Increasing Replayability with Deliberative and Reactive Planning

Michael van Lent, Mark O. Riedl, Paul Carpenter, Ryan McAlinden, Paul Brobst

Institute for Creative Technologies
University of Southern California
13274 Fiji Way, Los Angeles, CA 90292
{vanlent, riedl, carpenter, mcalinden, brobst}@ict.usc.edu

Abstract

Opponent behavior in today's computer games is often the result of a static set of Artificial Intelligence (AI) behaviors or a fixed AI script. While this ensures that the behavior is reasonably intelligent, it also results in very predictable behavior. This can have an impact on the replayability of entertainment-based games and the educational value of training-based games. This paper proposes a move away from static, scripted AI by using a combination of deliberative and reactive planning. The deliberative planning (or Strategic AI) system creates a novel strategy for the AI opponent before each gaming session. The reactive planning (or Tactical AI) system executes this strategy in real-time and adapts to the player and the environment. These two systems, in conjunction with a future automated director module, form the Adaptive Opponent Architecture. This paper describes the architecture and the details of the deliberative and reactive planning components.

Introduction

In most of today's computer and video games the behavior of AI opponents is controlled by a static script or some other form of fixed behavior encoding. This well-controlled but limited approach to opponent behavior has both advantages and disadvantages. Since the behavior is tightly controlled it is easier for developers to guarantee that the AI opponents will behave predictably and therefore ensure the intended gaming experience. Also, scripted or fixed AI techniques can be less computationally and memory intensive since a range of strategies don't need to be stored or considered during execution. However, the limited nature of fixed AI opponents can have a negative impact on the long-term replayability of games. The player's early experiences, while the AI's behaviors are still novel, are enjoyable. But as the player learns to predict and counter the AI opponent's single approach, the experience starts to feel stale. After a fairly small number of game sessions the game experience switches from exploring how to counter the AI to simply applying the same tried and true counter-strategy yet another time.

As games and game technology are used more and more frequently for non-entertainment applications, the current static techniques for generating opponent behavior show some additional limitations. For example, Full Spectrum Command (FSC) (van Lent, Fisher and Mancuso 2004) is a game-based training aid developed to help U.S. Army and Singapore Armed Forces company commanders learn the cognitive skills involved in commanding troops. In each game level, or mission, the AI opponent is controlled by a script created by the human mission designer. The first time a student plays a mission this script is unknown and the students must practice the cognitive skills FSC seeks to train. However, the third or fourth time through the mission the player knows the AI script and can use this knowledge to win rather than exercising the target cognitive skills. This becomes particularly apparent when students start selecting actions that are, in general, very poor choices but work well against the specific AI script for that mission.

The work described here seeks to use a combination of deliberative and reactive planning to move away from AI opponents that rely on a single, fixed strategy. A deliberative planner (DPOCL) is used as the central component of the Strategic AI system while a reactive planner (Soar) is the core of the Tactical AI system. An Automated Director that tracks player history and tailors the experience is a planned future component. Collectively this system is the *Adaptive Opponent Architecture*. The Strategic AI subsystem operates before each game session to create a novel strategy for the AI opponent based on the scenario environment, parameters and the AI opponent's goals. The Tactical AI subsystem controls the execution of this strategy through real-time interaction with the game environment during the game session. The Tactical AI subsystem is also responsible for purely reactive actions and local modifications to the strategy. This combination of deliberative and reactive planning results in dynamic, unpredictable behavior by AI opponents that challenges the player to constantly adapt and adjust their own strategies. The Adaptive Opponent Architecture increases the replayability of entertainment-based games and the learning value of education-based games.

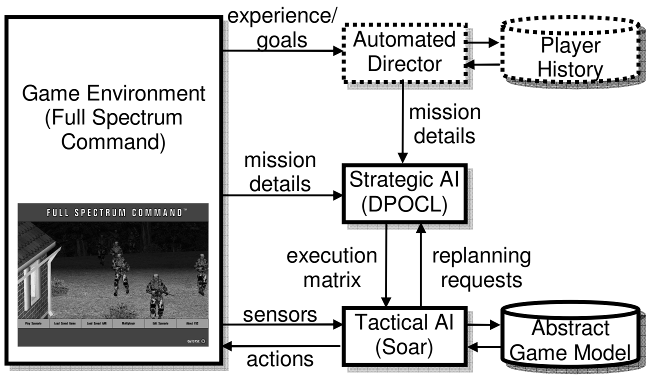

Figure 1: The Adaptive Opponent Architecture.

