, that proactive mediation makes no commitment to the specific techniques used to generate the user's hypothesized plan. Any algorithm that produces a plan containing the proper notations outlined in Section III-A should suffice.[1]

Generating responses to unanticipated change in an environment has been addressed by a number of research efforts. Firby's reactive action packages define various action sequences that a robot can perform for a given task in case of failure [19]. Gordon and Iuppa [20] introduce storyline adaptation strategies which define the ways that a story can change in response to unanticipated user action at choice points in a story. Steve, an animated pedagogical agent, monitors user activity and appropriately responds if a user interrupts the current task being demonstrated [21]. While these approaches deal with the generation of responses to unexpected behavior as it arises, none of these systems exploit expectations about likely future events to alter the unfolding action.

D. Zocalo Architecture

This section contains a brief overview of Zocalo, the system in which our approach is implemented. More details regarding the implementation of our work within Zocalo is provided in Section V.

The Zocalo system architecture [2] is a distributed, service-oriented approach to the generation and execution of interactive narratives within virtual environments. Components of the system communicate via XML across the internet to reason about high-level narrative structure. Two of these components are central to the discussion here: a story planning system based on the Longbow planning system [22] and the component that serves as the virtual world interface between a user and the rest of Zocalo. This component, called the MWorld, contains logic for translating declarative descriptions of the narrative produced by the planning system into function calls that execute directly in the user's virtual environment.

The planning component of Zocalo is used to generate the story structure executed by the characters within the story world. Before an interactive session begins, the planning system builds a story plan which represents the actions of all the agents in the story world, including those of the user. Longbow plan structures are similar to those used in partial-order, causal link, and HTN-style planning systems [23], [24]. The plans contain annotations that explicitly mark the temporal relationships between all actions in the story plan, defining a partial order indicating the steps' order of execution. Other annotations, called causal links, mark all causal relationships between the actions in the plan as well. A causal link connects plan steps s_1 and s_2 via condition e, written link $s_1 \rightarrow^e s_2$ just when s_1 establishes the condition e needed by subsequent action s_2 in order to execute.

Once the planning system has generated a plan for a story, a scheduler component called the execution manager translates the plan structure into a directed acyclic graph (DAG) representing the temporal dependencies between the steps in the plan. As steps in the DAG are ready to execute, the manager sends commands to the MWorld invoking the corresponding function calls. The MWorld provides updates to the execution manager regarding the status of each function's execution; as each function call completes, the manager updates the temporal dependencies within the execution DAG and sends commands to execute the next set of plan actions.

E. Reactive Mediation

As described above, Zocalo drives the action within its story world based on the structure of a plan produced by a narrative planner. Plan execution is complicated, however, because users are relatively unconstrained with respect to the actions that they can perform in the world as the plan is being executed. The plans used by Zocalo are dependent upon user actions, both because some user actions are required for the plans to progress and because the consequences of user actions may inadvertently interfere with the world state on which the plan structure depends. As users issue commands for their characters to perform actions within the story world, these actions must be checked against the narrative plan to determine how they fit with the plan's structure. This process, called reactive mediation, is described in detail in [2]. We provide a summary of the process below as background for the extension to mediation that is the main contribution of this paper.

1) Characterizing User Actions: As the user performs an action, Zocalo must characterize the act with respect to the story's requirements; actions that the story is depending upon must be identified in order for the story to progress, while actions that interfere with the story's structure must be identified so that the damage that they might cause to the narrative can be avoided or minimized.

By comparing each action executed by the user to the structure of the story world plan, Zocalo automatically characterizes user actions into one of three categories: constituent, consistent, and exceptional.

A *constituent* action is one that maps directly to a step in the story world plan. The action's type, arguments, and temporal and causal structure all match the corresponding constraints on a step specified for user execution.

A *consistent* action is one that is not constituent and whose effects do not alter the virtual world in a way that interferes with the successful execution of the story world plan. Specifically, an action a is consistent just when it is not constituent and, for each of its effects e, there is no causal link in the story world plan that spans the point in time where a is being executed and that is labeled $\neg e$. In practice, most user actions fall into this category.

[1]This also applies to cases in which plan recognition is not used at all, and some other means of proposing a user's plan is employed.

An *exceptional* action is neither constituent nor consistent, that is, at least one of the effects of the action threatens a causal link in the narrative plan. Formally, action a with effect $\neg e$ threatens causal link $s_1 \rightarrow^e s_2$ when a is performed after s_1 and before s_2. Exceptional actions, if allowed to execute, break the causal dependencies on which a story plan is based, making the plan impossible to execute.

2) Responding to Exceptional Actions: When a user initiates exceptional actions, their execution changes the state of the story world in such a way that the story plan is no longer executable. In order to prevent this consequence, the Zocalo execution manager monitors each command sent by the user to the virtual world, characterizing it immediately as consistent, constituent, or exceptional. When an exception is detected, the system determines an appropriate response before the user's command is queued for execution. The Zocalo execution manager responds to each exception either by preventing the exception's threatening effect to be established or by adjusting the narrative such that the action's performance poses no threat. These outcomes are achieved by *accommodation* or *intervention*, described briefly below.

When an exceptional action is accommodated, it is allowed to execute, and the remaining plan is restructured so that no causal links are threatened. This restructuring can often be slight, such as selecting a different character to perform a task. However, in certain cases, the revised narrative plan may be substantially different from the original, requiring significant computation on the part of the planner. Further discussion of this replanning process is beyond the scope of this paper.

When an exceptional action is handled by intervention, an alternative action is executed in its place. This alternative action, instantiated from a set of predefined failure modes, is similar in appearance and function to the exceptional action, but has a different set of preconditions and effects.

3) Policy Tables: The process of revising plans and finding suitable failure modes is complex, and cannot reliably execute in an acceptable amount of time if performed when the exception occurs. In order to provide a satisfactory response time when an exception does occur, accommodations and interventions are generated in advance, and held in the *mediation policy table*.

After generating a narrative plan but before its execution, Zocalo examines the plan's causal structure and identifies which user actions can cause exceptions at every point in the plan where a user may act. For every possible exception, a queue of appropriate responses (interventions and accommodations) is computed. This action/response queue pair is inserted as an entry into a mediation policy table, along with the interval in the narrative plan when the action can be performed. The queue of responses is sorted by a heuristic function that determines a qualitative measure of the responses' effectiveness.

