Believable Agents and Intelligent Scenario Direction for Social and Cultural Leadership Training

Mark O. Riedl
Institute for Creative Technologies
13274 Fiji Way, Marina del Rey, CA 90292
riedl@ict.usc.edu

Andrew Stern
Procedural Arts LLC
Portland, OR
andrew@proceduralarts.com

Keywords:
Automated Scenario Direction; Believable Agents; Leadership Training; Interactive Storytelling

ABSTRACT: *Simulation provides an opportunity for a trainee to practice skills in an interactive and reactive virtual environment. We present a technique for social and cultural leader training through simulation based on a combination of interactive synthetic agents and intelligent scenario direction and adaptation. Social simulation through synthetic characters provides an engaging and believable experience for the trainee. In addition, the trainee is exposed to a sequence of relevant learning situations where the trainee can practice problem-solving under particular conditions. An Automated Scenario Director provides high-level guidance to semi-autonomous character agents to coerce the trainee's experience to conform to a given scenario. When the trainee performs actions in the virtual world that cause the simulation state to deviate from the scenario, the Automated Scenario Director adapts the scenario to resolve any unexpected inconsistencies, thereby preserving the trainee's perception of self control while still retaining any relevant learning situations.*

1. Introduction

Research on leadership development shows that expertise is gained through experience and by taking the time to reflect on the lessons learned from an episode. The problem with learning by experience in a military context, however, is that some decisions and behaviors can have devastating or fatal consequences if mistakes are made, making such lessons very expensive. One way that one can learn from experience without "being there" is by playing a role in a game or interactive simulation. While a simulation may provide an excellent opportunity for a trainee to practice skills in an interactive and reactive virtual environment, evidence suggests that "guided discovery" is a powerful method of learning (vanLehn, 1996). Consequently, we believe the capability to manage the experience of a trainee in a virtual simulation may enhance the effectiveness of virtual training environments by increasing the likelihood that the trainee will be exposed to a specified sequence of relevant learning situations in which lessons can be learned and/or skills can be practiced.

While live training exercises are an essential part of military leader training, such exercises are primarily used to train procedural skills well defined by doctrine and best practices. In the current operating environment, however, there is a downward migration of leadership tasks (McCausland and Martin, 2001) such that the decisions and actions of small unit leaders and soldiers can take on strategic significance. For example, a small unit leader may be placed in a situation where he or she must interact with civilians and non-government organizations while deployed to foreign lands. In the research we present here, we are working in the domain of leader training in social and cultural awareness contexts. In such a domain, the trainee needs to experience scenarios in which he or she is forced to interact with foreign cultures and make local decisions with strategic and 2nd and 3rd order effects. The emphasis is not as much on procedural learning objectives, but on placing the trainee in situations where difficult decisions must be made.

We believe that effective social and cultural leadership training can be tackled by a combination of reactive social simulation and automated scenario direction. The domain of social and cultural leadership training naturally places the trainee in situations in which he or she must interact with other people. In a computer simulation that supports this domain, it is beneficial to

have synthetic humanoid agents that personify characters – friendly or otherwise – capable of interacting with the trainee through natural language and gesture (Swartout et al., 2001). Our approach uses intelligent semi-autonomous character agents that are designed to be interactive and believable. A believable agent is one that possesses behavioral traits that facilitate one's suspension of disbelief that the agent is a real person (Bates, 1992).

While social simulation is sufficient for an engaging experience we wish for the trainee's experience to be structured. We require the non-player characters (NPCs) to enact a scenario, particularly one that possesses one or more relevant learning situations. To accomplish structuring over the trainee's experience, an *Automated Scenario Director* provides high-level guidance to the social agents that inhabit the simulation. The Automated Scenario Director ensures a coherent progression towards relevant learning situations. When necessary the Automated Scenario Director can adapt the scenario to protect the coherence of the trainee's experience.

Our approach to managing the experience of a trainee in a social and cultural simulation is not dissimilar to the role of an Observer/Controller (O/C) in a live training exercise. However, our approach affords a greater degree of flexibility and adaptability than typically seen in exercises based around learning objectives.

2. Example Scenario

To motivate the problem, we present the following simple training scenario in which a trainee takes the role of a leader placed in a situation in which he or she must interact socially and culturally with a foreign population. The background is that the trainee is part of a peacekeeping mission deployed to a foreign country. As a measure of good will, the peacekeeping forces have set up a new marketplace that is modern and secure from insurgent attacks. The trainee plays the role of a captain in charge of maintaining the security of civilian merchants and buyers in the new marketplace.

The scenario is expected to unfold as follows. While the trainee is engaged in daily procedures concerning the new marketplace, a heated argument breaks out between two merchants that do business near each other. One merchant, named Saleh, is accusing the other, named Mohammed, of luring away his customers. Afterwards, Saleh approaches the trainee and complains that the presence of the peacekeeping troops is impinging on his ability to do business in the marketplace. He makes a dire prognostication that violence could ensue. Mohammed, in contrast appears to be nothing but friendly. Later that day, Mohammed slips away from the marketplace and returns concealing an improvised explosive device (presumably acquired from his insurgent conspirator). When Saleh steps away from his place of business, Mohammed plants the bomb there. Shortly afterwards, the bomb goes off. Fortunately for all involved, the bomb is a dud, but the marketplace is nonetheless left in a state of chaos, panic, and confusion.

