A Scalable and Extensible Interactive Scenario Architecture for Distributed Command and Control Simulations

Magy Seif El-Nasr	**Rashaad E.T. Jones**	**Michael McNeese**
Assistant Professor	PhD Candidate	Associate Professor
School of Information Science and Technology	School of Information Science and Technology	School of Information Science and Technology
Pennsylvania State University	Pennsylvania State University	Pennsylvania State University
314 IST Building	314 IST Building	314 IST Building
University Park, PA 16802	University Park, PA 16802	University Park, PA 16802
magy@ist.psu.edu	rjones@ist.psu.edu	mmcneese@ist.psu.edu

A Scalable and Extensible Interactive Scenario Architecture for Distributed Command and Control Simulations

Magy Seif El-Nasr	Rashaad E.T. Jones	Michael McNeese
Assistant Professor	PhD Candidate	Associate Professor
School of Information Science and Technology	School of Information Science and Technology	School of Information Science and Technology
Pennsylvania State University	Pennsylvania State University	Pennsylvania State University
314 IST Building	314 IST Building	314 IST Building
University Park, PA 16802	University Park, PA 16802	University Park, PA 16802
magy@ist.psu.edu	rjones@ist.psu.edu	mmcneese@ist.psu.edu

Abstract

Interactive virtual environments are becoming increasingly popular for their utility in virtual training, distributed decision-making and collaborative environments. Some of these applications rely on a scenario that is revealed to the user as he/she interacts with synthetic objects and characters that inhabit virtual worlds. The development and authoring of interactive dynamic scenarios is often hard and difficult to accomplish using current techniques. Many interactive scenario developers use decision trees, which yield very limiting and unfulfilling training experiences, because they do not stimulate learning or thinking beyond the scripted paths. Some researchers proposed plan-based interactive narrative architectures, which, although superior to decision trees, do not scale and do not address user's goals and intentions, yielding inflexible scenarios that do not adapt suitably to players' goals or behaviors. In this paper, we propose a dynamic scenario architecture that aims at enhancing scalability and reuse by using a multi-agent layered problem solving technique. Additionally, the interactive scenario architecture will automatically adapt to users' goals by integrating a user model and a user monitoring technique.

1. Introduction

Elaborate interactive 3D virtual environments populated with embodied synthetic characters are becoming a significant component for many military and command and control applications, including training, assessment, and experimental distributed decision-making and learning environments. In this paper, we focus on the design and development of effective user-centric dynamic scenario architecture that automatically adapts to users' behaviors.

We argue that for a dynamic scenario system to be effective in training, assessment, or team decision-making assessment, the architecture should address several issues: (1) dynamic modulation of the scenario to adapt to users' behaviors, (2) modulation of the scenario to adapt to the scenario objectives and learning strategies, and (3) integration of

user feedback and adaptation of scenarios through agents' behaviors based on user's feedback and actions.

Addressing these problems is hard due to many reasons. First, current techniques for interactive scenario rely on the use of decision trees, which, due to their exponential growth, are hard to author and modify. In addition, predicting users' behaviors is difficult. Thus, a decision tree-based architecture leads to very limiting and unfulfilling experiences, because they do not stimulate learning or thinking beyond the scripted paths. This drawback has hindered many interactive scenario-based applications.

Some researchers recognized this fact and searched eagerly to find a solution. Mateas and Stern proposed a HAP-based architecture for interactive narrative (Mateas and Stern 2000, 2001). Young proposed a plan-based interactive narrative architecture, where plans are revised depending on users' actions (Young 2000). Peter Weyhrauch proposed an interactive drama manager that uses game theory to select a story event given the current story state (Weyhrauch 1997).

These explorations present major achievements to break the norm of decision tree-based interactive scenarios. Although these projects have succeeded in developing interactive dynamic scenarios that are more adaptable and maintainable than decision tree-based scenarios, they still suffer from several drawbacks. They are difficult to debug and maintain. The use of plan-based techniques limits the architecture's scalability. In addition, they do not integrate user's goals or intentions in their scenario generation system, which limits their scenario variations and adaptation.