III. Extending Mediation

One significant limitation of reactive mediation is that it responds to user activity at the last possible moment. While this approach localizes the point in a story where a user's agency may need to be restricted, intervention and accommodation at the point of an exception can be problematic. Consider a scenario in which the user executes a long series of actions that clearly lead to an exceptional action, for instance, besieging the castle of a story's central character, capturing him, and then attempting to kill him. If the system allows the user to spend the time and resources to capture the nobleman, but then intervenes repeatedly as the user swings his sword, the user will not only be frustrated with his apparent inability to hit his target but also with the failure of the system to have guided the story more effectively. The user has put in significant effort in pursuit of a particular course of action, and yet the system has done nothing to deter the user until the action sequence's very end.

This problem is addressed by proactive mediation, which restructures the narrative plan to better account for anticipated user activity. Rather than monitoring the user's activity only as it occurs, a proactive mediator also examines the user's anticipated plan of action provided by an external plan recognition component (e.g., [17] and [18]). Steps in the user's anticipated plan[2] are characterized as constituent, consistent, or exceptional, just as individual actions are categorized under reactive mediation. Proactive mediation extends the notions of intervention and accommodation to avoid threats from exceptional steps that have not yet occurred. While the basic objectives of reactive and proactive mediation are the same, a proactive mediator's knowledge of expected future steps is utilized to allow a wider range of responses to user activity.

A. Proactive Mediation Input

As described above, Zocalo uses a plan representation to describe story events in a virtual environment. Proactive mediation uses the same plan representation to describe likely future event sequences that the user may perform. This plan is supplied by a plan recognition component, which, upon recognizing a user's plan, submits it to the proactive mediator. A plan is defined formally below.

1) Plan: A plan is a tuple $\langle S, B, O, L\rangle$ where S is the set of steps, B is the set of binding constraints on the steps in S, O is the set of ordering constraints between steps in S, and L is the set of causal links between steps in S. The proactive mediator takes as input a *narrative* plan, a *recognized* plan, and an *operator library*, defined formally below.

2) Narrative Plan: A narrative plan is a plan $N : \langle S_N, B_N, O_N, L_N\rangle$ generated by the Zocalo system which describes all of the steps to be performed by the characters in a story, including those of the user. We say that a step s is a narrative step just when $s \in S_N$.

3) Recognized Plan: A recognized plan is a plan $R : \langle S_R; B_R; O_R; L_R\rangle$ that is proposed by a plan recognition system. This plan hypothesizes the sequence of steps that the user intends to perform, and that the user expects to occur. We say that a step s is a recognized step just when $s \in S_R$.

4) Operator Library: An operator library is a collection of operators characterizing the actions available for the given story world domain, instantiated as steps. An operator is a tuple $\langle P, E\rangle$ where P is a set of preconditions that must hold true

[2]Here, it is appropriate to make the distinction between a step and an action. A step refers to a data structure that describes an event in the virtual world. A plan contains a set of steps, whose referent events, at the time of planning, have not yet occurred. An action refers to an event that is occurring at the present time, regardless of whether it is described in a plan.

before the step is executed, and E is a set of effects that are true after the step is executed. Each step in either the narrative plan or the recognized plan can be a system step (any action executed by system resources, characters, etc.) or a user step (one initiated by the user and performed by the user's character) and is identified by a unique ID. The set of system steps is denoted S_{SYS}, where $S_{\mathrm{SYS}} \subseteq (S_N \cup S_R)$. The set of user steps is denoted S_{USER}, where $S_{\mathrm{USER}} \subseteq (S_N \cup S_R)$ and $S_{\mathrm{SYS}} \cap S_{\mathrm{USER}} = \varnothing$.

B. Generating the Mediated Plan and Identifying Steps

The first step in the proactive mediation process is the creation of a working plan that encapsulates the actions of both input plans. This new plan, called the *mediated* plan, is formed by merging the narrative and recognized plans in the following manner. To simplify the current discussion, a step in the narrative plan and a step in the recognized plan whose IDs are identical are assumed to refer to the same event. The mediated plan is thus a tuple $M = \langle S_M, B_M, O_M, L_M \rangle$ where $S_M = S_R \cup S_N$, $B_M = B_R \cup B_N$, $O_M = O_R \cup O_N$, and $L_M = L_R \cup L_N$.

Once the mediated plan is created, the steps in the recognized plan are then categorized as *inclusive* or *exclusive*. Inclusive steps occur in both plans (i.e., N and R), while exclusive steps only occur in the recognized plan. Inclusive steps may be either user steps or system steps, while exclusive steps are assumed to be only performed by the user. That is, it is assumed that the user will not plan for system steps to occur which are not actually part of the narrative plan.

A step s is an inclusive step just when $s \in S_R \cap S_N$. The set of inclusive steps is denoted S_{IN}. A step s is an exclusive step just when $s \in (S_R - S_N)$. The set of exclusive steps is denoted S_{EXL}.

User steps can be constituent, consistent, or exceptional, just as user actions. The definitions of these steps are similar to their action counterparts, with the understanding that the steps refer to future events. The one deviation is the definition of an exceptional step, which can be described as a potential exceptional action given the ordering constraints of the mediated plan. All inclusive steps that are performed by the user are identified as constituent, and all exclusive steps are identified as either consistent or exceptional.

1) Constituent Step: A step s is constituent just when $s \in S_{\mathrm{IN}} \cap S_{\mathrm{USER}}$.

2) Exceptional Step: A step s with effect $\neg e$ is exceptional just when:

- $s \in S_{\mathrm{USER}}$;
- there exists a causal link $s_1 \rightarrow^e s_2 \in L_N$;
- s is not required to come before s_1 or after s_2, based on the transitive closure of the ordering constraints within O_M.

The set of exceptional steps is denoted S_{EXP} .

3) Consistent Step: A step s is consistent just when $s \in S_{\mathrm{EXL}} - S_{\mathrm{EXP}}$

C. Handling Exceptional Steps

Once the steps in the mediated plan have been characterized, the mediator then determines how to respond to each exceptional step. We say that an exceptional step is avoided when the mediator alters the narrative plan in a manner that deals with the harmful effects of the exceptional step. For each exceptional step s_x, $s_x \in S_M$ with effect $\neg e$ that threatens some causal $s_1 \rightarrow^e s_2 \in L_M$, s_x can be avoided in one of three ways:

- proactive intervention: stopping the user from performing step s_x in the mediated plan;
- proactive accommodation: eliminating the need for the causal link $s_1 \rightarrow^e s_2$ in the mediated plan;
- proactive reordering: enforcing orderings such that s_x cannot occur between s_1 and s_2.