The scenario is not contingent on the trainee completing any particular learning objective, but instead is designed to expose the trainee to certain relevant learning situations in which the trainee can practice certain skills (possibly associated with learning objectives) or practice decision-making and problem-solving. The primary relevant learning situation occurs at the end of the scenario when the dud explosion plunges the marketplace into chaos and uncertainty. It is the trainee's job to restore order, provide assurances to the people doing business there, assess blame, and possibly even make arrests. Given the expectations created by the scenario, it would justifiable if the trainee were to suspect Saleh of instigating the attack. Consequently, the lead up to this situation is just as important as the situation. Otherwise, the training simulation could start at the point of the explosion. The assumption is that if the trainee is engaged socially and culturally and comes to be familiar with the primary characters in the marketplace, the trainee may be able to recognize that Mohammed has a deep-rooted animosity towards the peacekeeping force and that Saleh is merely an outspoken but harmless detractor. Indeed, with initiative, the trainee may even be able to prevent the attack.

Note that if the trainee succeeds in preventing the attack by catching Mohammed with the explosive device then it will be impossible for Mohammed to plant the bomb and therefore impossible for the bomb to go off and establish the primary relevant learning situation. How would an Observer/Controller handle this dilemma in a live exercise? The O/C could do the realistic thing, removing the bombing from the scenario at the risk of eliminating an important learning situation. The O/C could alternatively "cheat" by changing the simulation to have the explosive device magically planted at the risk of making the trainee feel as if his or her actions were irrelevant.

In the remainder of the paper, we describe a novel technique for trainee experience management that is

designed to excel at such social and cultural simulation scenarios as that described above. The key is the ability to automatically recognize dilemmas like the one described here and handle them without clobbering relevant learning objectives or cheating.

3. Believable Characters

Social simulation is achieved through a collection of non-player characters (NPCs) that are reactive and appear intelligent, motivated, and reactive. Our agents are partly composed of a broad, general collection of local autonomous behaviors that are designed to afford suspension of disbelief. Local autonomous behaviors (LABs) such as working, running errands, shopping, etc. supply agents with a "rich inner life." The objective is not to have agents that are competent reasoning agents, but agents that *appear* to be intelligent, motivated, emotional, and consequently believable. This emphasis on appearance is referred to as a "broad but shallow" approach to agents (Bates, 1992). That is, agents can perform a wide repertoire of behaviors in a convincing manner but without performing "deep" reasoning. For our domain, this is sufficient because social and cultural interaction involves routine behavior and simple problem-solving instead of novel or complex cognitive tasks. In that sense, agent knowledge can be compiled into reactive behavior selection mechanisms.

For pedagogical reasons, it is important that NPCs are capable of acting to bring about a specific scenario. Scenario-specific interactive events such as confronting the player and acquiring, planting, and detonating an explosive device, are carried out by narrative directive behaviors (NDBs). Narrative directive behaviors are incorporated into the agents' behavior repertoires before run-time and triggered by high-level narrative direction from the Automated Scenario Director (see Section 4). These scenario-specific behaviors are designed to modulate, mix with, and/or override local autonomous behaviors.

3.1. Agent Architecture

To achieve the desired life-like qualities we implemented our agents using the reactive planning language ABL (A Behavior Language) (Mateas and Stern, 2004) using a behavioral infrastructure licensed from the Procedural Arts Behavior Library (PABL). The ABL language and PABL infrastructure were initially created for the interactive drama, *Façade* (Mateas and Stern, 2005). ABL is based on the Oz Project (Bates, 1992) believable agent language Hap (Loyall, 1997). ABL and its parent language Hap are designed to support the detailed expression of artistically-chosen personality, automatic control of real-time interactive animation, and architectural support for many of the requirements of believable agents (Loyall, 1997).

```
sequential behavior Foreshadow_Bomb ()
{
  precondition { (LocationWME
                      agent == me
                      location == my_booth)
                 (LocationWME
                      agent == player
                      location != my_booth) }
  subgoal Stage_To_Player();
  subgoal Speak_Bomb_Dialogue();
}

parallel behavior Speak_Bomb_Dialogue ()
{
  subgoal Speak_Dialog("I wish someone
                  would blow this place up");
  subgoal Gesture_Agitated();
}

sequential behavior Gesture_Agitated ()
{
  precondition { ... }
  act Play_Gesture(3);
}

sequential behavior Gesture_Agitated ()
{
  precondition { ... }
  act Play_Gesture(5);
}
```

Figure 3.1: Example ABL behaviors

In ABL, an activity (e.g., walking to the user, or speaking a line of dialog) is represented as a goal, and each goal is supplied with one or more behaviors to accomplish its task. An active goal chooses one of its behaviors to try. A behavior is a series of steps, that can occur sequentially or in parallel, that accomplish a goal. Preconditions are used to determine behavior applicability by matching against working memory elements (WMEs) that make up the agent's subjective knowledge about the world. A behavior may itself have one or more subgoals.

Figure 3.1 shows a simple example of ABL behavior specifications that cause an agent to walk to the player's location and then simultaneously speak a line of text while gesturing. The precondition of the `Foreshadow_Bomb` behavior matches the behavior against the agent's knowledge of the world to see if the behavior is applicable. The precondition clause searches the agent's working memory for two working memory elements that signify that the agent knows (a) it is at location `my_booth` and (b) the player is not. If applicable, this behavior sequentially tries to achieve the two subgoals in the given order. The `Stage_To_Player` behavior moves the agent to the player's location and orients it for conversation. The `Speak_Bomb_Dialogue` behavior is a parallel behavior, meaning the agent will simultaneously