Alternatively, we are proposing a dynamic interactive scenario architecture that integrates a user modeling approach and divides problem solving among several layers. The contribution of the methods described in this paper will impact the research on interactive scenario architectures in two directions. First, we present an interactive scenario architecture that enhances scenario adaptation by integrating two constructs that have been used by screenwriters and actors: (1) a user model to adapt the scenario dynamically to users' goals (Mckee 1997), and (2) a user monitoring technique to evaluate the failure or success of character behavior (Bennedetti 1994); some example user monitoring techniques include monitoring users' actions to infer agreement or to evaluate that his/her attention is directed towards the desired object/character. In a second direction, the paper proposes an architecture that enhances scalability and reuse by using a multi-agent layered problem solving technique. This technique enhances scalability by dividing the problem into smaller problems, and thus reducing the number of rules and predicates processed at each layer. The architecture also enhances reuse since it decouples scenes, scenario events, and character behaviors into components. It is worth noting that scalability and reuse are important qualities for interactive scenario architectures, especially considering the complexity of the authoring process.

In this paper, we will first discuss previous research. We will then describe the proposed dynamic scenario architecture defining the scenario structure and its representation. In the following section, we will show the integration and implementation of this architecture in a command and control-based simulation called, *Neocities*.

2. Previous Research

The need for scalable and effective interactive narrative or dynamic scenario architectures that adapt to users' interactions provides an incentive for many researchers to explore the utility of applying AI-based problem solving techniques to interactive scenarios. In this section, I will describe a few attempts.

Some researchers developed character-centric interactive narrative architectures where the narrative emerges as a product of user's interaction with an environment populated with synthetic agents. Examples of these architectures include *The Sims*, *Creatures*, *Catz*, *Dogz*, and *Woggles* (loyall 1997). Researchers focusing on the character-centric approach usually focus on character development (loyall 1997, Bates 1992, Bates et al. 1992, O'Rielly 1996, Aristotle 1967). Since the narrative or scenarios are not represented within the architecture, the utility of such architecture for authored interactive scenarios[1] is unclear.

Mateas and Stern proposed a HAP-based architecture for interactive narratives (2001). Young proposed a plan-based interactive narrative architecture, where plans are revised depending on users' actions (2000). Peter Weyhrauch proposed the use of game theory to adapt the scenario to user's behaviors (1997).

Young proposed an interactive narrative architecture that uses planning (Young 2000). Young's approach relies on re-planning to accommodate user's behaviors. For example, if the user attempts to shoot an important character, then the system, recognizing that the character is important, will adopt a 'gun out of bullets' routine to prevent the user from killing the important character. This technique is problematic and may lead to user frustration, because it deliberately obstructs the user from his/her goal in order to keep him/her on the story track. In addition, many performance issues may arise due to the use of planning.

These explorations, and many others, present major achievements to break the norm of decision tree-based interactive scenarios/narratives. Although these projects have succeeded in developing interactive scenarios that are more maintainable than decision trees, they do not scale and are time consuming to author. In addition, these architectures do not integrate users' goals or intentions or use such information to select scenario events; thus, these architectures yield inflexible scenarios that do not adapt suitably to players' goals or behaviors.

3. Interactive Dynamic Scenario Architecture

Figure 1 depicts the components of the proposed interactive dynamic scenario. As shown in the figure, the dynamic scenario architecture is composed of several components. The scenario generation engine takes as inputs: scenario goals, the events and actions sensed by the scenario manager from the environment, and the inferred user model. It then generates a scenario event. This event is then given to the scenario manager who dictates several behavioral goals to the respective agents within the simulation. Given the behavioral goals and the current state, each agent forms a plan that achieves these behavioral goals. This plan along with timing constraints for the actions within the plan is

---

[1] Authored interactive scenarios describe scenarios where an author, a director, or a designer has a set of goals that he/she wants the trainee, player, or participant to achieve or reach.

given to the scenario manager, who then synchronizes the plans to a uniform plan of action and distributes the behavioral plan that includes timing constraints to agents. The simulation engine takes the actions from the agents and adequately renders them with the supervision of the scenario manager.