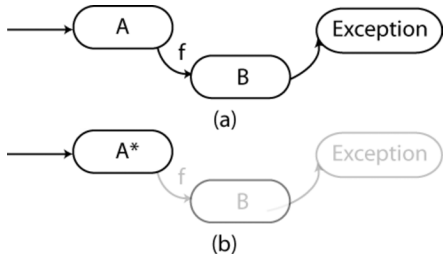

Fig. 1. Substitution: (a) a portion of a mediated plan with user steps A, B, and *exception*; (b) using substitution, step A is replaced with its failure mode A^* in order to stop *exception* from being executed. A^* does not establish the effect f, a precondition of the contributory step B, and the causal chain is broken. Removed plan structure is shown using faded lines.

1) Proactive Intervention: In general, the purpose of intervention is to prevent the exception's threatening condition from being established. As described earlier, reactive intervention achieves this by replacing the execution of the exceptional action with the execution of one of its failure modes that does not have the threatening condition as an effect. This solution is also possible under proactive intervention. However, since an exception considered by the proactive mediator has not yet occurred, additional action can be taken by the system to make one or more preconditions of the exception false, thus making the action itself inexecutable. This can be achieved by executing a system step which makes a condition false, or by removing a step that causally leads to the exception.

Proactive intervention prevents the user from performing some exceptional step s_x in question. This can be done by stopping the execution of s_x itself, or of any step that contributes to s_x. Step s_1 contributes to s_x just when $s_1 \rightarrow^e s_x \in L_M$ or some other step s_2 contributes to s_x and $s_1 \rightarrow^e s_2 \in L_M$. The execution of each contributory or exceptional step s in the mediated plan can be avoided via intervention by one of the following measures (schematic examples are shown for each in Figs. 1–4).

2) Substitution—Preventing the Exceptional Step from Establishing the Threatening Condition by Replacing s with One of Its Failure Modes s^*: If s is contributory, s^* must be selected so that at least one of the conditions established by s and used in a contributory causal link is not asserted by s^*. If s is exceptional, the effects of the failure mode must not threaten any causal link in the narrative plan. If the substituted failure mode has any preconditions that are not in the original step, those preconditions are considered open, and appropriate flaws are added to the plan. Fixing these flaws requires additional plan construction in order to make the resulting plan complete.

3) Aversion—Preventing the Execution of s by Making One or More Preconditions of s False at the Point Immediately Prior

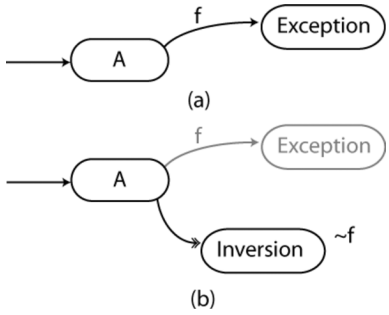

Fig. 2. Aversion: (a) a portion of a mediated plan with step A contributing to *exception*; (b) aversion is used, adding an *inversion* step to the plan, which establishes $\sim f$, removing the condition required to execute *exception*. The ordering link $A <$ *inversion* (shown as a double-headed arrow) ensures that the *inversion* indeed negates f. Removed plan structure is shown using faded lines.

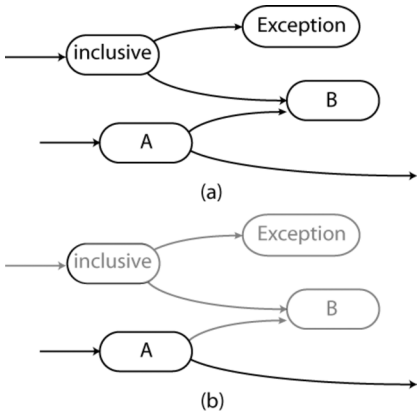

Fig. 3. Disablement: (a) a portion of a mediated plan with story steps A and B, an *inclusive* step, and an *exception*; (b) the *exception* is disabled by removing the *inclusive* step, which contributes to *exception*. Step B is also removed from the plan as part of the replanning process because it causally relies on *inclusive*. Removed plan structure is shown using faded lines.

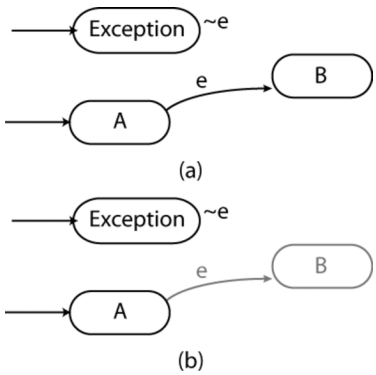

Fig. 4. Accommodation: (a) a portion of a mediated plan containing narrative steps A and B and an *exception*; (b) the *exception* is accommodated by eliminating the casual link AB, along with step B, as part of the replanning process. Removed plan structure is shown using faded lines.

to Its Execution: This is achieved by inserting a system step s_i, called an inversion step, into the plan. Step s_i has an effect $\neg f$, where f is a precondition of s. Additional ordering constraints are added such that s_i must come before s and s_i must come after all steps which establish f, including the original source of the causal link. If the resulting plan's ordering constraints are inconsistent, aversion using s_i cannot be performed.

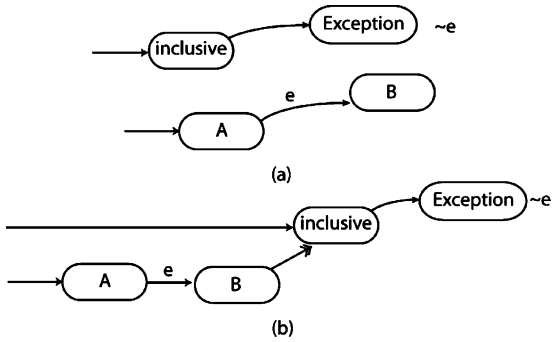

Fig. 5. Reordering: (a) a portion of a mediated plan with narrative steps A and B, an *inclusive* step, and an *exception*; (b) the ordering link B $<$ *inclusive* (shown as a double-headed arrow) is inserted into the plan so that the *exception* cannot be executed between A and B.

Although the inversion step s_i is ordered before s, there is no guarantee that the condition $\neg f$ will be established before the user executes s. A race condition exists between the system's execution of s_i and the user's execution of s; if s appears early in the recognized plan, or if a long sequence of steps is required to establish $\neg f$, then s could be executed first. This race condition is avoided by the system "instantaneously" executing the aversion steps, unconstrained by rules in the virtual world such as animation times or physics. For instance, the system can programmatically shut and lock a door without requiring that a character perform the act. This approach will work, however, only if 1) the user does not already know the status of any aversion effects, and 2) the user cannot directly observe the state changing from these actions.