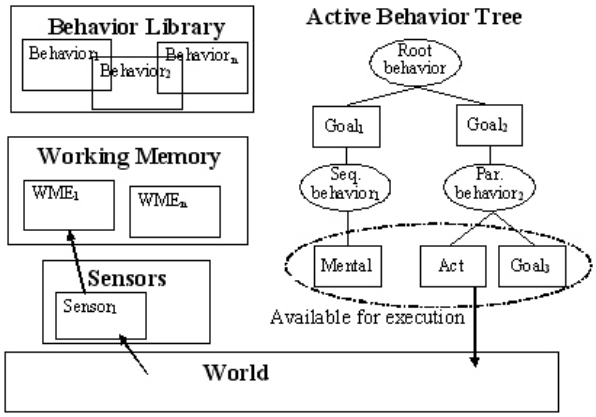

Figure 3.2: ABL runtime architecture (Mateas and Stern, 2004)

attempt to achieve the given subgoals: speaking a line of dialogue and gesturing agitatedly. Note that there may be many appropriate behaviors for gesturing, of which the agent picks the most applicable behavior. In this case, the preconditions help the agent pick the most appropriate alternative given the specific context. The subgoaling encapsulates this decision so that the parent behavior doesn't have to encode the logic of determining which gesture to perform for all possible conditions. The `Gesture_Agitated` behaviors specify *acts*, which, unlike subgoals, invoke function calls to the low-level graphics and animation engine controlling the agent's avatar, in this case playing a pre-specified indexed gesture animation.

To keep track of all the active goals and behaviors and subgoal relationships, ABL maintains an active behavior tree (ABT) and working memory. The ABT is a tree rather than a stack because some behaviors execute their steps in parallel, thus introducing parallel lines of expansion in the program state. The agent's working memory contains WMEs that are updated dynamically and continuously through sensor routines engaged with the virtual world. The architecture of an ABL agent appears in Figure 3.2. Typically, once a behavior completes all of its steps, the behavior and the goal that spawned it succeed. However if any of the behavior's steps fail, then the behavior itself fails and the goal attempts to find a different behavior to accomplish its task, finally failing if no such alternative behavior can be found.

Further, to harness the dramatic power of multi-agent teams of characters, ABL supports authoring of joint goals and behaviors (Mateas and Stern, 2004). When a goal is marked as joint, ABL enforces coordinated entry into and exit from the team members' behaviors chosen to accomplish the goal. This coordination is transparent to the programmer and analogous to the STEAM multi-agent coordination framework (Tambe, 1997). The driving design goal of joint behaviors is to combine rich semantics for individual expressive behavior with support for the automatic synchronization of behavior across multiple agents.

In contrast to standard imperative languages one might use to control agents (e.g. C++, Java), in ABL an author can, in relatively few lines of code, specify collections of goals and behaviors that can cleanly intermix solo and/or joint character actions, modulate their execution based on the continuously sensed state of the world, and perform local, context-specific reactions to a user's actions. While ABL serves a similar purpose to other reactive planners such as Soar and RAPS (Firby, 1989), the automated management of joint goals and the ability for an agent to perform parallel behaviors makes it particularly flexible for life-like agents. Additionally, ABL gives behaviors reflective access to the current state of the ABT, supporting the authoring of meta-behaviors that match on patterns in the ABT and dynamically modify other running behaviors. This is particularly useful for creating behaviors that can react immediately to the user. Supported ABT modifications include succeeding, failing or suspending a goal or behavior, and modifying the annotations of a subgoal step, such as changing the persistence or priority.

3.2. Behavior Authoring

There are two broad categories of agent behaviors that must be authored: local autonomous behaviors and narrative directive behaviors. Local autonomous behaviors (LABs) are the somewhat generic, re-usable "inner life" activities such as working, running errands, shopping, etc. Narrative directive behaviors (NDBs) are scenario-specific and are triggered by the Automated Scenario Director.

Local Autonomous Behaviors. Local autonomous behaviors (LABs) are implemented as a loosely structured collections of sub-behaviors called "LAB goals", that depend on and assert simple events in episodic memory. For example, the *opening the store* LAB may involve the agent *unlocking the store*, *unpacking boxes*, *chatting with assistant*, and *displaying new goods*. Each of these parts is implemented as its own simple LAB goal. *Displaying new goods* may depend on *unlocking the store* having been completed. Note that there can be partial ordering such that *chatting with assistant* is just as likely to occur after *unlocking the store* as *unpacking boxes*. Or, in such a case, *chatting* may end up getting skipped altogether since no other LAB goals depend on it. User interactions, should they occur, can easily get inserted during or in between the loosely organized LAB goals.

Each individual agent is responsible for selecting and sequencing their local autonomous behaviors. LABs manage their own sequencing. Whenever a new LAB needs to run, either upon start-up or once the previous LAB completes, each LAB may make a bid for how important it is to run next. Bids are numeric values calibrated in a range from low importance to extremely urgent. A LAB chooses a bid strength depending upon current world conditions, such as time of day, and episodic memory as needed; if the LAB does not care to run, it does not bid at all. A simple arbitration behavior makes a weighted probability choice among the bids, and the chosen LAB begins executing.

Narrative Directive Behaviors. By contrast, narrative directive behaviors (NDBs) are more tightly structured collections of sub-behaviors, intended to perform more important and more sophisticated parts of the scenario. Further, user interaction afforded in NDBs usually needs to be richer and more responsive than in LABs. The collection of sub-behaviors that constitute an NDB are organized around the dramatic beat (Mateas and Stern, 2005), a component of the PABL infrastructure. In the theory of dramatic writing, the beat is the smallest unit of dramatic action (McKee, 1997). A beat is a ~60-second-long dramatic interaction between characters such as a shared experience (e.g., witnessing a bombing), or a brief conflict about a topic (e.g., the user questioning an agent), or the revelation of an important secret. Beats are organized around a collection of "beat goal" behaviors, the dramatic content that the beat is designed to communicate to the user through animated performance.