Full Spectrum Command has been selected as the first target application of the Adaptive Opponent Architecture. Since FSC straddles the fence between computer game and training tool, it allows us to demonstrate the value of deliberative and reactive planning in both entertainment and educational applications. FSC is a one or two player PC-game in which the player commands a company of 120 light infantry soldiers divided into three platoons. A game session in FSC consists of three phases: planning, execution and after-action review. In the planning phase the player reads the mission description and creates a high-level plan coordinating the actions of the three platoons to achieve the mission goal. This emphasis on pre-mission planning sets FSC apart from most games that jump to the action as quickly as possible. In the execution phase the player monitors the platoons as they execute the plan and modifies/updates the plan as necessary through fragmentary orders (Fragos). In the after-action review phase the player reviews mission statistics and their plan, gets to see the opponent's plan for the first time, and can replay the mission with key points highlighted. The Adaptive Opponent Architecture fits well into this three phase design with the Strategic AI operating during the planning phase and the Tactical AI operating during the execution phase.

The next section of this paper describes the Adaptive Opponent Architecture in more detail. The following two sections describe the deliberative planning-based Strategic AI and the reactive planning-based Tactical AI. The next section describes the current status of the work and a number of next steps currently being explored and then wraps up with some conclusions.

Adaptive Opponent Architecture

The Adaptive Opponent Architecture, shown in Figure 1, consists of a three-tiered collection of AI modules that communicate with the game environment and each other to control the behavior of the AI opponents in support of entertainment or other game goals. The boxes with dashed outlines show components designated for future work. The top-tier module is an automated director that tracks the player's history over a series of game sessions and tailors the scenario details of each new game session to make each new session entertaining and consistent with previous sessions. For example, the automated director might adjust the level of difficulty to match the player's abilities and vary the details to ensure the player experiences some novel aspects of the environment each game session. The automated director module is planned future work. We hope to capitalize on the ongoing research in this area (e.g. Magerko and Laird 2003; Young et al. 2004).

The next module is the Strategic AI module which is based on the DPOCL domain-independent deliberative planner (see next section for more details). The Strategic AI module takes as input a set of mission details from the game environment (and eventually the automated director). These game details include the initial state of the game session, the AI opponent's goal for the game session, and the actions available to the AI opponent. The Strategic AI module then uses an abstracted model of the game to create a mission plan for the AI opponent. This mission plan is encoded as an execution matrix in which each row represents a sequence of actions by a specific platoon and each column is a slice of time in the mission. Transitions between time slices in the mission can be fixed to a specific time or variable depending on environmental conditions. Determining when the conditions indicating a step transition are satisfied is one of the roles of the Tactical AI. The deliberative planning process performed by the Strategic AI is the same process a player performs during the planning stage (for the player's side) or a mission designer performs during the authoring of a mission (for the AI opponent's side).

The final module of the Adaptive Opponent Architecture is the Soar-based Tactical AI. The Tactical AI takes the execution matrix as input and interacts with the game environment in real-time through a pre-defined set of sensors and actions. Sensor input, received multiple times a second, gives the Tactical AI information about the immediate state of the game environment. Based on this sensor input both goal-based actions (directed by the execution matrix) and reactive actions are selected and passed back to the game environment to control the AI opponent entities. Goal-based actions primarily focus on executing the mission plan generated by the Strategic AI. Example goal-based actions might include moving to a checkpoint, clearing a building, or ambushing an enemy unit. Reactive actions are selected in response to events in the environment and are influenced by, but separate from, the mission plan. Example reactive actions might include returning fire when fired upon, seeking cover, or fleeing when heavily outnumbered.

Deliberative Planning: Strategic AI

The Strategic AI module requires the ability to generate any number of high-level plans for the AI opponent that are novel, meaning significantly different in some recognizable way from previous plans used by the system. To promote variability, each game session may provide