4) Disablement—Removing a Contributory Inclusive Step s from the Mediated Plan: This prevents s from establishing causal links that contribute to an exceptional step. Once s has been removed (along with all causal links, variable bindings, and step orderings relating to s), conditions in the narrative plan that were satisfied by s are no longer satisfied, and some replanning will be required to reestablish those conditions.

5) Proactive Accommodation: Proactive accommodation allows the user to perform the exceptional step s_x; in response, the system replans the narrative to reestablish any causal links threatened by s_x's effects. In this regard, there is no difference between proactive accommodation and reactive accommodation, and the basic mechanism does not differ between the two. The fundamental difference between proactive accommodation and reactive accommodation is that proactive accommodation can alter the narrative steps which occur prior to s_1.

By modifying steps prior to s_1, proactive accommodation can eliminate extraneous steps from being executed to establish the original causal link, resulting in a narrative plan that is potentially more coherent. Further, plans generated with proactive accommodation can use effects of exclusive steps in the recognized plan to satisfy any open preconditions, potentially giving the user a stronger sense of participation in the storyline.

6) Proactive Reordering: Proactive reordering (Fig. 5) introduces additional orderings to the narrative plan that make it temporally impossible for s_x to be executed between s_1 and s_2. Conceptually, there are two ways to enforce this: add the ordering $s_x < s_1$, or add the ordering $s_2 < s_x$. Both of these

options effectively reorder the steps to prevent s_x from being executed between s_1 and s_2, but neither option is particularly viable. If the ordering $s_x < s_1$ is introduced into the mediated plan, Zocalo would wait for the user to perform s_x before executing s_1. This rigid requirement can easily stall the entire narrative if the plan recognition component proposed an inaccurate plan, or if the user simply changed her mind about performing s_x.

Enforcing the ordering $s_2 < s_x$ can be problematic as well, for the simple reason that s_x is to be performed by the user. Stopping the user from executing s_x if attempted before s_2 is already accomplished by reactive intervention.

There is a case, however, in which system steps can be reordered such that s_x can only be executed after s_2. If $s_c \in S_{\text{IN}} \cap S_{\text{SYS}}$ and s_c is ordered before s_x, adding the ordering $s_2 < s_c$ implicitly ensures that s_x cannot occur before s_2 (assuming that the ordering is consistent with the mediated plan).

D. Replanning

A number of mediation strategies described above involve removing elements of the mediated plan and filling in the resulting gaps with alternative plan structure. The replanning process used to fill in the missing plan structure is similar to the plan generation process we use, with one notable difference: no ordering link $s_n < s_e$ can be added to the plan, where s_n is a narrative step and s_e is an exclusive user performed step. This restriction is in place because the system has no direct control over when s_e will be performed, since its execution is left up to the user. As a result, s_e's execution cannot be guaranteed to follow that ordering. Ordering links and causal links are, however, allowed from exclusive steps to narrative steps, which can act to further involve the user in the narrative by effectively making exclusive actions inclusive.

E. Mediation Algorithm

A single mediated plan can potentially contain multiple exceptions, which must all be avoided before the plan can become the active storyline and begin execution. The mediation algorithm used to avoid all of the exceptions is similar in many respects to our planning algorithm, which can be characterized as a refinement search through a plan space graph [24]. A node in the graph represents a partial plan, and a node's children are refinements of a specific flaw in the parent. Similarly, the mediation algorithm is a refinement search through a mediation space graph, where a node represents a plan and its corresponding policy table, and a node's children are responses to a specific exception. Search through this mediation space is guided by an author-defined heuristic, which is used to qualitatively evaluate each plan/policy table pair. This heuristic could be derived from the same planning heuristic used to generate the original story plan, so that proactive responses that coincide with the author's intended story are favored.

While expanding the mediation space tree, choosing an exception to avoid is not completely arbitrary. Avoiding an earlier exception can (in the case of intervention) implicitly avoid any exceptions that are causally dependent on the former, so only candidate exceptions that have no contributory exceptions are chosen.

IV. Example

The following is an example of a mediated plan just after the narrative plan and recognized plan have been merged and the steps identified. The original narrative plan describes the events in a single chapter of a larger story, in which a secret agent played by the user is attempting to acquire secret documents by posing as an employee of a corporation. The chapter ends with the user being caught in her boss's office looking for the documents. The scenario begins with the user being given a master key to the building by a collaborator on the inside (the *spy*), and then walking to her boss's office and using the master key to enter it. At the same time, the boss arrives in his car, takes the elevator to the seventh floor, and then enters his office. This narrative corresponds to steps 0–8 in the merged plan depicted in Fig. 6.

The clever user, knowing that the boss rides the elevator every morning, decides to sabotage the elevator so that she will not be caught. This threatens the narrative plan, as the goal of this portion of the story is for the user to be caught. Specifically, the system identifies the Move step (step 9) as exceptional because one of its effects, ¬At(user, hallway), threatens the causal link between the InitialState (step 0) and another Move (step 4). Similarly, Sabotage (step 11) is identified as exceptional because one of its effects, ¬Working(elevator), threatens the causal link between the InitialState (step 0) and RideElevator (step 3).

The first exceptional step, Move (step 9), must be allowed to execute, because no failure modes have been defined for Move, and no inversion steps exist which can move the user to a different location (she must do this on her own). The only viable option is accommodation, which in this case can remove Move (step 4) and all associated causal links and ordering constraints. This does not introduce any new flaws to plan because Move (step 13) also establishes the condition At(user, officeDoor) which was originally established by Move (step 4).

Substitution can be used to stop the user from sabotaging the elevator, by replacing Sabotage (step 11) with a failure mode, or by replacing any exclusive contributory step (steps 9 or 10) with a failure mode. Performing a substitution on the Sabotage step is handled by reactive mediation (since that single action is exceptional), and in this world no failure modes have been defined for the Move operator. The Enter operator, however, does have a failure mode defined, which is called JammedDoor. This failure mode results in the door not opening and the user not being in the power room.

Aversion can also be used to mediate this plan by inserting a Move step followed by an inversion step, StandWatch, which moves the seventh floor door's security guard to watch over the power room. This has an effect of ¬Unguarded(powerRoom), which prevents the user's character from sabotaging the elevator. Note that the animations for walking the security guard to the power room do not need to play out if the user is not in the area (the action is not observable). The system can simply make the effects of the step true, in effect "warping" the guard to his new post.