The PABL authoring strategy for handling user interaction within a beat is to specify the "canonical" beat goal behavior logic (i.e., what dramatic performance the author intends the beat to accomplish), as well as a collection of beat-specific handler behaviors that modify this default logic in response to user interaction. In order to modify the default logic, the handler behaviors make use of meta-ABL functionality to modify the ABT state. While handler behaviors can in principle arbitrarily modify the ABT state, most fall into one of two general classes: mix-in handler behaviors that add an additional beat goal behavior in the middle of a beat while keeping the rest of the sequencing the same, and re-sequencing handler behaviors that more radically reorganize the beat goal sequence. The ability to factor behavior into sequences that achieve longer-term temporal structures, and meta-behaviors that modify these longer-term temporal structures, is a powerful idiom enabled by ABL's support for reflection.

Each interaction handler behavior is a demon that waits for some particular type of user interaction and "handles" it accordingly. User interaction includes dialogue interaction (the user can speak to the characters at any time by entering discourse acts, e.g. "disagree Saleh") and physical interaction (e.g. the user takes action such as "arrest Saleh"). Every NDB specifies some beat-specific handlers; additionally, there are more generic LAB handlers for handling interactions for which there are no beat-specific responses supplied by the current beat.

4. The Automated Scenario Director

There is a trade-off between scenario coherence and the trainee's perception of self-control (Riedl, Saretto, and Young, 2003). On one hand, instructional designers will want to ensure a coherent progression of scenario events that lead the trainee through a sequence of relevant learning situations. On the other hand, in a training simulation, the trainee needs to observe a realistic, populated social environment and be able to problem-solve – to perform actions and make decisions. In our system, the trainee is allowed to perform any of a wide repertoire of communicative and physical actions at any time. Simulation with social agents alone, however, is not enough to ensure that the trainee is exposed to relevant learning situations in an appropriate and contextual order.

To ensure that the trainee's experience is managed and that the appropriate sequence of relevant learning situations occur, an agent called the Automated Scenario Director acts as an unseen over-mind to coerce the trainee's experience to conform to a given scenario. Specifically, the Automated Scenario Director maintains a representation of the expected sequence of events that make up the scenario. From this representation, the scenario director derives and distributes directives to the NPCs to achieve certain conditions necessary to drive the scenario forward. For example, the Automated Scenario Director would direct the agent representing the Saleh character (from the example in Section 2) to establish the condition that the trainee distrusts the character – something that that agent might not choose to do if left to it's own devices.

Consistent with the design decision that the trainee can perform any of a wide repertoire of actions at any time, the automated scenario director has a second responsibility: to monitor the simulation environment, detect inconsistencies between the simulation state and the expected scenario, and to reconcile any inconsistencies. This is essential in balancing the trade-off between scenario coherence and trainee self-

```
(operator Plant-Bomb
  :parameters (?character ?bomb)
  :precondition ((has ?character ?bomb)
                 (:not (detained ?character))
  :effect ((armed ?bomb) (planted ?bomb)
           (:not (has ?character ?bomb))))

(operator Acquire-Object
  :parameters (?acquirer ?object ?owner)
  :precondition ((has ?owner ?object)
                 (:not (detained ?acquirer))
                 (:not (detained ?owner)))
  :effect ((has ?acquirer ?object)
           (:not (has ?owner ?object))))
```

Figure 4.1: Example planning operators

control because the trainee may perform actions that make it impossible for the scenario director to progress towards the desired relevant learning situations. For example, the trainee could decide to apprehend Mohammed before he plants the improvised explosive device in the marketplace. In this instance, the trainee has created an inconsistency between the simulation state (e.g. Mohammed is detained) and the expected scenario representation (e.g. it must not be the case that Mohammed is detained for the bomb to be planted). When inconsistencies arise, the scenario is adapted to reconcile the inconsistencies.

4.1. Computational Representation of Scenario

Scenario adaptation requires that the scenario be represented computationally in a way that can be reasoned over by an artificial intelligence system. Following (Riedl, Saretto, and Young, 2003), we represent scenarios as partially-ordered plans. A plan contains steps – events that change the state of the world – and annotations that explicitly mark the temporal and causal relationships between all steps in the plan, defining a partial order indicating the steps' order of execution (Weld, 1994).

Figure 4.1 shows un-instantiated operators from a domain operator library that could be used in the example scenario. The parameters are variables that will be bound as appropriate to such things as character or object instances. Preconditions are conditions in the world that must be true for the operator to be applicable. Effects are conditions in the world that become true after successful execution of the instantiated operator. The first operator in Figure 4.1 describes an action where a character plants a bomb. Preconditions state that that character must have the bomb in his or her possession and not be detained. The effects of the operator are that the bomb is planted and armed and that the character no longer has the bomb in his or her possession. The second operator in Figure 4.1 describes an action whereby one character acquires an object from another character. Preconditions state that one character, the owner has the object and that both characters are not detained. The effects of the operator are that the acquiring character has the object and the original owner no longer has the object.

Other annotations, called *causal links*, are used to mark all causal relationships between the steps in the plan. In a plan, a causal link relates the effect of one plan step to a precondition of another plan step that is temporally constrained to occur later than the first operator. Specifically, a causal link connects two plan steps s_1 and s_2 via condition e (written $s_1 \rightarrow^e s_2$) when s_1 establishes the condition e in the world needed by subsequent action s_2 in order for s_2 to execute (Weld, 1994). A plan is not considered complete unless every precondition of every plan step is satisfied by a causal link.

Causal dependency planning (Weld, 1994) operates in a backward chaining fashion as a process of flaw repair. A flaw is an annotation on an incomplete plan that specifies how the plan will fail to execute. One type of flaw is an open condition, where a plan step has a precondition that is not causally satisfied by a preceding step or the initial world state. Open conditions are repaired by extending a causal link from a preceding step in the plan that has an effect that unifies with the open condition. If an applicable step does not exist, a new step is instantiated from a library of operators.