3.1 Scenario Generation Engine

The scenario generation engine keeps track of its current state including history of selected scenario events, character actions, and character relationships. Given a number of authored scenario events, the user model, and the scenario state, the scenario generation engine first selects a scene that will satisfy the scenario objective, and then it will select a scenario event that will satisfy the scene objective. We use a reactive planning algorithm to select the scenes and scenario events (Firby 1989). In order to clearly describe the scenario generation engine, it is important to explain the scenario structure and representation. Therefore, we will first describe the scenario structure and then detail the rest of the interactive dynamic scenario architecture.

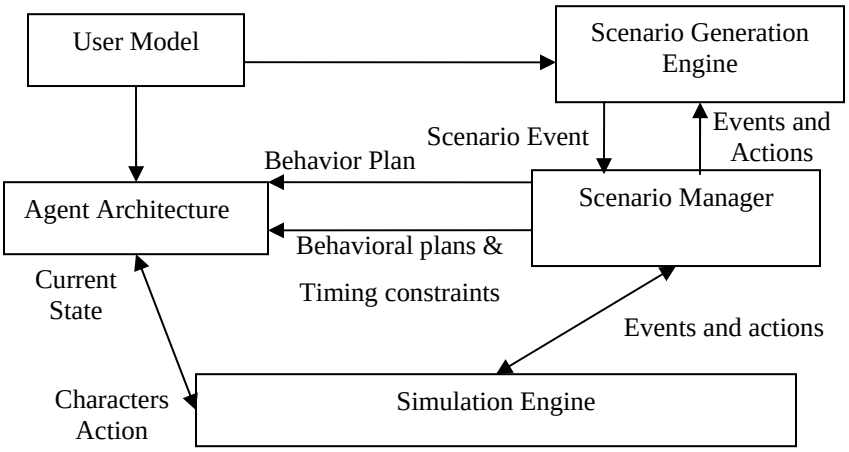

Figure 1: Dynamic Scenario Architecture

3.1.1 Scenario Structure

The scenario is represented in two separate levels: scenes and scenario events. Scenes are represented using the following structure:

Scene Objective: (scene-goal ?sg), where ?sg is a list of states connected by ands or ors. When the statement ?sg is true the objective becomes true. For example:

(scene-goal (Told Electra Archemedis (story-of Electra))

where the scene's goal is to make Electra tell her story to Archemedis.

Scene Preconditions: list of conditions concerning the scenario state, user model, and user actions. If true they qualify the scene for being fired given that the story engine is looking for a scene that achieves the scenic goal listed by the scene.

Scene Posteffects: list of states that become true upon success.

Scene Effects: list of states that become true upon firing.

Scene subgoals: (sgoal ?z)

Where sgoal is

(sequence sgoal) | (parallel sgoal) | (beat-goal ?y)

3.1.2 Scenario Events

Scenario events are represented using the following structure:

Event Goal: (event-goal ?bg), similar to scene goals

Event Preconditions: list of conditions concerning the scenario state, user model, and user actions. If true they qualify the scenario event for being fired given that the scenario generation engine is looking for an event that achieves the event goal listed above.

Event Posteffects: list of states that become true upon success

Event Effects: list of states that become true upon firing

Event subgoals: (agoal ?y)

where agoal is

(sequence agoal) | (parallel agoal) | (character-goal ?y) | (camera-goal ?c) | (lighting-goal ?x)

3.1.3 Selecting scenario events

A scenario event is then a collection of sequence and/or parallel event subgoals or character, camera, or lighting goals. To get a sequence of simple goals (such as character-goals, lighting-goals, and camera-goals), the scenario generation engine includes an algorithm that iteratively loops selecting scenes and breaks them into simpler scenes until a plan of simple scenes is selected. It then selects a scene from the plan to execute. The second step is then concerned with searching for a plan of scenario events that achieves the scene objective of the selected scene. It then selects scenario events and breaks them into simpler scenario events, constructing a tree of event-goals and simple-goals. Once a path is found with only simple goals, the scenario generation engine passes these simple goals to the scenario manager, who then relays them to appropriate agents. This deconstruction of goals to simpler goals is similar to the method described in (Loyall 1997, Firby 1989, Forbus and Kleer 1992).