The third form of intervention, disablement, can be applied to Give (step 1), because it is the only inclusive step that contributes to Sabotage (step 11). After removing the step from the

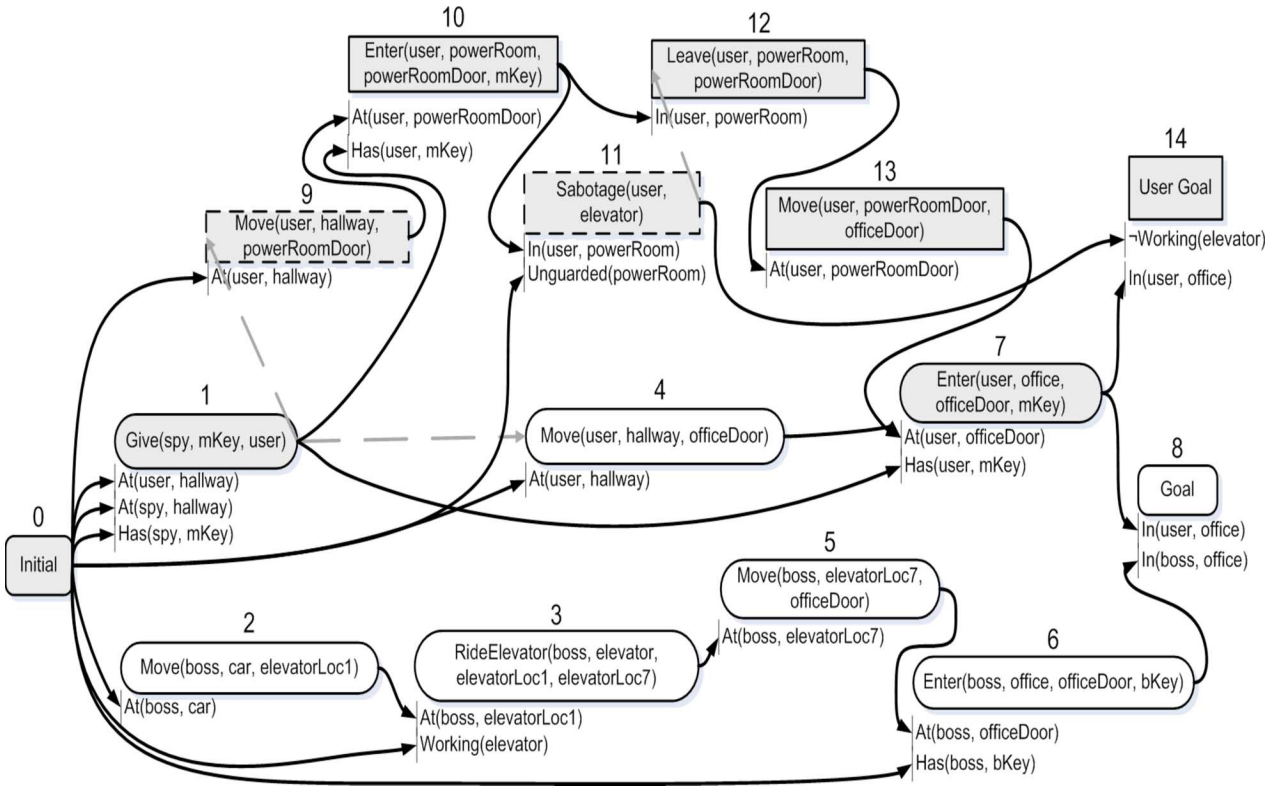

Fig. 6. Example of a merged plan, with all steps identified: the rounded nodes indicate narrative steps, the gray nodes indicate recognized steps, the box nodes indicate exclusive steps, and the dashed nodes indicate exceptional steps. Solid arrows represent causal links, and dashed arrows represent ordering constraints.

mediated plan, two preconditions are left open (in steps 10 and 7). Some additional planning is required to reestablish Has(user; ?key-7) with the additional constraint that Has(user; ?key-10) is not reestablished. In this particular world, the spy has a second key which only opens the boss's door, so one alternate plan replaces Give (step 1) with a different Give, in which this second key is given to the user.

Accommodation allows the user to sabotage the elevator, planning around the Working(elevator) causal link between the InitialState (step 0) and RideElevator (step 3). Alternate plans can include a repairman fixing the elevator, or the boss taking the stairs instead.

The Sabotage step can also be allowed if it occurs after the boss is on the seventh floor. This can be achieved by adding the ordering constraint $3 < 1$ to the plan, which waits for the boss to be on the seventh floor before the spy gives the user the master key.

V. Implementation

A. Zocalo Architecture

The Zocalo system architecture is a distributed, service-oriented approach to the generation and execution of interactive narratives within virtual environments. Components of the system communicate via XML across the Internet to reason about high-level narrative structure. In its current implementation, Zocalo comprises three components.

- Fletcher: the web service providing narrative planning functionality using Crossbow, a narrative planning system based on the Longbow planner [22].
- Execution Manager: the central component that serves as the gateway between the Zocalo web services and the execution environment. The Execution Manager contains logic for translating declarative descriptions of the narrative produced by Fletcher into execution messages for the environment and schedules when to send these messages. It also receives messages from the environment concerning user activity and enforces mediation policies provided by the two mediation services.
- CrossWind: the web service providing reactive mediation functionality [22]. CrossWind takes as input an initial narrative plan, an operator library (including failure modes), and an action specifiers document, which identifies environment-initiated actions to be considered for mediation. Upon initiation, CrossWind can then begin the mediation session.

When a mediation session begins, CrossWind constructs the initial policy table, populating it with interventions only. Once the Execution Manager receives this policy table, execution of the narrative can begin. At this point, CrossWind supplements the initial table with accommodations, calling on Fletcher to fill in missing plan data to restructure the narrative. When all accommodations have been generated, the Execution Manager updates its working policy table to the revised table that includes those accommodations. CrossWind then generates policy tables for accommodation plans in a similar fashion, producing a policy tree. In its current implementation, CrossWind expands this policy tree using an iterative depth-first search, pruning portions of the tree when they are no longer needed (when the user has passed the point in the narrative that contingent plans addressed).

The final key component (though not a part of Zocalo) is an execution environment, which actualizes execution messages from the Execution Manager and reports on user activity, as well as the success or failure of requested actions. It acts as the "front-end" to Zocalo and handles interactivity with the user such as rendering graphics and processing keyboard/mouse events. Supported execution environments currently range from text-based applications (e.g., MUDs) to web-based environments (Java applets, Flash pages, DHTML, etc.) to handheld platforms (Palm, PocketPC) to commercial video game engines (UT2004, Half Life 2, Neverwinter Nights) to full Virtual Reality simulations.