Besides satisfying open conditions, the planner also resolves causal threats. A *causal threat* occurs when the effect of some step, s_t, negates condition e of a causal link relating two steps, s_1 and s_2. s_1 establishes some condition e in the world which s_2 relies on for execution. But after s_1 occurs and before s_2 occurs, step s_t may occur, causing e to become false in the world and jeopardizing s_2's ability to succeed. Causal threats are repaired by temporally ordering s_t before s_1 or after s_2.

4.2. Anticipating Necessary Scenario Adaptations

Using planning structures to model scenarios is advantageous because a plan can be analyzed for points in which failure can occur due to unpredictable and interactive behaviors performed by the trainee. We use a technique similar to that described in (Riedl, Saretto, and Young, 2003) to analyze the causal structure of the scenario to determine all possible inconsistencies between plan and simulation state that can occur during the entire duration of the scenario. For every possible inconsistency that can arise that threatens a causal link in the plan, an alternative scenario plan is generated. We have modified the

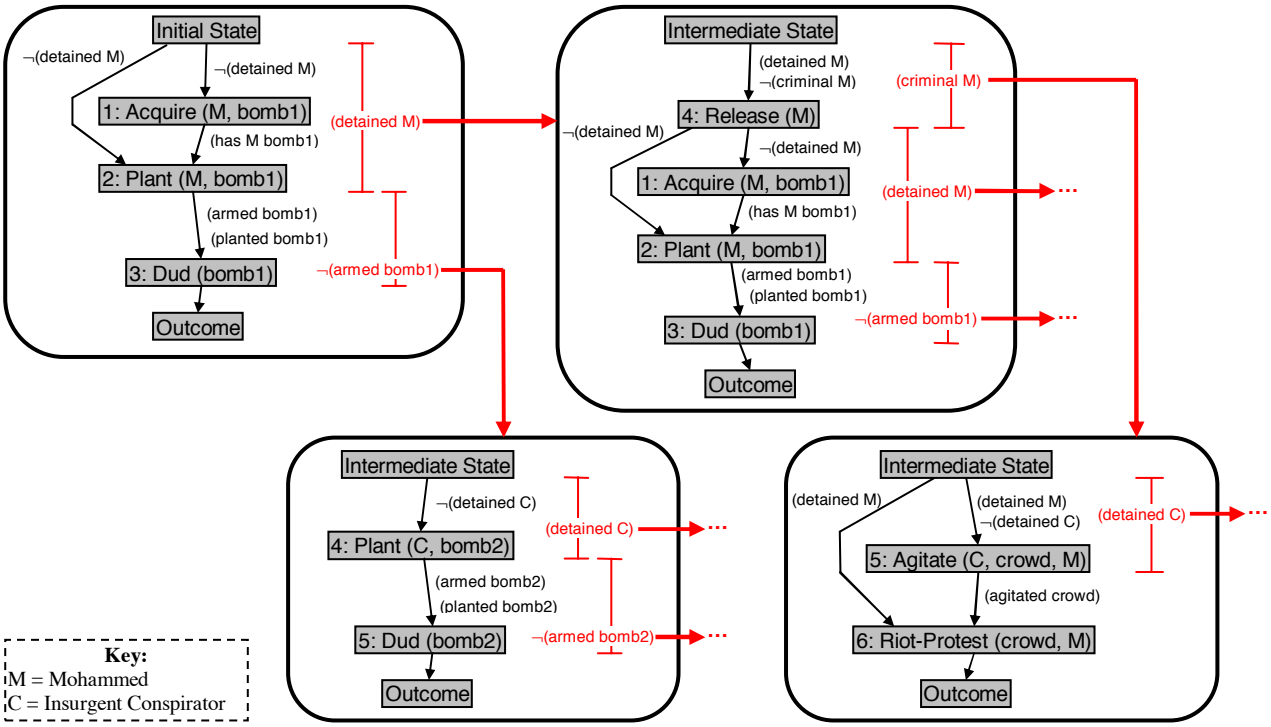

Figure 4.2: Portion of the generated branching scenario plan

original algorithm to use a tiered replanning approach. For each potential inconsistency that can arise, first the system attempts to repair the causal link that is threatened by the inconsistency. Barring that, the system attempts to remove any events that were dependent on the threatened causal link and then repair the plan by filling in events required to restore causal coherence. Finally, if all else fails, the system attempts to select new goals and relevant learning situations and rebuild the scenario plan.

Scenario replanning can be performed offline to avoid delays due to computation (Riedl, Saretto, and Young, 2003). The result of this process is a tree of contingency plans in which each plan represents a complete scenario starting at the initial world state or at the point in which an inconsistency can occur (Riedl and Young, 2006). If the user performs an action that causes an inconsistency that threatens the scenario plan, the system looks up the appropriate branch in the tree of contingencies and seamlessly begins directing the believable agents based on the new scenario plan.

4.3. Example of Scenario Adaptation

Continuing the example scenario from Section 2, suppose the training system is instantiated with a pre-constructed plan (shown in the upper left-hand corner of Figure 4.2). Except for the plan steps and causal links in the original scenario plan, everything in the tree of contingency plans is automatically generated, including the potential inconsistency annotations of the original scenario itself. For the purposes of this discussion we have simplified the scenario further into the following steps:

1. `Mohammed` acquires `bomb1`
2. `Mohammed` plants `bomb1` in the marketplace
3. `bomb1` goes off as a dud.

The directed connections between steps are causal links indicating what must be true for a step to be applicable and which preceding step establishes that condition. This plan represents the scenario that will execute if the trainee does not inadvertently (or intentionally) cause an inconsistency between the simulation state and the plan structure. The plan is annotated with two intervals in which inconsistencies will threaten the causal coherence of the plan:

- `Mohammed` is detained before he completes the planting of `bomb1`
- `bomb1` is disarmed before it goes off.