Abstracting some elements that define the important focus and problem difficulty level in the scenario is an important design element. Thus, the scenario engine allows authors to write rules to identify shifts in difficulty level through scenario event changes. These rules allow the scenario engine to identify the difficulty level of a specific moment, which is represented, as discussed above, as a number (0-100); an example is as follows:

if beat#2 is followed by beat #5

and Electra is using the threatening tactic on the user

then increase dramatic intensity by 10 increments,

These rules are extremely important, because they serve as catalysts for adapting the scenario and visual presentation to enhance and support the scenario.

3.2 Scenario Manager

The scenario manager is designed with the same functionality as the drama manager in the Oz Project (Mateas, 1999). The scenario manager is the essential component of dynamic scenario architecture; it serves as a hidden "intelligent agent" that controls and monitors the scenario execution. It coordinates agents plans. These agents are not restricted to characters in the virtual world, but they are extended to include visual agents, such as lighting, camera, and staging.

The agents propose behavioral plans and timing constraints to the scenario manager who is then responsible for coordinating these plans. The scenario manager uses several rules to synchronize the scheduling of actions within the behavioral plans. It then relays the plan with some synchronization constructs to the respective agents. The agents then proceed to execute the actions within their plans abiding by the synchronization constructs.

Synchronization constructs are defined as preconditions to the actions. For example, the scenario manager may decide to have character *x* move from position *x1* to position *x2* only when character *y* finishes speaking. Thus, the synchronized plan given to character *x* will be of the following form:

```
(action (walk-to (position x2))
        :precondition (not (speaking (character y)).
```

Furthermore, the scenario manager also coordinates the execution of the actions by monitoring the execution and passing messages back and forth between agents.

3.3 Agent Architecture

Once a character-goal is given to an agent, the agent uses a similar technique to scenario deconstruction to deconstruct the behavioral goals into simple actions (Forbus and Kleer 1992, Loyall 1997).

3.3.1 Behavior Representation

We are representing behaviors as follows:

Behavior goal: (goal ?cg), similar to scene goals

Behavior Preconditions: list of conditions concerning the scenario state, user model, and user actions. If true they qualify the behavior for being fired given that the scenario generation engine is looking for a behavior that achieves the objective listed above.

Behavior Posteffects: list of states that become true upon success

Behavior Effects: list of states that become true upon firing

Behavior subgoals: (bgoal ?y)

where bgoal is

(sequence bgoal) | (parallel bgoal) | (action ?y) | (say ?c)

where *action* and *say* are terminal actions.

Actions are represented by an action, an adverb, and an actor; for example (Walk Electra slowly) is a behavior where the action is walk, the actor is Electra, and the manner in which an action is performed is slowly. Therefore, an action can be animated in different manners defined by the adverb. For example, 'take the sword' is an action that is defined as three animations 'take sword eagerly', 'take sword hesitantly', and 'take sword regretfully'.

3.3.2 User behavior analysis and dynamic character improvisation

Justine Cassell has argued for the use of feedback and user modeling for conversational agents (Cassell 1998). Likewise, we argue for the use of such techniques for believable characters. By studying acting and observing actors improvise their roles, I concluded that current techniques do not advocate dynamic user monitoring or adapt behaviors according to dynamic user feedback. In contrast, actors continuously monitor each other within a scene looking for clues to evaluate their behaviors. They improvise by adjusting their behaviors to other actors' goals and behaviors.

Therefore, we propose the use of: (1) a mechanism for using a user model to choose behaviors and (2) a mechanism for dynamic evaluation of success and/or failure of the behavior using an analysis of user's behavior. To evaluate failure or success of their behaviors, agents will continuously monitor user's state and actions for signals. Thus, the interactive narrative representation is extended to include:

Failure condition:

```
e.g. (and (talking (character ?z))
     (not (attending user (character ?z))) )
```

Failure Tolerance: e.g. 90%

The failure condition example defined above identify when a character is talking, she/he will monitor the user to ensure his attention is directed towards him/her. The tolerance for violating this condition is high.