B. Kyudo

The implementation of the proactive mediation algorithm is named Kyudo, and is written in C#. Due to its coupling with plan recognition software, it can be referenced as a library or as a separate web service. In either configuration, the plan recognition software receives notifications of user action from the Execution Manager as they occur, and uses this history to propose a recognized plan as input to Kyudo.

1) Inputs: The specific inputs to Kyudo can be divided into two categories: session inputs and event inputs. Session inputs are presented when an interactive session begins and remains constant throughout its duration, while event inputs are called periodically when a user's plan has been recognized.

The session inputs are as follows.

- A domain specification: the set of narrative operators, described in XML, that are used to read in the narrative plan as well as to replan the narrative when responding with accommodation or disablement. The domain specification also defines failure modes for these operators.
- A problem specification: the initial state of the narrative world and the desired goal state, also described in XML.
- A user actions specification: an XML document describing sets of operators that constitute user performed steps. Kyudo uses this document to identify user steps in the narrative and recognized plan inputs.
- A set of directive operators: the set of directive operators that are used to instantiate inversion steps and to replan the narrative when responding with substitution or aversion. The file format is identical to that of the domain specification. However, since all directives are system steps, there is no need for failure modes and hence none are included.
- Planning heuristic: a string indicating which set of predefined heuristics to use to control search during replanning. In its current implementation, Kyudo also uses this heuristic as the basis for the mediation heuristic, so that mediation responses that coincide with the original narrative plan are favored. While this practice may be a sensible rule of thumb, it is certainly not a requirement, and future iterations will also allow the mediation heuristic to be specified independently.
- Replanning search size: an integer representing the maximum number of nodes to search in the plan space when replanning. Planning in general is computationally complex [25], and limiting the search space keeps the mediation algorithm from impractically long or large computations.

The event inputs are as follows.

- A narrative plan: the steps, ordering links and causal links that constitute the narrative plan, described in XML. This represents the desired narrative, not necessarily the one in progress. It can be the result of a reactive accommodation, a previous Kyudo output, or a different plan altogether.[3]
- A user plan: the steps, ordering links and causal links that constitute the proposed plan that the user is pursuing, also described in XML.

2) Algorithm: A single mediated plan can potentially contain multiple exceptions, which must all be avoided before the plan can become the active storyline and begin execution. The mediation algorithm used to avoid all of the exceptions is similar in many respects to our planning algorithm (described above), which can be characterized as a refinement search through a plan space graph [22].

The process of expanding the mediation space tree in effect determines the order in which the responses to exceptions are determined. Choosing an exception to avoid is not completely arbitrary. Avoiding an exception that occurs earlier in the story plan can (in the case of intervention) implicitly avoid any exceptions that are causally dependent on the former. Consequently, the search algorithm selects only candidate exceptions that themselves have no contributory exceptions.

3) Proactive Mediation Within Zocalo: Kyudo's strength lies in addressing a user's sequence of actions within the context of a narrative plan. This functionality is meant to augment, not replace, existing reactive mediation in the Zocalo framework. Both methods have their benefits, as a single action performed by the user may be an exception, whether part of a larger plan or not.

The Zocalo architecture with proactive mediation is depicted in Fig. 7. The Execution Manager relays all user activity, along with the currently executing plan, to the plan recognition component. When the plan recognition component formulates a plan hypothesis, it proposes this plan to Kyudo and submits the output as the new narrative plan. The Execution Manager adopts the plan portion of this mediation node as the current plan, and the entire node is submitted to CrossWind. If the node's policy table is empty (no substitutions were made in the proactive mediation process), then the process of building the policy table is the same as that at the beginning of the Zocalo session. Otherwise, the policy table's existing entries are preserved, and additional entries are added to preserve the new plan's causal links as described in Section II.

When CrossWind has constructed the initial policy table, execution of the revised narrative can begin.

VI. Future Work

The current implementation of Kyudo can effectively mediate a user's actions within the context of a narrative plan (the Kyudo algorithm is described in Fig. 8). However, new questions have been raised, which can serve to guide the future direction of

[3]One example of a different plan might be a modification of a previous call to Kyudo. This modification might be the removal of some plan structures (such as directive steps) that are no longer needed due to the user no longer pursuing a previously recognized plan.

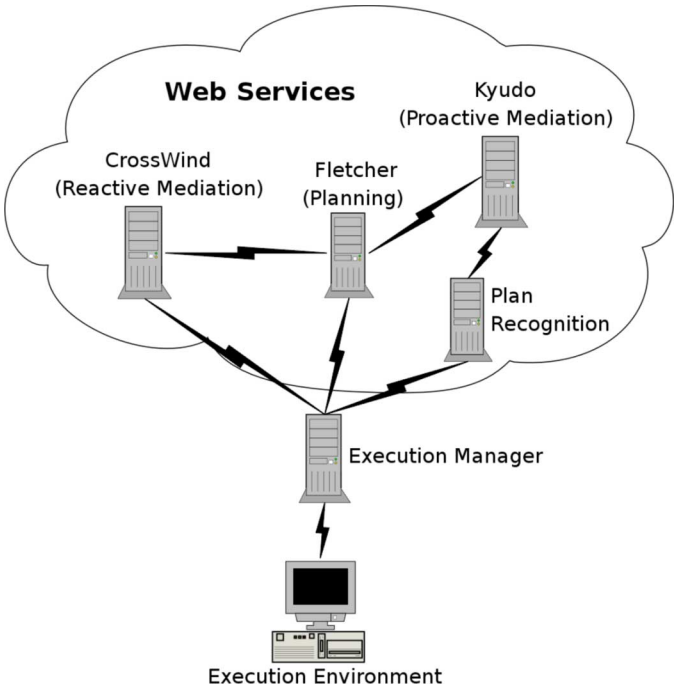

Fig. 7. Zocalo service-oriented architecture with proactive mediation.

Start of Kyudo
Step 1: Merge narrative and recognized plans into the mediated plan
Step 2: Set mediated plan as root of the mediation space
Step 3: Best plan = mediated plan
Step 4: If best plan has no exceptions, return best plan
Step 5: Pick an exception from the best plan
Step 6: Add to the mediation space all fixes for the exception
Step 7: If the mediation space fringe is the empty set, return FAIL
Step 8: Best plan = lowest ranked plan from the mediation space fringe
Step 9: Go to Step 4

Fig. 8. Kyudo algorithm.

research in this area. The following are potential areas to refine and extend this work.