The former can occur if the trainee has Mohammed arrested. The latter can occur if the trainee finds the bomb and calls in a bomb squad.

Each potential inconsistency annotation links to a contingency plan that repairs the inconsistency, should it occur. The detainment of Mohammed links to a plan (upper right of Figure 4.2) that is repaired by the addition of a step that releases Mohammed from custody, ostensibly because he has not committed a crime. If the trainee searches Mohammed and

Figure 5.1: Screenshot of trainee (central avatar) confronting a non-player character (right)

Mohammed in fact has `bomb1` in his possession, he will be marked as a criminal. This does not cause an inconsistency that threatens the original plan. But once detained, if Mohammed is marked as a criminal, he cannot be released, instantly causing a second transition to the plan in the lower right of Figure 4.2.

The plan in the lower left of Figure 4.2 represents the scenario in which `bomb1` is found and disarmed by the trainee. In this case, the outcome of the original scenario cannot be achieved. This plan repairs the original scenario by having an insurgent conspirator step in and complete the attack on the marketplace with a second bomb.

The plan in the lower right of Figure 4.2 represents the scenario in which Mohammed is caught with a bomb and permanently detained. Like the previously described contingency, the outcome of the original scenario can still be achieved by having an insurgent conspirator complete the attack with a second bomb. However the planner can find a more appealing variation. The insurgent conspirator uses the detention of Mohammed to agitate a crowd. The crowd then riots in protest of the player's actions.

5. Putting it all Together

Believable character agents, the Automated Scenario Director, and a variant of the example scenario from Section 2 have been combined into a prototype called IN-TALE (the Interactive Narrative Tacit Adaptive Leader Experience). The system is built on top of a 3D computer game engine. Figure 5.1 shows a screenshot of the trainee (central avatar) being confronted by an NPC. A set of believable social agents are implemented as semi-autonomous, intelligent agents in ABL. Each ABL agent controls a single virtual avatar in the game engine. The Automated Scenario Director receives state updates from the game engine via an interface described in (Riedl, 2005).

5.1. High-Level Direction

As dictated by the scenario plan, the Automated Scenario Director sends high-level directives to the ABL agents in one of two forms:

- Direction to achieve some world state.
- Direction to avoid causing states that are inconsistent with the scenario plan.

Both types of direction are essential. The first type of direction is the primary mechanism through which the automated director pushes a scenario forward and is necessary because the NPCs cannot be relied on to autonomously make decisions that are always favorable to the automated director. The second type of direction is important because the autonomy of NPCs means that, without constraints, NPCs could perform behaviors that create inconsistencies between the simulation state and the scenario plan. The ability to perform actions that cause such inconsistencies is a privilege of the trainee only.

The directives from the Automated Scenario Director to the NPCs are not detailed instructions. Directives are goals the NPC must adopt, in terms of declarative world state change. Character agents are left to determine the best way to achieve the directives, barring any behavior case world states explicitly prohibited by the Director. This gives the agents the leeway to engage in behaviors that are believable, achieve scenario goals, and take full advantage of the current world situation, e.g. executing one or more NDBs as described in Section 3.

The paradigm of high-level directives and low-level agent autonomy opens up the possibility of an agent selecting joint behaviors. A joint behavior is a method – one of many methods known to the agent for achieving a goal – that co-opts the participation of other NPCs in the world for close coordination of activity and/or dramatic effect. For example, if the agent representing the Mohammed character is directed to acquire an explosive device (e.g. `(has Mohammed bomb1)`), that character might not be able to do so while the trainee is in close proximity. Assume there is a directive that `(knows player (has Mohammed bomb1))` should never become true. The agent may select a joint behavior – one of many known methods for acquiring an object – that in conjunction with another agent, possibly a bystander, creates a diversion. The joint behavior provides coordination so that Mohammed knows to slip off while the player is distracted.

5.2. Mixing Autonomy and Scenario Directives

To avoid the appearance of agents that schizophrenically switch between goals (Sengers, 2003) – in this case goals autonomously selected for believability and goals demanded by the Scenario Director – behaviors selected to achieve the appearance of believability (LABs) and behaviors that achieve scenario goals (NDBs) must mix seamlessly. Local autonomous behaviors can run in parallel and/or interleave with behaviors selected to achieve scenario goals. "Real life" behaviors can be modulated to believably blend with the high-level scenario behaviors imposed on them. Modulations of LABs include: timing alteration to accommodate the needs of the scenario; reducing the number of physical resources required to avoid conflicts with scenario-driven behaviors; and avoiding actions that would violate overall believability in any way.

Mixing NDBs with LABs involves annotating NDBs and LABs with the resources they require: location requirements, object requirements, emotional state requirements and so on. When an NDB prepares for execution in order to fulfill a directive from the Automated Scenario Director, the NDB announces its resource requirements. LABs are coded with behavior variations in order to gracefully degrade their performance to accommodate the needs of the more important NDBs, while still behaving believably. When a LAB is required to interrupt or even abort its execution to serve the NDB's needs, it selects from a variety of short transition-out sub-behaviors to believably "glue", i.e., explain why. For example, if a *cleaning the store* LAB needs to be truncated or aborted in order for the agent to participate in NDBs to acquire and plant a bomb, the agent may choose to insert some dialog to the effect of, "Hmmm, the store is pretty clean today… I think sweeping can wait till tomorrow". Similarly, whatever LAB begins after the NDB ends can select from transition-in sub-behaviors similarly "gluing" the agent's behavior back into its daily routine.