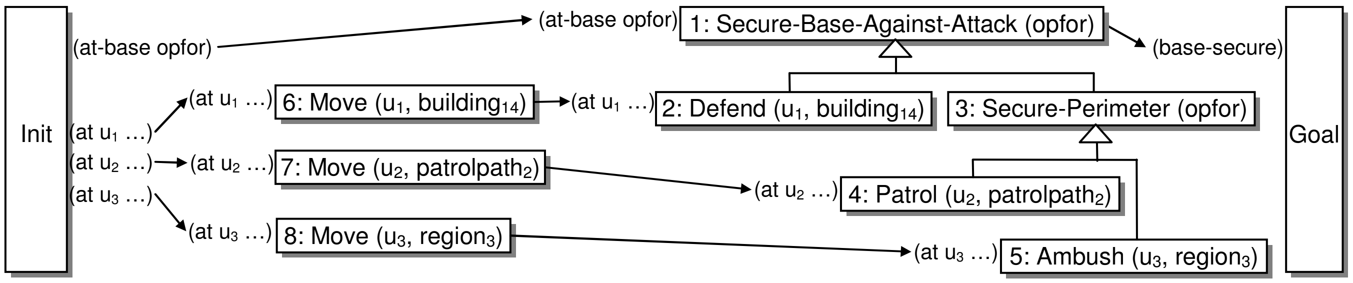

Figure 2: A hierarchical strategic plan.

different goals for the AI opponent to achieve and different force configurations. Furthermore, the player may be opposed by different levels of opponent sophistication such as an opponent with a well-defined military doctrine or an asymmetric opponent using unpredictable and improvised tactics.

Planning is one technique for generating novel strategic-level tactics for RTS games in which the opponent is expected to carry out goal-based behaviors. The advantages of planning include:

- Variability can be achieved by initializing the planner with different initial world states, different goals, and different sets of operations (possibly encoding doctrine) that can be carried out by opponent forces.
- Complete[1] planners can find and choose among any number of solutions for achieving a goal.
- Planning and execution can be interleaved so that the opponent, like the player, is not required to have a completely specified plan.
- Explicit models of plan failure exist so that opponent tactical execution can be monitored. There are well-known algorithms for replanning in the event that plan execution does fail.

Partial-order planning algorithms such as UCPOP (Penberthy and Weld 1992) allow for unrestricted parallelism (Knoblock 1994). Parallel execution is essential for strategic force planning because it enables unit forces to perform tasks in parallel while still retaining loose coordination. Hierarchical task network (HTN) planners (Sacerdoti 1977) plan on multiple levels of abstraction achieving a degree of cognitive plausibility due to the application of schema-like task networks. Task networks can be used to encode doctrine as well as more free-form tactics. HTN planners, however, rely on pre-defined task networks that cannot be readily adapted to novel circumstances.

The Decompositional Partial Order Causal Link (DPOCL) planner (Young, Pollack, and Moore 1994) is a partial-order planner based on UCPOP but extended to perform hierarchical decomposition of abstract operators. DPOCL meets all the requirements of a strategic deliberative planner for an opponent AI in an RTS game. DPOCL uses an operator library that contains both abstract and primitive operators. Abstract operators are decomposed into successively more primitive operators by applying decomposition rules.

Partial-order planning is a process of searching for a sequence of operations – abstract and primitive – that achieve a particular goal. This is achieved by non-deterministically backward-chaining from the goal conditions to be achieved. Operators have preconditions – facts about the state of the world that must be true for the operator to be applicable – and effects – facts about the state of the world that are changed by successful execution of the operator. New operators are instantiated by the planner to satisfy the goal conditions as well as the preconditions of operators already in the plan. Decomposition rules are applied to abstract operators. Unlike task networks, decomposition rules can be partial; the DPOCL planning algorithm is applied recursively to fill in missing details. Partial decomposition rules can be applied to a wide variety of circumstances and allow for innovation whereas a task network has a very specific set of circumstances that it can be applied to.

Strategic AI Example

Suppose the opponent force (OPFOR) is an asymmetric force occupying a base. The game scenario defines the OPFOR's goal – to keep the base secure – and initial configuration. For simplicity, suppose the OPFOR consists of three units: two of which carry rifles and one of which carries a grenade launcher.

Without a script, the AI opponent must develop a novel plan for keeping the base secure. At the time of writing, the strategic AI system can generate nearly 30 unique plans in under five minutes for the example domain model described here. Many of these plans however involve similar action sequences with different permutations of force allocations. The following discussion traces the generation of just one plan.

The goal can be achieved by a single operator `Secure-Base-Against-Attack`. The operator is abstract because it specifies what needs to be done, but not at a level that can be tactically executed by unit forces. The `Secure-Base-Against-Attack` has a single precondition – the OPFOR team must be at the base – that is established by the initial conditions of the scenario.

[1] Completeness is a property of planners such that the search space of a complete planner contains all possible solution plans.