- One of the inputs to the proactive mediation algorithm is a plan that the human user of the Zocalo system is pursuing, proposed by a plan recognition system. The first step for future work might be to explore the integration of a plan recognition component into Zocalo with the specific role of feeding user plans into Kyudo. There have been many approaches to plan recognition, and selecting and customizing a suitable approach for our needs would allow for further study of the practicalities of mediating various environments, as well as highlighting potential areas for refinement and extension.
- While this work defines a mediation space of possible avoidances of harmful actions, it is up to the authors or knowledge engineers of a particular narrative to provide heuristics for exploring that space. Developing tools and representations for these heuristics, as well as performing empirical studies evaluating the effectiveness of responses can aid in efficiently producing desirable solutions.
- The current approach taken when replanning the narrative removes all plan structure that is causally dependent on the original step(s) being removed. While this approach is taken to maximize the chances of finding a suitable plan, it has its drawbacks. First, there is the possibility that the modified plan will be very similar to the original, with only a few small modifications. This can be inefficient, as a large portion of the original plan is cleared out only to be reinserted by the replanning process. In addition, for very large plans this can be impractical. The cost of replanning can be too great to use in this setting, as it can be critical to put a new narrative plan in place to stop deviant user behavior. Future work could explore more efficient methods of removing plan structure, or incorporating adaptive planning [25] to generate the new narrative.
- When exceptional steps in the recognized plan are identified, only ordering constraints are taken into account. That is, a step is marked as exceptional if it can occur during the interval of a threatened causal link in the narrative plan. This can be overly cautious if, for example, the causal link begins much later in the plan and the exceptional step is hypothesized to occur very soon. The use of temporal knowledge in the plan inputs can not only aid in determining the likelihood of exceptions, but also in loosening the "instantaneous" execution requirement of directive operators.
- In this paper, a method for altering a narrative in anticipation of a user's activity is presented. As one may expect from the corporate espionage example given, one application of this technology would be in the field of interactive entertainment. Video games and virtual reality environments can better adapt a storyline to fit the user's behavior as it is being played out, engaging the user while maintaining the author's vision. In addition, proactive mediation could be used in tutoring and training software, such as Steve [20], or intelligent help systems (IHSs), such as SmartAide [27]. Responding to a user as they begin to move toward harmful behavior can not only help to avoid it, but also inform the user as to why their intended course of action should be avoided.

Finally, a target for future work focuses on what is perhaps its most significant limitation: its dependence upon an external model of narrative plan production that has not yet been fully developed by interactive narrative researchers. The motivation for this work makes an explicit assumption that plans produced using such a model reflect narrative content and the plans' narrative structure will be apparent to users interacting with worlds being driven by those plans. To date, however, very little work has been done to incorporate notions of important narrative phenomena explicitly into the plan generation process.

This limitation affects the resulting system in several ways. First, the plans being played out may have attractive formal properties from a planning perspective, but may fail to show interesting character dynamics, rising action, conflict, or other common and significant narrative elements. Focusing on the process of proactive mediation, a lack of ability to reason about narrative structure in the process of restructuring plans will likely result in revisions to story plans that lack strong narrative

coherence. Revised plans are then likely to break some aspect of a user's relationship with what he or she sees as the current unfolding storyline.

The choices for response to exceptional actions detailed in Section III-C all affect a user's experience of a storyline in context-specific ways. Critical future work will seek to explore the relationship between those choices and their cognitive and effective impact on a user when they occur. In related work [29], [30], we have previously adapted experiments that probe human subjects' cognitive models of narrative comprehension (e.g., [31]) to measure the similarity between our plan data structures and intended mental models of a generated story represented in text or video. These approaches may be adapted to gauge the consequences of mediation on a user's mental model of a story as it unfolds.

VII. Conclusion

In many interactive environments, a human user is permitted to manipulate the environment in a variety of ways. In a plan-based narrative environment, this manipulation may disrupt the actions of other agents or even the actions that the system intends the user to perform. Proactive mediation expands upon reactive mediation to generate a variety of responses to a user's proposed sequence of actions in the environment. Having a hypothesis about future user actions allows proactive mediation to generate a broader range of responses to user actions that can be temporally distributed over the course of a plan.

There are, however, additional factors to consider in evaluating our approach. Proactive mediation relies on effective plan recognition and can be sensitive to the frequency of change in input from the plan recognition component. In addition, planning is computationally complex; in cases where many proactive responses require replanning, there is no guarantee that an appropriate response can be generated within a reasonable amount of time. However, our preliminary consideration indicates that plans containing potential exceptions occur relatively rarely compared to the number of actions a user is, in practice, likely to perform at any given moment within a story. Most user actions are consistent or constituent; it is the effects of exceptions on the coherence of the story rather than their frequency that motivates the need to address them.

While computation performed by proactive mediation can be costly in some cases, the approach we outline is readily implemented as an anytime algorithm: should proactive mediation fail to generate a response in time to address an exception, reactive mediation can still be used. Similarly, should reactive mediation fail to generate an accommodation in time, intervention (which typically amounts to a straightforward lookup) can be invoked.

Finally, the restructuring of a narrative plan required by mediation may result in system-controlled agents performing actions that are not clearly motivated. Our current work includes integration of the proactive mediation component with an intent-driven planner [27] that generates plans where agents' actions can be understood in terms of their own beliefs, desires, and intentions.

References

[1] M. Cavazza, F. Charles, and S. Mead, "Planning characters' behaviour in interactive storytelling," *J. Vis. Comput. Animat.*, vol. 13, pp. 121–131, 2002.

[2] M. Riedl, C. Saretto, and R. M. Young, "Managing interaction between users and agents in a multi-agent storytelling environment," in *Proc. 2nd Int. Conf. Autonom. Agents Multi-Agent Syst.*, 2003, pp. 741–748.

[3] P. M. Weyhrauch, "Guiding interactive drama," Ph.D. dissertation, Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, 1997.

[4] A. Lamstein and M. Mateas, "Search-based drama management," in *Proc. AAAI Workshop Challenges Game AI*, 2004, pp. 32–41.

[5] A. S. Gordon and N. V. Iuppa, "Experience management using storyline adaptation strategies," in *Proc. Technol. Interactive Digit. Storytelling Entertain. Conf.*, 2003, pp. 19–30.

[6] M. Mateas and A. Stern, "Architecture, authorial idioms and early observations of the interactive drama facade," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-02-198, 2002.