6. Authoring

One advantage of the generative scenario adaptation technique described in Section 4 is that only a single exemplar scenario plan must be authored. The system in an offline process analyzes the causal dependencies of events in the scenario plan determines all ways in which trainee actions can conflict with the scenario and generates contingency scenarios. The result is analogous to a branching story (Riedl and Young, 2006). Pre-scripted branching stories such as those used in computer games typically have either few decision points or low branching factors (the number of alternatives in any given decision point). One reason for this is the combinatorial complexity of authoring branching stories (Bruckman 1990); as the number of decision points grows, the amount of story content that must be authored grows exponentially. The generative planning approach used here mitigates this effort by taking the effort scenario adaptation out of the hands of the system designers.

The planner used to re-plan scenarios however cannot operate in a vacuum – it requires knowledge about the virtual world in which the scenario is set. The more knowledge the planner has, the greater the number of content variations can be generated. The planner requires a library of plan operators and schemata (referred to as a domain theory). A domain theory describes in abstract terms all actions that are possible for story world characters to perform. It may be the case that the amount of knowledge required by the system is greater than the length of the plan generated. However, the advantage is that the same knowledge is used recursively over and over as each branch in the tree of scenario contingencies is generated. For branching narratives that are long and/or have a high branching factor, the amount of generated content can quickly exceed the size of the world domain authoring effort.

How hard is it to author the initial exemplar scenario? It is not trivial because the scenario must be represented as a plan with causal links. However, tools that link into the planner's domain theory knowledge-base can ease the process. Furthermore, by hooking into the planning algorithm itself, the authoring process could in the future be a mixed-initiative human-computer process (Burstein and McDermott, 1996) where the human author expresses creative intent and an intelligent authoring tool fills in the technical details, including causal links. Unfortunately, no such tool has been developed for this specific system.

Character agent authoring is knowledge-intensive. However, the "broad, shallow" approach means that autonomous character behaviors are easy to author. Knowledge engineers need only focus on the appearance of correct behavior without regard for agent reasoning or deep decision making. Local autonomous behaviors (LABs) can be specific to an individual NPC in the case that the character expresses a very unique personality or characteristic. But mostly, we anticipate that LABs will be general enough be reused or can be customized from generic templates. For example, a *greeting a customer* behavior could involve the same physical actions from

agent to agent, with only dialog variations customized to the agent. Further, some low-level behaviors, such as locomoting, operating a cash register, cleaning the store and so on, are generic and can be re-used from agent to agent, with little or no customization per agent.

Narrative directive behaviors (NDBs) can be considered a general pre-compiled plan or method for accomplishing world state change. The challenge of authoring NDBs is to consider alternative methods for achieving the same world state condition for a wide variety of possible circumstances. It is possible that narrative directive behaviors can be pre-computed by intelligent agent-based planning systems although we have yet to explore this possibility in any depth. Once authored, we anticipate a wide degree of re-use.

7. Related Work

There has been previous work on building systems that could be described as automated Observer/Controllers. The Virtual Observer/Controller (Banta et al., 2005) provides coaching feedback on low-level trainee skill attempts. The VOC has a limited ability to change the scenario to take advantage of trainee mistakes to show the negative consequences of actions. The Advanced Embedded Training System (AETS) (Zachary et al., 1999) applies intelligent tutoring to simulation-based training of air defense radar operators. AETS monitors the trainee for correct procedural actions when responding to simulated radar contacts and can provide feedback but otherwise doesn't adapt the scenarios.

Much of the work described here is influenced by research in the computer game and entertainment domain. Interactive Storytelling Systems attempt to tell a story in which the user is able to make decisions and perform actions that dynamically affect the direction and/or outcome of the story. In particular, the Mimesis system (Riedl, Saretto, and Young, 2003; Young et al., 2004) uses a generative approach to interactive story. A story plan is generated and executed by non-autonomous NPCs. When the user performs an action that threatens to undo a condition in the story plan, the system dynamically re-plans the remainder of the story to bring about dramatic goals. *Façade* (Mateas and Stern, 2005) uses a reactive engine to dynamically analyze the current scene according to dramatic principles and select the next mini-scene (called a beat) for the NPCs to act out. *Façade* uses the ABL behavior specification language to endow the NPCs with reactivity and believability. The Interactive Drama Architecture (IDA) (Magerko, 2005) takes human-authored story scripts and uses that to direct NPCs. IDA uses predictive modeling of the user to prevent the user from deviating from the prescribed story.

Interactive Storytelling techniques have been recently been applied to training and education. The Tactical Language Training system (Johnson, Marsella, and Vilhjálmsson, 2004) trains foreign languages through dramatic, interactive missions. The Interactive Storytelling Architecture for Training (ISAT) (Magerko et al., 2005) is based on IDA. Like the work presented here, ISAT uses a scenario director. However, the ISAT director agent selects from a pool of pre-authored scenes according to heuristics based on dramatic progression and trainee success or failure on learning objectives. For example, if the trainee fails a learning objective such as applying a tourniquet, the scenario director will select the next scene to be one in which another soldier becomes wounded in a way requiring a tourniquet.

8. Conclusions

Training cognitive leader skills in social and cultural domains requires seemingly conflicting requirements: the need for a virtual world populated by believable, interactive characters and the need to guide the trainee's experience through a sequence of relevant learning situations. The approach that we demonstrate in the IN-TALE prototype is to apply artificial intelligence technologies adopted from entertainment computing research to computer-based social simulation. Social simulation is the result of agents designed to broadly imitate the behaviors of believable denizens of a virtual world. Pedagogical needs are handled through intelligent scenario direction and scenario adaptation.