`Secure-Base-Against-Attack` must be decomposed into primitive-level operations before the plan can be considered complete. The planner non-deterministically tries all decomposition rules that apply and uses a heuristic function to rate the desirability of each possibility. The heuristic evaluates the plan-so-far on optimality, the effective use of unit capabilities, whether or nor the plan is similar to ones used in previous game session, and on pedagogical and entertainment objectives (from the automated director). Suppose a decomposition rule is chosen that has the OPFOR secure the base by defending a central building, `building14`, and securing the base's perimeter. Securing the perimeter is further decomposed into one unit patrolling the perimeter and another setting up an ambush for the player's forces.

The decompositions in the example are partial; they do not specify how `unit1` gets to `building14` or how `unit2` and `unit3` get to their assigned stations. That is, `Defend(unit1, building14)` has a precondition that `unit1` is at the building that is not specified a-priori. The decomposition is filled in by satisfying the open preconditions on actions in the sub-plan. The complete opponent AI plan is shown in Figure 2. It is beneficial for the Strategic AI to do some high-level path planning even though the Tactical AI could accomplish this without guidance so that the planner can heuristically evaluate the effectiveness of certain paths based on the global impact of the entire strategic plan.

Interfacing Strategic AI with Full Spectrum Command

Once a strategic-level plan has been constructed, the AI opponent must execute it. Full Spectrum Command uses an execution matrix where each row represents an opponent force unit. The strategic plan is converted to an execution matrix by discarding the abstract operators and collecting the primitive operators for each unit. The total ordering of primitive operators for each unit is preserved. Thus {6, 2} is the sequence in the matrix for `unit1` (see Figure 2). Each unit's sequence is guaranteed to execute correctly (assuming no unanticipated interference, e.g. from the player's forces) by virtue that the strategic plan is sound[2].

Limitations of Deliberative Planning

Full Spectrum Command provides a strategic planning phase that allows the AI opponent sufficient time to build a strategic plan before the game begins. The tactical, reactive AI substrate means that much uncertainty can be handled by unit entities without replanning and allows for higher-level strategic planning, reducing the frequency that plans fail. The computational complexity of partial-order planning, however, is problematic when plan failure does occur because a new plan must be built while game execution is ongoing. Application of decomposition rules increases efficiency. However, to make replanning feasible, the AI opponent can build contingency plans ahead of time for the (likely) event of plan failure. Young et al. (2004) demonstrate a technique for preemptively building contingency plans for possible failures before the execution phase begins in the context of interactive stories.

[2] Soundess is a property of partial-order plans such that a sound plan is one that is guaranteed to execute correctly in a world with no uncertainty. DPOCL plans are sound (Young, Pollack, and Moore, 1994).

Reactive Planning: Tactical AI

The Tactical AI module addresses the execution-related details of the plans created in the Strategic AI. These plans specifically target unit-level (platoon) tasks that can then be decomposed into single entity (soldier) actions in the game, such as `Move-To`, `Fire-At`, and `Change-Stance`. Deliberative planners use an abstracted model of the task and environment to create a plan in which all the necessary steps are represented. Executing these steps in the actual game environment and responding to the player's actions and environmental cues is the role of the Tactical AI. In addition, the decisions required to execute entity actions must occur very quickly after the triggering cues in order to maintain a synchronized, immersive game experience. This requires a more reactive approach not supported by most deliberative planners. Full Spectrum Command includes an AI system that uses a hierarchy of C++ objects to reactively execute the player's (or mission designer's) plan (van Lent, Fisher, and Mancuso 2004). While this hard-coded approach is reliable and makes quality assurance easier, it lacks the variability and unpredictability in entity actions that can surprise and entertain the player across multiple game sessions.

There are many ways of achieving agent reactivity. Examples include finite state machines, rule-based systems, reactive planners (e.g. Firby 1989), and cognitive models (e.g. Lehman, Laird, and Rosenbloom 1998). Due to cognitive plausibility and historical success modeling tactical military behavior, we chose Soar (Lehman, Laird, and Rosenbloom 1998) as the core of the Tactical AI system. In addition to remaining focused on its current goal, Soar is also distractible – or interruptible – reacting to changes in its environment such as unexpected enemy positions. Because Soar is not constrained to a linear search through a single problem space, it may change context in reaction to new or unexpected events in the environment (Newell 1990). Soar maintains a goal stack for each unit which dynamically grows and shrinks in response to orders from the Strategic AI and "distractions" from the environment. Each Soar rule is capable of firing at any time, allowing deviations from the current deliberative tasks due to changes in the environment.