[7] M. Cavazza, F. Charles, and S. Mead, "Planning characters' behaviour in interactive storytelling," *J. Vis. Comput. Animat.*, vol. 13, pp. 121–131, 2002.

[8] J. E. Laird, "It knows what you're going to do: Adding anticipation to a quakebot," in *Proc. 5th Int. Conf. Autonom. Agents*, J. P. Meuller, E. Andre, S. Sen, and C. Frasson, Eds., Montreal, QC, Canada, 2001, pp. 385–392.

[9] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artif. Intell.*, vol. 33, no. 3, pp. 1–64, 1987.

[10] R. M. Young, M. Riedl, M. Branly, R. J. Martin, and C. J. Saretto, "An architecture for integrating plan-based behavior generation with interactive game environments," *J. Game Develop.*, vol. 1, pp. 52–70, 2004.

[11] D. Thue, V. Bulitko, and M. Spetch, "Making stories player-specific: Delayed authoring in interactive storytelling," in *Proc. 1st Joint Int. Conf. Interactive Digit. Storytelling*, Erfurt, Germany, Nov. 26, 2008, pp. 230–241.

[12] J. Laaksolahti and M. Boman, "Anticipatory guidance of plot," ArXiv Computer Science e-prints, 2002 [Online]. Available: arXiv:cs/02060419

[13] B. Magerko, "Evaluating preemptive story direction in the interactive drama architecture," *J. Game Develop.*, vol. 2, no. 3, pp. 51–70, 2007.

[14] S. Carberry, "Techniques for plan recognition," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1–2, pp. 31–48, 2001.

[15] N. Lesh, "Adaptive goal recognition," in *Proc. Int. Joint Conf. Artif. Intell.*, 1997, pp. 1208–1214.

[16] M. Bauer, "Acquisition of user preferences for plan recognition," in *Proc. 5th Int. Conf. User Model.*, 1996, pp. 105–112.

[17] D. Albrect, I. Zuckerman, and A. Nicholson, "Bayesian models for keyhole plan recognition in an adventure game," *User Modeling and User-Adapted Interaction*, vol. 8, no. 1–2, pp. 5–47, 1998.

[18] M. Fagan and P. Cunningham, "Case-based plan recognition in computer games," in *Proc. 5th Int. Conf. Case-Based Reason.*, 2003, pp. 161–170.

[19] R. J. Firby, "Adaptive execution in complex dynamic worlds," Ph.D. dissertation, Dept. Comput. Sci., Yale Univ., New Haven, CT, 1989.

[20] A. S. Gordon and N. V. Iuppa, "Experience management using storyline adaptation strategies," in *Proc. Technol. Interactive Digit. Storytelling Entertain. Conf.*, 2003, pp. 19–30.

[21] J. Rickel and W. Johnson, "Animated agents for procedural training in virtual reality: Perception, cognition, and motor control," *Appl. Artif. Intell.*, vol. 13, pp. 343–382, 1999.

[22] R. M. Young, M. Pollack, and J. Moore, "Decomposition and causality in partial-order planning," in *Proc. 2nd AI Planning Scheduling Conf.*, 1994, pp. 188–193.

[23] J. S. Penberthy and D. S. Weld, "UCPOP: A sound, complete, partial order planner for ADL," in *Proc. 3rd Int. Conf. Principles Knowl. Represent. Reason.*, B. Nebel, C. Rich, and W. Swartout, Eds., San Mateo, CA, 1992, pp. 103–114.

[24] E. Sacerdoti, "The nonlinear nature of plans," in *Advance Papers 4th Int. Joint Conf. Artif. Intell.*, 1975, pp. 206–214.

[25] S. Kambhampati, C. A. Knoblock, and Q. Yang, "Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning," *Artif. Intell.*, vol. 76, pp. 167–238, 1995.

[26] S. Hanks and D. S. Weld, "A domain-independent algorithm for plan adaptation," *J. Artif. Intell. Res.*, vol. 2, pp. 319–360, 1995.

[27] M. Riedl and R. M. Young, "An intent-driven planner for multi-agent story generation," in *Proc. 3rd Autonom. Agents Multi-Agent Syst. Conf.*, 2004, pp. 186–193.

[28] A. Ramachandran and R. M. Young, "Providing intelligent help across applications in dynamic user and environment contexts," in *Proc. 10th Int. Conf. Intell. User Interfaces*, New York, NY, 2005, pp. 269–271.

[29] M. Riedl and R. M. Young, "An objective character believability evaluation procedure for multi-agent story generation systems," in *Proc. Int. Conf. Virtual Agents*, Kos, Greece, 2005, pp. 278–291.

[30] D. B. Christian and R. M. Young, "Comparing cognitive and computational models of narrative structure," in *Proc. 19th Nat. Conf. Artif. Intell.*, 2004, pp. 385–390.
[31] A. C. Graesser, K. L. Lang, and R. M. Roberts, "Question answering in the context of stories," *J. Experiment. Psychol., General*, vol. 120, pp. 254–277, 1991.

Justin Harris was born in Chapel Hill, NC, in 1981. He received the B.S. and M.S. degrees in computer science from North Carolina State University, Raleigh, in 2003 and 2005, respectively.

He has worked as a Software Architect at TransLoc, and as an AI Programmer at Virtual Heroes. He is currently a Senior Software Applications Engineer at Red Hat, Raleigh, NC. His professional interests include open source software, web technologies, and applied artificial intelligence.

R. Michael Young (M'98–SM'07) was born in Salinas, CA, in 1961. He received the B.S. degree in computer science from the California State University at Sacramento, Sacramento, in 1984, the M.S. degree in computer science with a concentration in symbolic systems from Stanford University, Stanford, CA, in 1988, and the Ph.D. degree in intelligent systems from the University of Pittsburgh, Pittsburgh, PA, in 1998.

He has worked as a Software Engineer at Hewlett-Packard, a Research Scientist at FMC Corporation, and a Research Scientist at Rockwell Palo Alto Research Laboratory. He worked as a Postdoctoral Fellow at the Robotics Institute at Carnegie Mellon University. He is currently an Associate Professor of Computer Science and Co-Director of the Digital Games Research Center, North Carolina State University, Raleigh, NC. His research interests span the computational modeling of narrative, the use of artificial intelligence techniques within computer games and virtual worlds, planning, computational linguistics, games-based learning, and automated cinematography.

Dr. Young is a senior member of the Association for Computing Machinery and a member of the Association for the Advancement of Artificial Intelligence and of the International Game Developers' Association.