Our system serves the same conceptual role as an Observer/Controller in a more conventional exercise. However, an O/C is more appropriate for exercises that emphasizes procedural learning objectives. In a domain such as social and cultural leadership training that emphasizes problem-solving, deviations from the scenario script can occur. An O/C has few means for handling such deviations such as eliminating portions of the scenario that no longer make sense or "cheating." Our approach, using believable agents in conjunction with an Automated Scenario Director capable of intelligent scenario replanning can head off inconsistencies by adapting the scenario to achieve relevant learning situations in a plausible manner. In this way, our system achieves the seemingly conflicting requirements of allowing variability of the trainee while also maintaining pedagogical validity of a managed training experience.

9. Acknowledgements

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). 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. Special thanks to Jason Alderman, Don Dini, and Julia Kim.

10. References

Banta, H.G., Troillet, D.B., Heffernan, N.T., Plamondon, B., and Beal, S.A. (2005). *The Virtual Observer/Controller (VOC): Automated Intelligent Coaching in Dismounted Warrior Simulations* (Research Note 2005-01). U.S. Army Research Institute.

Bates, J. (1992). Virtual Reality, Art, and Entertainment. *Presence: The Journal of Tele-operators and Virtual Environments*, 1(1).

Bruckman, A. (1990). *The Combinatorics of Storytelling:* Mystery Train *Interactive*. Unpublished manuscript.

Burstein, M.H. and McDermott, D.V. (1996). Issues in the Development of Human-Computer Mixed-Initiative Planning Systems. In B. Gorayska and J.L. Mey (Eds.) *Cognitive Technology: In Search of a Humane Interface*. Elsevier.

Firby, R.J. (1989). *Adaptive Execution in Complex Dynamic Worlds*. Ph.D. Dissertation, Yale.

Johnson, W.L., Marsella, S., and Vilhjálmsson, H. (2004). The DARWARS Tactical Language Training System. In *Proc. of the 2004 Interservice/Industry Training, Simulation, and Education Conference*.

Loyall, A.B. (1997). *Believable Agents: Building Interactive Personalities*. Ph.D. Dissertation, Carnegie Mellon University.

Magerko, B. (2005). Story Representation and Interactive Drama. In *Proc. of the 1st Conf. on AI and Interactive Digital Entertainment*.

Magerko, B., Wray, R.E., Holt, L.S., and Stensrud, B. (2005). Customizing Interactive Training Through Individualized Content and Increased Engagement. In *Proc. of the 2005 Interservice/Industry Training, Simulation, and Education Conference*.

Mateas, M. & Stern, A. (2004). A Behavior Language: Joint Action and Behavior Idioms. In H. Prendinger & M. Ishizuka (Eds.) *Life-like Characters: Tools, Affective Functions and Applications*. Springer.

Mateas, M. and Stern, A. (2005). Structuring Content in the *Façade* Interactive Drama Architecture. In *Proc. of the 1st Conf. on AI and Interactive Digital Entertainment*.

McCausland, J. & Martin, G., (2001). Transforming Strategic Leader Education for the 21st-Century Army. *Parameters*, Autumn 2001.

McKee, R. (1997). *Story: Substance, Structure, Style, and the Principles of Screenwriting*. Harper-Collins.

Riedl, M.O. (2005). Towards Integrating AI Story Controllers and Game Engines: Reconciling World State Representations. In *Proc. of the 2005 IJCAI Workshop on Reasoning, Representation and Learning in Computer Games*.

Riedl, M.O, Saretto, C.J., and Young, R.M. (2003). Managing Interaction Between Users and Agents in a Multi-Agent Storytelling Environment. In *Proc. of the 2nd Int. Conf. on Autonomous Agents and Multi Agent Systems*.

Riedl, M.O. and Young, R.M. (2006). From Linear Story Generation to Branching Story Graphs. *IEEE Computer Graphics and Applications*, to appear.

Sengers, P. (2003). Schizophrenia and Narrative in Artificial Agents. In M. Mateas and P. Sengers (Eds.) *Narrative Intelligence*. John Benjamins.

Swartout, W., Hill, R., Gratch, J., Johnson, W.L., Kyriakakis, C., LaBore, C., Lindheim, R., Marsella, S., Miraglia, D., Moore, B., Morie, J., Rickel, J., Thiebaux, M., Tuch, L., Whitney, R., & Douglas, J. (2001). Towards the Holodeck: Integrating Graphics, Sound, Character and Story. In *Proc. of the 5th Int. Conf. on Autonomous Agents*.

Tambe, M. (1997). Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*, 7.

vanLehn, K. (1996). Conceptual and Meta Learning during Coached Problem Solving. In *Proc. of the 3rd Int. Conf. on Intelligent Tutoring Systems*.

Weld, D. (1994). An Introduction to Least Commitment Planning. *AI Magazine*, 15(4).

Young, R.M., Riedl, M.O., Branly, M., Jhala, A., Martin, R.J., and Saretto, C.J. (2004). An Architecture for Integrating Plan-Based Behavior Generation with Interactive Game Environments. *Journal of Game Development*, 1(1), 51-70.

Zachary, W., Cannon-Bowers, J., Bilazarian, P., Krecker, D., Lardieri, P., and Burns, J. (1999). The Advanced Embedded Training System (AETS): An Intelligent Embedded Tutoring System for Tactical Team Training. *International Journal of Artificial Intelligence in Education*, 10, 257-277.

Author Biographies

MARK O. RIEDL is a Research Scientist at the ICT. He completed his Ph.D. in Computer Science at North Carolina State University in 2004. His research focuses on automated narrative generation, interactive narrative, and intelligent virtual agents for entertainment, education, and training.

ANDREW STERN is a designer, researcher, writer and engineer of personality-rich, AI-based interactive characters and stories. Previous to co-developing the award-winning interactive drama *Façade*, Andrew was a lead designer and software engineer of the Virtual Babyz, Dogz, and Catz from PF.Magic, which sold over 2 million units worldwide.