The traditional approach to controlling multiple units or entities with Soar agents would involve instantiating one instance of the Soar architecture per unit or entity. This approach allows each unit to perceive and act in the world

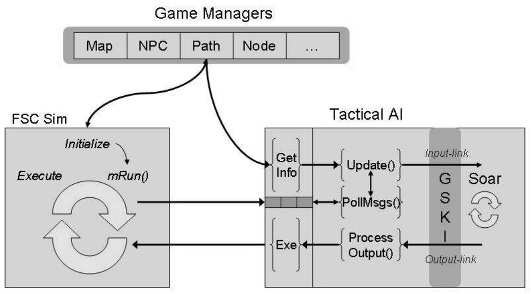

Figure 3: The tactical AI interface.

independently and to use Soar's internal operator stack to encode their goals. However, this approach involves a more complex interface to the game and makes communication between units and entities more difficult (although possibly more realistic). Instead, the Tactical AI system uses a *Forest of Goals* approach (Taylor and Wray 2004). In this approach one Soar agent (representing the overall commander) controls each unit and entity and maintains a separate goal stack (not using Soar's operator stack) for each. This approach has three advantages. First, it supports a hierarchical, parent-child task structure while allowing certain child nodes to act completely reactively. Second, the tasks that can execute reactively can also be triggered in a goal-directed fashion to achieve the Strategic AI's goals. Thus, an entity move command might be a reaction to incoming enemy fire or a part of a move-to-building plan step. Finally, the Forest of Goals approach allows the system to manage many more units than if each unit were controlled by its own instance of Soar.

The plan-based goals given by the Strategic AI are broken down into unit directives which are in turn broken down into entity level directives. Certain tasks also have the ability to fire reactively and independently of the current goal-directed task sequence. In this way the Tactical AI system will be able to respond very quickly to unexpected events such as enemy fire without having to go through a deliberative planning phase. After responding reactively it will be able to either revert to executing its prior plan step or run through the goal-decision process again to select an alternative.

Tactical AI Example

Continuing with the example presented in the Strategic AI section, suppose the OPFOR plan has decomposed securing the base into `Defend(unit1, building14)` and `SecurePerimeter(opfor)`. `SecurePerimeter` is decomposed into `Patrol(unit2, patrolpath2)` and `Ambush(unit3, region3)`. Full Spectrum Command assigns tasks at the platoon level, which in this case would be Defend, Patrol, and Ambush. It is at this point that the Strategic AI populates FSC's task stack and the Tactical AI takes these tasks as input. Once inside of Soar, the decomposition of these three tasks can occur. For example, the Patrol action will decompose into a series of `Move-To(unit, checkpoint)` actions. From here, the unit move-to's are further decomposed into `Move-To(entity, checkpoint)`, which is an action that the game can interpret and execute through the internal path planning mechanism. If there was no additional input from the game (i.e. state updates), it is at this point that the action is sent to the game for execution.

The introduction of variability comes with sensor information updating the Tactical AI's internal state. While a platoon-level task is decomposed, sensor information is arriving from the game and also populating the input-link. It is this state information that dictates how Soar executes the unit-level tasks, all the time polling the game for additional updates that can change the current goal stack as well as fire reactive actions.

Interfacing Tactical AI with Full Spectrum Command

The interface of Soar with FSC is based off an event-driven cycle, whereby messages from the simulation dictate how the AI makes decisions. Traditionally, games and simulations use a time-based ("tick") cycle in which the AI takes input and sends output back to the game for execution at some time interval. This event-driven approach reduces the amount of information that is sent to Soar and reduces extraneous state information that can adversely increase decision cycle time slices.

The complete flow of information into and out of the Tactical AI system is shown in Figure 3. The General Soar Kernel Interface (GSKI) is used as the middleware between the Soar kernel and Full Spectrum Command as it provides a clean and efficient API for working memory sensor and action I/O. Because an important aspect of games is a smooth frame-rate it was decided to keep the AI in a separate, asynchronous thread to reduce any delays in the processing of actions. Each time a message (e.g. `NPCMoved`, `NPCArrived`) arrives in the AI thread, the interface pushes it through the input-link and updates the state of the world through sensors. These sensors include unit/entity positions, current activity (moving, firing), task (ambush, patrol), and terrain information (building locations, exits, rooms). This information is encoded in working memory and used in the next set of decision cycles. After completing a decision cycle entity-level actions are sent out on the output-link directly to the game for execution. The entity-level actions include, but are not limited to, move, fire, stance, follow, and stop.