3.3.3 Behavioral plan generation algorithm

One major difference between this architecture and the previous work, in addition to the use of user modeling, is in the agent's ability to dynamically adapt to the user's behaviors. Agents continuously monitor mouse movements, mouse clicks for clues to infer the direction of user's attention and user's goals and intentions. Given these inferred values, the agent will continuously adjust its success or failure rates until failure reaches the tolerance limit. If that occurs, the character will (1) declare the behavior a failure, (2) update the user model, and (3) choose another behavior to solve the character-goal. The algorithm can be summarized as follows:

```
1. Choose behavior plan given
     user stereotype,
     character goal,
     failed behaviors
2. each time tick
```

2.1. monitor user action assessing current behavior
2.1.1 if failure limit reached,
2.1.1.1 fail behavior
2.1.1.2 go to step 2.
2.1.2. Update user model

3.4 User-Modeling

A user modeling technique is used to further enhance the adaptability aspect of the interactive dynamic scenario architecture. This component will appropriately and automatically adapt to users' goals by integrating a user model to adapt the scenario dynamically to player's goals (McKee, 1997).

The user-modeling approach we adopted was based on casual relationship between user ability and character personality (which is inferred by the user's actions) and scenario events. Given a user action, the history of user actions, and the scenario state, a rule-based system is used to adapt the user model to reflect the user's choice. A simple example can be "if user is unsuccessful in responding to a terrorist-related event and is easily frustrated (judging from previous interactions), then lower the difficulty and severity of all future events".

The user-modeling component was first implemented and evaluated within an interactive story called *Mirage*. *Mirage* is a full scale 3-D interactive virtual story, where a user interacts with embodied animated agents and make choices that impact his/her life and the simulation (Seif El-Nasr to appear). This model is currently being extended for use in *NeoCITIES*.

4. Results

An initial prototype of the interactive scenario architecture was implemented in an interactive story called *Mirage* (El-Nasr to appear). The result from incorporating this architecture in *Mirage* yielded a more scalable and adaptable narrative when compared to current techniques, including goal-based narrative without user modeling or monitoring, and decision-trees. To evaluate the utility of the proposed approach and its validity for Command and Control, we are currently integrating the architecture within an interactive simulation called *NeoCITIES* (McNeese et al. 2003; McNeese, in press)

Neocities is a scaled-world simulation specifically designed for a command and control (C^2) environment. The NeoCITIES's task is both a renovation, as well as an expansion, of the original CITIES task (Wellens & Ergener, 1988). As with the original CITIES task, the Neo Command and Control (C^2) Interactive Task for Identifying Emerging Situations (NeoCITIES) has been designed and developed for the purpose of studying group collaborative decision-making processes, knowledge acquisition and knowledge management using information acquired from multiple sources within a command and control (C^2) setting. As such, NeoCITIES is essentially an adaptable problem interface, which allows the close examination of semiautonomous, spatially distributed decision-making teams. These decision-making teams can be presented with several different overarching, dynamic and detailed resource allocation problem scenarios, for which they are required to find suitable solutions meeting the needs of given constituents and

working around various problem space constraints existing within the task. Figure 2 is a snapshot of the NeoCITIES interface.

The scenarios designed and types of teams selected to be used in this simulation were purposefully designed to be contextually relevant to the concerns of the Department of Homeland Security in the areas of crisis management. These scenarios have each been taken and adapted from news stories of recent world events or from stories in popular media. Additionally, experts from the field (crisis management personnel that includes DHS intelligent analysts, police and fire authorities, etc) were interviewed to assist in the development of scenarios to provide a highly-realistic and context-relevant setting. Following the scenario representation discussed above, we are establishing a scenario goal for each scenario, then authoring several scenes and scenario events with preconditions and postcondition predicates.

NeoCITIES effectively mimics distributed situated cognition in 911 and 9-11 events and storyline. Thus, the simulation emulates the management of a city wherein crisis management is appropriated through the joint interaction of three distinct teams (fire, police, and hazard materials). This involves potential terrorist activities as they emerged in New York City in 9-11 that are unexpected, complex, stressful, and very time-pressured. Conversely, it also involves more routine activities that go on in a 911 dispatch center.