Limitations of Reactive Planning

It should be noted that the AI is inherently limited by the simulation it is tied to. What is and is not simulated should be taken into consideration when designing the available actions to agents. A good example of this is the fact that our Tactical AI does not contain the concept of suppressing fire due to the fact that it is not simulated. This concept could easily be added but it would likely result in an AI that performs worse than before since it will have expected outcomes that are vastly different from what the simulation will deliver.

Status and Next Steps

The current instantiation of the Adaptive Opponent Architecture consists of the Strategic AI module (based on the DPOCL planner) and the Tactical AI module (based on Soar) integrated and operating with Full Spectrum Command. The Strategic AI system takes input from FSC, generates a plan (see Figure 2 for an example), and populates FSC's internal task representation with that plan. Either FSC's pre-existing AI system or the Tactical AI module (based on Soar) can then execute this plan. Currently the implementation of unit and entity-level tasks is the weakest element of the system. Six entity-level actions have been implemented from a total of three platoon-level tasks, six unit-level tasks and seven entity-level tasks. However, a full end-to-end scenario (with limited Tactical AI tasks) using both the Strategic and Tactical AI modules has been demonstrated.

Immediate next steps include completing the Tactical AI task implementation, fully implementing strategic AI replanning, and conducting some evaluations of the Adaptive Opponent Architecture. One planned evaluation will compare the pre-existing game-industry AI system with the more research-influenced Tactical AI system. Criteria for comparison will include computational and memory efficiency, approximate development time, variability of behavior, and easy of extension. Another planned comparison will compare the range of plans generated by the Strategic AI to the range of plans generated by human players and mission designers.

Longer term next steps will include the investigation of the Automated Director module and the application of the Adaptive Opponent Architecture to other game environments. In the current year the Adaptive Opponent Architecture will be integrated with a second game-based training simulation, the Joint Fires and Effects Training System, currently in use to train soldiers at Ft. Sill.

Conclusions

The Adaptive Opponent Architecture is a three-tiered approach to making an AI opponent more unpredictable and adaptable while still achieving certain pedagogical or entertainment goals. This approach also increases replayability in training and entertainment games. The Automated Director focuses on adapting the experience over many sessions. The Strategic AI focuses on adapting the high-level strategic decisions of the AI opponent. The Tactical AI focuses on adaptive behavior at the unit execution level.

The Adaptive Opponent Architecture is being applied to Full Spectrum Command, a game-based training aid that contains elements of both training and entertainment. As primarily a training-aid, FSC needs to be replayable so that student player is challenged to exercise their cognitive skills instead of learning the script and making choices that are, in general, poor choices but work well against the AI opponent. Replayability is also an important property for games with entertainment value because the player continues to be challenged and motivated.

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.

References

Firby, J.R. 1989. *Adaptive execution in complex dynamic worlds*. Ph.D. dissertation, Yale University.

Knoblock, C.A. 1994. Generating parallel execution plans with a partial-order planner. *Proc. of the 2nd Int. Conf. on Artificial Intelligence Planning Systems*.

Lehman, J., Laird, J., & Rosenbloom, P. 1998. A gentle introduction to soar, an architecture for human cognition. In S. Sternberg & D. Scarborough (Eds.) *Invitation to Cognitive Science (Second Edition), Volume 4: Methods, models, and conceptual issues*. Cambridge, MA: The MIT Press.

Magerko, B. & Laird, J. E., 2003. Building an Interactive Drama Architecture, *1st Int. Conf. on Technologies for Interactive Digital Storytelling and Entertainment*.

Newell, A. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Penberthy, J.S. & Weld, D. 1992. UCPOP: A sound, complete, partial-order planner for ADL. *Proc. of the 3rd Int. Conf. on Knowledge Representation and Reasoning*.

Sacerdoti, E.D. 1977. *A Structure for Plans and Behavior*. New York: Elsevier.

Taylor, G. & Wray, R.E. 2004. *A Behavior Design Pattern Library*. Ann Arbor, MI: Soar Technology, Inc.

van Lent, M., Fisher, B. and Mancuso, M., 2004. "An Explainable Artificial Intelligence System for Small-unit Tactical Behavior", *The 16th Innovative Applications of Artificial Intelligence Conference*.

Young, R.M., Pollack, M.E., & Moore, J.D. 1994. Decomposition and causality in partial-order planning. *Proceedings of the 2nd Int. Conf. on Artificial Intelligence and Planning Systems*.

Young, R.M., Riedl, M.O., Branly, M., Jhala, A., Martin, R.J., & Saretto, C.J. 2004. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1, 51-70.