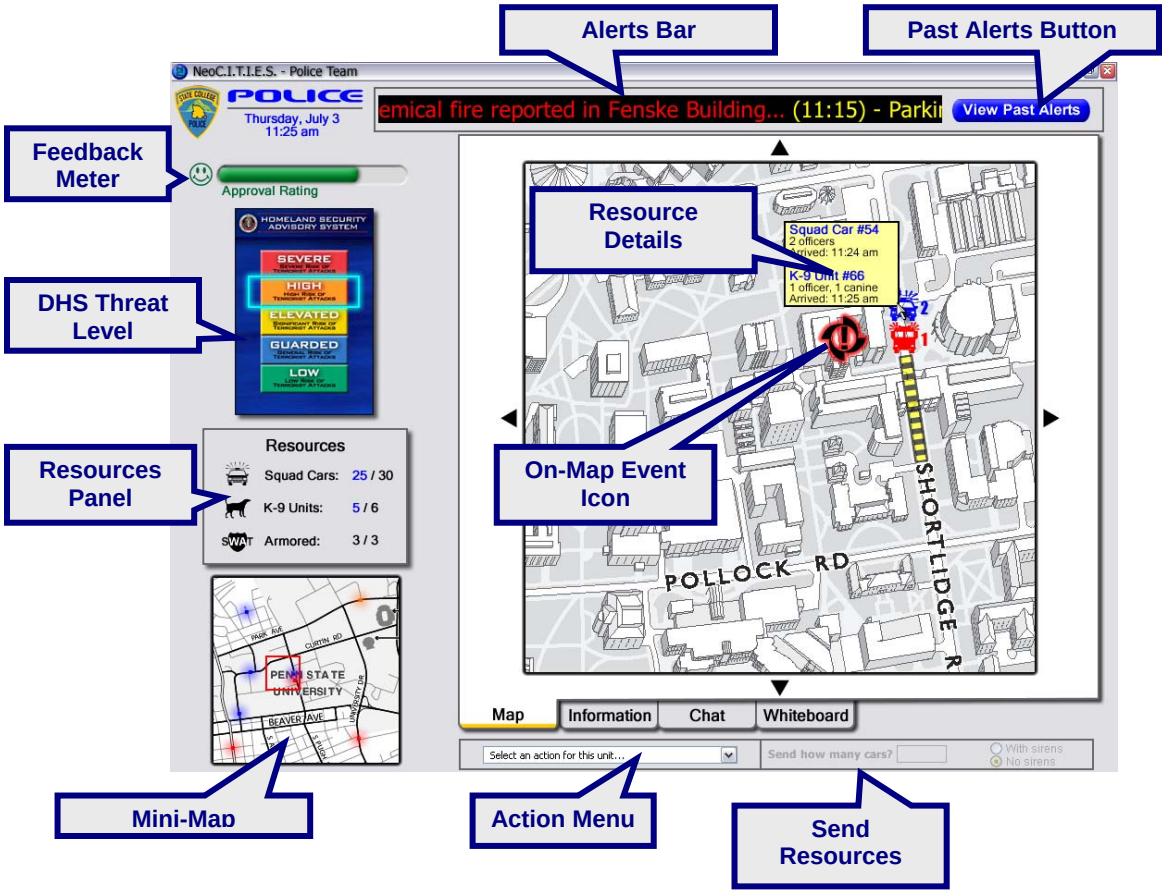

Figure 2: NeoCITIES interface

Each team has limited resources that can be placed on events to do certain commands (user actions). Events may be single team (i.e., they only require a given team to respond), or they can be more complex and require two or even three teams to pool resources to assuage the activity under investigation. As the simulation begins, very few events occur, but as it moves forward in time many events begin to popup that have to be addressed or they start to run out of control (e.g. a fire spread from a single trash-can to an entire building if not processed). Thus, the events in the storyline *emerge*, which is defined by the user modeling component, as time moves forward.

NeoCITIES is termed a team resource allocation problem and this captures part of the simulation functionality. *Within team* and *across team* cognition can get complicated as decisions start to pile up potentially creating information overload problems. Behind this basic-level logic of the scaled world is another layer however. The *hidden layer* consists of teams coming to recognize a pattern--given a series of events--which something ominous may be going on.

5. Conclusions

A dynamic scenario approach that integrates problem solving techniques within a multi-agent architecture is presented. The proposed architecture incorporates and integrates a user model formed by dynamically monitoring and modeling users' behaviors. The paper proposes a dynamic scenario architecture that decomposes problem solving or scenario generation among several levels and uses reactive planning to generate a sequence of goals or behaviors that will achieve the scenario goals. We argue that this decomposition enhances scalability and reuse. The system uses user's actions and inferred goals and intentions to guide its decisions, thus forming a user centric approach to interactive scenario generation and adaptation. In addition, the proposed architecture advocates the use of agents that dynamically monitor user's behaviors for feedback to evaluate their behaviors. This architecture is currently being implemented in NeoCITIES, which is a simulation specifically designed for studying group collaborative decision-making processes, knowledge acquisition and knowledge management using information acquired from multiple sources within a command and control (C^2) setting.

References

Aristotle. (1967). *Peotics*.

Bates, J. (1992).The Role of Emotion in Believable Agents. *Communications of the ACM*, *37* (7). 122-125.

Bates, J., Loyall, B. and Reilly, S. (1992). An Architecture for Action, Emotion, and Social Behavior, Carnegie Mellon University, Pittsburgh.

Bates, J., Loyall, B., and Reilly, S. (1992). An Architecture for Action, Emotion, and Social Integrating Reactivity, Goals and Emotion in a Broad Agent, Carnegie Mellon University, Pittsburgh.

Benedetti, R. (1994). *Actor at Work*, 6th ed. Englewood Cliffs: Prentice-Hall.

Cassell, J. (1998). A Framework for Gesture Generation and Interpretation. in Pentland, R.C.a.A. ed. *Computer Visiion in Human-Machine Interaction*, Cambridge University Press, New York, 191-215.

Jones, R. E. T., McNeese, M. D., Connors, E. S., Jefferson Jr., T., & Hall, D. (2004). A distributed cognition simulation involving homeland security and defense: The development of NeoCITIES. *Proposal submitted to HFES Conference*, LA.

K. Forbus, K., & de Kleer, J. (1993). *Building Problem Solvers.* MIT Press.

Loyall, A. B., & Bates, J. (1997). *Personality-Rich Believable Agents That Use Language.* Paper presented at the Autonomous Agents, Marina Del Rey.

Mateas, M. (1999). An Oz-Centric review of interactive drama and believable agents. *AI Today: Recent Trends and Developments.*

Mateas, M. (2001). Interactive drama. *Research Proposal*, Carnegie Mellon University, Pittsburgh.

Mateas, M., & Stern, A. (2000). Towards integrating plot and character for interactive drama, *Socially Intelligent Agents: The Human in the Loop AAAI Fall Symposium*.

McKee, R. (1997). *Story:Substance, Structure, Style, and the Principles of Screenwriting*. New York: HarperCollins.

McNeese, M. D. (in press). How video informs cognitive systems engineering: Making experience count. To be published in the *International Journal of Cognition, Technology, and Work.*

McNeese MD and the MINDS Group (2003) Acquiring, accessing, assessing and sharing knowledge in distributed cognition: A cognitive science perspective. Presentation for the 2003 ONR Review. Penn State University, University Park, PA.

Seif El-Nasr. (to appear). An Interactive Narrative Architecture based on Filmmaking. International Journal for Intelligent Games and Simulation.

Wellens, R & Ergener, D. (1988). The C.I.T.I.E.S. game: A computer-based situation assessment task for studying distributed decision making. Simulation & Games, 19(3), 304-327.

Weyhrauch, P. (1997). Guiding interactive drama. *PhD Thesis*. Pittsburgh: Carnegie Mellon University.

Young, M. (2000). Notes on the use of plan structures in the creation of interactive plot, *AAAI Fall Symposium on Narrative Intelligence*, MA.